Dual Space Graph Contrastive Learning

Haoran Yang†
haoran.yang-2@student.uts.edu.au
University of Technology Sydney
Sydney, NSW, Australia

Hongxu Chen†
hongxu.chen@uts.edu.au
University of Technology Sydney
Sydney, NSW, Australia

Shirui Pan
shirui.pan@monash.edu
Monash University
Melbourne, VIC, Australia

Lin Li
cathylilin@whut.edu.cn
Wuhan University of Technology
Wuhan, Hubei, China

Philip S. Yu
psyu@cs.uic.edu
University of Illinois at Chicago
Chicago, Illinois, U.S.A

Guandong Xu†
guandong.xu@uts.edu.au
University of Technology Sydney
Sydney, NSW, Australia

ABSTRACT
Unsupervised graph representation learning has emerged as a powerful tool to address real-world problems and achieve huge success in the graph learning domain. Graph contrastive learning is one of the unsupervised graph representation learning methods, which recently attracts attention from researchers and has achieved state-of-the-art performances on various tasks. The key to the success of graph contrastive learning is to construct proper contrasting pairs to acquire the underlying structural semantics of the graph. However, this key part is not fully explored currently, most of the ways generating contrasting pairs focus on augmenting or perturbing graph structures to obtain different views of the input graph. But such strategies could degrade the performances via adding noise into the graph, which may narrow down the field of the applications of graph contrastive learning. In this paper, we propose a novel graph contrastive learning method, namely Dual Space Graph Contrastive (DSGC) Learning, to conduct graph contrastive learning among views generated in different spaces including the hyperbolic space and the Euclidean space. Since both spaces have their own advantages to represent graph data in the embedding spaces, we hope to utilize graph contrastive learning to bridge the spaces and leverage advantages from both sides. The comparison experiment results show that DSGC achieves competitive or better performances among all the datasets. In addition, we conduct extensive experiments to analyze the impact of different graph encoders on DSGC, giving insights about how to better leverage the advantages of contrastive learning between different spaces.

CCS CONCEPTS
• Computing methodologies → Semantic networks; Unsupervised learning.

†Both authors contributed equally to this research.
†Corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

WWW ’22, April 25–29, 2022, Virtual Event, Lyon, France
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9096-5/22/04...
https://doi.org/10.1145/3485447.3512211

1 INTRODUCTION
Graph neural networks (GNNs) [5, 9, 12, 25, 28] leverage the expressive power of graph data in modeling complex interactions among various entities under real-world scenarios including social networks [12] and molecules [7]. GNNs are able to process variable-size graph data and learn low-dimensional embedding via an iterative process of transferring and aggregating the semantics from topological neighbors. They have achieved huge success on various tasks. GNNs require a large amount of manually labelled data to learn informative representations via the supervised learning protocol. However, it is not always realistic to obtain enough effective labels due to expensive labour work and privacy concerns in some context. Thus, unsupervised approaches such as reconstruction based methods [24] and graph contrastive learning methods [10, 21, 31] are coupled with GNNs to tackle the problem in a self-supervised manner. Recently, graph contrastive learning has emerged as a successful method for unsupervised graph representation learning and achieved state-of-the-art on various tasks including node classification, graph classification as well as applications in recommender systems [8, 30] and drug discovery [27].

The key idea of graph contrastive learning is to construct proper contrasting pairs and try to learn the underlying important structural semantics of the input graph. However, existing approaches to generate contrasting pairs are quite limited, the majority of these methods are graph perturbations including node dropping, edge perturbation and attribute masking [31, 33]. As a result, these methods introduce noisy signals to produce corrupted views, which degrade the model performance [10]. Therefore, it is always a confronting challenging problem to design effective ways to generate contrasting pairs. Let us look closer, what graph contrastive learning actually does is that it pushes negative pairs far away from each other and make positive pairs as similar as possible to each other. For negative pairs, it is easier to distinguish them against irrelevant graphs derived from other datasets [21] or other irrelevant graphs
in the same training batch [31], which will not involve significant noises. In contrast, the positive samples are usually derived from the input graph via graph perturbation methods, and the noise is inevitable. To avoid introducing too much noise, many researchers instead adopt sub-graph sampling to generate positive samples [19, 21, 31]. The intuition is that each sub-graph has unique focus on certain aspects of the graph semantics and different sub-graphs or local structures can hint the full spectrum of the semantics [31] carried on the graphs. Nevertheless, how to generate sufficient views containing unique and informative features is still an open problem.

