공학석사학위논문

이종 및 계층 구조 교차 문맥 그래프 합성곱 신경망

H_2C_2GCN: Heterogeneous and Hierarchical Cross-Context Graph Convolution Network

2021년 2월

서울대학교 대학원

컴퓨터공학부

박채흠

이종 및 계층 구조 교차 문맥 그래프 합성곱 신경망

H_2C_2GCN: Heterogeneous and Hierarchical Cross-Context Graph Convolution Network

지도교수 강 유

이 논문을 공학석사 학위논문으로 제출함

2020년 10월

서울대학교 대학원

컴퓨터공학부

박채흠

박채흠의 석사 학위논문을 인준함

2020년 12월

위 원 장	김 선	(인) [illegible]
부위원장	강 유	(인) [illegible]
위 원	김 건 희	(인) [illegible]

Abstract

H_2C_2GCN: Heterogeneous and Hierarchical Cross-Context Graph Convolution Network

Chaeheum Park
Department of Computer Science & Engineering
The Graduate School
Seoul National University

Given attributed graphs, how can we accurately classify them using both topological structures and node features? Graph classification is a crucial task in data mining, especially in the bioinformatics domain where a chemical compound is represented as a graph of attributed compounds. Although there are existing methods like graph kernels or truncated random walks for graph classification, they do not give good accuracy since they consider features present at a single resolution, i.e., nodes or subgraphs. Such single resolution features result in a biased view of the graph's context, which is nearsighted or too wide, failing to capture the comprehensive properties of each graph.

In this paper, we propose H_2C_2GCN (**H**eterogeneous and **H**ierarchical **C**ross-**c**ontext **G**raph **C**onvolution **N**etwork), an accurate end-to-end framework for graph classification. Given multiple input graphs, H_2C_2GCN generates a multi-resolution tree that connects the given graphs by cross-context edges. It gives a unified view of multiple graphs considering both node features and topological structures. We propose a novel hierarchical graph convolutional network to extract the representation of each graph. Extensive experiments on real-world datasets show that H_2C_2GCN provides the state-of-the-art accuracy for graph classification.

Keywords : Graph Classification, Graph Convolutional Network, Hierarchical Graph, Attributed Graph

