Stochastic Subgraph Neighborhood Pooling for Subgraph Classification

Shweta Ann Jacob
shweta.jacob@ontariotechu.net
Ontario Tech University

Paul Louis
paul.louis@ontariotechu.net
Ontario Tech University

Amirali Salehi-Abari
abari@ontariotechu.ca
Ontario Tech University

ABSTRACT

Subgraph classification is an emerging field in graph representation learning where the task is to classify a group of nodes (i.e., a subgraph) within a graph (e.g., identifying rare diseases given a collection of phenotypes). Graph neural network (GNN) solutions for node, link, and graph tasks fail to perform well on subgraph classification as they do not capture the external topology of the subgraph (i.e., how the subgraph is located within the larger graph). The current state-of-the-art models address this shortcoming through either labeling tricks or multiple message-passing channels, which are computationally expensive and not scalable to large graphs. To address the scalability issue while maintaining generalization, we propose *Stochastic Subgraph Neighborhood Pooling (SSNP)*, which jointly aggregates the subgraph and its neighborhood (i.e., external topology) information while removing the need for any computationally expensive operations (e.g. labeling tricks). Our extensive experiments demonstrate that *SSNP* outperforms or is comparable to state-of-the-art methods while being up to 13× faster in runtime.

CCS CONCEPTS

• **Computing methodologies → Neural networks**.

KEYWORDS

Graph Neural Networks, Subgraph Classification.

ACM Reference Format:
Shweta Ann Jacob, Paul Louis, and Amirali Salehi-Abari. 2023. Stochastic Subgraph Neighborhood Pooling for Subgraph Classification. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3583780.3615227

1 INTRODUCTION

Graph-structured data is prevalent in many domains such as social networks [1], recommender systems [21], and drug discovery [9]. Graph representation learning has continuously progressed in recent years with the advent of more expressive graph neural networks (GNNs) [7, 11, 17, 19]. Subgraph classification is an emerging problem in GRL where one intends to predict the properties associated with a group of nodes (i.e., a *subgraph*) of the larger observed *base* graph [3, 18]. Subgraph classification finds application in various domains such as finding toxic (or violence-inciting) communities in social networks, group recommendation, drug discovery, diagnosis of rare diseases, etc. As subgraphs may contain any number of nodes ranging from one node to all nodes of the base graph, typical downstream tasks (e.g., node classification, link prediction, or graph classification) can be considered as specific instances of subgraph classification.

Subgraph classification requires solutions that can learn, combine, and contrast topological properties and the connectivity between the nodes within and outside the subgraph. Learning these complex intra-connectivity and inter-connectivity patterns of the subgraph and the base graph renders this problem challenging. The direct application of traditional GNNs is an inferior solution as they ignore the external topology of the subgraph [18]. Recent state-of-the-art work (e.g., GLASS [18] and SubGNN [3]) alleviates this shortcoming of the lack of external topology information through the use of labeling tricks [18] or artificially-crafted message passing channels [3], which are computationally intensive, especially when dealing with larger (sub)graphs.

We introduce a computationally-light model for subgraph classification that operates on the original graph while capturing external topologies of subgraphs. The crux of our solution is *Stochastic Subgraph Neighborhood Pooling (SSNP)*, which aggregates the node representations of the subgraph and its neighborhood to generate the topologically-rich subgraph embeddings. The addition of subgraph neighborhood information in *SSNP* facilitates capturing the external topology of a subgraph within a base graph. To prevent neighborhood explosion in large graphs, our *SSNP* uses random walks to sample the neighborhood of subgraphs. For a higher extent of scalability, our sampling method can be conducted multiple times in a pre-processing stage as a data augmentation strategy, to create multiple sparse views of the subgraph neighborhood. Our comprehensive experiments on real-world datasets show that our solution offers a runtime speedup up to 13× while matching or outperforming various state-of-the-art baselines.