Inspired by recent progress of geometric graph mining in hyperbolic space [3, 4], which has achieved satisfactory results under various real-world scenarios, e.g., recommendation systems [15], we innovatively utilize the greater expressiveness of different embedding spaces including the hyperbolic space and the Euclidean space to generate multiple views of the input graph. There are two major advantages of the hyperbolic space. The first is that the hyperbolic space takes smaller space to accommodate a given graph with complex structures, which means hyperbolic embedding has unique expressiveness compared to its counterpart in Euclidean space. The most obvious uniqueness is that Euclidean space expands polynomially, while the hyperbolic space expands exponentially. Therefore, we can leverage that with much lower dimensional embeddings to represent a graph, which will result in purer, compact but powerful embedding spaces. According to the experimental results that showcase the impact of hidden dimensions on performances of our proposed DSGC framework, it is shown that our model achieves stable and better performances within smaller dimensional hyperbolic embedding spaces. The second advantage is that the hyperbolic space is more capable of capturing hierarchical structures exhibiting in graph data, which implies that hyperbolic space is also suitable to be used as a special view for graph contrastive learning paradigm.

Despite the advantages of hyperbolic space embedding, one cannot ignore the power of the traditional Euclidean space [9, 12, 25, 28]. Compared to the hyperbolic space, vector calculation in Euclidean space is more efficient as distance metrics in Euclidean space does not need inverse trigonometric functions. Moreover, metrics in Euclidean space is able to differentiate the relative distances among the data points (i.e., satisfying the triangle inequality), and the superiority is evidenced by various famous clustering methods, including k-NN [6] and HDBSCAN [2]. Thus, introducing both the hyperbolic and Euclidean spaces together to generate distinct views could illustrate the hierarchical topology and relative distances among the nodes in the original graph, which are beneficial to graph contrastive learning. To fully leverage such advantages, we propose to select different graph samplers to sample sub-graphs in different spaces. Here we argue that random sub-graph extraction cannot fully leverage the advantages of contrasting between Euclidean space and the hyperbolic space. As mentioned previously, different spaces have different advantages, it is better to fully leverage both of them. Hence, we should have different strategies to sample sub-graphs. For Euclidean space, we need to sample a sub-graph that is able to demonstrate the skeleton of the input graph (i.e., illustrating relative locations among nodes). For hyperbolic space, a sub-graph carrying hierarchical information and topology structure is expected. Further, we note that all the current works [10, 19, 21, 31] on graph contrastive learning adopt the same graph encoder to encode different views. We question that such a strategy will limit the distinctions among various views, which may undermine the performances of graph contrastive learning. To avoid this, we also propose different graph encoders for different views.

The main contributions of this paper are summarized as follows:

- We first utilize graph representations in different spaces to construct contrasting pairs, which could leverage the advantages from both the hyperbolic space and the Euclidean space.
- We innovatively propose to select different graph encoders for representation learning in different spaces, which is also verified by the experiments as feasible way to generate different views of the input graph.
- We conduct extensive experiments to show the superiority of the proposed DSGC on all the datasets comparing to all the baselines. Detailed study is executed to analyze the strategy about selecting different graph encoders for the graph representation learning in different spaces. We also illustrate if the model can achieve better performances with lower hidden dimensions to verify whether DSGC leverages the advantages of the hyperbolic space.

2 RELATED WORK

2.1 Graph Contrastive Learning

Graph contrastive learning methods are currently state-of-the-art in unsupervised graph representation learning. Deep Graph Infomax (DGI) [26] first introduced Deep Infomax [11] into graph learning and achieve satisfying results via maximize the mutual information between local structure and global context. Some other works generalized such a strategy and try to measure the mutual information between different instances. For example, GCC [21] conducted contrastive learning among different sub-graphs extracted from different input graphs, which can be regarded as the mutual information estimation among different local structures. GraphCL [31] is another type, which are different from GCC. It maximized mutual information between the input graph and the augmented graph, which is contrasting between two different global views. We note that most current graph contrastive learning methods including [10, 20, 33] all conduct contrastive learning between corrupted or augmented graphs or sub-graphs. The methods to generate different view of the input graph for contrastive learning are not rich, most of them generate different views via perturbing graph structures.

2.2 Hyperbolic Representation Learning

Recently, the hyperbolic space has been explored as a latent space, which is suitable for various tasks. It has shown its impressive performance comparing to the Euclidean space and achieves better performances in many domains including graph representation learning [4, 16, 23] and recommendation systems [14, 32]. One key advantage of the hyperbolic space is that it expands faster than the Euclidean space. The Euclidean space expands polynomially, while the hyperbolic space expands exponentially. It suggests that the hyperbolic space would take smaller space to embed a group of data than the Euclidean space does. Another property of the
hyperbolic space is that it can preserve the hierarchies of data. Many real-world graph data including social networks and e-commercial networks exhibit hierarchical tree-like structures. The hyperbolic space is suitable to serve as a tool to describe such graph data [1, 17]. However, we think both the hyperbolic space and the Euclidean space have their own advantages, and we should try to leverage both of them instead of solely utilizing one of them.