Student Number : 2019-20235

Contents

List of Figures

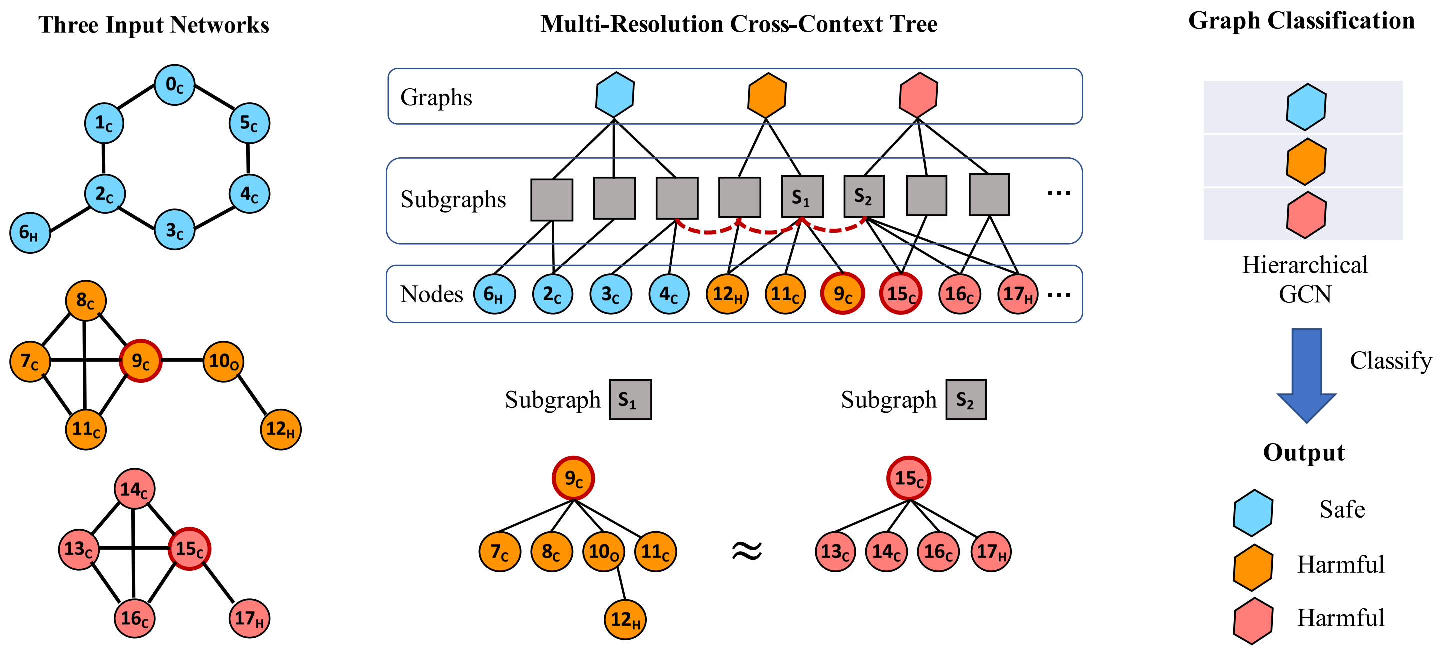
List of Tables

Chapter 1

Introduction

Graph-structured data have become ubiquitous recently in the fields of web analysis, social networks, bioinformatics, and chemoinformatics. There are many applications on networks where machine learning models have achieved a great performance in both supervised and unsupervised fashions including node classification [1], anomaly detection [2], link prediction [3], and recommendation [4]. Graph classification, which is to classify an entire graph to a discrete label, is an essential task with many applications. For instance, in bio- and chemo-informatic domains, chemical compounds are represented as graphs of atoms whose element types are represented as node attributes; graph classification is used for detecting virus mutations, solubility, or its effect toward cancer.

For graph classification the following two techniques have been proven successful: graph kernels [5, 6] and random walks [3, 1, 7, 8, 9]. They summarize the properties of each graph as a single *embedding* vector to compute the similarity between graphs or to learn typical classifiers such as support vector machines. However, the previous works face the following three challenges. First, they focus on a single resolution when extracting an embedding vector, i.e., nodes or subgraphs, resulting in a nearsighted or too wide view of the graph's context. Second, they fail to capture the common characteristics of multiple graphs that contain the general knowledge of the domain. Third, they ignore node attributes present at each graph, focusing only on the structural characteristics.



(a) Input graphs

(b) Multi-resolution cross-context trees

(c) Graph classification

Figure 1: An overview of our proposed method H_2C_2GCN. (a) We are given three graphs colored in blue, orange, and peach, where the number in each node represents its index and the subscript represents its attribute. Our goal is to classify each graph (e.g., a chemical compound) to a category (e.g., safe or harmful). (b) We map *objects* of multiple resolutions, i.e., nodes, subgraphs, or entire graphs, of each graph to a new tree named multi-resolution cross-context tree ($MRCCT$). We make cross-context edges between subgraphs from different graphs based on their similarities. (c) We extract graph embeddings and classify the graphs using our proposed Hierarchical GCN, a graph neural network specialized for hierarchical graphs.

In this work, we propose H_2C_2GCN (**H**eterogeneous and **H**ierarchical **C**ross-**C**ontext **G**raph **C**onvolution **N**etwork), an accurate approach for graph classification that addresses the aforementioned challenges. H_2C_2GCN consists of three steps. First, H_2C_2GCN creates multi-resolution trees from given graphs whose nodes represent graph objects of three different resolutions: nodes, subgraphs, and graphs. Then, H_2C_2GCN connects the trees with cross-context edges by comparing the subgraphs from different graphs. Finally, H_2C_2GCN runs Hierarchical GCN, a novel graph neural network that we propose to extract embeddings of multipartite graphs having partial attributes, on the generated tree. The overall structure of H_2C_2GCN is illustrated as Figure 1, assuming three input graphs as an example, for classifying the property of chemical compounds.

Our contributions are as follows:

- **Multi-Resolution Objects.** We decompose each graph into *objects* of multiple resolutions, making a balanced view of the graph's context, and utilize the relationships between all objects for end-to-end learning.

- **Cross-Context Information.** We utilize the cross-context information of multiple graphs by connecting subgraphs with similar degree sequences. This allows us to consider comprehensive information of all given graphs.

- **Hierarchical GCN.** We propose Hierarchical GCN, a new graph neural network for hierarchically structured graphs having partial node attributes. We apply it to extract graph embeddings specialized for downstream tasks considering both the structural information and node attributes.

- **Experiments.** Through extensive experiments on five benchmark datasets, we show that H_2C_2GCN outperforms five baselines for graph classification, achieving up to 2.7 % points higher accuracy than the best competitors.

The rest of this paper is organized as follows. We first describe related works of graph classification in Section 2. Then, we introduce H_2C_2GCN in Section 3 and describe experimental results for five real-world datasets in Section 4, comparing it to the previous methods. We conclude at Section 5.

Chapter 2

Related Works

We review related works on four categories: single-resolution embedding, multi-resolution embedding, graph kernels, and graph convolutional networks.

Single-Resolution Embedding. There are various approaches to obtain embeddings of different resolutions from a graph. Deepwalk [1] and node2vec [3] extract the embeddings of nodes using truncated random walks. subgraph2vec [10] and graph2vec [8] focus on the embeddings of subgraphs and graphs, respectively, by utilizing the structural information of graphs from a broader perspective than that of node-based approaches. Our goal in this work is to generalize such approaches and to consider multi-resolution information.

Multi-Resolution Network Embedding. MrMine [9] is a recent approach that considers multi-resolution information for learning the embeddings of graphs. MrMine combines multiple input graphs as a new network and applies Deepwalk to the generated network to extract the embeddings of every resolution such as nodes, subgraphs, and graphs simultaneously while having the same embedding size for all resolutions. However, MrMine shows a limited performance for graph classification, because it ignores the node attribute information and it learns the embeddings and the classifier separately rather than in an end-to-end way.