2 BACKGROUND AND RELATED WORK

Let $G = (V, E)$ represent a simple, undirected graph where $V = \{1, \ldots, n\}$ is the set of nodes (e.g., users, proteins, etc.), and $E \subseteq V \times V$ represents the edge set (e.g., friendships, interactions, etc.). We sometimes represent G by the adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ where $a_{ij} = 1$ if an edge exists between nodes i and j, and 0 otherwise. We also assume each node $i \in V$ possesses a d-dimensional feature $\mathbf{x}_i \in \mathbb{R}^d$ (e.g., user information, protein characteristics). We sometimes stack all nodal features, row-by-row in the feature matrix $\mathbf{X}$ whose i-th row contains $\mathbf{x}_i$. We consider a subgraph $S = (V_S, E_S)$ in base graph G where $V_S \subseteq V$ and $E_S \subseteq (V_S \times V_S) \cap E$.

CIKM '23, October 21–25, 2023, Birmingham, United Kingdom

ACM ISBN 979-8-4007-0124-5/23/10…$15.00
https://doi.org/10.1145/3583780.3615227

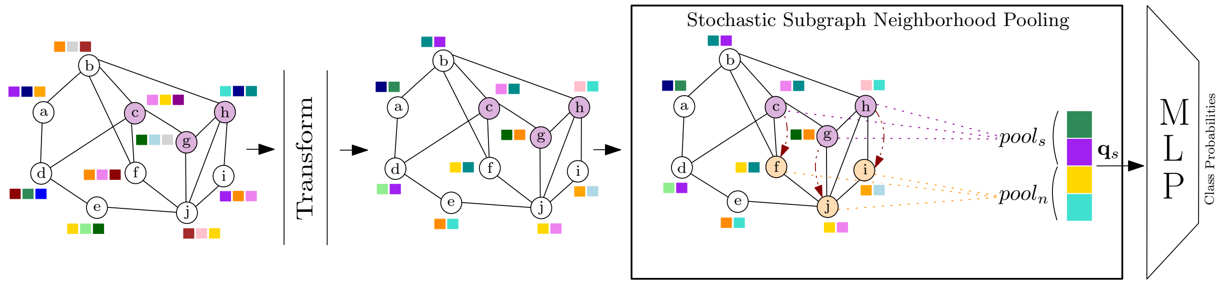

Figure 1: Architecture of our model. Subgraph nodes are shaded in purple. The initial node features are transformed using transformation layers. The stochastic subgraph neighborhood pooling $\text{pool}_{\text{SSNP}}$ is applied in multiple steps. The subgraph neighborhood nodes (shaded in brown) are sampled by rooted random walks (red dashed arrows). The subgraph and its sampled neighborhood are separately pooled by $pool_s$ and $pool_n$, which are simple graph pooling operators (e.g., mean, sum, etc.). The pooling outputs are concatenated to form the subgraph representation $\mathbf{q}_s$ and is passed to an MLP to generate class probabilities.

Subgraph Classification Problem. The goal is to learn a mapping function $f(G, \mathbf{X}, S)$ which takes the base graph G, its node feature matrix $\mathbf{X}$, and a subgraph S as an input, and outputs the subgraph class label $y \in \{1, \ldots, C\}$, where C is the number of classes.

Related Work SubGNN [3], an early work on subgraph classification, samples anchor patches from the base graph and propagates messages between anchors to the subgraph in multiple channels to learn the internal and external topologies of subgraphs. The current state-of-the-art GLASS [18] uses the zero-one (or its variant max zero-one) labeling trick [24] to differentiate between the internal and external nodes of a subgraph and thereby encode various topological properties of the subgraph. Sub2Vec [2], deployed for community detection and graph classification, is also adopted for the subgraph classification task. PADEL [12] uses data augmentation and contrastive learning techniques along with position encodings of nodes during message passing. These methods are either fast but suboptimal (e.g., Sub2Vec [2]) or effective but computationally-expensive (e.g., SubGNN and GLASS). Our work offers a fast yet effective solution for subgraph classification.