3 PRELIMINARIES

Dual space graph contrastive learning aims to capture multi-level graph semantics in different spaces. We mainly focus on contrasting between Euclidean space and non-Euclidean spaces. Specifically, we contrast graph semantics in Euclidean space and hyperbolic space, which are both widely used spaces for graph representation learning. To better understand the proposed method, in this section, we give introductions to preliminaries about hyperbolic space and the transformation between Euclidean space and hyperbolic space.

3.1 Hyperbolic Space

Various hyperbolic models are created by researchers to solve problems under different scenarios. The Poincaré ball model \( \mathbb{D} \), the Klein model \( \mathbb{K} \), the half-plane model \( \mathbb{P} \), the hyperboloid model \( \mathbb{H} \), and the hemisphere model \( \mathbb{J} \) are five well-known hyperbolic models [3]. All of them have different properties. Among them, the Poincaré ball model is the most suitable one for representation learning since it can be tuned by gradient-based optimization [18]. The definition domain of the Poincaré ball model with a constant negative curvature \( -c \) is:

\[
\mathbb{D} = \{(x_1, ..., x_n) : x_1^2 + \cdots + x_n^2 < \frac{1}{c^2}\}.
\] (1)

It is an \( n \)-dimensional open ball in \( \mathbb{R}^n \). The straight lines in Euclidean space are corresponding to the geodesics in the Poincaré ball. Given two points in the Poincaré ball \( u, v \), the similarity between these two points is the reciprocal of the length of geodesics linking the two points, which is defined as:

\[
d_\mathbb{D}(u, v) = \frac{1}{\text{arccosh}(1 + \frac{2||u-v||^2}{1-||u||^2(1-||v||^2)})},
\] (2)

when the constant negative curvature is \(-1\), where \( || \cdot || \) is the Euclidean norm and \( \text{arccosh}(\cdot) \) is the inverse hyperbolic cosine function. Note that, there is only one center in the Poincaré ball, which is the origin. Let us take the origin as the root node, the leaf nodes will spread from origin layer by layer, capturing the tree topology and hierarchical structure information of the graph [32].

3.2 Space Mapping

We cannot directly apply Euclidean gradient descent optimization methods to the hyperbolic space, because the gradient in the hyperbolic space is different from the Euclidean gradient. It means that existing graph representation learning methods in Euclidean space is not suitable for graph learning in the hyperbolic space. To leverage existing methods for graph learning, there is a solution, which is mapping the hyperbolic embeddings to the Euclidean embeddings. Here, we introduce the mechanism of such mappings.

There are two mappings, which are called exponential mapping and logarithmic mapping, respectively. The mapping from tangent space \( T_o \mathbb{D}_c \), which is an Euclidean space with \( o \) as the origin, to the hyperbolic space \( \mathbb{D}_c \) with the constant negative curvature \( -c \) is exponential mapping, and the mapping from hyperbolic space \( \mathbb{D}_c \) to tangent space \( T_o \mathbb{D}_c \) is logarithmic mapping [4]. Taking origin \( o \) as the target point ensures the simplicity and symmetry of the mapping [32]. With \( t \in T_o \mathbb{D}_c \) and \( u \in \mathbb{D}_c \), the exponential mapping \( \exp^c_o : T_o \mathbb{D}_c \to \mathbb{D}_c \) and the logarithmic mapping \( \log^c_o : \mathbb{D}_c \to T_o \mathbb{D}_c \) are defined as:

\[
\exp^c_o(t) = \tanh(\sqrt{c||t||}) \frac{t}{\sqrt{c||t||}},
\] (3)

\[
\log^c_o(u) = \text{artanh}(\sqrt{c||u||}) \frac{u}{\sqrt{c||u||}}.
\] (4)

With two mapping functions above, we can map hyperbolic embeddings to Euclidean space for further process. Matrix multiplication, bias addition, and activation are three common embedding processing methods in neural network models.

\[
y = \sigma(W \cdot u + b).
\] (5)

The weight matrix and bias are usually defined in the Euclidean space, which means that they cannot be directly processed together with hyperbolic embeddings. To achieve that, we must first map hyperbolic embeddings to Euclidean embeddings. For hyperbolic matrix multiplication, we have:

\[
W \otimes u = \exp^c_o(W \cdot \log^c_o(u)),
\] (6)

for hyperbolic bias addition, we have:

\[
u \circ b = \exp^c_o(\log^c_o(u) + b),
\] (7)

and for activation, we have:

\[
y = \exp^c_o(\sigma(\log^c_o(W \otimes u + b))).
\] (8)

Leverage the transformations mentioned above, we can utilize the existing graph learning methods in the Euclidean space to process the graph learning problems in the hyperbolic space.

4 METHODOLOGY

In this section, we introduce details of the proposed Dual Space Graph Contrastive learning, a semi-supervised graph representation learning framework, namely DSGC.