Graph Kernels. Graph Kernels use hand-crafted kernel functions to measure the similarity of a pair of graphs. A widely used method is the Weisfeiler-Lehman (WL) Kernel [5] that relabels nodes, compresses its labels, and uses a kernel function to measure the similarity. Graph kernels cannot be used effectively for downstream tasks except for computing the similarity of graphs. Moreover, graph kernel methods are generally slow, since they have exponential computational time with regard to node counts unless approximated [11].

Graph Convolutional Networks. Graph convolution networks [12] (GCN) have been proven to be successful at extracting explicit node embeddings using a convolution operation that extracts features from a node and its neighbors. The main advantage of GCNs is the ability to consider both node attributes and structural information simultaneously by a single operation. We propose Hierarchical GCN, a modified version of GCN capable of extracting node embeddings from hierarchically structured graphs with multi-resolution views.

Chapter 3

Proposed Method

We propose H_2C_2GCN, an accurate approach for graph classification that considers both node attributes and cross-context features to maximize its accuracy. We first provide a brief overview in Section 3.0.1. Then we describe how to generate multi-resolution trees and how to connect similar subgraphs in different graphs in Sections 3.0.2 and 3.0.3, respectively. Finally, we explain how H_2C_2GCN classifies given graphs using a newly proposed Hierarchical GCN in Section 3.0.4.

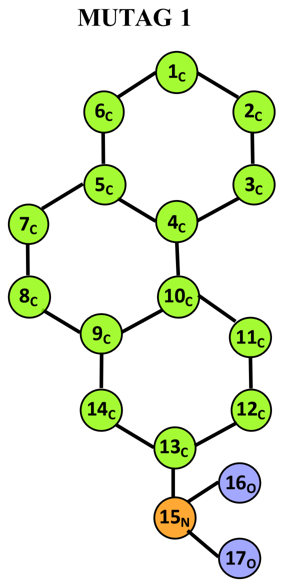
3.0.1 Overview

Given multiple attributed graphs, our goal is to accurately classify them. There are several challenges for achieving the goal.

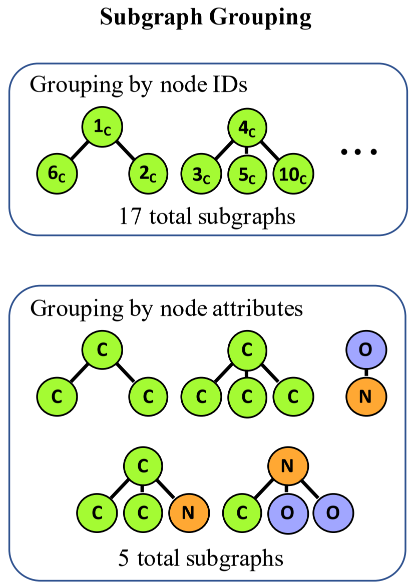
1. How can we consider features from multiple resolutions such as nodes, subgraphs, and graphs for extracting the embeddings of graphs?
2. How can we utilize *cross-context features* between different graphs? In other words, how can we propagate the information of a single graph to the others to improve the overall quality of embedding vectors?
3. How can we learn task-specific embeddings of graphs in an end-to-end fashion without the help of separate modules such as random walks?

Our ideas to solve the aforementioned challenges are as follows. The overview of our H_2C_2GCN is illustrated in Figure 1.

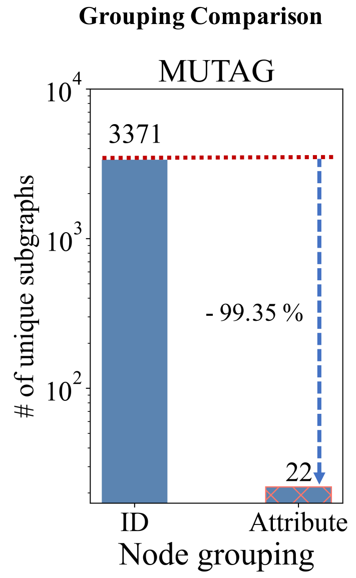
1. **Multi-resolution mapping (Section 3.0.2).** In a graph there are multiple *objects* having different resolutions: nodes, subgraphs, and the graph itself. We generate a new tree having the objects of each graph as its nodes, as in Figure 1b, to make them exchange multi-resolution information.
2. **Cross-context edges (Section 3.0.3**). We connect the subgraphs of multiple graphs if their similarity is larger than a threshold, as the red dotted lines of Figure 1b, combining the trees as a single multi-resolution cross-context tree ($MRCCT$) that contains information of all graphs simultaneously.
3. **Hierarchical GCN (Section 3.0.4).** We propose Hierarchical GCN, a graph neural network specialized for hierarchically structured graphs having initial features only for a subset of nodes. We apply the Hierarchical GCN to extract graph embeddings suitable for graph classification.



