Transferable and Differentiable Discrete Network Embedding for Multi-domains with Hierarchical Knowledge Distillation

Tao He[a,b], Lianli Gao[c],Jingkuan Song[c], Yuan-Fang Li*[b]

[a] *School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731*

tao.he01@hotmail.com

[b] *Faculty of Information Technology, Monash University, Clayton, VIC 3800*

yuanfang.li@monash.edu

[c] *School of Computer Science, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731*

lianli.gao@uestc.edu.cn, jingkuan.song@gmail.com

Highlights

- The first discrete network embedding with the transferability for multi-domains
- A hierarchical knowledge distillation strategy to transfer rich knowledge on the source domain to a novel domain in an unsupervised fashion
- A novel differentiable discretization technique
- State-of-the-art performance of discrete network embedding for multi-domains

Transferable and Differentiable Discrete Network Embedding for multi-domains with Hierarchical Knowledge Distillation

Tao He[a,b], Lianli Gao[c], Jingkuan Song[c], Yuan-Fang Li*[b]

[a] *School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731*
[b] *Department of Data Science and AI, Faculty of Information Technology, Monash University, Clayton, Victoria 3800*
[c] *Center for Future Media, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731*

Abstract

Discrete network embedding aims to learn compact and low-dimensional discrete representations for network (graph) nodes by preserving semantical and structural information of networks. In practice, there are numerous domains for network data, but expensive semantical labels may only be available for some of them. This demands those embedding algorithms to be equipped with transferability, i.e., unsupervisedly adapting to other domains. However, the prevailing conventional discrete network embedding methods force on a single domain. This work bridges this gap by developing a transferable discrete network embedding method. We focus on solving two problems: knowledge transfer from a labeled domain to a new unlabeled domain and

Email addresses: tao.he01@hotmail.com (Tao He), lianli.gao@uestc.edu.cn (Lianli Gao), jingkuan.song@gmail.edu (Jingkuan Song), yuanfang.li@monash.edu (Yuan-Fang Li*), Corresponding Author. (Yuan-Fang Li*)

non-differentiable discretization in the discrete embedding. Specifically, for the former, we propose a hierarchical knowledge distillation strategy to mitigate the knowledge gap between the domains. At the same time, for the latter, we novelly treat the discretization as a classification problem instead of the conventional regression problem. Finally, we conduct a wide range of experiments to evaluate our method on a suite of tasks, including link prediction, node classification, and neighbor recommendation. Our evaluation results demonstrate that our model performs better than the conventional discrete embedding methods over all the tasks.
Keywords: Discrete Network Embedding, Hierarchical Knowledge Distillation, Transfer Learning, Differentiable Discretization

1. Introduction

Discrete network embedding tries to learn an encoding paradigm to embed high-dimensional graphical nodes into low-dimensional, discriminative and compact discrete codes. In this line of research, learning to hash [34, 28] or quantisation [12] are the two mainstream approaches to effective data compression since it enables efficient storage (in terms of space) and fast retrieval (in terms of time) with the explosive increase in the amount of available digital data. This work mainly targets the former, i.e., binarized (hash) embeddings for networks. In practice, however, since the networks are versatile from variant domains, e.g., paper citation networks from different periods [31], this poses a question: *can we develop a discrete network*

embedding method with transferability for multiple domains? It is worth noting that the research scope of this work lies in the conventional *closed-set* domain adaptation [46, 35], which assumes that both the source and target domains have the same label set, but the more challenging open-set domain adaption [22, 19] is not the focus in this paper.

Recently, although discrete embedding techniques, e.g., hashing [25, 37], have been widely investigated, they dominantly focus on single-domain networks. Those methods only work well on the source domain but would fail on a target (new) dataset with a distributional shift from the source. In other words, these methods do not adequately address the knowledge transfer among the multiple domains. Moreover, the new domains are always unlabeled, requiring the knowledge transfer to be conducted in an unsupervised fashion.

A natural solution to mitigate the knowledge discrepancy is to finetune the model learned on the source domain with the target domain. However, the problem of finetuning lies in that retraining the model on new datasets requires the availability of human annotations of the new domain, which is expensive to obtain and thus may not be available. Therefore, this motivates us to devise an unsupervised knowledge transfer mechanism to bridge the distribution gap of multiple domains. To this end, the mainstream technique is Generative Adversarial Networks (GANs) [36, 32]. In a nutshell, they leverage a discriminator network to distinguish the source of embeddings. When the discriminator cannot determine the origin of the embeddings, they

believe the domain disparity has been suppressed to a relatively low level. However, GAN cannot well handle the distribution of discrete embeddings, as evidenced in [42]. To sidestep this issue, we borrow the idea of knowledge distillation (KD) [30] that transfers the knowledge of a mature teacher model to a student model and propose a hierarchical knowledge distillation strategy to enhance the transferability of the embedding model by dividing the transferred knowledge into three hierarchical levels from coarse to fine. Specifically, we define the three types of transferred knowledge as 1) prototype-based centroids that present the high-level semantics of each domain; 2) regional pointwise feature ensuring multi-domain features in the embedding space are regionally aligned on top of the prototype-based centroids; and 3) cross-domain prediction scores guaranteeing the aligned embeddings with the same semantics, that is, the supervision on the source domain is transferred to the new domain in a pointwise manner.