3 PROPOSED SOLUTION

The steps of our proposed solution are depicted in Figure 1. The initial node features $\mathbf{X}$ are transformed to learned embeddings $\mathbf{Z} = f_T(G, \mathbf{X})$. The transformation function f_T can be multi-layers of graph convolutions for feature smoothing or a simple multi-layer perceptron for dimensionality reduction. After obtaining node embeddings $\mathbf{Z}$, our proposed $\text{pool}_{\text{SSNP}}$ function is used to aggregate the target subgraph's internal and external topological properties into a subgraph representation:

$$\mathbf{q}_s = \text{pool}_{\text{SSNP}}(\mathbf{Z}, G, S) \tag{1}$$

This subgraph representation $\mathbf{q}_s$ is fed to an MLP to output class probabilities for the subgraph classification task. The MLP also learns how to mix the pooled subgraph and its neighborhood representations. Our proposed solution does not require computationally-expensive labeling tricks (as opposed to GLASS [18]), or artificially-crafted message passing channels (as opposed to SubGNN [3]).

Subgraph Neighborhood Pooling and Variants. Our proposed pooling is built on the idea that the representations of subgraphs and their neighborhoods are both important for capturing the internal and external topology of subgraphs. We first define the h-hop subgraph neighborhood as:

Definition 1 (h-hop Subgraph Neighborhood). *Given the base graph $G = (V, E)$ and its subgraph $S = (V_S, E_S)$, the h-hop subgraph neighborhood $N_S^{(h)}$ is the induced subgraph created from the node set $\{j \in V_N | min_{i \in S} d(i, j) \leq h\}$, where $d(i, j)$ is the geodesic distance between node i and j, and $V_N = V \setminus V_S$ are nodes of G that do not belong to S.*

In simple words, the h-hop subgraph neighborhood is the subgraph of G whose nodes do not belong to S and are within a distance of h to at least one of the nodes of S. Our h-hop subgraph neighborhood can be viewed as an extension of the enclosing subgraphs for pair of nodes [22] but with two distinctions: (i) the h-hop neighborhood is defined for any subgraph size (rather than just a pair of nodes) and (ii) the subgraph S is excluded from its neighborhood subgraph. Given this h-hop subgraph neighborhood definition, we first consider a simple *subgraph neighborhood pooling*:

$$\text{pool}_{\text{SNP}}(\mathbf{Z}, G, S, h) = \text{pool}_s(\mathbf{Z}_S, S) \oplus \text{pool}_n\left(\mathbf{Z}_N, N_S^{(h)}\right), \tag{2}$$

where $\mathbf{Z}_S$ and $\mathbf{Z}_N$ denote the matrix node embeddings of the subgraph S and its neighborhood $N_S^{(h)}$. Here, $\oplus$ is the concatenation operator, and $pool_s$ and $pool_n$ can be any order invariant graph pooling function (e.g., sum, mean, max, or SortPooling [23]). The main idea here is to treat the subgraph and its neighborhood as two separate graphs, pool their information, and then concatenate their representations to capture the topology of the subgraph. Current subgraph representation learning models (e.g., GLASS, SubGNN) only use $pool_s$, while ignoring the rich information of the neighborhood subgraph. However, consuming the complete subgraph neighborhoods is problematic as these neighborhoods can become extremely large with many uninformative and noisy nodes, thus hindering the model's learning and slowing down the running time. To overcome this limitation, we define:

Definition 2 (h-hop Sparsified Subgraph Neighborhood). *Given the base graph $G = (V, E)$ and subgraph $S = (V_S, E_S)$, we define a h-hop sparsified subgraph neighborhood $\hat{N}_S^{(h,k)}$, as the subgraph induced from the nodes in $\hat{V}_S^{(h,k)} \in \{W_S^{(h,k)} \setminus V_S\}$, where $W_S^{(h,k)}$ is a set of nodes visited by k many h-length random-walk(s) from the nodes in V_S.*

Compared to the exact subgraph neighborhood, the sparse neighborhoods are bounded by hk (i.e., the product of the length and

Table 1: Statistics of all real-world datasets.

