Subgraph Sampling Strategy for Equivariant Subgraph Aggregation Network Based on Weisfeiler-Lehman Similarity

Yu Tingyang (155141476@link.cuhk.edu.hk)
2022-23 Term 1 Final Year Project Thesis for ESTR4998
Supervisor: Professor Irwin King

Abstract

Subgraph augmentation is a widely used method for graph classification when the original graph can not be efficiently distinguished for network. With the rapid development of message-passing graph neural network (MPNN), many MPNN architectures are designed to process graphs augmented by their subgraphs. In this work, we present a new subgraph sampling strategy EGO+WL based on Weisfeiler-Lehman similarity. It achieves superior classification accuracy on TU datasets compared to the existing state-of-the-art strategies. It further reduces the space complexity up to 55% on dataset IMDB-MULTI compared to the second best strategy and reduces 10-20% training time on dataset NCI1. The code is available at https://github.com/YistYU/ESAN_WLS.

1 Introduction

Message-Passing Neural Network (MPNNs) are the leading Graph Neural Network (GNN) architecture for deep learning because of their strong simlicity. However, [Morris *et al.*, 2019]; [Xu *et al.*, 2018] have shown that these architectures are at most expressive as the Weisfeiler-Lehman (WL) graph isomorphism test ([Weisfeiler and Leman, 1968]). The Weifeiler-Leman (WL) test [Weisfeiler and Leman, 1968] is an classic algorithmic test of graph isomorphism, possibly with categorical node attributes. Although the proposed version of the algorithm is parameterized by dimension k, we only explain the 1-dimensional case, which is mostly discussed in machine learning architectures.

Assume we are given two graphs G and G' as shown in Figure 1 and we would like to test whether they are isomorphic. The key idea of the algorithm is to augment the node labels by the sorted set of node labels of neighborhood nodes, and compress these augmented labels into new, short labels. These steps are then repeated until the node label sets of G and G' differ, or the number of iterations reaches the maximum iteration. We take the graph G in Figure 1 as an example. If G has no node attributes, we set $f(v) = 1$ for all $v \in V(G)$. Then, we update the node attributes in "stages", once for all nodes at each stage. The updated attribute is the pair of itself and the set of attributes of its neighbors. For further iterations, the new attributes may be re-labeled via an injective mapping, *i.e.*, $(1, \{1, 1\}) \to 2$ and $(1, \{1, 1, 1\}) \to 3$.

After a fixed number of iterations, we compare the set of resulting attributes to that from another graph. If two sets differ, then the two graphs are non-isomorphic and are $distinguishable$ by the WL-test. Many indistinguishable non-isomorphic pairs of graphs exist. However, asymptotically, the WL-test uniquely identifies almost all graphs; c.f. [Arvind *et al.*, 2020]. As a consequence, MPNNs cannot distinguish between very simple graphs.

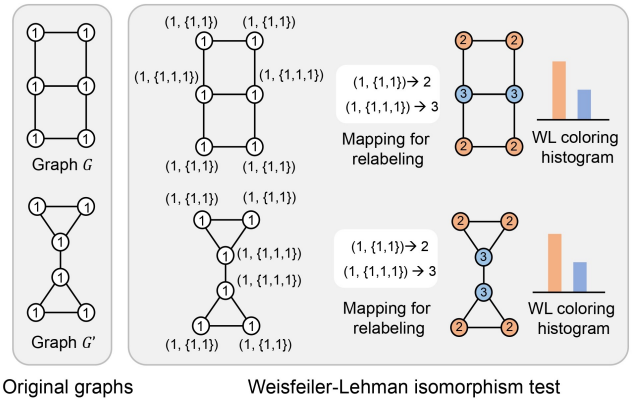

Figure 1: A pair of graphs not distinguishable by the WL test.

In light of this limitation, a question naturally arises: is it possible to improve the expressiveness of MPNNs? Broadly speaking, the efforts in enhancing the expressive power of GNNs can be clustered along three main directions: (1) Aligning to the k-WL hierarchy ([Morris *et al.*, 2019]; [Maron *et al.*, 2019]); (2) Augmenting node features with exogenous identifiers ([Sato, 2020]; [Dasoulas *et al.*, 2019]; [Abboud *et al.*, 2020]); (3) Leveraging on structural information that cannot provably be captured by the WL test ([Bouritsas *et al.*, 2021]; [Thiede *et al.*, 2021]; [de Haan *et al.*, 2020]; [Bodnar *et al.*, 2021]). Bevilacqua *et.al.* further presented Equivariant subgraph aggregation networks (ESAN) which is provably expressive with scalable and non-domain specific policies [Bevilacqua *et al.*, 2021]. ESAN advocated representing each graph as a bag of subgraphs chosen according to

some predefined policy, *e.g.*, all graphs that can be obtained by removing one edge from the original graph. Therefore, the following question occurs: how to effectively select subgraphs for the network?