(a) A graph in the MUTAG dataset



(b) Grouping subgraphs by node IDs or attributes



(c) A comparison of the numbers of subgraphs

Figure 2: An illustration of our subgraph grouping method that utilizes node attributes. (a) A graph in the MUTAG dataset, where the numbers inside the nodes represent node IDs and their subscripts represent node attributes; green, orange, and blue nodes represent node attribute C, N, and O, respectively. (b) Grouping subgraphs by node IDs results in many subgraphs that focus only on the structural characteristics. On the other hand, grouping by node attributes results in a fewer number of subgraphs that consider both on structural and attribute characteristics. (c) Grouping by node attributes results in 99.35% fewer subgraphs in MUTAG dataset compared to grouping by node IDs.

3.0.2 Multi-Resolution Mapping

Our idea to learn latent representations of graphs while considering different objects from multiple resolutions is to transform the given graphs into a single tree which we call Multi-Resolution Cross-Context Tree ($MRCCT$). We present how to generate a multi-resolution tree from each graph in this section, and then how to combine the trees into the single $MRCCT$ in Section 3.0.3.

We decompose each graph into objects with different resolutions: nodes, subgraphs, and the entire graph. Specifically, we extract subgraphs from the graph by selecting a root node and extracting its k-hop neighborhood. Thus, the number of subgraphs that we extract is the same as the number of nodes in the graph. Then, we connect the extracted objects based on the *membership* relationships; there exists an edge between a node object and a subgraph object if the node is included in the subgraph. Likewise, all subgraphs are connected to the graph object, as they are all included in the graph. This results in a hierarchical tree that contains the objects with multiple resolutions in each graph.

Although this method carefully extracts key information of each graph with multiple resolutions, it has a limitation of not considering the nodes' attributes which are important for understanding the graph. Thus, we group the subgraphs based on their node attributes, greatly reducing the number of unique subgraphs. If two subgraphs have the same structure and node attribute information, they are treated as the same object in the tree even though they contain different nodes in the original graph. Figure 2 shows that grouping by node attributes decreases the number of unique subgraphs by $153\times$.

Algorithm 1 describes how to generate a multi-resolution tree from a graph. We use object mapping functions m_v, m_s, and m_g for generating the node, subgraph,

Algorithm 1: Generating a multi-resolution tree from a graph.

```
Input: A graph $G = (\mathbb{V}, \mathbb{E})$, where $\mathbb{V}$ and $\mathbb{E}$ represent the sets of nodes and edges,
    respectively, and object mapping functions $m_v(\cdot)$, $m_s(\cdot)$, and $m_g(\cdot)$
Output: A multi-resolution tree $T$
 1: $\mathbb{S} \leftarrow$ Set of $k$-hop subgraphs extracted from $G$
 2: $\mathbb{V}_T \leftarrow \{m_v(u) \mid u \in \mathbb{V}\} \cup \{m_s(S) \mid S \in \mathbb{S}\} \cup \{m_g(G)\}$
 3: $\mathbb{E}_T \leftarrow \{\}$
 4: for each subgraph $S \in \mathbb{S}$ do
 5:     for each node $u \in S$ do
 6:         $\mathbb{E}_T \leftarrow \mathbb{E}_T \cup \{(m_v(u), m_s(S))\}$
 7:     end for
 8:     $\mathbb{E}_T \leftarrow \mathbb{E}_T \cup \{(m_s(S), m_g(G))\}$
 9: end for
10: $T \leftarrow (\mathbb{V}_T, \mathbb{E}_T)$ # a multi-resolution tree to be returned
```

and graph objects in the multi-resolution tree, respectively. While m_v and m_g simply map objects to a new node in the tree, the subgraph mapping function m_s transforms two subgraphs into the same node in the tree if they are identical: $m_s(S_1) = m_s(S_2)$ if and only if S_1 and S_2 have the same structural and node attribute information. The number of subgraphs is greatly reduced as we transform each subgraph S by m_s before putting it into the tree T.

3.0.3 Cross-Context Mapping

We combine the multi-resolution trees to construct a single $MRCCT$ that includes cross-context features between graphs. We first combine the subgraph objects having identical structural and attribute information from multiple trees. Specifically, if two subgraph objects from different graphs are identical we redirect associated edges to one combined subgraph object node. As a result, most of the multi-resolution trees are connected as they share subgraph objects, and the number of subgraph objects in the whole dataset is greatly reduced.

Algorithm 2: Generating $MRCCT$ by connecting cross-context edges.

```
Input: Given a set $\mathbb{T} = \{T_1, T_2, ..., T_a\}$ of multi-resolution trees, a threshold $\tau$,
    and a window size $w$
Output: The $MRCCT$ network $G'$
 1: $\mathbb{V}_{G'}, \mathbb{E}_{G'} \leftarrow$ Combine all nodes and edges in $\mathbb{T}$ with subgraph aggregation
 2: $\mathbb{S} \leftarrow$ Set of subgraph objects existing in $\mathbb{V}_{G'}$
 3: $\mathbb{S}' \leftarrow$ Sort the subgraphs in $\mathbb{S}$ by the sum of values in the degree sequences
 4: for each subgraph $S_i \in \mathbb{S}'$ do
 5:     for each subgraph $S' \in \{S_{i-w}, S_{i-w+1}, \cdots, S_{i+w}\}$ do
 6:         if $f(Q_S, Q_{S'}) < \tau$ then
 7:             $\mathbb{E}_{G'} \leftarrow \mathbb{E}_{G'} \cup \{(S, S')\}$
 8:         end if
 9:     end for
10: end for
11: $G' \leftarrow (\mathbb{V}_{G'}, \mathbb{E}_{G'})$ # the final $MRCCT$ to be returned
```

