Scalable Deep Metric Learning on Attributed Graphs

Xiang Li[1], Gagan Agrawal[2], Ruoming Jin[3], and Rajiv Ramnath[1]

[1] The Ohio State University, Columbus OH 43210, USA
[2] University of Georgia, Athens GA 30602, USA
[3] Kent State University, Kent OH 44242, USA

Abstract. We consider the problem of constructing embeddings of large attributed graphs and supporting multiple downstream learning tasks. We develop a graph embedding method, which is based on extending deep metric and unbiased contrastive learning techniques to 1) work with attributed graphs, 2) enabling a mini-batch based approach, and 3) achieving scalability. Based on a multi-class tuplet loss function, we present two algorithms – DMT for semi-supervised learning and DMAT-i for the unsupervised case. Analyzing our methods, we provide a generalization bound for the downstream node classification task and for the first time relate tuplet loss to contrastive learning. Through extensive experiments, we show high scalability of representation construction, and in applying the method for three downstream tasks (node clustering, node classification, and link prediction) better consistency over any single existing method.

Keywords: Attributed Graph · Deep Metric Learning · Graph Embedding · Graph Convolutional Network · Scalability

1 Introduction

Last several years have seen much interest in developing learning techniques on *attributed graphs*, i.e., graphs with features associated with nodes. Such graphs are seen in multiple domains such as recommendation systems [28], analysis of citation or social networks [22, 11], and others. Of particular interest are the deep learning based graph embedding methods [21, 29, 31, 6, 32] that encode graph structural information and node features into low-dimensional representations for multiple downstream tasks. Current approaches use Graph Convolutional Networks (GCN) [32] or graph filters [29, 31, 6], but either way, the methods do not scale to large graphs. At a high level, these graph embeddings are designed with the primary objective of pulling examples with distinct labels apart from each other, while pushing the ones sharing the same label closer. It turns out that the spirit of deep metric learning [19, 16] is also almost the same, though to date this idea has been primarily applied to learn visual representations [2, 30, 13]. However, besides the challenges of tailoring these methods for attributed graphs, scalability is also a concern. Specifically, deep metric learning requires: 1) explicit sampling of tuplets such that one or more negative examples is against a single

positive example [16], and 2) expensive search to increase negative hardness of samples, which is needed for enhanced learning power [19, 7, 15].

This paper addresses these problems in applying deep metric learning to attributed graphs in a scalable fashion. First, we employed an extended version of *multi-class tuplet loss* function [20] capable of working with multiple positive samples, building on a similar loss function has been discussed in [10] for image classification. Next, we use (approximate) Generalized PageRank (GPR) [3] as a scalable graph filter, which also leads to a compact node representation and, as we observe, increased negative sample hardness. Finally, we further achieve scalability by mini-batch training; specifically with each batch serving as a natural tuplet comprising multiple positive and negative samples; and eliminate the cost of sampling. With this basic framework, we build multiple algorithms, specifically, **D**eep Metric Learning with **M**ulti-class **T**uplet Loss (**DMT**) for semi-supervised learning and **DMAT-i** for unsupervised conditions.

To summarize the novelty of our contributions – we connect DMAT-i with an extensively applied contrastive loss [4] and theoretically establish how it leads to a bound on the generalization error of a downstream classification task. Equally important, our theoretical analysis explains why contrastive learning is successful for graph representation learning from a deep metric learning perspective. On the experimental side, we compare our methods with the state-of-the-art baselines in semi-supervised node classification, node clustering, and link prediction, and show more consistent level of accuracy as compared to any existing method, and state-of-the-art results in several cases. Finally, we also show greater scalability with our methods.

2 Preliminaries

Deep Metric Learning. We denote $x \in \mathcal{X}$ as the input data, with corresponding labels $y \in \mathcal{Y}$. Let $\mathcal{C}$: $\mathcal{X} \to \mathcal{Y}$ be the function of assigning these labels, i.e., $y = \mathcal{C}(x)$. In deep metric learning, we denote x^+ as a *positive sample* of x (i.e., $\mathcal{C}(x^+) = \mathcal{C}(x)$) and x^- as the *negative sample* (i.e., $\mathcal{C}(x^-) \neq \mathcal{C}(x)$). Define $p_x^+(x')$ to be the probability of observing x' as a positive sample of x and $p_x^-(x')$ the probability its being a negative sample. We assume the class probabilities are uniform such that probability of observing y as a label is τ^+ and probability of observing any different class is $\tau^- = 1 - \tau^+$. Then the data distribution can be decomposed as $p(x') = \tau^+ p_x^+(x') + \tau^- p_x^-(x')$.

Deep metric learning uses a neural network $f : \mathcal{X} \to \mathbb{R}^d$ to learn a d-dimensional nonlinear embedding $f(x)$ for each example x based on objectives such as tuplet loss [20] or triplet loss [19]. [20] proposed a $(\mathcal{N}+1)$-tuplet loss, where for a tuplet $(x, x^+, \{x_i^-\}_{i=1}^{N-1})$ we optimize to identify a single positive example from multiple negative examples as:

$$L_{\text{tuplet}}^{\mathcal{N}+1}(f) = \log\left(1 + \sum_{i=1}^{\mathcal{N}-1} exp\{f(x)^\top f(x_i^-) - f(x)^\top f(x^+)\}\right) \quad (1)$$

This softmax function based objective is *hardness known* where the hard negative samples receive larger gradients [8].

Contrastive Learning. In fact, $L_{\text{tuplet}}^{\mathcal{N}+1}$ is mathematically equal to the ideal *unbiased contrastive loss* $\widetilde{L}_{\text{Unbiased}}^{\mathcal{N}+1}(f)$ proposed in [5], where they introduced:

$$\widetilde{L}_{\text{Unbiased}}^{\mathcal{N}+1}(f) = -\log \frac{exp\{f(x)^\top f(x^+)\}}{exp\{f(x)^\top f(x^+)\}+(\mathcal{N}-1)\mathbb{E}_{x^-\sim p_x^-} exp\{f(x)^\top f(x^-)\}} \tag{2}$$

In contrastive learning, the positive sample (and negative samples) are obtained through perturbation and mainly used in the unsupervised setting (where class label is not available). Thus, p_x^- is usually not accessible and negative samples x_i^- are generated from the (unlabeled) $p(x)$ [5]. Thus, the typical contrastive loss [4] now becomes:

$$\widetilde{L}_{\text{Contrast}}^{\mathcal{N}+1}(f) = -\log \frac{exp\{f(x)^\top f(x^+)\}}{exp\{f(x)^\top f(x^+)\}+(\mathcal{N}-1)\mathbb{E}_{x^-\sim p} exp\{f(x)^\top f(x^-)\}} \tag{3}$$

Since x_i^- is drawn from $p(x)$, it also has a probability of τ^+ of being a positive sample. Thus, the contrastive learning is closely related to, and can even be considered a variant of, deep metric learning, where the positive/negative samples are generated through different perturbation mechanisms. To facilitate our discussion, we use the notations $L_{\text{tuplet}}^{\mathcal{N}+1}$ and $\widetilde{L}_{\text{Unbiased}}^{\mathcal{N}+1}$ interchangeably in the rest of the paper. More related works are reviewed in appendix.