The other challenge in discrete network embedding is non-differentiable nature of discretization functions [2], mainly caused by the widely-applied and non-differentiable *sign* function. This problem has been mitigated by replacing it with the differentiable and continuous *tanh* function or using the alternative coordinate optimisation strategy [40]. However, those techniques could produce undesirable relaxation errors [33], degrade learned discrete codes' quality, and hardly integrate the whole embedding network into an end-to-end scheme. To address this issue, we treat the binarized procedure as a classification problem instead of the conventional regression problem [28].

Concretely, we deploy a linear classifier for each dimension of a discrete embedding based on the Gumbel softmax function [4]. We do not involve any discretized function, such as *sign* or *tanh*, eventually avoiding any relaxation error produced.

In summary, we propose a transferable and differentiable discrete network embedding (TD-DNE) method in this work for multi-domains. This is the first discrete network embedding method with the transferability for multi-domains to the best of our knowledge. Specifically, we develop a hierarchical knowledge distillation strategy to transfer the multiple knowledge to the new domains from coarse to fine in an unsupervised manner and novelly treat the discretization function as a composition of multiple linear classifiers based on the Gumbel-Softmax function to efficiently train the model without involving relation errors during discretizing. Last, we evaluate our method on three multi-domains datasets on the standard tasks of network embedding, such as link prediction, node classification, node recommendation, and an ablation study. All results demonstrate that our method can promisingly outperform conventional discrete network embeddings focusing on a single domain.

A preliminary version of this paper has appeared at IJCAI 2019 [14]. The preliminary version concentrated on the domain-adaptive hashing problem for the image modality by generative adversarial networks (GANs). In this manuscript, we have made the following major extensions:

1. We first propose a transferable discrete network embedding method for multi-domain networks.

2. We develop a hierarchical knowledge distillation to mitigate the knowledge gap among multiple domains instead of the conventional GANs.
3. We leverage a reparameterization technique to design a differentiable discretization function.

The rest of this paper is arranged as follows. Sec. 2 presents several highly related works and our research backgrounds. The following Sec. 3 introduces our proposed TD-DNE method accompanying our three main techniques. Then, we conduct extensive experiments in Sec. 4, including visualization examples. The conclusion is drawn in Sec. 5.

2. Related Work

We briefly survey relevant literature from three aspects: domain-adaptive learning, discrete network embedding, and knowledge distillation.

2.1. Domain Adaptation Learning

The primary purpose of domain adaptation is to manipulate supervised information on the source domain to guide model training on the target domain without ground-truth labels. In the computer vision community, domain adaptation has been applied in image segmentation [47] and image retrieval [14]. Unlike traditional training datasets consisting of single-domain data, domain-adaptive learning aims to train a unified model to handle multiple domains (e.g., digits and handwritten numbers). The main challenge

of domain adaptation lies in the distributional discrepancy between different domains, also named as domain shift in some other fields. To this end, many unsupervised strategies has been proposed to diminish the domain distribution semantical mismatch. A fundamental idea is to supervisedly train a classifier on the source domain and then finetune it on the new domain. Hu *et al.* [16] developed an autoencoder-based duplex network to bridge the distribution gap between the two domains by equipping two generator networks with the ability to reconstruct both domains' data from their latent space. A standard practice of dealing with adaptive learning is to project the source and target domains into a common space and then reduce the domain disparity [8] by a distribution alignment component or loss function.

2.2. Discrete Network Embedding

Network embedding [43] aims to map each node or edge in a network (e.g., heterogeneous network, homogeneous network, attributed network, etc.) into a low-dimension vector and simultaneously preserve the network's information, including structure and semantics, as much as possible. However, with the explosive growth of networks, the conventional continuous embeddings [11] require ample space or memory to store their embeddings. To overcome this issue, a suit of researchers propose the other line of research, i.e., discrete network embedding [40, 12]. Specifically, Yang *et al.* [40] first proposed a binarized attributed network embedding model to encode network structure and attributes by leveraging the Weisfeiler-Lehman proximity

matrix. Nearly at the same time, Lian *et al.* [20] based on the matrix factorization develop an information network hashing method to preserve the high-order proximity, which can achieve almost comparable performance with the corresponding continuous embedding. Wu *et al.* [37] first leverage the random hash techniques to embed both content and structure information of attributed networks. Unlike the previous discrete embeddings using binarized codes to present networks, [13] propose a semisupervised quantisation-based method for attribute networks to alleviate the performance degradation of hashing codes.

