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 the network. With the rapid development of message-passing graph neural networks (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 Networks (MPNNs) are the leading Graph Neural Network (GNN) architecture for deep learning because of their strong simplicity. 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 WL test [Weisfeiler and Leman, 1968] is a 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.

Suppose we have two graphs G and G', as shown in Figure 1, and we want to determine whether they are isomorphic. In essence, the algorithm augments node labels by sorting neighbor node labels and compresses augmented labels into short labels. The steps are repeated until G and G' node label sets differ, or the number of iterations reaches a maximum number. In Figure 1, we take graph G as an example. We set $f(v) = 1$ for all $v \in V(G)$ if G has no node attributes. We then update the node attributes in stages and each node is updated once only. The updated attribute is the pair of its own attributes and the 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. Otherwise, the two graphs are isomorphic by the 1-WL test. Many indistinguishable non-isomorphic pairs of graphs exist. As a consequence, MPNNs cannot distinguish between very simple graphs.

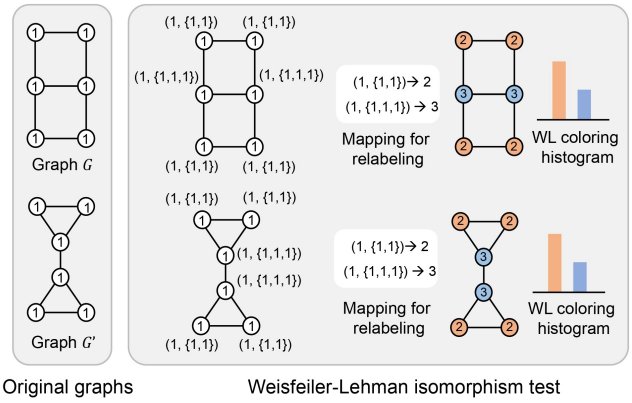

Figure 1: A pair of graphs not distinguishable by the WL test.

To address this limitation, many efforts have been paid to work on improving the expressiveness of MPNNs. In general, the architectures can be clustered into three categories in enhancing the expressive power of GNNs: (1) Augmenting node features with different identifiers ([Sato, 2020]; [Dasoulas *et al.*, 2019]; [Abboud *et al.*, 2020]); (2) Aligning to the k-WL hierarchy ([Morris *et al.*, 2019]; [Maron *et al.*, 2019]); (3) Leveraging on structural information, *i.e.,* subgraphs, that cannot 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.* presented Equivariant subgraph aggregation networks (ESAN) which belongs to the category (3) with non-domain specific policies [Bevilacqua *et al.*, 2021]. ESAN represented each graph as a bag of subgraphs chosen according to some predefined policy. In detail, they presented four strategies as the baselines: node-deleted subgraphs (ND), edge-deleted subgraphs (ED), and

ego-networks (EGO, EGO+) as demonstrated 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 deleting a single node. Similarly, the **edge-deleted policy** is defined by deleting 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 neighborhood with the induced connectivity). They also consider a variant of the ego-networks policy where the center node holds a feature that is different from other nodes (EGO+).

However, their approach is related to multiple techniques in graph learning. For example, DropEdge is a semi-supervised classification architecture that can be considered as a stochastic version of the Edge-deleted policy ([Rong *et al.*, 2019]). Ego-GNNs resemble the EGO policy, as messages are passed within the ego-net of each node, and are aggregated from the ego-net that a node is contained in ([Sandfelder *et al.*, 2021]). ID-GNNs can be seen as a variation of 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 ([You *et al.*, 2021]). FactorGCN, which learns the subgraphs containing edges that capture latent relationships, is a model that disentangles relations, especially for graph data ([Yang *et al.*, 2020]).

We can find 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 in 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

In this section, we present how to formulate the subgraph sampling problem in our work and the vector representation of a (multi-)set.

2.1 Problem formulation

We assume a standard graph classification setting. We represent a graph with n nodes as a tuple $G = (V, A, X)$ 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 $y^0(v)$ and the successively updated ones as $y^1(v)$, $y^2(v)$, *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 after applying σ.

2.2 Representation of a multiset

Given sets of points $X = \{x_i\}_{i=1}^m$ and $Y = \{y_i\}_{j=1}^n$. Next, we will represent a set of vectors by the sum of the vectors after applying *feature map* which is a transformation. Suppose $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a function between X and Y, *i.e.*, the Gaussian kernel $\exp\left(\frac{\|x-y\|^2}{2}\right)$.