3 Methodology

3.1 Problem Statement

We are given an attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \widetilde{X})$, where $\mathcal{V} = \{v_1, v_2, \cdots, v_N\}$ and $\mathcal{E}$ represent node set and edge set, respectively, and $\widetilde{X}$ denotes the node attributes (i.e., each node is associated with a feature vector). Each vertex v_i belongs to a single class (or a cluster) and we apply all notations defined in deep metric learning to graph representations. The input data for deep metric learning $\mathcal{X}$ is calculated by a graph filter $\mathcal{H}$: $\mathcal{X} = \mathcal{H}(\widetilde{X}, A)$, where A is the adjacency matrix. Our objective is to learn an encoder $f : \mathcal{X} \to \mathbb{R}^d$ to obtain a d-dimensional embedding $f(\mathcal{X})$.

To develop deep metric learning (or contrastive learning) on graphs, we need to consider and address the following problems: (1) How to establish a unified approach to cover both semi-supervised and unsupervised settings for graphs? (2) How to scale the learning process for large-scale graphs by taking advantage of mini-batch training?

To elaborate on the second point, the existing contrastive learning for graph representation, particularly GCA [32], is built upon a GCN architecture and uses a typical contrastive loss [4]. It perturbs the graph topology and node attributes separately, which are fed to GCN to generate augmented views for contrasting. The transformation by GCN limits both accuracy (due to over-smoothing [12]) and scalability.

3.2 DMT Algorithm

We first propose the learning framework, **D**eep Metric Learning with **M**ulti-class **T**uplet (**DMT**), for semi-supervised node classification task. By applying a multiclass tuplet loss [20, 10] which can recognize multiple positive samples from

the tuplet, DMT addresses the aforementioned batch and scalability problem with the following distinguishing advantages: 1) high scalability and efficiency is achieved by using each shuffled node batch as a natural tuplet – this choice also alleviates the need for explicit (and expensive) sampling; 2) enhanced and faster representations construction through graph filtering, which we show later increases *negative sample hardness*.

Specifically, **DMT** employs a GPR-based graph smoothing filter $\mathcal{H}$ – as described earlier, the goal is to smooth node attributes $\widetilde{X}$ by graph structure via $\mathcal{X} = \mathcal{H}(\widetilde{X}, A)$ such that each $x \in \mathcal{X}$ contains information from its neighborhood as well. The details of this filtering, and how it can be done on large graphs, is presented in the appendix. This approach can also help increase negative sample hardness, a property that has been shown to accelerate training and enhance the discriminative power [19, 7, 15] - details again are captured in the appendix.

DMT employs an extended version of the multi-class tuplet loss from the deep metric learning [20]. Training is conducted in mini-batches and we consider each train batch $\mathcal{X}_B$ of size $\mathcal{B}$ as a $\mathcal{B}$-tuplet $(x, \{x_i^+\}_{i=1}^m, \{x_i^-\}_{i=1}^q)$ with m positive samples x^+ and q negative samples x^- of x respectively (m and q are batch dependent). Furthermore, we define $h(x, x'; f) = exp\{\frac{f(x)^T \cdot f(x')}{t}\}$, where we apply the cosine similarity as a metric distance such that each feature vector $f(x)$ is normalized before performing the Cartesian product. Temperature t is the radius of hypersphere where the representations lie [25] and can control penalty degree on hard negative samples as inspired by [24].

Now, the *multi-class tuplet loss* function is:

$$L_{\mathrm{DMT}}^{\mathrm{m,q}}(x; f) = -\log \frac{h(x,x;f)+\sum_{i=1}^{m} h(x,x_i^+;f)}{h(x,x;f)+\sum_{i=1}^{m} h(x,x_i^+;f)+\sum_{i=1}^{q} h(x,x_i^-;f)} \tag{4}$$

Here, x is counted as one positive sample of itself to avoid zero-value inside the *log* function. The loss function above shares a close mathematical form of supervised contrastive loss as proposed in [10] and enables us to create efficient mini-batch versions, while preserving the essential ideas behind metric or contrastive learning. One important aspect is because the function can work with varying m and q across batches, we can simply use all the positive and negative samples associated with any given batch.

3.3 DMAT-i Algorithm

In the unsupervised cases, $\{x_i^+\}$ and $\{x_i^-\}$ are no longer recognizable. To deal with this problem, we adopt the idea of contrastive learning, which includes multiple views of graph embeddings through *augmentation*, while assuming that the labeling still exists initially (thus, drawing from the deep metric learning framework). Then, we will show we can drop out the labels of the loss, which leads to the format of the contrastive learning loss.

Specifically, for one batch of samples $\mathcal{X}_B$ of size $\mathcal{B}$ together with their augmented counterparts, we have a $2\mathcal{B}$-tuplet $(x, \bar{x}, \{x_i^+\}_{i=1}^m, \{x_i^-\}_{i=1}^q)$ with m positive pairs and q negative pairs – here, $\bar{x}$ denotes the augmented counterpart (trivial positive sample) of x. Thus, we introduce an immediate DMAT tuplet loss $L_{\mathrm{DMAT}}^{\mathrm{m,q}}(x, \bar{x}; f)$ following the similar form of Eq.4:

$$L^{\mathrm{m,q}}_{\mathrm{DMAT}}(x,\bar{x};f) = -\log \frac{h(x,\bar{x};f)+\sum_{i=1}^{m} h(x,x_i^+;f)}{h(x,\bar{x};f)+\sum_{i=1}^{m} h(x,x_i^+;f)+\sum_{i=1}^{q} h(x,x_i^-;f)} \tag{5}$$

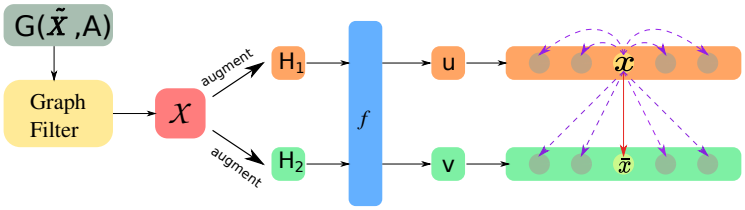

Fig. 1: Schematic of DMAT-i architecture. The graph filter generates smoothed node attributes $\mathcal{X}$ by incorporating graph structural information. A pair of views (H_1, H_2) of $\mathcal{X}$ are produced by augmentation and fed to the subsequent encoder f to generate latent representations $U = f(H_1)$ and $V = f(H_2)$. Metric distance measurement is performed on $U \bigcup V$. For each sample $x \in U$, its counterpart $\bar{x} \in V$ is the only recognizable positive sample.