	# nodes	# edges	# Subgraphs	# Classes	Multi-label
ppi-bp	17080	316951	1591	6	No
hpo-metab	14587	3238174	2400	6	No
hpo-neuro	14587	3238174	4000	10	Yes
em-user	57333	4573417	324	2	No

number of random walks). The rooted random walks allow sampling "important" external nodes to a subgraph (similar to rooted PageRank [4]), which encapsulates information on the border neighborhood. The randomness in the neighborhood subgraph also adds some regularization effect to the training of the model (similar to what was observed in ScaLed [13]). Given the computational and learning advantages of sparsified neighborhood subgraphs, we introduce *stochastic subgraph neighborhood pooling (SSNP)* by a slight modification of Eq. 2:

$$\text{pool}_{\text{SSNP}}(\mathbf{Z}, G, S, h, k) = \text{pool}_s(\mathbf{Z}_S, S) \oplus \text{pool}_n\left(\mathbf{Z}_{\hat{N}}, \hat{N}_S^{(h,k)}\right), \quad (3)$$

where $\mathbf{Z}_S$ and $\mathbf{Z}_{\hat{N}}$ denote the matrix node embeddings of the subgraph S and its sparsified neighborhood $\hat{N}_S^{(h,k)}$ by k-many h-length random walks. In the absence of distinguishing node features, our model with $\text{pool}_{\text{SSNP}}$ is potentially more expressive than a plain GNN (which only pools subgraph without its neighbors).

Random walks are effective in approximating and sparsifying subgraphs around a node [13, 20]. However, the sampling of the sparsified subgraph neighborhood in each training epoch might introduce undesirable instability and stochasticity in gradient computations and optimization procedures. To address this instability and reduce the sampling overhead, we introduce three different stochastic subgraph neighborhood sampling strategies.

Online Stochastic Views (OV): The h-hop sparsified subgraph neighborhood is sampled in each epoch. This stochasticity over training intends to add regularization to the model but might have undesirable outcomes of gradient instability. Also, the epoch-level sampling adds computational overhead to the training.

Pre-processed Stochastic Views (PV): To overcome the additional overhead created by sampling during training, we propose *pre-processed stochastic views (PV)* for which a fixed number n_v of sparsified subgraph neighborhood is sampled for each subgraph during preprocessing. These sampled neighborhood subgraphs can be viewed as data augmentation that provides n_v views of the subgraph neighborhood. Similar to other data augmentation strategies, PV improves the generalization of our model and makes it more robust to noise and overfitting. However, the dataset size and training time grows linearly with the number of views n_v.

Pre-processed Online Stochastic Views (POV): To reduce the training time on the augmented datasets, we propose *pre-processed online stochastic views (POV)* that leverages both the pre-processed and online subgraph neighborhood sampling method. In the preprocessing stage similar to PV, POV creates n_v multiple sparsified subgraph neighborhoods for each subgraph. But, during each training epoch, for each subgraph only n_{ve} of the precomputed views are randomly sampled. POV allows data augmentation with multiple views while keeping the number of training instances per epoch independent of the number of views n_v.

Table 2: The micro-F1 scores (average of 10 runs) for all models. The top 3 are First, Second, and Third.

Model	ppi-bp	hpo-metab	hpo-neuro	em-user
MLP	0.445±0.003	0.386±0.011	0.404±0.006	0.524±0.019
GBDT	0.446±0.000	0.404±0.000	0.513±0.000	0.694±0.000
GNN-plain	0.613±0.009	0.597±0.012	0.668±0.007	0.847±0.021
Sub2Vec	0.388±0.001	0.472±0.010	0.618±0.003	0.779±0.013
GNN-seg	0.361±0.008	0.542±0.009	0.647±0.001	0.725±0.003
SubGNN	0.599±0.008	0.537±0.008	0.644±0.006	0.816±0.013
GLASS	0.618±0.006	0.598±0.014	0.675±0.007	0.884±0.008
SSNP-MLP	0.591±0.006	0.571±0.006	0.669±0.004	0.853±0.012
SSNP-GCN	0.607±0.005	0.553±0.011	0.667±0.003	0.843±0.014
SSNP-NN	0.636±0.007	0.587±0.010	0.682±0.004	0.888±0.005

Table 3: Our model vs GLASS: dataset preprocessing time, training and inference time per epoch and average runtime in seconds (mean over 10 runs). The min/max speedup is the ratio of time taken by GLASS to the time of the slowest/fastest *SSNP* model (in italics/bold).