Then, we extend the cross-context information by comparing and connecting subgraph objects with similar characteristics. We connect two subgraph objects when their hierarchical degree sequences are similar. For example, subgraph S_1 of Figure 1b has a hierarchical degree sequence as $Q_{S_1} = ((1), (1, 1, 2, 1), (1))$, where each number represents the degree of a node and the nodes in each hierarchy level are combined together. We generalize the observation from previous work [9] that two subgraphs are similar if the hierarchical degree sequences are similar in terms of the Spearman's footrule distance.

Specifically, the distance between two subgraphs S_i and S_j with hierarchical degree sequences of Q_{S_i} and Q_{S_j}, respectively, is defined as follows:

$$f(Q_{S_i}, Q_{S_j}) = \sum_{h=1}^{H} \sum_{t=1}^{T(h)} \left| \tilde{Q}^h_{S_i}(t) - \tilde{Q}^h_{S_j}(t) \right|,$$

where $H = \max(|S_i|, |S_j|)$ is the maximum number of levels in the subgraphs,

$T(h) = \max(|S_i^h|, |S_j^h|)$ is the maximum number of nodes for level h, and $\tilde{Q}_S^h$ is a sorted degree sequence of subgraph S at level h. For instance, $\tilde{Q}_{S_1} = ((1), (1, 1, 1, 2), (1))$ in our example of Figure 1b; note that the order of nodes in level 2 is different from that of Q_{S_1}. We also make the lengths of two degree sequences the same for the comparison by adding zero paddings in front of each degree sequence whose length is smaller than the other.

However, it is computationally expensive to compare all pairs of subgraphs existing in the trees. Thus, we approximate the all-pairs comparison by introducing a window parameter. We first sort all subgraphs by the sum of node degrees ignoring the hierarchy. Then, we compare only the subgraphs in the sorted list whose distance is less than or equal to w; we ignore the distance between subgraphs whose sums of node degrees are not similar. This results in speeding up the whole process significantly without hurting the result.

Algorithm 2 describes how to connect many multi-resolution trees by generating cross-context information between subgraphs. In line 1, we combine all nodes and edges in a set $\mathbb{T}$ of multi-resolution trees by combining identical subgraph objects existing within different trees. In lines 2 to 3, we extract the set of unique subgraph objects and sort them by the sum of node degrees in the hierarchical degree sequences. In lines 4 to 10, we select a subgraph S' within the window w to compare with the currently selected subgraph S and add the edge when the distance is smaller than a predefined threshold τ.

3.0.4 Hierarchical GCN

H_2C_2GCN uses a graph convolutional network (GCN) [12] to extract latent representations from $MRCCT$ in an end-to-end fashion. Let $H^{(l)}$ be the feature representations of the lth layer. Then, the feature $H^{(l+1)}$ of the next layer is computed from the following propagation rule, which is called a graph convolution:

$$H^{(l+1)} = \sigma(D^{-\frac{1}{2}}AD^{-\frac{1}{2}}H^{(l)}W^{(l)}),$$

where A is the adjacency matrix, D is the degree matrix of A such that $D_{ii} = \sum_j A_{ij}$, $W^{(l)}$ is a trainable weight matrix, and $\sigma(\cdot)$ is an activation function such as ReLU.

However, a typical GCN cannot be directly used for the $MRCCT$ network $\mathbb{G}'$, which results from Algorithm 2, because the graph- and subgraph-level tree nodes in $\mathbb{G}'$ do not contain initial feature vectors. We propose Hierarchical GCN, which extends traditional GCNs to multipartite graphs where node features are given only for a subset of nodes. Our Hierarchical GCN effectively generalizes an existing approach [13] designed for bipartite graphs, making it possible to learn graph embeddings in an end-to-end fashion. For simplicity, we call the node-, subgraph-, and graph-level nodes as level 1, 2, and 3 nodes, respectively.

First, we use only the level 1 nodes to propagate their initial features to the level 2 nodes. Thus, $H^{(2)}$ is newly computed only for the level 2 nodes. Then, the level 1 and 2 nodes both participate in the second propagation, computing $H^{(3)}$ for all nodes in $\mathbb{G}'$. This is simply done by using different adjacency matrices for the hierarchical propagations. Unlike the traditional GCNs, we do not add self-loops to the adjacency matrix, because the main characteristic of multipartite graphs is that no edges exist in a single independent set.