Next, we extend DMAT to unsupervised cases where $\{x_i^+\}$ and $\{x_i^-\}$ are no longer recognizable. Here, the resulting method, DMAT-i, involves further simplification by extracting $\bar{x}$ as the only positive sample of x while ignoring all other positive ones. The loss function is (mathematically equal to Eq.3):

$$L^{\mathrm{m,q}}_{\text{DMAT-i}}(x,\bar{x};f) = -\log \frac{h(x,\bar{x};f)}{h(x,\bar{x};f)+\sum_{i=1}^{m} h(x,x_i^+)+\sum_{i=1}^{q} h(x,x_i^-)} \tag{6}$$

Note $\{x_i^+\}_{i=1}^{m}$ and $\{x_i^-\}_{i=1}^{q}$ are explicitly denoted for ease of analysis, but they remain unknown during the training. Eq. 6 is in fact calculated without knowing any labels as:

$$L^{\mathrm{m,q}}_{\text{DMAT-i}}(x,\bar{x};f) = \log\left\{\sum_{\substack{x'\in\mathcal{X}_B \\ x'\neq x}} h(x,x';f)/h(x,\bar{x};f)\right\}$$

Complete Algorithm: The general idea is illustrated in Figure 1. Augmented views are generated on the fly from $\mathcal{X}$ by masking certain columns – the consequence is that the node features and structural information (encoded inside $\mathcal{X}$) are "distorted" simultaneously. A subsequent DNN based module can abstract information and perform metric similarity measurements (as in Eq. 6) between each pair of views. In real implementation, we use $\mathcal{X}$ as the anchor view and each augmented view as the counterpart to calculate an average of training loss. Thus, the encoder will be optimized to learn robust characteristics of representations across different views. The overall objective to be maximized is defined as the average agreement $L_{\text{DMAT-i}}(x,\bar{x};f)$ over all interchangeable view pairs as follows:

$$\mathbf{J} = \frac{1}{2\mathcal{B}} \sum_{x\in\mathcal{X}_B} [L^{\mathrm{m,q}}_{\text{DMAT-i}}(x,\bar{x};f) + L^{\mathrm{m,q}}_{\text{DMAT-i}}(\bar{x},x;f)] \tag{7}$$

The entire training process is presented in Algorithm 1. As input, $\mathcal{X}$ is generated using random-walk based GnnBP (Graph neural network via Bidirectional Propagation [3]) as graph filtering. In line 3, multiple (n_{view}) augmented embedding will be generated from one batch of filtered feature $\mathcal{X}_B$ by masking certain columns in $\mathcal{X}_B$. In line 5, the generated graph embedding views will be input into the DNN based encoder f to produce the latent representations. The deep metric learning in line 6 is performed in batches between encoded representations u of the anchor view $\mathcal{X}_B$ and v of each augmented view H_B. The obtained embedding Z in line 8 will be used for the downstream learning tasks.

Algorithm 1 DMAT-i Training

Input data: GnnBP filtered attributes $\mathcal{X}$, Graph G, number of views: n_{view}

1: **for** *epoch* $= 1, 2, \cdots$ **do**
2: **for** $\mathcal{X}_B$ in $\mathcal{X}$ **do**
3: Generate n_{view} augmented views of $\mathcal{X}_B$: $\{H_B\}$
4: **for** $i = 1, 2, \cdots$ **do**
5: $u \leftarrow f(\mathcal{X}_B)$; $v \leftarrow f(H_B^i)$
6: Compute multi-class tuplet loss **J** (Eq.7)
7: **end for**
8: SGD update on f to minimize **J**
9: **end for**
10: **end for**
11: $Z \leftarrow f(\mathcal{X})$

4 Theoretical Analysis

DM(A)T and Contrastive Learning $\widetilde{L}^{\mathcal{N}+1}_{\text{Unbiased}}(f)$ (Eq. 2) contrasts one positive sample against multiple negative samples and has been recognized as the ideal loss to optimize [5]. $L^{\text{m,q}}_{\text{DM(A)T}}(f)$ improves $\widetilde{L}^{\mathcal{N}+1}_{\text{Unbiased}}(f)$ by recognizing multiple positive samples at the same time. It turns out that it can be shown as a lower bound of $L^{\mathcal{N}+1}_{\text{Unbiased}}(f)$, specifically:

Lemma 1 *For any embedding f, given the same size of tuplets sharing one positive sample x_0^+, i.e. $(x, x_0^+, \{x_i^-\}_{i=1}^{N-1})$ for $L^{\mathcal{N}+1}_{\text{Unbiased}}$ and $(x, x_0^+, \{x_i^+\}_{i=1}^{m}, \{x_i^-\}_{i=1}^{q})$ for $L^{\text{m,q}}_{\text{DM(A)T}}$, we have: $L^{\text{m,q}}_{\text{DM(A)T}}(f) \leq \widetilde{L}^{\mathcal{N}+1}_{\text{Unbiased}}(f)$*

Now, as we know, both $\widetilde{L}^{\mathcal{N}+1}_{\text{Unbiased}}(f)$ and $L^{\text{m,q}}_{\text{DM(A)T}}(f)$ require p_x^+ and p_x^-, which can only be accessed from training data (i.e., during supervised learning). For unsupervised conditions, our $L^{\text{m,q}}_{\text{DMAT-i}}$ considers $\bar{x}$ as the only available positive sample. Next, we will show how $L^{\text{m,q}}_{\text{DMAT-i}}$ contributes to a downstream learning task.

DMAT-i Generalization Bound on Node Classification We relate $L^{\mathrm{m,q}}_{\text{DMAT-i}}$ to a supervised loss and present how $L^{\mathrm{m,q}}_{\text{DMAT-i}}$ leads to a generalization bound for a supervised node classification task. Consider a supervised node classification task with K classes, we fix the embedding $f(\mathcal{X})$ from DMAT-i representation learning and train a linear classifier $\psi(\mathcal{X}) = f(\mathcal{X})W^\top$ with the standard multi-class softmax cross entropy loss $L_{\text{Softmax}}(\psi)$. We define the supervised loss for the representation $f(\mathcal{X})$ as: $L_{\text{Sup}}(f) = \inf_{W\in\mathbb{R}^{K\times d}} L_{\text{Softmax}}(fW^\top)$

[5] has proved $\widetilde{L}^{\mathcal{N}+1}_{\text{Unbiased}}(f)$ as an upper bound of $L_{\text{Sup}}(f)$. What we contribute here is to bound the difference between $\widetilde{L}^{\mathcal{N}+1}_{\text{Unbiased}}(f)$ and $L^{\mathrm{m,q}}_{\text{DMAT-i}}$.