	ppi-bp			
Model	Preproc.	Training	Inference	Runtime
SSNP-NN	*8.94±0.54*	0.38±0.02	0.02±0.00	129.35±3.27
SSNP-GCN	8.89±0.71	*0.42±0.02*	*0.03±0.00*	*142.38±3.85*
SSNP-MLP	**8.79±0.63**	**0.06±0.02**	**0.00±0.00**	**16.00±0.94**
GLASS	3.93±0.10	0.78±0.02	0.05±0.00	207.99±24.76
Speedup	0.44/0.45	1.86/13	1.67/25	1.46/13

	hpo-metab			
Model	Preproc.	Training	Inference	Runtime
SSNP-NN	25.20±0.84	0.73±0.02	0.05±0.001	159.56±18.86
SSNP-GCN	*26.13±1.53*	*0.94±0.03*	*0.06±0.00*	*209.20±43.15*
SSNP-MLP	**24.81±0.75**	**0.10±0.02**	**0.00±0.00**	**35.00±1.72**
GLASS	15.99±0.88	2.15±0.03	0.13±0.00	239.48±33.22
Speedup	0.61/0.64	2.29/21.5	2.17/43.33	1.14/6.84

	hpo-neuro			
Model	Preproc.	Training	Inference	Runtime
SSNP-NN	*29.67±1.54*	1.27±0.03	0.05±0.00	202.28±26.01
SSNP-GCN	**28.14±0.81**	*1.58±0.05*	*0.06±0.00*	*344.14±44.14*
SSNP-MLP	28.37±1.13	**0.21±0.01**	**0.01±0.00**	**50.00±1.05**
GLASS	16.56±0.84	4.20±0.04	0.25±0.00	511.54±94.40
Speedup	0.56/0.59	2.66/20	4.17/25	1.49/10.23

	em-user			
Model	Preproc.	Training	Inference	Runtime
SSNP-NN	*27.93±1.41*	*3.00±0.04*	*0.08±0.00*	*156.81±32.10*
SSNP-GCN	27.62±0.91	1.61±0.04	*0.08±0.00*	108.30±18.62
SSNP-MLP	**27.52±1.54**	**0.16±0.01**	**0.00±0.00**	**44.00±1.71**
GLASS	25.11±1.61	4.93±0.04	0.56±0.00	212.28±23.51
Speedup	0.90/0.91	1.64/30.81	7/140	1.35/4.82

4 EXPERIMENTS

We compare our solutions against different baselines on four real-world datasets to evaluate their performance and scalability.[1]

Datasets. We perform experiments on four publicly-available real-world datasets that have been the subject of other studies [3, 18] (see

[1]More experimental results are available in the longer version of this paper [8].

Table 1). We follow the same dataset split as GLASS [18]: 80/10/10 for train, validation, and test splits.

Baselines. Our baselines include the state-of-the-art GLASS [18], SubGNN [3], graph-agnostic MLP, GBDT [6], GNN-plain, Sub2Vec [2], and GNN-seg [18]. The baseline results, except for GLASS, are taken from [18]. We rerun GLASS to capture the runtime values and verify that our setup is identical to their setup of reported results.

Setup. For GLASS, we use its best-performing reported hyperparameters. For our model, we set the transformation functions to either MLP, Nested Network (NN) [15], or Graph Convolution Network (GCN) [11], and the corresponding models are called *SSNP*-MLP, *SSNP*-NN and *SSNP*-GCN, respectively. We always set the number of walks per node $k = 1$, and let the pooling method for the subgraph and neighborhood be the same (i.e., $\text{pool}_s = \text{pool}_n$). Unless noted otherwise, we use the POV for creating subgraph neighborhood views with the number of views $n_v = 20$ and the number of views per epoch $n_{ve} = 5$. The other hyperparameters are searched over validation datasets to maximize micro-F1 scores. The search spaces are $\text{pool}_s \in \{sum, size\}$, length of walks $h \in \{1, 5\}$, and the number of transformation layers $\in \{1, 2, 3\}$. As with GLASS, we set the learning rate to 0.0005 for ppi-bp, 0.002 for hpo-neuro and 0.001 for hpo-metab and em-user, and use pre-trained 64-dimensional nodal features as the initial node features. We use Adam optimizer [10] paired with ReduceLROnPlateau learning rate scheduler. We set dropout [16] to 0.5 for all models. We use a single-layer MLP to output the class probabilities and the cross-entropy loss in our model. Our model is implemented in PyTorch Geometric [5] and PyTorch [14].[2] Our results are reported with an average F1-score over 10 runs with different random seeds.