In ESAN, they presented four strategies as the baselines: node-deleted subgraphs (ND), edge-deleted subgraphs (ED), and ego-networks (EGO, EGO+) as described next. In the **node-deleted policy**, a graph is mapped to the set containing all subgraphs that can be obtained from the original graph by removing a single node. Similarly, the **edge-deleted policy** is defined by removing a single edge. The **ego-networks policy** EGO maps each graph to a set of ego-networks of some specified depth, one for each node in the graph (a k-Ego-network of a node is its k-hop neighbourhood with the induced connectivity). They also consider a variant of the ego-networks policy where the room node holds an identifying feature (EGO+). However, their approach is related to multiple techniques in graph learning.

Introduced for semi-supervised node-classification tasks, DropEdge ([Rong *et al.*, 2019]) can be considered as a stochastic version of the ED policy that processes one single edge-deleted subgraph at a time. Ego-GNNs ([Sandfelder *et al.*, 2021]) resemble our EGO policy, as messages are passed within each node's ego-net, and are aggregated from each ego-net that a node is contained in. ID-GNNs ([You *et al.*, 2021]) resemble the EGO+ policy, the difference being the way in which the root node is identified and the way the information between the ego-nets is combined. RNPGNNs ([Tahmasebi *et al.*, 2020]) learn functions over recursively-defined node-deleted ego-nets with augmented features, which allows for high expressive power for counting substructures and computing other local functions. FactorGCN ([Yang *et al.*, 2020]) is a model that disentangles relations in graph data by learning subgraphs containing edges that capture latent relationships. The graph automorphic equivalence network ([Xu *et al.*, 2021]) compares ego-nets to subgraph templates in a GNN architecture. Furthermore, Hamilton *et al.* ([Hamilton *et al.*, 2017]) introduce a minibatching procedure that samples neighborhoods for each GNN layer - in large part for the sake of efficiently processing large graphs. Other works like Cluster-GCN ([Chiang *et al.*, 2019]) and GraphSAINT ([Zeng *et al.*, 2019]) have followed by developing methods for sampling whole subgraphs at a time, and then processing these subgraphs with a GNN.

We can found that the current subgraph sampling strategies are mostly in a very trivial manner. For gigantic datasets, it could be extremely costly if all the subgraphs are preserved and processed by the existing strategies. In our work, we present subgraph sampling strategies EGO+WL which efficiently select the most representative subgraphs for network training. This paper offers the following main contributions:

(1) We propose new graph sampling strategies EGO+WL based on WL similarity which shows up to 3% improvement on the graph classification accuracy on TU datasets.

(2) EGO+WL has a lower space complexity and time complexity. Our method reduces up to 55% GPU memory consumption and 10-15% network training time compared to the second-best strategy on TU datasets.

(3) EGO+WL can efficiently select the most representative subgraphs. It achieves superior performance than baseline methods using only 50% subgraphs while the baseline methods use all the subgraphs for training.

2 Preliminaries

Problem formulation We assume a standard graph classification/regression setting. We represent a graph with n nodes as a typle $G = (V, A, X)$ where V is the set of nodes of G, $A \in \mathbb{R}^{n\times n}$ is the graph adjacency matrix and $X \in \mathbb{R}^{n\times d}$ is the node feature matrix. We assume each node $v \in V$ is assigned an *attribute* $f(v)$, which is either a categorical variable from a finite set or a vector in $\mathbb{R}^d$. If we update the attribute on v, the original attribute is written as $f^0(v)$ and the successively updated ones as $f^1(v)$, $f^2(v)$, *etc.* The set of nodes adjacent to v is denoted as $\mathcal{N}(v)$. The edge that connects u and v is denoted as uv. We denote the concatenation operator as . We write a *multiset* as a *set* in the following discussion.

The main idea behind the subgraph sampling strategy is to represent the graph G as a bag $S_G = \{G_1, G_2, \ldots, G_m\}$ of its subgraphs. Let $\mathcal{G}$ be the set of all graphs with n nodes or less, and let $\mathbb{P}(\mathcal{G})$ be its power set, *i.e.*, the set of all subsets $S \in \mathcal{G}$. A subgraph selection policy is a function $\pi : \mathcal{G} \to S(\mathcal{G})$ that assigns to each graph G a subset of the set of its subgraphs $\pi(G)$. We require that π is invariant to permutations of the nodes in the graphs, namely that $\pi(G) = \pi(\sigma \cdot G)$, where $\sigma \in S_n$ is a node permutation, and $\sigma \cdot G$ is the graph obtained afeter applying σ.