4.1 Overview

The overview of the proposed framework is shown in Figure 1. The entire framework can be devided into two parts, one for labeled graph and one for unlabeled graph. In each part, there are two different views, which are Euclidean and hyperbolic, respectively. More than the supervision signals from labels, we leverage graph contrastive learning between Euclidean space and hyperbolic space to obtain more informative self-supervised signals to enhance the ability to learn graph representations. Algorithm 1 summarizes how we combine the information from Euclidean space and hyperbolic space via updating model parameters according to the training objective of DSGC.

The figure and the algorithm provide basic understandings about our proposed framework DSGC. In the following subsections, we will give you more details.
Figure 1: The overview of the proposed method. DSGC first adopts two different graph samplers to sample sub-graphs for the two different sapces on all graphs. Then, the generated sub-graphs will be fed into different graph encoders and processed in different spaces. After obtaining embeddings of all the graphs, we conduct graph contrastive learning, which is maximizing the similarity between the Euclidean embedding and the hyperbolic embedding of the same graph and minimizing the similarity between the hyperbolic embeddings of different graphs. To introduce the supervised signals into the model, the Euclidean embeddings of the labeled graphs will be fed into downstream predictor to predict the labels and the cross-entropy loss function will be utilized to update the model.

4.2 Graph Sampling
The success of graph contrastive learning mainly relies on how to generate proper contrasting pairs. As what we discussed in the introduction, we adopt sub-graph sampling, as many works do [21, 31], to generate the initial different views. Many methods can derive sub-graphs from a graph, including Depth First Search, Breath First Search, RandomWalk. Plus, sub-graph sampling is widely used as a preprocessing method for graph representation learning since it can discard noises in graphs [9]. Give a graph $G = (V, E, X)$, where $V$ denotes node set, $E$ denotes edge sets, $x$ denotes features, and a graph sampler $S(\cdot)$, for the graph $G$, we can have its sub-graph as follows:

$$SG = S(G) \quad (9)$$

In our work, specifically, we take DiffusionSampler [22] to acquire a more complete and unbiased view comparing to other RandomWalk based sampling methods to serve as a skeleton of the input graph, which will be fed into Euclidean space to process and learn. For hyperbolic space, we utilize CommunityStructureExpansion-Sampler [22] to extract hierarchical information and tree-topology structures of the input graph.

4.3 Graph Encoder
Graph encoders are critical to graph representation learning since we cannot convert input graphs to embeddings without them. GNN models are a satisfying branch of graph encoder leveraging the advantages of neural networks. In our proposed DSGC, we apply various GNN models to learn graph embeddings. Researchers have developed various GNN models, which can be categorized as different types, for example, convolutional GNN models in spectral domain including GCN [12], message passing based GNN models including GraphSAGE [9], attention mechanism based GNN models including GAT [25] and , and structure information aware GNN models including GIN [28]. Note that, for hyperbolic GNN models, they actually have the same structures as the Euclidean GNN models. The different point is that the output embeddings of hyperbolic GNN models will be mapped to hyperbolic space, and the hyperbolic embeddings will formulate loss function in hyperbolic space. The hyperbolic loss function trains the hyperbolic GNN models to enforce them to output proper hyperbolic embeddings.

Give a graph $G = (V, E, x)$, and a graph encoder $g(\cdot)$, for the graph $G$, we can have its embeddings in Euclidean space and hyperbolic space, respectively, as follows:

$$H^E = g_E(G). \quad (10)$$

$$H^H = \exp_C(g_H(G)). \quad (11)$$

For graph-level learning tasks, we need an extra readout function to summarize the updated node embeddings to output the graph representation.

Graph embeddings are critical to the whole framework due to their strong representative ability. We now can feed them to the following components and move forward to the next step.

4.4 Space Transformation
As mentioned in the abstract, different spaces have their own advantages and features, it is unwise to pick up only one of them to
Algorithm 1 DSGC algorithm

Input: Labeled graph $G_l$ and its label $p_l$; unlabeled graph $G_u$; graph sampler $S_l(\cdot, \cdot)$; graph sampler $S_u(\cdot, \cdot)$; sampling rate for graph processed in hyperbolic space $\alpha_H$; sampling rate for graph processed in Euclidean space $\alpha_E$; trainable hyperbolic GNN model $g_H(\cdot)$; trainable Euclidean GNN model $g_E(\cdot)$; trainable predictor $P(\cdot)$; cross-entropy function $C(\cdot)$; unlabelled loss weight $\lambda_u$.