2.3. Knowledge Distillation

Knowledge distillation (KD) [10] has been received extensive attention from the research community, particularly in model compression [30]. Without loss of generality, a KD-based method always consists of two parallel networks, a large teacher network that can be pretrained on massive data or corpus in a supervised manner and a small student model jointly trained by the teacher model with the unlabeled data. The main idea of KD is to encourage the student model to mimic the teacher model so that the student can achieve comparable performance with the teacher model. Based on the distilled knowledge types, we could divide them into three main categories: response-based knowledge [15], feature-based knowledge [18] and relation-based knowledge [23]. A very close to our work is that Song *et al.* [29] proposed to divide the distillation knowledge into different important levels

and distill the knowledge in a in a coarse-to-fine manner. Fang *et al.* [7] developed a data-free knowledge distillation via an adversarial training so that the student model can be trained without access to real-world data. In this work, we typically focus on the first two types: the response of the last prediction layer and the representation of a middle layer of the teacher model as the guided knowledge, respectively.

3. Methodology

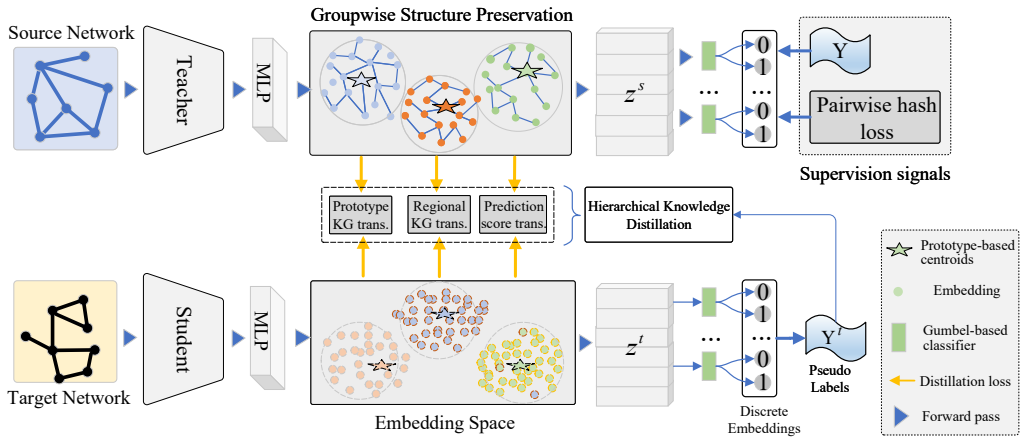


Figure 1: The overview of our TD-DNE, which consists of two parallel networks, a teacher network trained on the labeled source domain and a student network trained on the target domain without supervised signals. Note that the small circles in the embedding space denote different nodes and we use different colors to distinguish their semantic labels. Both networks mutually interact through our proposed hierarchical knowledge distillation. Specifically, we propose three coarse-to-fine knowledge-specific transfer techniques, i.e. prototype-based, regional feature based and cross-domain prediction score based, to enable the supervised information in the source domain to guide the adaptive learning in the target domain.

Figure 1 shows the overview framework of our proposed TD-DNE and all used notations in this paper are defined in Table 1. Specifically, our model

consists of a supervisedly trained teacher network and a parallel student network. TD-DNE attempts to solve the two problems: transferring the rich knowledge supervision knowledge to the novel domain and alleviating the discretization errors. To this end, we develop two techniques, hierarchical knowledge distillation and differentiable discretization embedding. Before introducing them, we first elaborate the network structure preservation for both domains.

Table 1: Notations.