Vector representation of a set In this section, we briefly introduce the kernel distance between the point sets, focusing only on what is required in this paper. To summarize, we represent a set of vectors by the sum of the vectors after applying a transformation called a feature map. Let $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a function. For example, we may think of the Gaussian kernel $\exp\left(\frac{\|x-y\|^2}{2}\right)$. Function K is a positive definite kernel (PD kernel) if, for any constants $\{c_i\}_{i=1}^n$ and points $\{x_i\}_{i=1}^n$ in $\mathbb{R}^d$, we have $\sum_i \sum_j c_i c_j K(x_i, x_j) \geq 0$. A PD kernel K has an associated reproducing kernel Hilbert space $\mathcal{H}$ with feature map $\phi : \mathbb{R}^d \to \mathcal{H}$ such that $K(x, y) = \langle \phi(x), \phi(y) \rangle$ for all $x, y \in \mathbb{R}^d$, where $\langle , \rangle$ denotes the inner product on $\mathcal{H}$.

A PD kernel K with feature map ϕ induces a (pseudo-)distance d_K on $\mathcal{R}^d$, which is defined by $d_K^2(x, y) = \|\phi(x) - \phi(y)\|_{\mathcal{H}}^2 = \langle \phi(x) - \phi(y), \phi(x) - \phi(y) \rangle = K(x, x) - 2K(x, y) + K(y, y)$. For sets of points $X = \{x_i\}_{i=1}^m$ and $Y = \{y_i\}_{j=1}^n$, the induced distace D_K is similarly defined as

$$
\begin{aligned}
D_K^2(X, Y) = &\sum_{x\in X} \sum_{x'\in X} K(x, x') - 2 \sum_{x\in X} \sum_{y\in Y} K(x, y) \\
&+ \sum_{y\in Y} \sum_{y'\in Y} K(y, y') \\
= &\| \sum_{x\in X} \phi(x) - \sum_{y\in Y} \phi(y) \|^2.
\end{aligned} \tag{1}
$$

Hence, $\phi(X) = \sum_i \phi(x_i)$ represents set X independently of Y , and the set distance D_K can be computed using the distance between the representation vectors. If points x_i

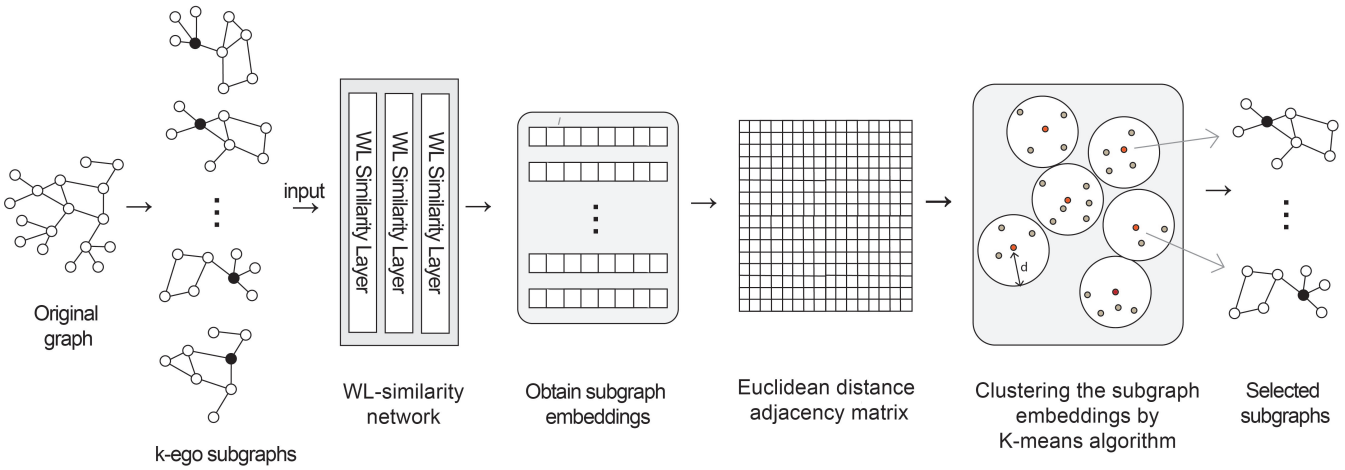

Figure 2: The architecture of EGO+WL subgraph sampling strategy. Tt selects the most representative subgraphs from a bag of k-ego subgraphs from the original graph, input them into the WL similarity kernel, build the adjacency matrix between these embeddings pairwisely and cluster these embedding points by K-means clustering algorithm. With the clustered results, we extract only the centroids from the algorithm and input them to the ESAN training architecture.

and y_j are associated with weights v_i and w_j in $\mathbb{R}$, then we replace $K(x_i, y_j)$ with $v_i w_i K(x_i, y_j)$ and obtain $\phi(X) = \sum_i v_i \phi(x_i)$ and $\phi(Y) = \sum_j w_j \phi(y_j)$ in the same manner. For many known kernels, the explicit feature maps are unclear or infinite-dimensional. However, we remark that when walking backward, for an arbitrary map $\phi : \mathbb{R}^d \to \mathbb{R}^D$, there is an associated set similarity using the representation map $\phi(X) = \sum_{x \in X} \phi(x)$. Its usefulness depends on specific problems at hand.