Output: Training objective of DSGC, $L$;

1. Sampling sub-graphs for labeled graph: $SG_l^H = S_H(G_l, q_H)$, $SG_l^E = S_E(G_l, q_E)$;
2. Sampling sub-graphs for unlabeled graph: $SG_u^H = S_H(G_u, q_H)$, $SG_u^E = S_E(G_u, q_E)$;
3. Generating embeddings for sampled sub-graphs via GNN models: $H_l^H = \exp_p^E(g_H(SG_l^H))$, $H_l^E = g_E(SG_l^E)$, $H_u^H = \exp_p^E(g_H(SG_u^H))$, $H_u^E = g_E(SG_u^E)$;
4. Mapping Euclidean embeddings of sub-graphs to hyperbolic space: $H_l^{E \rightarrow H} = \exp_p^H(H_l^E)$, $H_u^{E \rightarrow H} = \exp_p^H(H_u^E)$;
5. Hyperbolic distance calculation for positive pair: $d_l^H(H_l^H, H_l^H)$, $d_u^H(H_u^H, H_u^H)$;
6. Hyperbolic distance calculation for negative pair: $d_u^H(H_u^H, H_u^{E \rightarrow H})$;
7. InfoNCE loss for labeled and unlabeled data: $L_{NCE} = \frac{1}{N_l} \sum_{i=1}^{N_l} \log \frac{e^{d_l^H(H_l^H, H_l^H)} + e^{d_l^H(H_l^H, H_l^E)}}{e^{d_l^H(H_l^H, H_l^H)} + e^{d_l^H(H_l^H, H_l^{E \rightarrow H})}}$, $L_{NCE} = \frac{1}{N_u} \sum_{i=1}^{N_u} \log \frac{e^{d_u^H(H_u^H, H_u^H)} + e^{d_u^H(H_u^H, H_u^{E \rightarrow H})}}{e^{d_u^H(H_u^H, H_u^H)} + e^{d_u^H(H_u^H, H_u^{E \rightarrow H})}}$;
8. Generating probabilistic distribution for labeled graph: $p = \delta(P(H_l^H))$;
9. Cross-entropy loss function for labeled graph: $C(p, p_l)$;
10. return $L = C(p, p_l) + L_{NCE} + \lambda_u L_{NCE}$.

4.5 Probabilistic Predictor

The task of probabilistic predictor is to predict the label of the input graph. The predicted label will be fed into cross-entropy loss function to update model parameters to introduce supervision signals. Here, we take Multi-Layer Perceptron (MLP) as the predictor to map the Euclidean graph embedding to low-dimension space to acquire probabilistic distribution regarding all labels:

$$p = \delta(P(H_l^E^H)), p \in \mathbb{R}^K,$$

where $K$ denotes that there are $K$ different labels in the graph dataset and $\delta(\cdot)$ denotes sigmoid function.

4.6 Loss Functions

In this section, we aim to formulate a loss function to train the framework, which should include self-supervised loss and supervised loss.

Objective of self-supervised learning phase is graph contrastive learning between graph hyperbolic embedding $H_l^H$ and transformed graph Euclidean embedding $H_l^{E \rightarrow H}$ in the hyperbolic space. Since these two embeddings are different views of the same input graph, hence, they formalize a positive pair and should be similar, which means that the distance between the two embeddings should be small enough. So, one of our self-supervised learning objective is to minimize the distance between the two embeddings $d_u^H(H_l^H, H_u^{E \rightarrow H})$. Note that $d_u^H(\cdot, \cdot)$ is defined in Section 3.1. Moreover, with a positive pair, we also need negative pairs in order to fulfill the graph contrastive learning process. We borrow the idea from GraphCL [31], which is coupling other graphs in the same batch with the input graph to form the negative pairs. More specifically, assume that there are $N$ unlabeled graphs and a labeled graph in a batch. In our settings, we couple the labeled graph with $N$ unlabeled graphs in the same batch to form negative pairs for graph contrastive learning for the labeled graph and couple an unlabeled graph with the labeled graph in the same batch to form negative pairs for graph contrastive learning for each unlabeled graph. Consider that we first try to minimize the distance between the postive pair in the hyperbolic space, we need to minimize the agreement or maximize the distance between the negative pair in the hyperbolic space. The two-phase graph contrastive learning procedure mentioned above can be described as:

$$L_{contra} = L_{NCE}^l + L_{NCE}^u + \frac{1}{\tau} \log \frac{e^{d_l^H(H_l^H, H_l^{E \rightarrow H})}}{e^{d_l^H(H_l^H, H_l^{E \rightarrow H})} + e^{d_u^H(H_l^H, H_l^{E \rightarrow H})}} + \frac{1}{\tau} \log \frac{e^{d_u^H(H_u^H, H_u^{E \rightarrow H})}}{e^{d_u^H(H_u^H, H_u^{E \rightarrow H})} + e^{d_u^H(H_u^{E \rightarrow H}, H_u^{E \rightarrow H})}},$$

where we introduce $\tau$ to serve as a temperature hyperparameter.