Definition 2.1. *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 $\sum_i \sum_j c_i c_j K(x_i, x_j) \geq 0$.*

With a PD kernel and an associated Hilbert space $\mathcal{H}$, we can define the feature map as the following:

Definition 2.2. *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 can output a distance or pseudo-distance d_K on $\mathbb{R}^d$, which is defined by $d_K^2(x, y) = \|\phi(x) - \phi(y)\|_{\mathcal{H}}^2 = \langle \phi(x) - \phi(y), \phi(x) - \phi(y) \rangle = K(x, x) - 2K(x, y) + K(y, y)$.the induced distance D_K is in the same manner 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 and y_j are associated with weights v_i and w_j in $\mathbb{R}$, replacing $K(x_i, y_j)$ with $v_i w_i K(x_i, y_j)$, we can obtain $\phi(X) = \sum_i v_i \phi(x_i)$ and $\phi(Y) = \sum_j w_j \phi(y_j)$ similarly.

Remark 2.1. *For many known kernels, the explicit feature maps are unclear or infinite-dimensional [Ok, 2020]. However, for an arbitrary map $\phi : \mathbb{R}^d \to \mathbb{R}^D$, there is an associated set similarity using the representation map $\phi(X) = \sum_{x\in X} \phi(x)$ when walking backward.*

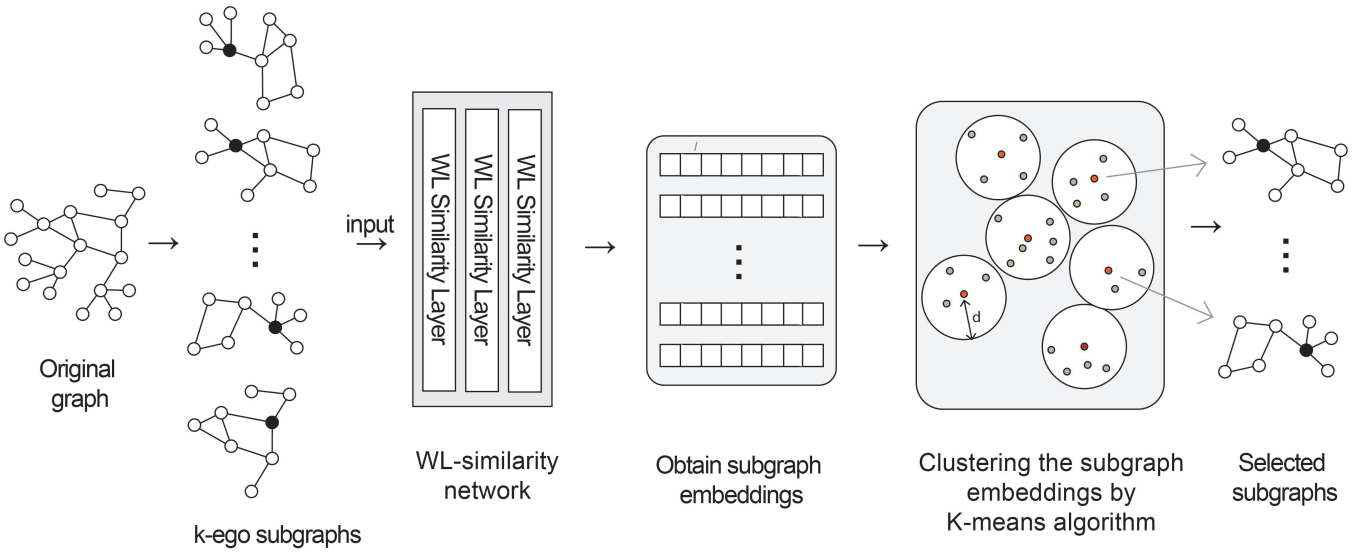

Figure 2: The architecture of EGO+WL subgraph sampling strategy. Tt selects the most representative subgraphs from a bag of k-ego subgraphs from the original graph, input them into the WL similarity network and cluster these embedding points by K-means clustering algorithm. With the clustered results,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

3 Method

In this paper, we explore a more advanced subgraph selection strategy that proves 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 utilize 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 into the ESAN training architecture. Next, we will go through the details of each stage.

Algorithm 1 Updating node attributes in Weisfeiler-Leman similarity [Ok, 2020]

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

for i from 1 to k **do**
  for $v \in V$ **do**
    $g^1(v) \leftarrow \phi_i(y^{i-1}(v))$;
    $\hat{y}^i(v) \leftarrow \sum_{u \in \mathcal{N}(v)} g^i(u)$;
    $y^i(v) \leftarrow \text{COMBINE}_i\left(y^{i-1}(v), \hat{y}^i(v)\right)$.
  end for
end for
return Updated attributes $y^k(v)$ for $v \in V$.