3 Method

In this paper, we explore a more advanced subgraph selection strategy that prove to strike a good balance between complexity (the number of subgraphs) and the resulting expressive power: Ego network under WL similarity kernel (EGO+WL), as described next. Intuitively, it selects the most representative subgraphs from a bag of k-ego subgraphs from the original graph. We utlize the WL similarity network to obtain the subgraph representation embeddings, build the adjacency matrix between these embeddings pairwisely by their Euclidean distance and cluster these embedding points by K-means clustering algorithm. With the clustered results, we extract only the centroids from the algorithm and input them to the ESAN training architecture. Next, we will go through the details in each stage.

Weisfeiler-Leman similarity After obtained the k-ego subgraphs, we borrow the idea from OK *et.al.* as shown in algorithm 1 which iteratively update the node attributes using the neighbors' information [Ok, 2020]. The focus of this algorithm is to reflect the similarity between the sets of neighbors' attributes into the node-wise updated attributes via the set representation vector. Feature maps ϕ_i can be ones from well-known kernels or problem-specific functions. If we use the concatenation as COMBINE_i in Algorithm 1, because $\|f^i(v) - f^i(v')\|^2 = \|f^{i-1}(v) - f^{i-1}(v')\|^2 + \|\hat{f}^i(v) - \hat{f}^i(v')\|^2$, both the similarities between f^{i-1} and between the sets of neighbors' attributes are incorporated into $f^i(v)$. The steps in a single iteration correspond to *transform-sum-cat* in this case.

Algorithm 1 Updating node attributes in Weisfeiler-Leman similarity

Input: Graph G, nodes V, initial attributes $f^0(v)$ for $v \in V$, iteration number k, and feature maps ϕ_i for $i = 1, 2, \ldots, k$.

1: **for** i from 1 to k **do**
2: **for** $v \in V$ **do**
3: $g^1(v) \leftarrow \phi_i(f^{i-1}(v))$;
4: $\hat{f}^i(v) \leftarrow \sum_{u \in \mathcal{N}(v)} g^i(u)$;
5: $f^i(v) \leftarrow \text{COMBINE}_i \left(f^{i-1}(v), \hat{f}^i(v) \right)$.
6: **end for**
7: **end for**
8: **return** Updated attributes $f^k(v)$ for $v \in V$.

After the node-wise update, they keep the set of updated attributes and discard the adjacency. To compare two graphs G and G', they measure the distance between $\{f^k(v) : v \in V(G)\}$ and $\{f^k(v') : v' \in V(G')\}$ using another kernel of choice. An example is to use the Gaussian kernel between the node attributes. After transform the subgraph to subgraph embeddings, we run the K-means algorithm on the subgraph embeddings and select the subgraphs whose embeddings are the centeriods of clusters. Intuitively, these centroids are the most representative subgraphs of the orignal graph as we can view other subgraphs as its variation after perturbation.

K-means clustering The WL-similarity network outputs the embeddings of the subgraphs. Then we build the Euclidean distance adjacency matrix of these graphs and cluster them by K-means clustering algorithm. Given a set of observations $(x_1, x_2, \ldots, x_n)$, where each observation is a d-dimensional real vector, k-means clustering aims to partition

the n observations into $k(\leq n)$ sets $S = \{S_1, S_2, \ldots, S_k\}$ so as to minimize the within-cluster sum of squares (*i.e.* variance). Formally, the objective is to find:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = \arg\min_{\mathbf{S}} \sum_{i=1}^{k} |S_i| \operatorname{Var} S_i,$$

where $_i$ is the mean of points in S_i. This is equivalent to minimizing the pairwise squared deviations of points in the same cluster: $\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \frac{1}{|S_i|} \sum_{\mathbf{x},\mathbf{y} \in S_i} \|\mathbf{x} - \mathbf{y}\|^2$. Since the total variance is constant, this is equivalent to maximizing the sum of squared deviations between points in different clusters (between-cluster sum of squares).

Take the centroids of clusters With the clusters from k-means, we take the centroid points of the clusters and input the subgraphs they represent to the ESAN network for training. We input up to 50% of the subgraphs from the full bag to the network to reduce the space complexity for computing.

4 Experiments

Table 2: Details of the TU datasets. We list their number of graphs, number of label classes, number of nodes and number of edges.

Dataset	# Graphs	# Classes	# Nodes	# Edges
MUTAG	188	2	97.9	202.5
PTC	344	2	14.29	14.69
PROTEINS	1113	2	39.06	72.82
NCI1	4110	2	29.87	32.3
NCI109	4127	2	29.8	32.13
IMDB-BINARY	1000	2	19.77	96.53
IMDB-MULTI	1500	3	13.00	65.94

We perform an extensive set of experiments to answer whether our approach is more expressive than the existing graph sampling strategy in practice and whether our approach decreases the time complexity and space complexity compared to the existing graph sampling strategy in practice. We use the same architecutre as ESAN [Bevilacqua *et al.*, 2021] and compare our strategy with the proposed strategies in their work as the baselines: node-deleted subgraphs (ND), edge-deleted subgraphs (ED), and ego-networks (EGO, EGO+).