Chapter 4

Experiments

We evaluate H_2C_2GCN on five real-world datasets to answer the following questions:

- **Q1. Classification accuracy (Section 4.0.2).** How well does H_2C_2GCN classify multiple attributed graphs compared to previous approaches?
- **Q2. Model depth (Section 4.0.3).** What is the best number of layers to include cross-context features in H_2C_2GCN?
- **Q3. Ablation study (Section 4.0.4).** Does considering cross-context features and attribute information help improve the accuracy?

4.0.1 Experimental Settings

All experiments are performed on a workstation with Intel(R) Xeon(R) E5-2630 v4 2.2GHz with 512GB of RAM and 4 GTX1080Ti GPUs.

Datasets. We use five benchmark datasets in Table 1 for graph classification. MUTAG [14] contains 188 aromatic and heteroaromatic nitro compounds, which are labeled as positive if they have a mutagenic effect on bacterium *Salmonella typhimurium*. PTC [15] consists of 344 compounds whose classes indicate the carcinogenicity of rats. PROTEINS [16] is a dataset whose nodes represent secondary structure elements and edges indicate neighborhood in the amino-acid sequence or in the 3D space. NCI1 [17] and NCI109 [5] are datasets of chemical compounds screened for

Table 1: Summary of benchmark datasets.

Dataset	# of Graphs	# of Classes	# of Attributes	Average # of Nodes
MUTAG[1]	118	2	7	17.9
PTC[1]	344	2	19	25.5
PROTEINS[1]	1,113	2	3	39.1
NCI1[1]	4,110	2	37	29.8
NCI109[1]	4,147	2	38	29.6

activity against non-small cell lung cancer and ovarian cancer cell lines, respectively. All of our datasets are publicly available.[1]

Experimental Setup. We use 80% of each dataset as a training set and the remaining 20% as a test set, while reserving 20% of the training set as a validation set. We use a three layer Hierarchical GCN for H_2C_2GCN except in Section 4.0.3 where we study the effect of the depth. We train each model for 100 epochs with a learning rate of 0.005, hidden layer size of 64, L2 regularization with weight decay of 0.001, and dropout rate of 0.5, while using the Adam optimizer.

Competitors. We compare H_2C_2GCN to the following competitors.

- **WL Kernel [5]** uses label compression to relabel its nodes and calculates the similarity of two graphs using a kernel function.
- **Deep WL Kernel [18]** extends the WL kernel by calculating an additional kernel matrix that considers the similarities between subgraphs.
- **node2vec [3]** builds a corpus of random walks from each graph and learns a skip-gram model to learn the embedding vectors of nodes. We average the

[1] https://ls11-www.cs.tu-dortmund.de/staff/morris/graphkerneldatasets

embeddings of all nodes in each graph to make graph embeddings.

- **graph2vec [8]** is also based on the skip-gram model, but it is designed to provide graph-level embeddings instead of node-level ones.
- **MrMine [9]** is a classification method for graphs without node attributes, and uses Deepwalk [1] to extract the embedding of objects.

Table 2: Accuracy of H_2C_2GCN and baselines for graph classification. The average and standard deviation are reported over five runs with different random seeds. Our H_2C_2GCN shows the best performance for all datasets.

	MUTAG	**PTC**	**PROTEINS**	**NCI1**	**NCI109**
WL Kernel	76.76 ± 1.82	56.47 ± 1.34	71.53 ± 1.10	79.66 ± 0.45	80.02 ± 2.17
Deep WL	79.46 ± 0.89	57.65 ± 0.45	72.16 ± 1.30	80.02 ± 1.79	80.36 ± 1.22
node2vec	66.49 ± 2.61	56.47 ± 1.95	54.95 ± 1.87	53.89 ± 2.92	51.47 ± 3.05
graph2vec	77.84 ± 1.92	60.59 ± 1.10	72.97 ± 1.22	73.07 ± 1.14	74.50 ± 2.51
MrMine	82.70 ± 1.52	58.24 ± 0.55	69.82 ± 1.10	68.15 ± 0.84	65.55 ± 0.45
H_2C_2GCN	**83.24 ± 1.30**	**60.88 ± 0.55**	**73.96 ± 0.84**	**82.73 ± 1.22**	**82.38 ± 1.82**

4.0.2 Classification Accuracy

Table 2 shows the classification accuracy of H_2C_2GCN and its competitors. H_2C_2GCN gives the highest accuracy for all five datasets, proving the effectiveness of its main ideas. The improvement over MrMine [9] is especially noticeable when the numbers of graphs and node attributes are large because H_2C_2GCN extracts the graph embeddings through Hierarchical GCN in an end-to-end way considering the node attributes, while MrMine does not consider the attributes and separates the embedding and classification steps.

At the same time, the previous approaches that use only a single resolution view perform worse than H_2C_2GCN. node2vec [3] produces near random performance for large datasets, as it considers only nearsighted views when extracting graph embeddings as the average of all node embeddings. WL Kernel [5], Deep WL Kernel [18], and graph2vec [8] perform better than node2vec, as they extract graph embeddings directly from the structure, but still show lower accuracy than that of H_2C_2GCN which utilizes multi-resolution features.