Notations	Definition
D^s	Source domain data points
D^t	Target domain data points
$\mathrm{Encoder}^{teacher}$	Teacher network
$\mathrm{Encoder}^{student}$	Student network
$\mathcal{P}^+$	Group of positive nodes
$\lvert\mathcal{P}^+\rvert$	Number of positive nodes
$\boldsymbol{z}^s$	Embedding vectors of source domain nodes
$\boldsymbol{z}^t$	Embedding vectors of target domain nodes
$\boldsymbol{b}$	Discrete embedding vectors
ϕ	Distance measurement function
$\mathbf{y}^{t'}$	Pseudo label of target nodes

3.1. Network Structure Preservation

Following a couple of network embedding techniques [46, 9, 1], we employ multi-layer perceptron (MLP)-based deep neural network [20, 40, 25] as our encoder network. Thus, the encoding process for both domains can be

formulated as:

$$
\begin{aligned}
h_i^s &= \text{Encoder}^{teacher}(\mathbf{w}_i h_{i-1}^s + \mathbf{b}_i) \\
h_i^t &= \text{Encoder}^{student}(\mathbf{w}_i h_{i-1}^t + \mathbf{b}_i)
\end{aligned}
\tag{1}
$$

where the $\text{Encoder}^{teacher}$ and $\text{Encoder}^{student}$ denotes the teacher model and student network consisting of MLPs, respectively. Specifically, both of them have three components: Dropout, LayerNorm and ReLU non-linearity; i denotes the i-th layer of the multi-layer perceptron, and $\mathbf{w}_i$ and $\mathbf{b}_i$ represent the i-th layer's weight and bias parameters respectively. It is worth noting that when $i = 1$, h_0^s (resp. h_0^t) is initialized by the source (resp. target) domain node features x^s (resp. x^t).

Then, we utilize the metric learning technique [48, 41] to equip embeddings with nodes' neighborhood information. Specifically, we define two types of nodes, the positive and negative, based on an anchor node. Formally, we consider the positive node group $\mathcal{P}^+$ only if they have a direct connection, otherwise as a negative group $\mathcal{P}^-$. The objective function can be formulated as:

$$
\mathcal{J} = \max_{j \in \mathcal{P}_i^+, k \in \mathcal{P}_i^-} (0, \lambda + \phi(z_i, z_j) - \phi(z_i, z_k)) \tag{2}
$$

where j denotes an arbitrary node sampled from the positive group $\mathcal{P}_i^+$ of the anchor node i while k denotes an arbitrary node sampled from the negative group $\mathcal{P}_i^-$ of the anchor node i; λ is a constant margin hyperparameter and $\phi(\cdot)$ is a function to measure the distance of two embeddings in the embedding

space, for which we choose the Euclidean distance (L_2 norm). However, as the number of negative pairs are orders of magnitudes more than the number of positive pairs, the model is prone to inclining heavily to the negative samples, leading to poor preservation of network structure.

To address this issue, we modify it into a hard scheme [38] based on positive and negative groups instead of pairs, i.e. our groupwise objective function aims at minimizing the maximal distance in the positive group whilst maximizing the minimal distance in the negative group. More concretely, the positive group of node i is defined as all of its direct neighbors, denoted as $j \in \mathcal{P}_i^+$, while the negative group of node i is those nodes not in the neighbors of node i, i.e., $j \in \mathcal{P}_i^-$. Hence, we could rewrite Eq. (2) as the groupwise contrastive loss as the below:

$$\mathcal{L}_1 = \frac{1}{N}\sum_{i=1}^{N} \max(0, \lambda + \max_{j \in \mathcal{P}_i^+} \phi(z_i, z_j) - \min_{k \in \mathcal{P}_i^-} \phi(z_i, z_k)) \tag{3}$$

Since $|\mathcal{P}_i^+|$ is not too large, for a give anchor node i, we sample all its positive neighbors as $\mathcal{P}_i^+$. On the other hand, due to the large size of $|\mathcal{P}_i^-|$, we randomly sample about $10\times|\mathcal{P}_i^+|$ negative pairs to construct $\mathcal{P}_i^-$ each time.

3.2. Differentiable Discrete Embedding

As aforementioned, the main dilemma of discrete embedding lies in the non-differentiable discretization function, e.g. *sign* [40, 39] that simply model the discretization processing as a regression relaxation problem, that is forcing continuous embeddings to be close to their corresponding discrete embed-

dings. Inevitably, those methods could involve the quantization errors [33], i.e. $|sign(\boldsymbol{z}) - \boldsymbol{z}|$, where $\boldsymbol{z}$ denotes the continuous embeddings. This feature from $sign(.)$ could degrade the performance learned discrete embeddings.