Theorem 1 *For any embedding f and same size of tuplets,*

$$\left|\widetilde{L}^{\mathcal{N}+1}_{\text{Unbiased}}(f) - L^{\mathrm{m,q}}_{\text{DMAT-i}}(f)\right| \leq \sqrt{\frac{2(e^3-e)(\tau^0)^2\pi}{m}} + \sqrt{\frac{2(e^3-e)(\tau^-)^2\pi}{q}}$$

$$\tau^0 = \tau^+\left(\frac{\left|\frac{1}{m}\sum_{i=1}^m h(x,x_i^+) - \mathbb{E}_{x^-\sim p_x^-} h(x,x^-)\right|}{\left|\frac{1}{m}\sum_{i=1}^m h(x,x_i^+) - \mathbb{E}_{x^+\sim p_x^+} h(x,x^+)\right|}\right) \tag{8}$$

where $\sum_{i=1}^m h(x,x_i^+)$ represents the positive samples unrecognized by $L^{\mathrm{m,q}}_{\text{DMAT-i}}$, i.e., *false negative samples*. Hence τ^0 covers the side effects from these false negatives and an empirical evaluation in appendix has shown reasonable small values of τ^0 for most samples across our experimental datasets.

In practice, we use an empirical estimate $\widehat{L}^{\mathrm{m,q}}_{\text{DMAT-i}}(f)$ over N data samples $x \in \mathcal{X}$, each sample with a tuplet $(x, \bar{x}, \{x_i^+\}_{i=1}^m, \{x_i^-\}_{i=1}^q)$. The optimization process learns an empirical risk minimizer $\widehat{f} \in \arg\min_{f\in\mathbb{F}} L^{\mathrm{m,q}}_{\text{DMAT-i}}(f)$ from a function class $\mathbb{F}$. The generalization depends on the *empirical Rademacher complexity* $\mathcal{R}_{\mathcal{S}}(\mathbb{F})$ of $\mathbb{F}$ with respect to our data sample $\mathcal{S} = \{x_j, \bar{x}_j, \{x_{i,j}^+\}_{i=1}^m, \{x_{i,j}^-\}_{i=1}^q\}_{j=1}^N$. Let $f_{|\mathcal{S}} = (f_k(x_j), f_k(\bar{x}_j), \{f_k(x_{i,j}^+)\}_{i=1}^m, \{f_k(x_{i,j}^-)\}_{i=1}^q)_{j\in[N],k\in[d]} \in \mathbb{R}^{(m+q+2)dN}$ be the restriction of f onto $\mathcal{S}$, using $[N] = \{1,\ldots,N\}$ and $[d] = \{1,\ldots,d\}$. Then $\mathcal{R}_{\mathcal{S}}(\mathbb{F})$ is defined as: $\mathcal{R}_{\mathcal{S}}(\mathbb{F}) := \mathbb{E}_\sigma \sup_{f\in\mathbb{F}} \langle\sigma, f_{|\mathcal{S}}\rangle$ where $\sigma \sim \{\pm 1\}^{(m+q+1)dN}$ are *Rademacher random variables*. We provide a data dependent bound from $L^{\mathrm{m,q}}_{\text{DMAT-i}}(f)$ on the downstream supervised generalization error as follows.

Theorem 2 *With probability at least $1-\delta$, for all $f \in \mathbb{F}$ and $q \geq K-1$,*

$$L_{\text{Sup}}(\hat{f}) \leq L^{\mathrm{m,q}}_{\text{DMAT-i}}(f) + \mathcal{O}\left(\tau^0\sqrt{\tfrac{1}{m}} + \tau^-\sqrt{\tfrac{1}{q}} + \frac{\lambda\mathcal{R}_{\mathcal{S}}(\mathbb{F})}{N} + \Gamma\sqrt{\frac{\log\frac{1}{\delta}}{N}}\right)$$

where $\lambda = \frac{(m+q)e}{m+q+e}$ and $\Gamma = \log(m+q)$.

The bound states that if the function class $\mathbb{F}$ is sufficiently rich to contain embeddings for which $L^{\mathrm{m,q}}_{\text{DMAT-i}}$ is small, then the representation encoder $\hat{f}$, learned from a large enough dataset, will perform well on the downstream classification task. The bound highlights the effects caused by the false negative pairs with the first term and also highlights the role of the inherent positive and negative sample sizes m and q per mini-batch in the objective function. The last term in the bound grows slowly with $m+q = 2\mathcal{B}-2$, but the effect of this on the generalization error is small if the dataset size N is much larger than the batch size $\mathcal{B}$, as is common.

5 Experimental Results

Baselines: For the node clustering task, we compared the proposed DMAT-i model with multiple frameworks: 1) **KMeans** [9] (when applied to attributed graphs uses node attributes only); 2) **DeepWalk** [18], which uses topological information only, and seven recent frameworks that leverage both node attributes and graph structure: 3) **AGC (2019)**[29] that uses high-order graph convolution; 4) **DGI (2019)** [23] maximizes mutual information between patch representations and high-level summaries of graph; 5) **SDCN (2020)** [1] unifies an autoencoder module with a GCN module; 6) **AGE (2020)** [6] applies a customized Laplacian smoothing filter; 7) **SSGC (2021)** [31] is a variant of GCN that exploits a modified Markov Diffusion Kernel. 8) **GCA (2021)** [32] leverages a node-level contrastive loss between two augmented graph views to learn a graph representation; 9) **ProGCL (2022)** [26], on top of GCA, further proposed a more suitable measure for negatives hardness and similarity. To compare performance on node classification and link prediction, we select the most competitive graph embedding based frameworks correspondingly.

Scalability of Representation Construction As in Figure 2, all baselines hit specific ceilings as limited by the GPU memory capacity while DMAT-i can continuously scale with application of mini-batch training and use of random-walk to obtain approximate pagerank scores. Particularly, DMAT-i could handle 10^7 nodes, and no other frameworks could handle more than 10^6 nodes. The details of experimental settings are in appendix.