Results: F1-Score. Table 2 shows the mean micro-F1 results for all datasets. On ppi-bp, hpo-neuro, and em-user, our *SSNP*-NN model outperforms all others with a gain of 0.018, 0.011, and 0.004, respectively. For hpo-metab, *SSNP*-NN ranks third with a small margin of 0.011 compared to GLASS ranked first. This relatively low performance could be attributed to the fact that subgraphs in hpo-metab are dense and therefore, do not need external topological information. Surprisingly, both *SSNP*-NN and *SSNP*-GCN outperform SubGNN across all the datasets. *SSNP*-MLP (even without message passing) outperforms SubGNN in all datasets except for ppi-bp for which it has a comparable result. *SSNP*-MLP also appears to be relatively competitive by being ranked third in hpo-neuro and em-user as well as surpassing MLP by a significant margin. These results provide strong evidence in demonstrating how effective neighborhood pooling is for subgraph classification.

Results: Runtime. The average runtimes are reported in Table 3.[3] Our models for all datasets require at most twice the preprocessing times of GLASS due to the sampling of multiple subgraph neighborhood views. However, in return, the training and inference times are 1.5-137× faster depending on the model variations and datasets. Our best-performing *SSNP*-NN has a training speedup of 1.5-3.3× (min. for em-user and max. for hpo-neuro) and an inference speedup of 2.5-7× (min. for ppi-bp and max. for em-user). *SSNP*-MLP is the fastest with maximum training and inference (resp.) speedups of 30× and 140× (resp.) in em-user. Cross-examining Tables 2 and 3, we observe that *SSNP*-MLP vs. GLASS has a speedup of 13-140× (for both training and inference) with a small negative gain of 0.006–0.031 in F1-score and a runtime speedup of 4.8-13× (min. for em-user and max. for ppi-bp).

[2] Our code is available at https://github.com/shweta-jacob/SSNP. We run our experiments on servers with 50 CPUs, 377GB RAM, and 11GB GPUs.

[3] SubGNN with suboptimal runtime compared to GLASS [18], is excluded.

Table 4: F1-score (avg. over 5 runs), various sampling strategies, *SSNP*-NN.

Strategy	ppi-bp	hpo-metab	hpo-neuro	em-user
OV	0.527±0.008	0.443±0.055	0.681±0.002	**0.906±0.009**
PV (5 views)	0.628±0.007	0.569±0.015	0.680±0.003	0.878±0.015
PV (20 views)	0.635±0.003	0.553±0.013	0.671±0.003	0.902±0.007
POV	**0.638±0.008**	**0.577±0.017**	**0.686±0.004**	0.902±0.007

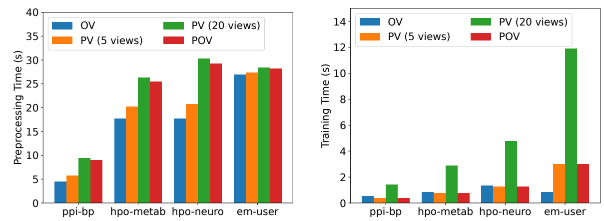

Figure 2: The effect of sampling strategies on pre-processing time (left) and training time per epoch (right) in *SSNP*-NN.

Results: Stochastic Sampling Strategies. We intend to study the effect of various stochastic sampling strategies (i.e., OV, PV, and POV) on both F1-score and runtime. For these experiments, we set the number of views per epoch n_v to 1 for OV, to 5 or 20 for PV, and to 20 for POV. For POV, we set the number of views per epoch $n_{ve} = 5$. For all datasets (except em-user), POV provides the best F1-scores (see Table 4). For em-user, OV suppresses POV with a small margin of 0.004. The training time for OV in ppi-bp, hpo-metab and hpo-neuro is higher than PV with 5 views and POV (see Figure 2). However, pre-processing of OV is faster than all other sampling strategies. Although the pre-processing times are comparable for PVs and POV, POV offers much faster training time and a higher F1-score (see Table 4). In hpo-metab and hpo-neuro, the F1 score of PV with 5 views is higher than that of PV with 20 views, implying that a higher number of views does not necessarily improve performance for PV. However, POV, with 5 views per epoch and a total of 20 views, has the highest F1 score meaning the stochasticity across epochs improves generalization for our model.