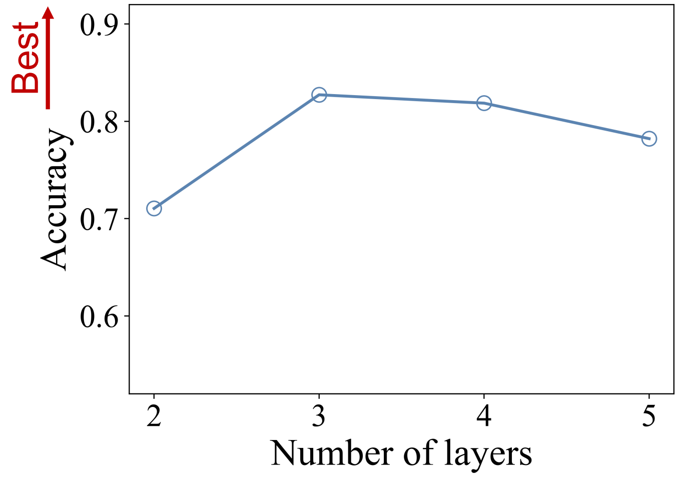
4.0.3 Model Depth

We investigate the effect of the number of convolution layers in Hierarchical GCN, which is a core module of H_2C_2GCN that extracts the graph embeddings for downstream tasks. We evaluate the accuracy of graph classification on the NCI1 dataset changing the number of layers from 2 to 5 in Figure 3a. Unlike a typical GCN that works the best when the number of layers is 2 [12], H_2C_2GCN shows its highest accuracy when the number of layers is 3. This is because our Hierarchical GCN is applied to tripartite graphs consisting of node, subgraph, and graph objects where only the node objects have initial features. H_2C_2GCN with two layers can propagate the node attributes to graph objects, but cannot utilize the cross-context edges between subgraph objects. With three layers, it is possible for H_2C_2GCN to utilize all connections in the $MRCCT$ by considering both multi-resolution and cross-context information. The accuracy slightly drops when more layers are added after 3, since the increased number of parameters leads to overfitting, but is still higher than that of two layers.

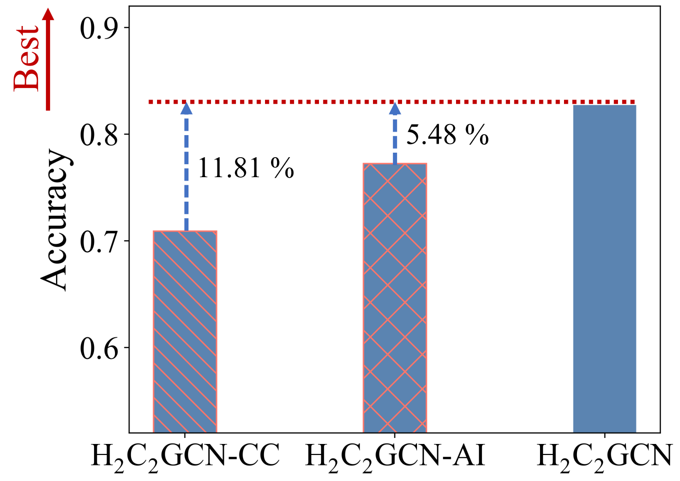
4.0.4 Ablation Study

We perform an ablation study of H_2C_2GCN on the NCI1 dataset to verify the effects of cross-context features and attribute information of nodes. We first examine the effect of cross-context features by removing the cross-context edges from $MRCCT$, ignoring lines 2 to 11 of Algorithm 2 when generating $MRCCT$ from the given trees. Figure 3b shows that considering the cross-context features enhances the accuracy by 11.81 percent points. This illustrates that considering cross-context features is crucial in graph classification.

We also examine the effect of attribute information by discarding the node attributes from the node objects of $MRCCT$ and transforming node IDs into one-hot vectors as their initial features. Figure 3b shows that considering the node attributes enhances the performance by 5.48 percent points, demonstrating that node attributes essential information for extracting accurate embeddings.



(a) Effect of model depth



(b) Effects of cross-context features and node attribute information

Figure 3: Accuracy of H_2C_2GCN with different settings on the NCI1 dataset. (a) H_2C_2GCN performs the best when the number of layers is three, which is the minimum number of layers to consider cross-context edges. (b) The cross-context features and node attributes are effective for improving the classification accuracy. $H_2C_2GCN\text{-}CC$ and $H_2C_2GCN\text{-}AI$ represent H_2C_2GCN without cross-context consideration and without attribute information, respectively.

Chapter 5

Conclusion

We propose H_2C_2GCN, a graph classification method which considers cross-context, multi-resolution, and attribute information to improve the classification accuracy. H_2C_2GCN maps multiple graphs to a new network $MRCCT$, and discovers features thanks to its cross-context and multi-resolution analysis. We propose Hierarchical GCN, a modified GCN capable of extracting embeddings on a hierarchically structured graph and apply it on $MRCCT$ to extract graph embeddings while adding support for attribute information to enrich the extracted embeddings. Extensive experiments show that H_2C_2GCN classifies multiple attributed graphs the most accurately.

References

[1] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710, 2014.

[2] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[3] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864, 2016.