To solve this issue, we inspired by the reparameterization trick [17, 26] view the discretization as a 0-1 classification problem. Specifically, we deploy l linear classifiers $\boldsymbol{W} \in \mathbb{R}^{l \times m \times k}$ after the embeddings $\boldsymbol{z} \in \mathbb{R}^{n \times d}$, where n is the number of nodes and d is the dimension of the embedding space, and $l \times m$ is equal to d. Note that k is set to 2 since there are only two options, 0 or 1, for each dimension. Then, we divide $\boldsymbol{z}$ into l components each of which is m-dimensional, i.e. $\boldsymbol{z} = [\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_l]$ where $\boldsymbol{z}_i \in \mathbb{R}^{n \times m}$. Formally, the classification processing is calculated by:

$$\boldsymbol{u} = [\boldsymbol{z}_1 \boldsymbol{W}_1; \boldsymbol{z}_2 \boldsymbol{W}_2; \ldots; \boldsymbol{z}_l \boldsymbol{W}_l] \tag{4}$$

where $[;]$ is the concatenation operation, and $\boldsymbol{u} \in \mathbb{R}^{n \times l \times k}$ is the prediction logits. Sequentially, we leverage the Gumbel-Softmax [4], which has gained great success in selecting categorical variables great, as the active function for the logits to generate prediction scores by:

$$\boldsymbol{s} = \left[\mathrm{g-softmx}(\frac{\boldsymbol{u}_1 + \boldsymbol{g}_1}{\tau}), \mathrm{g-softmx}(\frac{\boldsymbol{u}_2 + \boldsymbol{g}_2}{\tau}), \ldots, \mathrm{g-softmx}(\frac{\boldsymbol{u}_l + \boldsymbol{g}_l}{\tau}) \right] \tag{5}$$

where $\boldsymbol{g}_i \in \mathbb{R}^{1 \times k}$ is sampled from a Gumbel Distribution [17], i.e. $\boldsymbol{g}_i = \log(-\log(\mathrm{U}(0, 1))$ where U is a uniform distribution and $\tau \in (0, \infty)$ is a temperature parameter to adjust the approximation [4]. It is worth noting

that we omit the first dimension of $\boldsymbol{u}$ for ease of illustration, that is $\boldsymbol{u} \in \mathbb{R}^{l \times k}$. Additionally, the active function g-softmx is calculated by:

$$\mathrm{g-softmx}(\frac{\boldsymbol{u}_i + \boldsymbol{g}_i}{\tau}) = \frac{\exp\left(\left(u_{ij} + g_j\right)/\tau\right)}{\sum_{j=1}^{k} \exp\left(\left(u_{ij} + g_j\right)/\tau\right)} \tag{6}$$

After obtaining the prediction scores, we use argmax$(\cdot)$ to select the discrete codes, i.e. $\boldsymbol{b}_i = [\mathrm{argmax}(\boldsymbol{s}_1); \mathrm{argmax}(\boldsymbol{s}_2); \ldots; \mathrm{argmax}(\boldsymbol{s}_l)]$, where $\boldsymbol{s}_i \in \mathbb{R}^k$ and $\boldsymbol{b}_i \in \{0, 1\}^l$. To inject the supervision of the source domain, following the widely used pairwise constraint [14, 33], we employ the supervised pairwise loss to optimize $\boldsymbol{b}$:

$$\mathcal{L}_2 = \frac{1}{2} \sum_{y_{ij} \in \mathcal{Y}} \left(\frac{1}{l} \tilde{\boldsymbol{s}}_i^\top \tilde{\boldsymbol{s}}_j - y_{ij} \right)^2 \tag{7}$$

where $\tilde{\boldsymbol{s}}_i \in \mathbb{R}^l$ is the prediction score on the corresponding hash codes, that is $\tilde{\boldsymbol{s}} = [\max(\boldsymbol{s}_1); \max(\boldsymbol{s}_2); \ldots; \max(\boldsymbol{s}_l)]$, and $\mathcal{Y} \in \{-1, 1\}$ is the similarity matrix constructed from ground-truth labels of the source domain, that is, if two nodes have at least one same label, their similarity is defined as 1 and otherwise -1. Since Eq. (7) is differentiable, we can directly use gradient descent strategies to optimize the whole model.

3.3. Hierarchical Knowledge Distillation

Our ultimate goal is to transfer the supervision to the target domains, since labels are only available for the source domain network but not for new domains. To this end, we propose three coarse-to-fine knowledge distillation

components to hierarchically transfer the supervisedly learned knowledge from the source domain to the unlabeled target domain. Concretely, we focus on the following three types of knowledge: 1) prototype-based 2) regional feature-based and 3) cross-domain prediction score based.