For supervised learning task, it will be conducted only on labeled graph. In this paper, we mainly focus on graph classification tasks. Therefore, we adopt cross-entropy function to measure the gap between predicted results and the ground-truth. So, the loss function of supervised learning is:

$$L_{sup} = C(p, p_l)$$
Table 1: Statistics of datasets

| Name       | Num. of Graphs | Num. of Classes | Avg. Number of Nodes | Avg. Number of Edges |
|------------|----------------|-----------------|----------------------|----------------------|
| MUTAG      | 188            | 2               | 17.93                | 19.79                |
| REDDIT-BINARY | 978          | 2               | 243.11               | 288.53               |
| COLLAB     | 5,000          | 3               | 74.49                | 2457.78              |

where \( C(\cdot, \cdot) \) is the cross-entropy function, \( p \) denotes the predicted results and \( p_i \) denotes the ground-truth.

Then, we have the overall objective for DSGC:

\[
L = L_{sup} + \omega \cdot L_{contra}
\]

where \( \omega \) is the hyper-parameter to control the impact of graph contrastive learning on the model.

5 EXPERIMENTS

5.1 Datasets

To fully demonstrate the performances of DSGC, we choose three public and wildly-used datasets of three categories including chemical compounds, social networks, and citation networks. which are MUTAG [13], REDDIT-BINARY [29], and COLLAB [29]. MUTAG is a collection of nitroaromatic compounds and the goal is to predict their mutagenicity on Salmonella typhimurium. REDDIT-BINARY is a balanced dataset where each graph corresponds to an online discussion thread where nodes correspond to users, and there is an edge between two nodes if at least one of them responded to another’s comment. COLLAB is a scientific collaboration dataset. A graph corresponds to a researcher’s ego network.

All the datasets mentioned above are available on the website\(^1\). The statistics of datasets are shown in Table 1. Note that, to implement sampling algorithms and meet their requirements, we discard the unconnected graphs in all datasets.

5.2 Baselines

To verify the effectiveness of the proposed framework, we compare it with several baselines in two categories. The first one is graph neural network models, including GCN [12], GraphSAGE [9], GAT [25], and GIN [28]. The second one is novel graph contrastive learning methods, including GCC [21] and GraphCL [31]. To ensure the fairness of experiments, for GNN models, we take matrix completion task as the training objective on unlabeled data following the protocol in [24], in which case GNN models would receive enough self-supervised signals from unlabeled data.

5.3 Experimental Settings

For reproducibility, we introduce the detailed settings of the proposed DSGC. First, to maintain the reliability of the experimental results, we run 10 times on each dataset, each time we have 10% data as the testing set. Then, we follow the protocol of transductive learning and take all the data as the training set and have a samll ratio of labeled data. Note that labeled data in training set does not contain any data in testing set. The metric to measure the performances of all the methods is classification accuracy. Moreover, to implement graph sampling algorithms, we discard all the graphs containing isolated nodes. All the experiments were conducted on NVIDIA TITAN Xp. We utilized PyTorch (version 1.7.0) and PyTorch Geometric (version 1.6.3) to implement our method. The hyper-parameter settings of DSGC for comparison experiment is in the appendix.

5.4 Experimental Results

5.4.1 Does the framework have the superiority comparing to the baselines? The comparison experiment results for all baselines and our proposed methods on all three datasets are shown in Table 2. Generally, the proposed DSGC method outperforms the best baselines. Note that, DSGC has much lower standard error comparing to the baselines, which shows that our proposed method is more stable when facing the datasets having different distributions. We also find that DSGC has more advantages on dataset COLLAB. It shows that the proposed contrastive learning process is able to acquire more unsupervised signals providing the model more informative semantics to achieve the best performances.

As to the baselines, GCC and GraphCL consistently outperform GNN models, which indicates contrastive learning in graph representation learning domain is feasible and could achieve better performances. Moreover, GraphCL has superiority comparing to GCC on dataset REDDIT-BINARY and COLLAB, which are larger than dataset MUTAG. Obviously, GraphCL performs better on datasets with large scales. It is because that GraphCL has a Multi-Layer Perceptron (MLP) as the readout function to aggregate the processed node features, which is more capable to process complex data comparing to average readout function adopted by GCC. Note that, our proposed DSGC also adopted average readout function to have graph embeddings, but DSGC outperforms GraphCL. This phenomenon indicates that the contrastive learning settings in DSGC may be more effective than GraphCL.