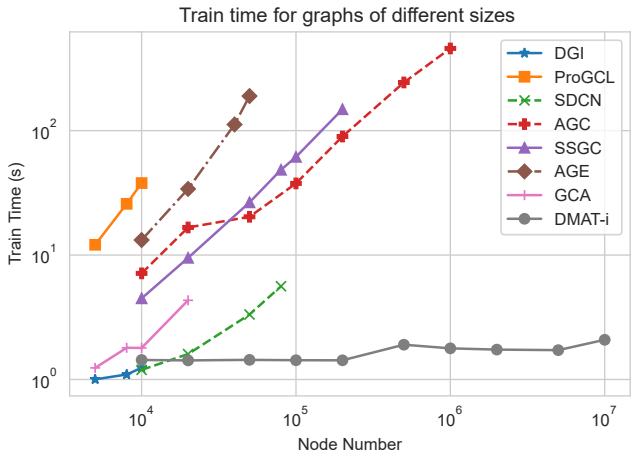

Fig. 2: Scalability of Different Frameworks: Training Time vs. No. of Nodes in Graph

5.1 Results on Downstream Tasks

DM(A)T is evaluated on performance of semi-supervised node classification while DMAT-i is evaluated for multiple tasks: node clustering, node classifi-

Table 1: Clustering performance on eight datasets (mean±std) where each experiment is performed for 10 runs. We employ six popular metrics: accuracy, Normalized Mutual Information (NMI), Average Rand Index (ARI), and macro F1-score are four metrics for ground-truth label analysis, whereas modularity [14] and conductance [27] are graph-level metrics. All metrics except conductance will indicate a better clustering output with a larger value. DMAT-i results highlighted in bold if they have the top 2 clustering performance. The asterisk indicates a convergence issue. Certain data points are missing when execution ran out of GPU memory. DGI can only handle five smaller datasets due to high GPU memory cost, and GCA also could not handle largest of these 8 datasets.