Weisfeiler-Leman similarity network After obtaining 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

$$\begin{aligned}\|y^i(v) - y^i(v')\|^2 =& \|y^{i-1}(v) - y^{i-1}(v')\|^2 \\ &+ \|\hat{f}^i(v) - \hat{f}^i(v')\|^2,\end{aligned} \tag{2}$$

both the similarities between y^{i-1} and between the sets of neighbors' attributes contributed to $y^i(v)$. After the update of every node, they keep the set of updated attributes. We use a graph neural network (GNN) to update the node attributes based on WL similarity which is equipped with three layers. Each layer consists of a linear transformation, a 1D normalization and a activate function (*i.e.*, ReLU). We input the adjacency matrices as well as the node feature matrices of the subgraphs into the network and obtain the graph latent representations $(x_1, x_2, \ldots, x_N)$ for N graphs based on WL similarity.

K-means clustering The WL-similarity network outputs the embeddings of the subgraphs. To compare two graphs G and G', we measure the Euclidean distance between $\{f^k(v) : v \in V(G)\}$ and $\{f^k(v') : v' \in V(G')\}$ using Gaussian kernel. Then we cluster them by K-means algorithm [MacQueen, 1967]. Given the set of m subgraph embeddings $(x_1, x_2, \ldots, x_m)$, where each of them is a d-dimensional real vector, we use K-means to partition the n embeddings into $g(\leq m)$ sets $S = \{S_1, S_2, \ldots, S_g\}$ to minimize the within-cluster sum of squares (*i.e.* variance). Formally, the objective is to minimize the pairwise squared deviations of points in the same cluster, or equivalently, to maximize the

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

sum of squared deviations between points in different clusters (between-cluster sum of squares):

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{g} \frac{1}{|S_i|} \sum_{\mathbf{x},\mathbf{y} \in S_i} \|\mathbf{x} - \mathbf{y}\|^2.$$

Take the centroids of clusters By default, we set g to be half number of the subgraphs. With the g 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

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.

Table 2: Details of the TU datasets. We list their number of graphs, number of label classes, number of nodes and number of edges. For the number of nodes and edges, their values are both in thousand.

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

4.1 Experiment settings

We use the same architecture 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+). In the paper of ESAN, they presented two architectures DSS-GNN and DS-GNN [Bevilacqua *et al.*, 2021]. We experimented with popular datasets from the TUD repository [Morris *et al.*, 2020]. We evaluated both DSS-GNN and DS-GNN on all the TUDatasets. We followed the widely-used hyperparameter selection and experimental procedure proposed by [Xu *et al.*, 2018].

4.2 Graph classification performance on TUDatasets.

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 the Adam optimizer with learning rate that 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 with a 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 (State-of-The-Art).

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 had a better performance in practice.

4.3 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 an RTX3080Ti GPU with 20G

Table 3: GPU memory allocation percentage of the process. Run on RTX3080Ti with 20G GPU memory. The highlighted entries corresponding to the different datasets are the results of the strategy that 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%**

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 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 is highly repeated, this strategy will hardly showcase its efficiency.

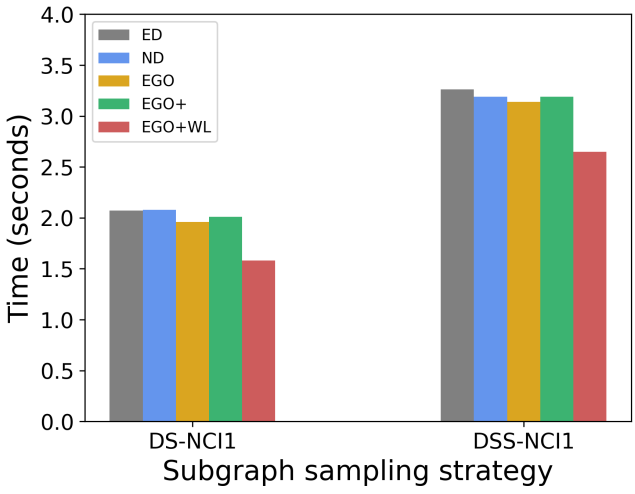

Figure 3: Timing comparison per epoch on an RTX3080 GPU. Time is taken for a single epoch with batch size 32 on the NCI1 dataset. All values are in seconds.

4.4 Time complexity

We focus our analysis on the 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 an 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. 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 Discussion

To summarize, 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 fewer subgraphs, lower space complexity, and time complexity for an equivariant subgraph aggregation network. Even though the strategy efficiently accelerates the training of ESAN, its time complexity still cannot be the most optimal since it maps the embeddings to the Gaussian kernel and computes the distance between float points.

In the future, we are thinking to replace the kernel from the WL similarity kernel with WL subtree kernel [Shervashidze *et al.*, 2009]. The WL subtree kernel maps a graph to a vector representation in a discrete manner. Inspired by this, we are working on another strategy EGO+WLSubtree as an updated version for EGO+WL. Since the subgraph embeddings from the 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 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.

[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.

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

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.

[MacQueen, 1967] J MacQueen. Classification and analysis of multivariate observations. In *5th Berkeley Symp. Math. Statist. Probability*, pages 281–297, 1967.

[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 benchmark 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.

[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.

[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.