With different ratio of labeled data, the differences among the results of GNN models are not significant. It is reasonable for unsupervised learning on the datasets without features. Because, on

\(^1\)https://ls11-www.cs.tu-dortmund.de/staff/morris/graphkerneldatasets
Table 2: Comparison experiment results of classification accuracies and standard error of all the comparing methods (the best results are in bold-face).

| Dataset | Label Ratio | Methods | GCN | GraphSAGE | GAT | GIN | GCC | GraphCL | DSGC |
|---------|-------------|---------|-----|-----------|-----|-----|-----|---------|------|
| MUTAG   | 0.1         | 56.11(std 19.02) | 52.78(std 19.14) | 58.89(std 18.23) | 50.56(std 20.98) | 61.67(std 16.15) | 57.78(std 13.44) | 62.22(std 15.30) |
|         | 0.3         | 62.78(std 15.54) | 57.22(std 18.05) | 62.78(std 14.22) | 54.44(std 18.01) | 63.89(std 14.06) | 62.78(std 12.99) | 66.11(std 12.41) |
|         | 0.5         | 61.10(std 14.60) | 63.33(std 15.57) | 58.89(std 17.60) | 60.56(std 19.88) | 55.56(std 13.57) | 59.44(std 16.76) | 66.67(std 12.37) |
| REDDIT-BINARY | 0.1       | 52.27(std 7.54) | 51.55(std 13.07) | 53.71(std 12.66) | 53.51(std 7.05) | 51.85(std 7.36) | 54.54(std 7.44) | 55.26(std 6.69) |
|         | 0.3         | 56.60(std 6.08) | 55.88(std 11.35) | 54.43(std 7.72) | 54.33(std 10.12) | 53.60(std 10.68) | 58.79(std 8.67) | 57.32(std 5.67) |
|         | 0.5         | 53.67(std 6.98) | 53.61(std 8.33) | 57.63(std 7.99) | 53.40(std 9.00) | 52.87(std 8.81) | 58.14(std 5.73) | 57.73(std 4.35) |
| COLLAB  | 0.1         | 38.98(std 13.75) | 38.56(std 14.20) | 38.74(std 11.77) | 38.48(std 10.88) | 37.68(std 13.38) | 46.72(std 7.78) | 50.00(std 3.79) |
|         | 0.3         | 38.54(std 9.07) | 42.90(std 13.44) | 42.24(std 11.38) | 38.56(std 6.62) | 37.78(std 13.26) | 48.12(std 7.51) | 50.48(std 5.14) |
|         | 0.5         | 35.14(std 10.13) | 36.96(std 12.19) | 42.64(std 9.51) | 40.24(std 6.41) | 38.74(std 6.81) | 46.76(std 7.20) | 52.00(std 1.39) |

The one hand, the features or the embeddings of the test instances are all trained via the contrastive learning part, supervised signals have no direct impact on this training phase. On the other hand, supervised signals impacts embedding training of the test data via updating graph encoders’ parameters. However, the graph encoders we adopt are simple, which have no deep or sophisticated structures. A small portion of labeled data is capable to train the encoders and too much labeled data may force the encoders to be overfitting. But the proposed DSGC method have no such phenomenon, we believe that the dual space contrastive learning protocol here is helpful to address the overfitting problem in the supervised learning phase.

5.4.2 Does the proposed DSGC work with different graph encoders encoding different views of the input graphs? The core idea of graph contrastive learning is how to construct proper contrasting pairs to conduct contrastive learning to obtain informative semantics. In this paper, we propose to utilize the different ability to learn representations of the Euclidean and Hyperbolic space to have the different but related views of the input graph. To fully leverage such different ability, we should select different graph encoders for different spaces. Adopting different graph encoders enriches the semantics among contrasting instances. It is critical to find out how to select graph encoders to fully leverage the advantages of DSGC. In this section, we have an initial exploration about the impact that different pairs of graph encoders have on the proposed DSGC framework. The experiment results are illustrated by Fig. 2, where the x-axis corresponds to the graph encoders for the hyperbolic space and the y-axis corresponds to the graph encoders for the Euclidean space. The block with deep colour represents higher accuracies.

The first insight is that the higher ratio of labeled training data the better overall performances DSGC has. But when ratio is low, it will be tricky to select different pairs of graph encoders for DSGC to achieve better performances. So, choosing different graph encoders to produce contrasting views of input graph to conduct graph contrastive learning is indeed feasible and would be useful when label ratio is lower. In other words, when we face the real-world problems about graph contrastive learning lacking of labels, we could carefully choose the pairs of different encoders to achieve better results.

Moreover, according to the figures, we note that the best performances will not only appear when both encoders are the same but also appear when the encoders are different. It means that some different combinations of graph encoders have the potential to achieve the best results. Therefore, under real-world scenarios, it is possible for us to find out the pairs of encoders with less complexity to improve the efficiency of the models. For example, in the second figure in Fig. 2(c), we can see that GAT-GAT combination achieves one of the best results. However, GIN-GCN combination also has good performances. Note that, GAT are more complex than both GIN and GCN because of its attention mechanism. So, in practice, we should adopt GIN-GCN combination to replace GAT-GAT combination to have higher efficiency.