Dataset	Metric	KMeans	DeepWalk	SDCN	AGC	SSGC	AGE	DGI	GCA	ProGCL	**DMAT-i**
ACM	Accuracy ↑	66.62 ± 0.55	50.59 ± 4.27	89.63 ± 0.31	78.21 ± 0.00	84.43 ± 0.29	90.18 ± 0.13	90.17 ± 0.28	89.91 ± 0.46	89.18 ± 1.70	**91.60 ± 0.70**
	NMI ↑	32.41 ± 0.34	16.12 ± 4.96	66.74 ± 0.75	46.31 ± 0.01	56.15 ± 0.51	66.92 ± 0.30	67.84 ± 0.72	66.58 ± 0.91	64.64 ± 3.04	**70.95 ± 1.44**
	ARI ↑	30.22 ± 0.41	18.56 ± 5.80	72.00 ± 0.75	48.02 ± 0.00	60.17 ± 0.60	73.12 ± 0.31	73.28 ± 0.66	72.49 ± 1.08	70.72 ± 3.88	**76.72 ± 1.75**
	macro F1 ↑	66.83 ± 0.57	46.56 ± 4.43	89.60 ± 0.32	78.26 ± 0.00	84.44 ± 0.29	90.18 ± 0.13	90.12 ± 0.27	89.89 ± 0.46	89.16 ± 1.71	**91.59 ± 0.70**
	Modularity ↑	31.20 ± 0.50	38.57 ± 9.51	60.86 ± 0.16	59.44 ± 0.02	60.19 ± 0.05	60.93 ± 0.08	59.79 ± 0.19	60.05 ± 0.12	60.14 ± 0.43	57.92 ± 0.20
	Conductance ↓	30.96 ± 0.23	1.79 ± 0.59	3.07 ± 0.17	2.51 ± 0.01	2.54 ± 0.11	3.64 ± 0.13	3.87 ± 0.14	3.85 ± 0.18	4.06 ± 0.15	6.68 ± 0.27
DBLP	Accuracy ↑	38.65 ± 0.58	38.99 ± 0.02	69.08 ± 1.95	69.06 ± 0.06	68.66 ± 1.95	*62.49 ± 0.76	59.72 ± 4.68	77.69 ± 0.39	73.79 ± 1.70	**80.30 ± 0.60**
	NMI ↑	11.56 ± 0.53	5.91 ± 0.02	34.64 ± 1.94	37.00 ± 0.07	33.89 ± 2.08	*37.32 ± 0.50	26.90 ± 4.43	46.24 ± 0.57	41.54 ± 1.27	**51.00 ± 0.81**
	ARI ↑	6.95 ± 0.39	5.83 ± 0.02	36.31 ± 2.86	33.69 ± 0.13	37.30 ± 3.13	*34.60 ± 0.71	25.12 ± 4.76	50.46 ± 0.81	43.30 ± 2.99	**55.42 ± 1.08**
	macro F1 ↑	31.81 ± 0.53	36.87 ± 0.02	67.81 ± 3.46	68.59 ± 0.05	65.91 ± 2.19	*59.16 ± 0.83	59.31 ± 4.69	77.29 ± 0.37	72.96 ± 2.03	**79.94 ± 0.59**
	Modularity ↑	33.83 ± 0.47	64.05 ± 0.03	63.38 ± 1.87	68.77 ± 0.01	62.02 ± 1.64	*48.62 ± 0.87	50.16 ± 3.77	63.01 ± 0.28	64.62 ± 1.02	55.67 ± 0.71
	Conductance ↓	36.20 ± 0.51	4.03 ± 0.02	7.56 ± 0.54	5.29 ± 0.01	3.24 ± 0.52	*11.15 ± 0.15	13.84 ± 1.12	9.51 ± 0.16	9.53 ± 0.29	16.52 ± 0.53
Cora	Accuracy ↑	35.37 ± 3.72	63.87 ± 2.14	64.27 ± 4.87	65.23 ± 0.93	68.50 ± 1.98	74.34 ± 0.42	68.47 ± 1.43	69.24 ± 2.92	68.17 ± 4.67	**70.57 ± 1.28**
	NMI ↑	16.64 ± 4.21	44.11 ± 1.33	47.39 ± 3.49	50.05 ± 0.49	52.80 ± 1.03	58.11 ± 0.58	52.60 ± 0.88	54.48 ± 1.94	54.37 ± 2.71	53.59 ± 1.22
	ARI ↑	9.31 ± 2.14	39.64 ± 1.68	39.72 ± 5.53	40.23 ± 0.95	45.70 ± 1.28	50.87 ± 0.96	45.63 ± 1.44	46.63 ± 3.25	45.37 ± 5.04	**47.34 ± 2.41**
	macro F1 ↑	31.49 ± 4.58	57.98 ± 2.43	57.88 ± 6.99	58.93 ± 1.68	64.38 ± 2.71	70.37 ± 0.29	65.79 ± 1.53	68.10 ± 2.68	67.12 ± 4.85	**69.33 ± 1.00**
	Modularity ↑	20.77 ± 3.37	72.98 ± 0.79	62.59 ± 5.18	69.98 ± 0.46	73.71 ± 0.45	71.89 ± 0.14	69.86 ± 0.29	74.18 ± 0.51	74.36 ± 0.38	**74.19 ± 0.39**
	Conductance ↓	59.77 ± 5.31	7.88 ± 0.35	18.32 ± 2.26	11.08 ± 1.61	9.41 ± 0.55	8.23 ± 0.11	13.64 ± 0.69	10.27 ± 0.31	9.47 ± 0.54	10.04 ± 0.52
Citeseer	Accuracy ↑	46.70 ± 4.33	43.56 ± 1.03	63.42 ± 3.31	67.18 ± 0.52	67.86 ± 0.26	66.06 ± 0.78	68.68 ± 0.76	66.23 ± 1.00	66.43 ± 1.16	67.46 ± 0.41
	NMI ↑	18.42 ± 3.26	16.02 ± 0.56	37.28 ± 2.19	41.37 ± 0.70	41.86 ± 0.22	40.56 ± 0.88	43.22 ± 0.91	40.81 ± 1.15	41.41 ± 1.03	41.75 ± 0.62
	ARI ↑	18.42 ± 3.26	16.37 ± 0.66	37.40 ± 2.79	42.10 ± 0.87	42.95 ± 0.30	39.84 ± 0.75	44.53 ± 1.02	41.24 ± 1.45	41.73 ± 1.52	42.48 ± 0.60
	macro F1 ↑	44.47 ± 4.44	40.37 ± 0.97	56.16 ± 4.53	62.68 ± 0.48	63.61 ± 0.23	60.80 ± 0.75	64.41 ± 0.70	62.16 ± 0.95	62.53 ± 1.11	62.83 ± 0.38
	Modularity ↑	43.57 ± 2.67	76.44 ± 0.20	70.83 ± 2.77	77.57 ± 0.21	78.03 ± 0.12	71.88 ± 0.45	72.42 ± 0.38	73.14 ± 0.36	74.54 ± 0.44	75.78 ± 0.23
	Conductance ↓	37.21 ± 2.19	2.98 ± 0.12	7.98 ± 1.99	1.72 ± 0.04	1.75 ± 0.03	4.84 ± 0.13	7.19 ± 0.55	6.96 ± 0.58	5.57 ± 0.23	3.02 ± 0.22
Pubmed	Accuracy ↑	59.50 ± 0.02	69.98 ± 0.04	59.95 ± 1.00	61.54 ± 0.00	70.71 ± 0.00	69.66 ± 0.09	-	64.10 ± 2.11	-	**70.90 ± 0.20**
	NMI ↑	31.21 ± 0.10	29.09 ± 0.11	17.78 ± 0.91	29.11 ± 0.00	32.12 ± 0.00	29.06 ± 0.16	-	28.50 ± 2.41	-	**32.49 ± 0.28**
	ARI ↑	28.08 ± 0.08	31.81 ± 0.13	16.39 ± 1.16	26.16 ± 0.00	33.26 ± 0.00	31.26 ± 0.12	-	26.15 ± 2.46	-	**33.52 ± 0.36**
	macro F1 ↑	58.15 ± 0.02	68.51 ± 0.06	60.29 ± 1.02	60.28 ± 0.00	69.91 ± 0.00	68.68 ± 0.08	-	63.69 ± 2.34	-	**70.10 ± 0.20**
	Modularity ↑	34.92 ± 0.06	57.25 ± 0.26	55.53 ± 0.86	50.40 ± 0.00	57.73 ± 1.35	57.48 ± 0.10	-	53.90 ± 1.76	-	57.56 ± 0.44
	Conductance ↓	17.27 ± 0.04	4.67 ± 0.03	7.50 ± 0.58	8.65 ± 0.00	3.93 ± 0.00	4.75 ± 0.16	-	9.51 ± 0.80	-	**4.10 ± 0.20**
Amazon Photo	Accuracy ↑	27.86 ± 0.81	77.27 ± 2.48	60.42 ± 3.36	55.93 ± 0.09	56.16 ± 1.05	66.96 ± 3.00	61.05 ± 2.48	77.21 ± 0.72	78.27 ± 0.97	76.53 ± 1.32
	NMI ↑	13.78 ± 1.19	68.97 ± 1.96	50.08 ± 3.28	53.35 ± 0.05	51.74 ± 1.66	56.73 ± 2.62	52.93 ± 2.11	66.48 ± 1.23	70.11 ± 1.32	66.94 ± 1.37
	ARI ↑	5.62 ± 0.42	58.64 ± 2.81	40.08 ± 3.95	25.31 ± 0.10	33.86 ± 1.36	46.48 ± 3.37	39.59 ± 2.74	56.09 ± 0.92	61.30 ± 1.71	58.84 ± 1.07
	macro F1 ↑	23.78 ± 0.48	71.59 ± 2.47	53.13 ± 5.99	51.56 ± 0.06	52.00 ± 0.67	62.13 ± 3.33	59.60 ± 2.94	76.23 ± 0.71	72.35 ± 1.49	70.05 ± 0.77
	Modularity ↑	8.38 ± 0.51	73.18 ± 0.12	59.25 ± 4.04	57.69 ± 0.04	62.07 ± 1.88	64.07 ± 1.45	61.12 ± 1.39	67.76 ± 0.83	70.81 ± 0.47	70.72 ± 0.19
	Conductance ↓	76.38 ± 0.58	8.47 ± 0.23	20.17 ± 3.88	4.42 ± 0.00	8.37 ± 2.15	15.81 ± 1.49	22.14 ± 1.65	15.27 ± 1.11	10.46 ± 0.72	10.98 ± 0.67
Coauthor CS	Accuracy ↑	27.96 ± 1.09	67.10 ± 2.98	56.86 ± 3.40	62.24 ± 1.81	66.19 ± 1.19	76.35 ± 3.14	-	72.02 ± 2.54	-	**76.92 ± 1.26**
	NMI ↑	15.42 ± 2.25	66.67 ± 0.86	54.79 ± 2.44	65.22 ± 0.44	70.06 ± 0.67	76.75 ± 1.66	-	73.95 ± 1.02	-	72.55 ± 0.41
	ARI ↑	1.02 ± 0.74	53.66 ± 2.91	40.41 ± 4.52	46.96 ± 3.54	58.50 ± 0.17	71.27 ± 5.46	-	63.92 ± 3.21	-	66.91 ± 1.38
	macro F1 ↑	11.68 ± 1.56	63.36 ± 2.84	29.36 ± 3.22	51.42 ± 1.27	60.17 ± 1.94	71.10 ± 1.96	-	63.63 ± 3.28	-	**70.48 ± 3.13**
	Modularity ↑	9.61 ± 1.88	72.88 ± 0.41	53.05 ± 2.02	69.58 ± 0.14	71.82 ± 0.14	70.45 ± 1.71	-	69.91 ± 0.58	-	69.79 ± 0.40
	Conductance ↓	37.12 ± 4.10	17.09 ± 0.66	*23.09 ± 1.89	19.80 ± 0.24	19.76 ± 0.22	14.41 ± 0.32	-	21.96 ± 0.60	-	21.13 ± 0.66
Coauthor PHY	Accuracy ↑	56.19 ± 0.75	87.97 ± 0.01	64.65 ± 6.92	77.41 ± 0.00	55.70 ± 2.26	92.04 ± 0.06	-	-	-	**89.30 ± 0.70**
	NMI ↑	11.72 ± 1.92	69.13 ± 0.02	50.60 ± 3.71	62.11 ± 0.02	57.71 ± 1.31	75.84 ± 0.13	-	-	-	**72.54 ± 0.80**
	ARI ↑	8.25 ± 1.26	79.15 ± 0.03	48.76 ± 9.58	72.43 ± 0.02	44.91 ± 1.58	84.44 ± 0.16	-	-	-	**77.68 ± 1.61**
	macro F1 ↑	24.74 ± 2.11	83.32 ± 0.02	48.51 ± 4.68	62.09 ± 0.00	55.26 ± 2.30	88.90 ± 0.08	-	-	-	**86.65 ± 0.91**
	Modularity ↑	5.74 ± 0.83	47.96 ± 0.00	44.97 ± 3.16	45.31 ± 0.00	60.70 ± 0.39	47.69 ± 0.07	-	-	-	50.56 ± 0.27
	Conductance ↓	10.56 ± 1.47	5.99 ± 0.00	19.86 ± 7.16	5.80 ± 0.00	13.47 ± 0.07	5.73 ± 0.03	-	-	-	7.31 ± 0.31