Graph classification performance on TUDatasets. We experimented with popular datasets from the TUD repository [Morris *et al.*, 2020]. We followed the widely-used hyperparameter selection and experimental procedure proposed by [Xu *et al.*, 2018]. Specifically, we conducted 10-fold cross validation and reported the validation performances at the epoch achieving the highest averaged validation accuracy across all the folds. We used Adam optimizer with learnin grate decayed by a factor of 0.5 every 50 epochs. The training is stopped after 350 epochs. As for DS-GNN, we implemented $R_{subgraphs}$ with summation over node features, while module E_{sets} is parameterized wit ha two-layer DeepSets with final mean readout. In DSS-GNN, we considered the mean aggregator for the feature matrix and use the adjacency matrix of the original graph. We implemented $R_{subgraphs}$ by averaging node representations on each subgraph. We considered the baseline model GIN [Xu *et al.*, 2018]. The results are reported in Table 1 where the best performing method for each dataset is reported as SoTA.

In the table, the best performance for each model From the results, we can find that EGO+WL outperforms other strategies. In dataset PTC, NCI1 and IMDB-MULTIPLE, it achieves superior performance compared to other strategies. Worth to mention that, our strategy only utilized half of the subgraphs but a better performance in practice.

Space complexity For the space complexity, we focus our comparison on TU datasets between EGO+ and EGO+WL. We used GIN as the base encoder and record the peak percentage of utilized GPU memory during the training on a RTX3080Ti GPU with 20G GPU memory. We kept the hyperparameters the same for all methods to allow a fair comparison. Results are reported in Table 3. As can be seen in the table, our strategy can mostly reduce the memory consumption and achieves a 55% reduction on IMDB-MULTI and IMDB-BINARY compared to the overall best baseline strategy: EGO+. It is observed that on dataset NCI109 and MUTAG, our strategy used slightly more spaces compared to EGO+, it might be caused by the highly repeated k-ego graphs as EGO+WL takes all the centroids subgraphs of the same topology from the full subgraph bag. If the topology of the selected centroids subgraphs are highly repeated, this strategy will hardly showcase its efficiency.

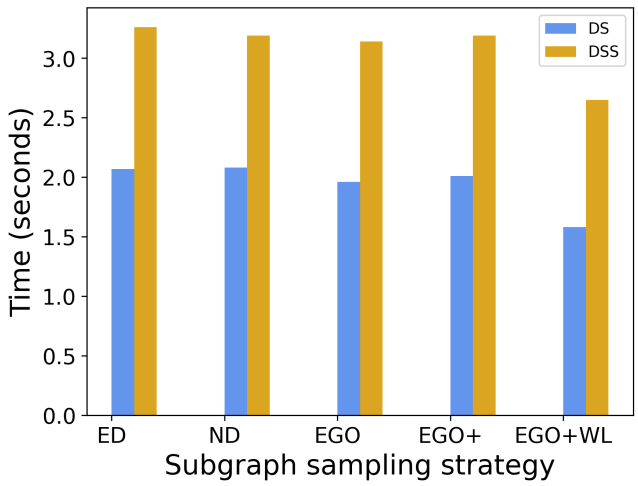

Figure 3: Timing comparison per epoch on a RTX3080 GPU. Time taken for a single epoch with batch size 32 on the NCI1 dataset. All values are in seconds.

Time complexity We focus our analysis on NCI1 dataset from [Morris *et al.*, 2020] (4110 graphs, 30 nodes/edges per graph on average) and estimated the time to perform a single training epoch on a RTX 3080 GPU. We used GIN as a base encoder and compared the times of DS-GNN and DSS-GNN without sampling. We kept the hyperparameters the same for all methods to allow a fair comparison. Note that these times consider a single process completing a single training epoch.

Table 1: TUDatasets. Red text indicates the highest classification accuracy for each category. SoTA line reports results for the best performing model for each dataset.