5 CONCLUSIONS AND FUTURE WORK

The state-of-the-art subgraph classification solutions are not scalable due to the use of labeling tricks or artificial message-passing channels. We propose a simple yet powerful model that has our proposed stochastic subgraph neighborhood pooling (*SSNP*) in its core. Leveraging *SSNP*, our model learns the internal connectivity and border neighborhood of subgraphs. We also present simple data augmentation techniques that help to improve the generalization of our model. Our model combined with our augmentation techniques outperforms or match current state-of-the-art subgraph classification models with a runtime speedup of up to 13×. For future work, we plan to explore alternative ways to approximate neighborhood subgraphs and perform contrastive learning on the different views of neighborhood subgraphs.

REFERENCES

[1] Lada A Adamic and Eytan Adar. 2003. Friends and Neighbors on the Web. *Social Networks* (2003), 211–230.

[2] Bijaya Adhikari, Yao Zhang, Naren Ramakrishnan, and B Aditya Prakash. 2018. Sub2vec: Feature learning for subgraphs. In *Advances in Knowledge Discovery and Data Mining: 22nd Pacific-Asia Conference*. Springer, 170–182.

[3] Emily Alsentzer, Samuel Finlayson, Michelle Li, and Marinka Zitnik. 2020. Subgraph Neural Networks. *Advances in Neural Information Processing Systems* (2020), 8017–8029.

[4] Sergey Brin and Lawrence Page. 2012. Reprint of: The anatomy of a large-scale hypertextual web search engine. *Computer Networks* (2012), 3825–3833.

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

[6] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

[7] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 1025–1035.

[8] Shweta Ann Jacob, Paul Louis, and Amirali Salehi-Abari. 2023. Stochastic Subgraph Neighborhood Pooling for Subgraph Classification. *arXiv preprint arXiv:2304.08556* (2023).

[9] Dejun Jiang, Zhenxing Wu, Chang-Yu Hsieh, Guangyong Chen, Ben Liao, Zhe Wang, Chao Shen, Dongsheng Cao, Jian Wu, and Tingjun Hou. 2021. Could graph neural networks learn better molecular representation for drug discovery? A comparison study of descriptor-based and graph-based models. *Journal of cheminformatics* (2021), 1–23.

[10] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*.

[11] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*.

[12] Chang Liu, Yuwen Yang, Zhe Xie, Hongtao Lu, and Yue Ding. 2023. Position-Aware Subgraph Neural Networks with Data-Efficient Learning. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*. 643–651.

[13] Paul Louis, Shweta Ann Jacob, and Amirali Salehi-Abari. 2022. Sampling Enclosing Subgraphs for Link Prediction. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 4269–4273.

[14] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*. Article 721, 12 pages.

[15] Xiang Song, Runjie Ma, Jiahang Li, Muhan Zhang, and David Paul Wipf. 2021. Network in graph neural network. *arXiv preprint arXiv:2111.11638* (2021).

[16] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* (2014), 1929–1958.

[17] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

[18] Xiyuan Wang and Muhan Zhang. 2021. GLASS: GNN with Labeling Tricks for Subgraph Representation Learning. In *International Conference on Learning Representations*.

[19] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *International Conference on Learning Representations*.

[20] Haoteng Yin, Muhan Zhang, Yanbang Wang, Jianguo Wang, and Pan Li. 2022. Algorithm and System Co-design for Efficient Subgraph-based Graph Representation Learning. *arXiv preprint arXiv:2202.13538* (2022).

[21] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 974–983.

[22] Muhan Zhang and Yixin Chen. 2018. Link Prediction Based on Graph Neural Networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 5171–5181.

[23] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. 2018. An End-to-End Deep Learning Architecture for Graph Classification. *Proceedings of the AAAI Conference on Artificial Intelligence* (2018).

[24] Muhan Zhang, Pan Li, Yinglong Xia, Kai Wang, and Long Jin. 2021. Labeling Trick: A Theory of Using Graph Neural Networks for Multi-Node Representation Learning. In *Advances in Neural Information Processing Systems*. 9061–9073.