Table 2: Accuracy for semi-supervised node classification task with different data usage for embedding generation: 1) using 10% of data with labels; 2) using all data without labels.

Data Usage	Method	Cora	Citeseer	Pubmed	ACM	DBLP	Amazon Photo	Coauthor CS	Coauthor PHY
train data (labeled)	DMT	84.30 ± 0.25	70.42 ± 0.33	86.46 ± 0.16	91.42 ± 0.36	77.59 ± 0.30	92.60 ± 0.42	93.30 ± 0.12	95.44 ± 0.03
	DMAT	83.92 ± 0.45	71.39 ± 0.38	86.19 ± 0.10	92.04 ± 0.16	79.80 ± 0.60	93.42 ± 0.11	93.44 ± 0.15	95.20 ± 0.04
	DMAT-i	81.99 ± 0.54	70.91 ± 0.27	83.52 ± 0.21	91.32 ± 0.38	78.39 ± 0.67	93.19 ± 0.19	92.90 ± 0.12	94.86 ± 0.07
all data (unlabeled)	DMAT-i	83.65 ± 0.71	72.40 ± 0.43	83.91 ± 0.25	92.55 ± 0.40	80.92 ± 0.50	92.97 ± 0.16	91.28 ± 0.17	94.66 ± 0.08
	SSGC	83.48 ± 0.06	68.15 ± 0.02	84.59 ± 0.01	89.71 ± 0.25	77.14 ± 0.12	89.80 ± 0.14	91.37 ± 0.03	94.88 ± 0.02
	GCA	83.89 ± 0.56	73.36 ± 0.34	83.38 ± 0.17	90.01 ± 0.27	79.73 ± 0.50	90.30 ± 0.47	90.91 ± 0.11	-
	ProGCL	85.04 ± 0.42	71.42 ± 0.39	-	88.98 ± 0.48	79.55 ± 0.41	92.13 ± 0.82	-	-
	AGE	83.78 ± 0.22	72.13 ± 0.92	80.18 ± 0.24	92.10 ± 0.18	80.02 ± 0.40	73.16 ± 2.53	91.40 ± 0.13	94.21 ± 0.08

cation, and link prediction. Our framework is compared with existing state-of-the-art appraoches on 8 real-world datasets (with details in appendix).

Node Clustering We set the number of clusters to the number of ground-truth classes and perform K-Means algorithm [9] on resulting embedding Z from DMAT-i following previous efforts [17, 1, 31, 6]. Table 1 summarizes clustering results. DMAT-i maintains either the state-of-the-art clustering results or is fairly close to the best. In particular, DMAT-i further reduced state-of-the-art accuracy gap between unsupervised learning and transductive supervised learning as presented later in Table 2 across datasets such as DBLP and Coauthor PHY. Not surprisingly, deep clustering methods that use both node attributes and graph structure appear to be more robust and stronger than those using either of them (KMeans and DeepWalk), although the latter shows good performance for certain datasets. Compared with GCN based methods like SDCN, DMAT-i shows significant performance gain due to solving over-smoothing issues through graph filtering. For clustering methods like AGC, SSGC or AGE with carefully designed Laplacian-smoothing filters, DMAT-i can still outperform them in most cases. The most competitive clustering performance comes from AGE on several datasets – however, it does not even converge for DBLP. DMAT-i achieves robust convergence across all real-word datasets – a detailed summary of convergence time across different datasets is presented in the appendix.

Table 3: Link prediction performance.

Dataset	Metrics	DMAT-i	SSGC	AGE	GCA	ProGCL
Cora	AP	92.41 ± 0.28	93.24 ± 0.00	92.26 ± 0.30	92.95 ± 0.41	92.87 ± 0.28
	AUC	92.62 ± 0.29	92.14 ± 0.00	92.07 ± 0.21	92.95 ± 0.34	93.60 ± 0.13
Citeseer	AP	95.52 ± 0.26	96.14 ± 0.00	92.22 ± 0.48	93.38 ± 0.39	95.65 ± 0.28
	AUC	95.19 ± 0.26	95.29 ± 0.00	92.66 ± 0.44	92.57 ± 0.49	95.59 ± 0.24
Pubmed	AP	95.42 ± 0.08	97.53 ± 0.00	84.67 ± 0.09	92.65 ± 0.44	-
	AUC	95.18 ± 0.10	97.84 ± 0.00	86.70 ± 0.12	93.81 ± 0.37	-
ACM	AP	97.55 ± 0.17	82.33 ± 0.00	98.14 ± 0.10	92.61 ± 1.05	97.02 ± 0.23
	AUC	97.41 ± 0.17	81.15 ± 0.00	97.51 ± 0.18	94.19 ± 0.75	97.31 ± 0.18
DBLP	AP	95.50 ± 0.37	95.88 ± 0.00	92.68 ± 0.31	93.41 ± 0.61	95.99 ± 0.24
	AUC	95.50 ± 0.50	95.22 ± 0.00	90.96 ± 0.43	92.47 ± 0.59	95.38 ± 0.23
Amazon Photo	AP	92.73 ± 0.23	83.93 ± 0.00	91.65 ± 0.23	76.26 ± 1.39	93.43 ± 0.65
	AUC	93.89 ± 0.20	89.21 ± 0.00	93.12 ± 0.19	81.28 ± 1.34	95.66 ± 0.42
Coauthor CS	AP	94.76 ± 0.14	88.96 ± 0.00	93.96 ± 0.18	82.70 ± 0.98	-
	AUC	95.03 ± 0.12	93.56 ± 0.00	93.61 ± 0.15	83.54 ± 0.74	-
Coauthor PHY	AP	91.25 ± 0.20	93.92 ± 0.00	94.35 ± 0.08	-	-
	AUC	92.75 ± 0.15	96.57 ± 0.00	95.20 ± 0.06	-	-

Node Classification For the transductive semi-supervised node classification task, we applied train-validation-test data split with fraction as train (10%), validation (10%), and test (80%). Following the experimental settings of SSGC [31] and GCA [32], we evaluate the classification performance of DM(A)T and DMAT-i by using a linear classifier to perform semi-supervised classification and report the accuracy. As shown in Table 2, the embedding generation methods are categorized based on the availability of labels, where DM(A)T learns from train data (10% of all data) with labels and DMAT-i can proceed in an unsupervised way on all data samples. For comparison, we apply DMAT-i in both settings.