Method ↓ / Dataset →	MUTAG	PTC	PROTEINS	NCI1	NCI109	IMDB-B	IMDB-M
SoTA	92.7 ± 6.1	68.2 ± 7.2	77.2 ± 4.7	83.6 ± 1.4	84.0 ± 1.6	77.8 ± 3.3	54.3 ± 3.3
GIN (Xu et al., 2019)	89.4 ± 5.6	64.6 ± 7.0	76.2 ± 2.8	82.7 ± 1.7	82.2 ± 1.6	75.1 ± 5.1	52.3 ± 2.8
DS-GNN (GIN) (ED)	89.9 ± 3.7	66.0 ± 7.2	76.8 ± 4.6	83.3 ± 2.5	83.0 ± 1.7	76.1 ± 2.6	52.9 ± 2.4
DS-GNN (GIN) (ND)	89.4 ± 4.8	66.3 ± 7.0	77.1 ± 4.6	83.8 ± 2.4	82.4 ± 1.3	75.4 ± 2.9	52.7 ± 2.0
DS-GNN (GIN) (EGO)	89.9 ± 6.5	68.6 ± 5.8	76.7 ± 5.8	81.4 ± 0.7	79.5 ± 1.0	76.1 ± 2.8	52.6 ± 2.8
DS-GNN (GIN) (EGO+)	91.0 ± 4.8	68.7 ± 7.0	76.7 ± 4.4	82.0 ± 1.4	80.3 ± 0.9	77.1 ± 2.6	53.2 ± 2.8
DS-GNN (GIN) (EGO+WL)	89.9 ± 5.7	68.8 ± 5.8	75.9 ± 4.1	83.9 ± 0.7	81.8 ± 2.0	77.0 ± 2.5	53.5 ± 2.2
DSS-GNN (GIN) (ED)	91.0 ± 4.8	66.6 ± 7.3	75.8 ± 4.5	83.4 ± 2.5	82.8 ± 0.9	76.8 ± 4.3	53.5 ± 3.4
DSS-GNN (GIN) (ND)	91.0 ± 3.5	66.3 ± 5.9	76.1 ± 3.4	83.6 ± 1.5	83.1 ± 0.8	76.1 ± 2.9	53.3 ± 1.9
DSS-GNN (GIN) (EGO)	91.0 ± 4.7	68.2 ± 5.8	76.7 ± 4.1	83.6 ± 1.8	82.5 ± 1.6	76.5 ± 2.8	53.3 ± 3.1
DSS-GNN (GIN) (EGO+)	91.1 ± 7.0	69.2 ± 6.5	75.9 ± 4.3	83.7 ± 1.8	82.8 ± 1.2	77.1 ± 3.0	53.2 ± 2.4
DS-GNN (GIN) (EGO+WL)	90.4 ± 4.3	71.4 ± 5.7	76.6 ± 3.3	84.4 ± 1.3	83.5 ± 2.3	78.1 ± 3.1	54.7 ± 0.7

Table 3: GPU memory allocation percentage of the process. Run on RTX3080Ti with 20G GPU memory. The highlighted entries corresponding the different datasets are the results from the strategy achieves lower space complexity.

	EGO+	EGO+WL
MUTAG	**30.38**%	30.60%
PTC	15.01%	**14.29**%
PROTEINS	65.2%	**47.9**%
NCI1	9.0%	**6.4**%
NCI109	**16.0**%	17.3%
IMDB-BINARY	21.1%	**9.6**%
IMDB-MULTI	26.0%	**11.5**%

In our implementation, we perform the cross validation in parallel, making use of multiprocessing. Results are reported in Figure 3. As can be seen in the table, our method reduces these times by 10-15%, showcasing how our EGO+WL strategy can be beneficial in practice.

5 Future Work

To summary, we present a new subgraph sampling strategy EGO+WL based on WL similarity. EGO+WL achieves overall superior graph classification performance on TU datasets with less subgraphs, lower space complexity and time complexity for equivariant subgraph aggregation network. Even though the strategy efficiently accerlerate the network training, its time complexity still cannot be the most optimal since it maps the embeddings to Gaussian kernel and computes the distance between float points.

For the future, we are thinking to replace the kernel from WL similarity kernel (in continuous field $\mathbb{R}$) to WL subtree kernel [Shervashidze *et al.*, 2009]. As shown in Figure 4, WL subtree kernel maps a graph to a vector representation in the discrete manner. Inspired from this, I am working on another strategy EGO+WLSubtree in Figure 5 as an updated version for EGO+WL. Since the subgraph embeddings from WL subtree kernel will be discrete, we can compute the Hamming distance between them and formulate the problem as a graph encoding problem. Each subgraph can be seen as a "code" of the original graph and all the subgraphs form a "codebook". There might have a large space for the theorems in Coding Theory to be applied here in this problem formulation.

References

[Abboud *et al.*, 2020] Ralph Abboud, Ismail Ilkan Ceylan, Martin Grohe, and Thomas Lukasiewicz. The surprising power of graph neural networks with random node initialization. *arXiv preprint arXiv:2010.01179*, 2020.

[Arvind *et al.*, 2020] Vikraman Arvind, Frank Fuhlbrück, Johannes Köbler, and Oleg Verbitsky. On weisfeiler-leman invariance: Subgraph counts and related graph properties. *Journal of Computer and System Sciences*, 113:42–59, 2020.

[Bevilacqua *et al.*, 2021] Beatrice Bevilacqua, Fabrizio Frasca, Derek Lim, Balasubramaniam Srinivasan, Chen Cai, Gopinath Balamurugan, Michael M Bronstein, and Haggai Maron. Equivariant subgraph aggregation networks. *arXiv preprint arXiv:2110.02910*, 2021.

[Bodnar *et al.*, 2021] Cristian Bodnar, Fabrizio Frasca, Yuguang Wang, Nina Otter, Guido F Montufar, Pietro Lio, and Michael Bronstein. Weisfeiler and lehman go topological: Message passing simplicial networks. In *International Conference on Machine Learning*, pages 1026–1037. PMLR, 2021.