Prototype-based Knowledge: The high-level prototype-based knowledge [27] is defined as the semantic centroids of a distribution. This knowledge tries to align the two domains' centers of the same class and guarantee the target domain's pivot centroids are consistent with the supervised semantics on the source domain. This can be viewed as a coarse alignment for two domains. As can be seen in Figure 1, we show three categories differentiating with three colors. When two domains' clusters (i.e., the star marks in both domains) are aligned, we could see their overall data distributions are also coarsely similar. In practice, we first use K-means clustering algorithm to calculate the centroids and leverage MSE loss to optimize their distance as:

$$\mathcal{L}_3 = \sum_{i=1}^{K} \text{MSE}(\frac{1}{m_i} \sum\nolimits_{y^s=i} \boldsymbol{z}^s, \frac{1}{n_i} \sum\nolimits_{y^{t'}=i} \boldsymbol{z}^t) \tag{8}$$

where $\boldsymbol{z}^s$ denotes an embedding of a source domain data point from the teacher network, while the $\boldsymbol{z}^t$ denotes an embedding of a target domain data point from the strudent network; K is the number of classes and m_i (resp. n_i) denotes the number of samples in the same cluster in the source (resp. target) domain. $y^{t'}$ is the pseudo label for the new domain and we will elaborate it latter.

Noticeably, due to the large time cost when calculating K-means on all samples, we do not precisely calculate the centers of all nodes in the training stage, but instead calculate the centers of mini-batches to approximate the global centers. It is worth noting that we usually set a large batch size to make sure the K-means clustering covers all categories during training.

Regional Feature-based Knowledge : Although the above prototype centroids ensure the two domains are globally aligned, the regional subspaces of the two domains in the embedding space might inconsistently distribute. For instance, as shown in Figure 1, even if the centroids of green points in the two domains are aligned, their other points do not distribute consistently. Hence, we further propose to impose restrictions on the alignment of both domains' regional features in a point-to-point manner. To this end, we feed the target data points with high-confidence labels into the teacher network and distill the pointwise feature knowledge from the teacher to the student network. Then, we use the MSE to optimize their distance , i.e., distilling the teacher model's knowledge to the student. Following [3], we could write our regional feature-based knowledge distillation as: $\mathcal{L}_4 = \sum_{i \in D^t} \text{MSE}(\boldsymbol{z}_i^t, \boldsymbol{z}_i^{s'})$, where $\boldsymbol{z}^{s'}$ denotes the embedding of target point i via the teacher network and $\boldsymbol{z}^t$ is the embedding of target point i via the student network. Compared to Eq.(8), their difference is that we remove the clustering process and this can be viewed as a global-to-regional knowledge alignment. Note that the target points with confidence labels are calculated by Eq. (9). The reason for that is we hope the features from the teacher network is relatively accurate

so that it can guide the feature learning of the student network.

Cross-domain Prediction Scores based Knowledge: The above modules constrains two domains' features are globally and regionally aligned, but their final semantical labels might be various. To this end, we propose to align the cross-domain prediction score to make sure the aligned pointwise embeddings with the same distributional semantics.

In practice, we let both networks intersectedly predict both domains' data. Specifically, the teacher model supervisedly classifies the source data by $\mathcal{L}_5 = -\frac{1}{N^s} \min \sum_{i=1}^{N^s} \mathbf{y}_i^s \log(\tilde{\mathbf{y}}_i^s)$, where $\tilde{\mathbf{y}}_i^s$ is the prediction probability of the source domain point i from D^s and $\mathbf{y}_i^s$ is the corresponding ground-truth label.

For the unlabeled target domain, we use the teacher model to generate highly precise pseudo labels for the target domain, and set a threshold T to select the labels, as shown below:

$$\mathbf{y}_i^{t'} = \begin{cases} \arg\max(\hat{\mathbf{y}}_i^t) & \text{if } \hat{\mathbf{y}}_i^t > T \\ -1 & \text{otherwise} \end{cases} \tag{9}$$

where $\mathbf{y}_i^{t'}$ denotes the pseudo label of the target domain point i, and $\hat{\mathbf{y}}_i^t$ is point i's prediction probability from the teacher model. After that, we treat the pseudo labels as the proxy ground-truth and train the student model in a supervised way, i.e. $\mathcal{L}_6 = -\frac{1}{N^t} \min \mathbf{y}_i^{t'} \log(\tilde{\mathbf{y}}_i^t)$, where $\tilde{\mathbf{y}}_i^t$ is the predicted score by the student model. Then, we constrain $\mathbf{y}_i^t$ and $\tilde{\mathbf{y}}_i^t$ to have the close

prediction score distribution by:

$$\mathcal{L}_7 = \frac{1}{N^s} \min \sum_i \mathrm{KL}\left(\hat{\mathbf{y}}_i^t \| \tilde{\mathbf{y}}_i^t\right) \tag{10}$$

where $\mathrm{KL}(\cdot)$ denotes the Kullback-Leibler divergence to measure the discrepancy between the two domains.