5.4.3 What is the impact of the hidden dimension of learned features on the whole framework? DSGC not only introduce the advantages of the Euclidean space but also leverages the representation ability of the hyperbolic space. One of the major advantages of the hyperbolic space is that it does not need larger hidden dimensions to acquire representative embeddings [32]. In this section, we explore the performances of DSGC with different hidden dimensions to verify if it leverages such an advantage. To achieve this goal, we conduct experiments on DSGC with different hidden dimensions, we present their performances in each fold on all the datasets in the Fig. 3.

As the figure shows, we note that smaller hidden dimensions, e.g, 8 and 16, do achieve more stable performances, which is that the average of the results of 10 folds and the standard error is continuous on the same dataset with different label ratio. But, when hidden dimension is 8, the performances is unsatisfying. We think it is too small to maintain semantics of graphs even if we have introduced the hyperbolic space. When hidden dimensions are large, e.g., 32 and 64, the results have not shown superiority. Note that, the performances of DSGC with different hidden dimensions are not stable on the dataset COLLAB, and the situations are more severe when hidden dimension is 32 or 64. Therefore, for DSGC, we should adopt relatively small hidden dimensions to have higher efficiency and better performances.

6 CONCLUSION

In this paper, we propose a novel graph contrastive learning framework, dual space graph contrastive learning (DSGC), which first utilizes the different representation abilities of different spaces to generate contrasting views. Moreover, to fully leverage advantages
Figure 2: We explore the possibility to apply different graph encoders for graph representation learning in different spaces. The experiment results show that it is feasible to adopt different graph encoders to acquire the contrasting views of the input graph in different spaces to conduct graph contrastive learning.

of different spaces, we choose different graph sampler to sample sub-graphs and explore the different combinations of graph encoders for different spaces to further reinforce the distinctions among embeddings of the sampled sub-graphs. The feasibility of such strategy is verified by the experiments and we conduct extensive experiments to analyze the properties of DSGC. Our explorations provide a novel way to conduct graph contrastive learning and extend the potential application scenarios of graph contrastive learning.

Figure 3: DSGC’s performances of each fold on all three datasets with different hidden dimensions.

ACKNOWLEDGMENTS

This work is supported by the Australian Research Council (ARC) under grant No. DP220103717, DP200101374, LP170100891, and LE220100078, and NSF under grants III-1763325, III-1909323, III-2106758, and SaTC-1930941.

REFERENCES

[1] Aaron B. Adcock, Blair D. Sullivan, and Michael W. Mahoney. 2013. Tree-Like Structure in Large Social and Information Networks. In 2013 IEEE 13th International Conference on Data Mining, Dallas, TX, USA, December 7–10, 2013, Hui
A HYPER-PARAMETER SETTINGS

For the reproducibility of our work, we list all the hyper-parameter settings of comparison study in Table 3.

Table 3: Hyper-parameter settings of the proposed DSGC in the comparison study on all the datasets with different ratio of labeled data in training set.

| Hyper-parameters               | Dataset & Label Ratio | MUATG  | REDDIT-BINARY | COLLAB |
|--------------------------------|-----------------------|--------|---------------|--------|
|                                |                       | 0.1    | 0.3           | 0.5    | 0.1   | 0.3  | 0.5  | 0.1   | 0.3  | 0.5  |
| Euclidean Encoder              | GraphSAGE             | GCN    | GCN           | GCN    | GraphSAGE | GCN | GCN  | GIN  | GraphSAGE | GCN | GraphSAGE | GIN |
| Hyperbolic Encoder             | GraphSAGE             | GCN    | GCN           | GCN    | GCN       | GIN | GIN  | GIN  | GIN       | GIN |
| Number of Encoder Layers       | 3                     | 3      | 3             | 1      | 1         | 1   | 1    | 1    | 1         | 1   |
| Temperature                    | 1                     | 1      | 1             | 100    | 100       | 100 | 100  | 100  | 100       | 100 |
| Learning Rate                  | 1e-4                  | 1.7e-4 | 5e-5          | 2e-5   | 1e-4      | 1e-5| 1e-5 | 2e-5 | 2e-5      | 2e-5|
| Weight Decay                   | 1e-5                  | 1e-5   | 1e-5          | 1e-5   | 1e-5      | 1e-5| 1e-5 | 1e-5 | 1e-5      | 1e-5|
| Number of Training Epoch       | 200                   | 200    | 200           | 200    | 200       | 200 | 200  | 200  | 200       | 200 |
| Weight of Contrastive Learning (w) | 0.01                 | 0.01   | 0.01          | 1e-5   | 0.01      | 0.01| 0.01 | 0.01 | 0.01      | 0.01|
| Batch Size                     | 8                     | 8      | 8             | 16     | 16        | 16  | 16   | 16   | 16        | 16  |
| Hidden Dimension               | 16                    | 16     | 16            | 16     | 16        | 16  | 16   | 16   | 16        | 16  |