[Bouritsas *et al.*, 2021] Giorgos Bouritsas, Andreas Loukas, Nikolaos Karalias, and Michael Bronstein. Partition and code: learning how to compress graphs. *Advances in Neural Information Processing Systems*, 34:18603–18619, 2021.

[Chiang *et al.*, 2019] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 257–266, 2019.

[Dasoulas *et al.*, 2019] George Dasoulas, Ludovic Dos Santos, Kevin Scaman, and Aladin Virmaux. Coloring graph

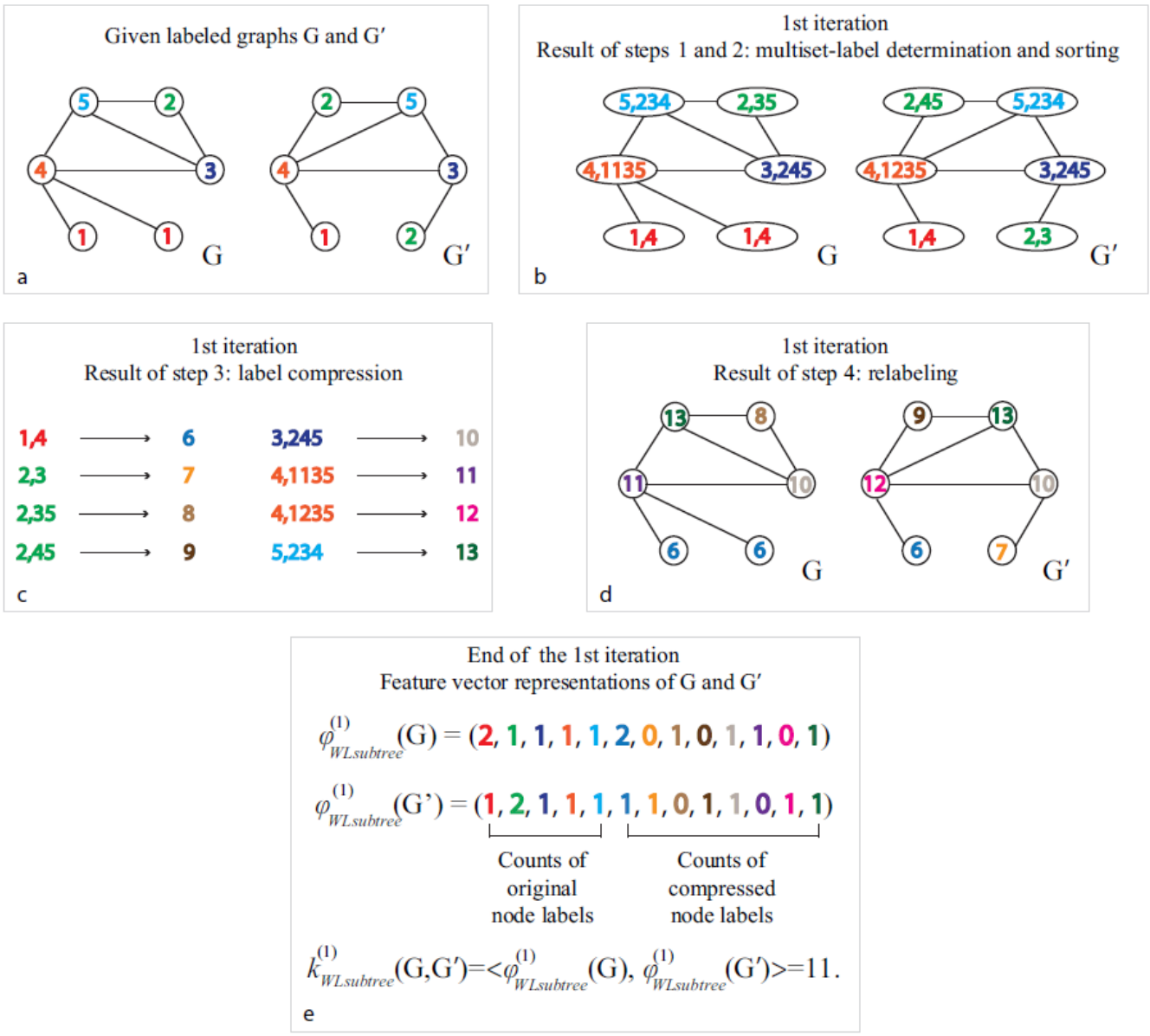

Figure 4: Illustration of the computation of the Weisfeiler-Lehman subtree kernel with $h = 1$ for two graphs. Here $\{1, 2, \ldots, 13\} \in \Sigma$ are considered as letters. Note that compressed labels denote subtree patterns: For instance, if a node has label 8, this means that there is a subtree pattern of height 1 rooted at this node, where the root has label 2 and its neighbours have labels 3 and 5 .

neural networks for node disambiguation. *arXiv preprint arXiv:1912.06058*, 2019.