3.4. Learning

Overall there are three types of objective functions in our model: network structure preservation in Sec. (3.1), discrete hash learning in Sec. (3.2) and hierarchical knowledge transferring in Sec. (3.3). Formally, we could write the overall objective function as:

$$\mathcal{L} = \mathcal{L}_1 + \alpha \mathcal{L}_2 + \beta \mathcal{L}_{3\sim 7} \tag{11}$$

where α and β are the two hyper-parameters to balance the terms. It is worth noting that for the three hierarchical knowledge we equally set their weights as 1. In the latter parameters analysis, we test the impacts of the hyper-parameters on the learned discrete embeddings.

4. Experiments

In this section we will evaluate our TD-DNE against some state-of-the-art models on networks. We begin by describing the benchmark datasets, baseline models and implementation details.

4.1. Datasets

Following UDA [35], we conduct our experiments on three citation networks obtained from ArnetMiner [31]. Brief statistics of the three networks are shown in Table 2. We sample three subsets from three large citation networks: DBLPv4 (D), ACMv8 (A) and Citationv1 (C). To reduce the overlap between different datasets, we extract published papers from different periods for these three datasets following UDA [35]. Papers are classified into eight categories: Engineering, Electronic, Software Engineering, Mathematics, Theory, Applied, Artificial Intelligence , and Computer Science. For the attributes, we extract the word frequency of each paper's abstract, which is represented as an $8,328$-dimensional vector.

Table 2: Brief statistics of the three datasets.

G	DBLPv4	ACMv8	Citationv1
$\|V\|$	6,209	8,173	4,350
$\|E\|$	8,056	22,753	8,513
Attr.	8,328	8,328	8,328
Labels	8	8	8

4.2. Baselines

We choose the following state-of-the-arts discrete hash methods for network embeddings as our baselines.

- SH [6] is a classical and widely-applied hash method for the approximate nearest neighbor search task.

- Discrete Collaborative Filtering (DCF) [44] is a principled hashing method able to tackle the challenging discrete optimization problem in hash learning and avoid large quantisation errors caused by two-step optimization.

- DNE [25] is the first work of discrete representation learning for single domain networks by preserving Hamming similarity.

- NetHash [37] utilises randomized hashing techniques to generate discrete hash codes by treating a network as a tree-like graph.

- Binarized Attributed Network Embedding (BANE) [40] develops a Weisfeiler-Lehman proximity matrix to preserve the dependence between node attributes and connections via combining the features from neighboring nodes.

- Information Network Hashing (INH) [20] is an embedding compression method based on matrix factorization and able to preserve high-order proximity into binary codes.

- Discrete Embedding for Latent Networks (DELN) [39] is an end-to-end discrete network embedding method to learn binary representations.

4.3. Implementation Details

The feature encoder network consists of three MLP modules and the last layer outputs 256-dimensional embeddings. It is worth noting that a wide

range of network embedding methods [5, 35, 45] use graph neural network (GNN) as the encoder network, and have demonstrated that the GNN-based encoder is superior to MLP. However, in this work, to fairly compared with the other domain-adaptive network embedding methods [20, 40, 39, 25] using MLP as their encoder network, we also employ the MLP as our backbone network to learn the embeddings. The detailed configuration of our teacher network is shown in Table 3 and the student network has the same configuration.

The detailed configuration of our teacher network is shown in Table 3 and the student network has the same configuration. The batch size is set to 400, ensuring the effectiveness of the K-means algorithm. For a fair comparison with baseline methods, all methods' hash code length is set to 128. The margin λ for groupwise contrastive loss is set to 5. The temperature τ in Equation (6) is set to 1 as [4]. The learning rate is set to 0.005 and the threshold T for the pseudo label selection is set to 0.85. The hyper-parameters of α and β are set to 0.1 and 0.7 by a grid search [21] from the range $[0.1, 1.0]$ and $[0.1, 1.0]$, respectively. The search step is set 0.1.

For the evaluation on the node classification task, we first generate all nodes' embeddings and then train a one-vs-rest logistic regression classifier to classify the embeddings, where all methods use the same-dimensional hash codes for training and testing. We measure the mean score of Micro F1 and Macro F1 metrics to evaluate the performance of node classification, following DANE [46], and use the area under curve (AUC) score to evaluate

Layer	Size of Filter	Operations
MLP_1	8,328 × 1,024	Dropout, LayerNorm, ReLu
MLP_2	1,024 × 512	Dropout, LayerNorm, ReLu
MLP_3	512 × 256	Dropout, LayerNorm, ReLu
$\boldsymbol{W}$	128 ×2 × 1	g-softmx
MLP_4	128 × 256	Dropout, LayerNorm, ReLu
MLP_5	256 × 512	Dropout, LayerNorm, ReLu
MLP_6	512 × 8	Dropout, Softmax

Table 3: Configuration details of our teacher network.

the performance of link predication, following Graph2Gauss [1]. For link prediction, we randomly select 5% and 10% edges as the validation and test set respectively, following Graph2Gauss [1]. All experiments were performed on a workstation with 256 GB memory, 32 Intel(R) Xeon(R) CPUs (E5-2620 v4 @ 2.10GHz) and 8 GeForce GTX 1080Ti GPUs.