[4] F. Fouss, A. Pirotte, J.-M. Renders, and M. Saerens, "Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation," *IEEE Transactions on knowledge and data engineering*, vol. 19, no. 3, pp. 355–369, 2007.

[5] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels.," *Journal of Machine Learning Research*, vol. 12, no. 9, 2011.

[6] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt, "Graph kernels," *The Journal of Machine Learning Research*, vol. 11, pp. 1201–1242, 2010.

[7] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, "struc2vec: Learning node representations from structural identity," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 385–394, 2017.

[8] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal, "graph2vec: Learning distributed representations of graphs," *arXiv preprint arXiv:1707.05005*, 2017.

[9] B. Du and H. Tong, "Mrmine: Multi-resolution multi-network embedding," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 479–488, 2019.

[10] A. Narayanan, M. Chandramohan, L. Chen, Y. Liu, and S. Saminathan, "subgraph2vec: Learning distributed representations of rooted sub-graphs from large graphs," *arXiv preprint arXiv:1606.08928*, 2016.

[11] U. Kang, H. Tong, and J. Sun, "Fast random walk graph kernel," in *Proceedings of the 2012 SIAM international conference on data mining*, pp. 828–838, SIAM, 2012.

[12] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[13] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 974–983, 2018.

[14] A. K. Debnath, R. L. Lopez de Compadre, G. Debnath, A. J. Shusterman, and C. Hansch, "Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity," *Journal of medicinal chemistry*, vol. 34, no. 2, pp. 786–797, 1991.

[15] C. Helma, R. D. King, S. Kramer, and A. Srinivasan, "The predictive toxicology challenge 2000–2001," *Bioinformatics*, vol. 17, no. 1, pp. 107–108, 2001.

[16] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. Vishwanathan, A. J. Smola, and H.-P. Kriegel, "Protein function prediction via graph kernels," *Bioinformatics*, vol. 21, no. suppl_1, pp. i47–i56, 2005.

[17] N. Wale, I. A. Watson, and G. Karypis, "Comparison of descriptor spaces for chemical compound retrieval and classification," *Knowledge and Information Systems*, vol. 14, no. 3, pp. 347–375, 2008.

[18] P. Yanardag and S. Vishwanathan, "Deep graph kernels," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1365–1374, 2015.

요 약

어떻게 구조적 특성과 노드의 레이블을 활용하여 속성 그래프를 분류 할 수 있을까? 그래프 분류는 데이터 마이닝 분야에서 중대한 과제로 여겨진다, 특히나 생물 정보 영역에서 화학 물질들이 속성 그래프로 표현되어 있는 경우에는 더욱 중요하다. 그러나 기존 연구들은 그래프 커널 방식이나 무작위 행보 방식을 사용하여, 그래프 내에 하나의 해상도 (노드 또는 부분그래프) 에 한정되어서 특징들을 고려한다. 이와 같이 하나의 해상도에 집중하여 특징을 고려할 경우 그래프 전체에 대한 편향된 시선으로 바라볼 수밖에 없다. 즉, 그래프들에 대하여 좁게 또는 넓게 바라보므로 그래프 간의 특징을 구분하는데 큰 어려움이 있다.

이 논문에서는 그래프 분류에 종단 간 학습이 가능한 H_2C_2GCN (**H**eterogeneous and **H**ierarchical **C**ross-**c**ontext **G**raph **C**onvolution **N**etwork)를 제안한다. 다수의 속성 그래프가 주어졌을 시, H_2C_2GCN 는 다수의 해상도를 지닌 교차 문맥 간선이 이어진 트리를 만든다. 이를 통하여 다수의 그래프 간의 노드 레이블 및 구조적 특성의 견해를 담을 수 있다. 만들어진 트리에서 그래프 합성곱 신경망을 사용하여 하여 각 그래프의 임베딩을 추출하게 된다. 실험을 생물 정보 데이터에 대하여 평가를 하여 H_2C_2GCN 가 기존 방법들에 비하여 높은 정확도를 가지는 것을 확인할 수 있다.

주요어 : 그래프 분류, 그래프 합성곱 신경망, 계층 그래프, 속성 그래프

학번 : 2019-20235

감사의 글

이 논문을 완성하기까지 많은 분들의 도움이 있었습니다. 도움을 주신 모든 분께 감사의 인사를 드립니다. 먼저 석사 학위 과정을 잘 마칠 수 있도록 연구에 대해 많은 가르침을 주시고 세심한 지도를 해주신 강유 교수님께 가장 큰 감사의 인사를 드립니다. 또한 바쁘신 와중에도 논문 심사를 맡아주시고 조언을 해주신 김선 교수님과 김건희 교수님께도 감사드립니다.

석사 과정 중 많은 도움을 주었던 데이터 마이닝 연구실 식구들에게도 감사의 말씀을 드립니다. 기쁜 일과 힘든 일을 함께 공유하면서 부족한 저에게 많은 조언을 줘서 많이 배울 수 있었습니다. 마지막으로 저에게 무한한 신뢰와 응원을 보내는 저희 가족에게도 감사하다는 말을 전합니다.