[de Haan *et al.*, 2020] Pim de Haan, Taco S Cohen, and Max Welling. Natural graph networks. *Advances in Neural Information Processing Systems*, 33:3636–3646, 2020.

[Hamilton *et al.*, 2017] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[Maron *et al.*, 2019] Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. *Advances in neural information processing systems*, 32, 2019.

[Morris *et al.*, 2019] 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. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4602–4609, 2019.

[Morris *et al.*, 2020] Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of bench-

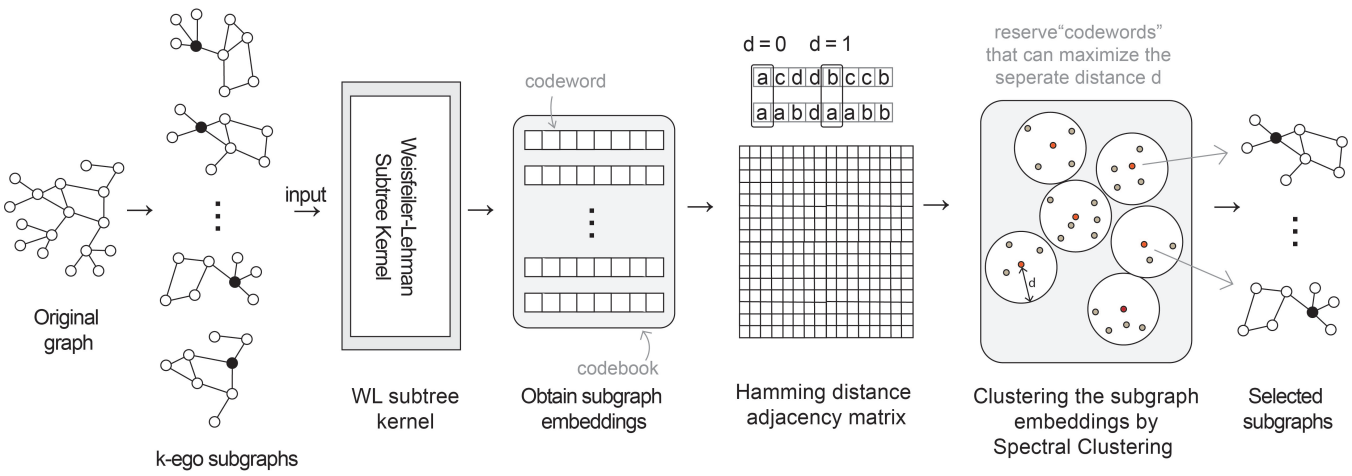

Figure 5: The architecture of Ego+WLSubtree strategy

mark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020.

[Ok, 2020] Seongmin Ok. A graph similarity for deep learning. *Advances in Neural Information Processing Systems*, 33:1–12, 2020.

[Rong *et al.*, 2019] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903*, 2019.

[Sandfelder *et al.*, 2021] Dylan Sandfelder, Priyesh Vijayan, and William L Hamilton. Ego-gnns: Exploiting ego structures in graph neural networks. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8523–8527. IEEE, 2021.

[Sato, 2020] Ryoma Sato. A survey on the expressive power of graph neural networks. *arXiv preprint arXiv:2003.04078*, 2020.

[Shervashidze *et al.*, 2009] Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. Efficient graphlet kernels for large graph comparison. In *Artificial intelligence and statistics*, pages 488–495. PMLR, 2009.

[Tahmasebi *et al.*, 2020] Behrooz Tahmasebi, Derek Lim, and Stefanie Jegelka. Counting substructures with higher-order graph neural networks: Possibility and impossibility results. *arXiv preprint arXiv:2012.03174*, 2020.

[Thiede *et al.*, 2021] Erik Thiede, Wenda Zhou, and Risi Kondor. Autobahn: Automorphism-based graph neural nets. *Advances in Neural Information Processing Systems*, 34:29922–29934, 2021.

[Weisfeiler and Leman, 1968] Boris Weisfeiler and Andrei Leman. The reduction of a graph to canonical form and the algebra which appears therein. *NTI, Series*, 2(9):12–16, 1968.

[Xu *et al.*, 2018] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[Xu *et al.*, 2021] Fengli Xu, Quanming Yao, Pan Hui, and Yong Li. Automorphic equivalence-aware graph neural network. *Advances in Neural Information Processing Systems*, 34:15138–15150, 2021.

[Yang *et al.*, 2020] Yiding Yang, Zunlei Feng, Mingli Song, and Xinchao Wang. Factorizable graph convolutional networks. *Advances in Neural Information Processing Systems*, 33:20286–20296, 2020.

[You *et al.*, 2021] Jiaxuan You, Jonathan M Gomes-Selman, Rex Ying, and Jure Leskovec. Identity-aware graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10737–10745, 2021.

[Zeng *et al.*, 2019] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931*, 2019.