With labelled training data, DM(A)T turns out to achieve high quality of representations and shows superior results. DMAT-i, however, fails to recognize part of positive samples as compared with DMAT in this condition and lose some accuracy. When labels are completely unavailable, we can see that competitive results have been observed from DMAT-i compared to other advanced baselines under unsupervised setting. More importantly, DMAT-i generally achieves better performance when generating embedding in unsupervised condition than "partially-supervised" condition (with partial labels available). That is because much more samples i.e. all data, are included during tuplet loss optimization.

Link Prediction To evaluate DMAT-i on this task, we remove 5% edges for validation and 10% edges for test while keeping all node attributes [21, 17, 6]. The reconstructed adjacency matrix $\hat{A}$ can be calculated as per the previous publication [21]: $\hat{A}{=}\sigma(ZZ^T)$, where σ denotes the sigmoid function. For comparison purposes, we report area under the ROC curve (AUC) and average precision (AP) following settings from previous works [21, 17, 6]. As shown in Table 3, DMAT-i is robust, i.e, produces high-quality link prediction (above 90% for both metrics for all datasets), whereas no other methods has a comparable consistency.

6 Conclusions

This paper has presented a scalable graph (node-level) learning framework. Employing a mutli-class tuplet loss function, we have introduced both semi-supervised learning and unsupervised algorithms. We have also established connections between tuplet loss and contrastive loss functions and also theoretically shown how our method leads to generalization error bound on the downstream classification task. The learned representation is used for three downstream tasks: node clustering, classification, and link prediction. Our extensive evaluation has shown better scalability over any existing method, and consistently high accuracy (state-of-the-art or very competitive in each case).

References

1. Bo, D., Wang, X., Shi, C., Zhu, M., Lu, E., Cui, P.: Structural deep clustering network. WWW (2020)
2. Chen, B., Li, P., Yan, Z., Wang, B., Zhang, L.: Deep metric learning with graph consistency. In: AAAI. vol. 35, pp. 982–990 (2021)
3. Chen, M., Wei, Z., Ding, B., Li, Y., Yuan, Y., Du, X., Wen, J.: Scalable graph neural networks via bidirectional propagation. In: NeurIPS (2020)
4. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: ICML (2020)
5. Chuang, C.Y., Robinson, J., Lin, Y.C., Torralba, A., Jegelka, S.: Debiased contrastive learning. NeurIPS (2020)
6. Cui, G., Zhou, J., Yang, C., Liu, Z.: Adaptive graph encoder for attributed graph embedding. In: KDD (2020)
7. Cui, Y., et al: Fine-grained categorization and dataset bootstrapping using deep metric learning with humans in the loop. In: CVPR (2016)

8. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016), http://www.deeplearningbook.org
9. Hartigan, J.A., Wong, M.A.: A k-means clustering algorithm. Journal of the Royal Statistical Society. Series C (Applied Statistics) **28**(1), 100–108 (1979)
10. Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., Krishnan, D.: Supervised contrastive learning. In: NeurIPS (2020)
11. Lazer, D., Pentland, A., Adamic, L., Aral, S., Barabasi, A., Brewer, D., Christakis, N., Contractor, N., Fowler, J., Gutmann, M.: Life in the network: The coming age of computational social science **323** (2009)
12. Li, Q., Han, Z., Wu, X.: Deeper insights into graph convolutional networks for semi-supervised learning. In: AAAI (2018)
13. Meyer, B.J., Harwood, B., Drummond, T.: Deep metric learning and image classification with nearest neighbour gaussian kernels. In: ICIP (2018)
14. Newman, M.E.J.: Modularity and community structure in networks. Proceedings of the National Academy of Sciences **103**(23) (May 2006)
15. Norouzi, M., Fleet, D.J., Salakhutdinov, R.R.: Hamming distance metric learning. Advances in neural information processing systems **25** (2012)
16. Oh Song, H., Xiang, Y., Jegelka, S., Savarese, S.: Deep metric learning via lifted structured feature embedding. In: CVPR. pp. 4004–4012 (2016)
17. Park, J., Lee, M., Chang, H., Lee, K., Choi, J.: Symmetric graph convolutional autoencoder for unsupervised graph representation learning. In: ICCV (2019)
18. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: KDD (2014)
19. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: A unified embedding for face recognition and clustering. CVPR (2015)
20. Sohn, K.: Improved deep metric learning with multi-class n-pair loss objective. In: Advances in Neural Information Processing Systems (2016)
21. Thomas, K.N., Max, W.: Variational graph auto-encoders. NIPS Workshop on Bayesian Deep Learning (2016)
22. Thomas, K.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2017)
23. Veličković, P., Fedus, W., Hamilton, W.L., Liò, P., Bengio, Y., Hjelm, R.: Deep Graph Infomax. In: ICLR (2019)
24. Wang, F., Liu, H.: Understanding the behaviour of contrastive loss. In: CVPR (2021)
25. Wang, T., Isola, P.: Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In: ICML (2020)
26. Xia, J., Wu, L., Wang, G., Chen, J., Li, S.Z.: Progcl: Rethinking hard negative mining in graph contrastive learning. In: ICML (2022)
27. Yang, J., Leskovec, J.: Defining and evaluating network communities based on ground-truth. In: KDD. MDS '12 (2012)
28. Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W.L., Leskovec, J.: Graph convolutional neural networks for web-scale recommender systems. In: KDD (2018)
29. Zhang, X., Liu, H., Li, Q., Wu, X.: Attributed graph clustering via adaptive graph convolution. In: IJCAI (2019)
30. Zhao, W., Rao, Y., Wang, Z., Lu, J., Zhou, J.: Towards interpretable deep metric learning with structural matching. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (2021)
31. Zhu, H., Koniusz, P.: Simple spectral graph convolution. In: ICLR (2021)
32. Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., Wang, L.: Graph contrastive learning with adaptive augmentation. In: WWW (2021)