4.4. Cross-domain Node Classification Results

Table 4 shows the node classification results (of discrete embeddings) compared with the state-of-the-art discrete network embedding methods. For a fair comparison, since SH cannot learn representations for networks, we use the feature learned from TD-DNE to train SH, i.e., the latent embeddings $\boldsymbol{z}$, while the other models use nodes' attributes to train. From Table 4, it can be observed that our method TD-DNE achieves the best performance over all but one domain-transfer tasks, except for C→A with DELN being 0.33 percentage points higher than ours. On average, TD-DNE is superior to the baseline methods, surpassing the second best method DELN by 2.42 per-

Table 4: Node classification results on six cross-domain tasks compared with the state-of-the-art discrete embedding methods in terms of the mean of Micro-F1 and Macro-F1 score (%).

Methods	A→D	A→C	C→D	C→A	D→A	D→C	Average
SH	21.25	16.51	20.37	17.52	19.68	19.34	19.13
DCF	20.47	18.13	21.52	19.11	20.24	23.06	20.42
DNE	23.51	24.81	22.38	21.91	22.40	25.06	23.35
NetHash	23.83	19.38	24.15	23.05	23.52	24.28	23.06
BANE	25.72	22.15	20.63	22.40	23.08	20.75	22.46
DELN	25.06	26.83	28.59	**27.12**	26.13	25.23	26.48
INH	21.40	25.14	24.36	25.60	24.41	25.40	24.55
TD-DNE	**25.41**	**29.82**	**31.37**	26.79	**29.25**	**30.14**	**28.90**

centage points. We consider that the main reason is the compared methods can only work well on single domains, but perform poorly on new domains with relatively large distribution disparity. For example, the majority of discrete embedding methods, including DNE, INH and DELN, leverage a matrix factorization technique to learn the binary codes. They decompose the input attribute matrix into hash codes under the constraint of reconstructing the original attribute matrix. Their common issue lies in the fact that the target network has a large different attributes distribution from the source network, which results in large reconstruction errors by the matrix decomposition operation, and finally leading to poor results on the domain transfer. In contrast, TD-DNE explores hierarchical knowledge distillation techniques to mitigate the problem of attribute distribution shift by aligning their embedding space and prediction scores. In the ablation study described later, we will further discuss how effective those techniques are.

Table 5: Link prediction results on six cross-domain tasks compared with the state-of-the-art discrete embedding methods in terms of the mean of AUC score (%).

Methods	A→D	A→C	C→D	C→A	D→A	D→C	Avg.
SH	65.24	67.51	64.37	61.82	62.43	65.29	64.44
DCF	68.47	66.38	65.12	63.22	67.59	64.18	65.83
NetHash	71.64	69.73	68.51	64.14	68.65	67.74	68.40
DNE	65.81	70.34	69.15	70.35	71.19	68.31	69.19
BANE	66.37	71.62	65.79	69.42	66.36	66.67	67.71
DELN	70.29	68.41	65.82	67.31	66.81	68.03	67.78
INH	74.51	76.14	70.38	69.60	72.43	72.40	72.31
TD-DNE	**77.02**	**78.52**	**74.27**	**71.79**	**75.37**	**74.36**	**75.17**

It is worth noting that SH gains comparable results with DCF and does not show catastrophic performance degradation as we originally anticipated, possibly because SH is trained by our learned continuous representations, which, to some extent, confirms that our network embedding strategy can learn high-quality embeddings for multiple domains.

4.5. Link Prediction Results

Link prediction evaluates the learned codes' ability to reconstruct the original network's neighbor structure. Following Graph2Gauss [1], the validation/test set consists of 5%/10% of edges randomly sampled from the network respectively, and we randomly select edges and an equal number of non-edges from the test set. Table 5 shows the link prediction results of discrete embeddings on the six cross-domain tasks.

From the results we can observe that our TD-DNE method exceeds other single-domain discrete embedding methods over all domain transfer tasks.

In particular, TD-DNE is 2.86 points better than the second best method INH. Since INH places emphasis to preserving high-order proximity, it shows superiority to other matrix factorization based models such as DNE. It is worth noting that DELN achieves poor performance in this task, although it obtains competitive results in node classification.

In summary, although single-domain embedding methods can handle unitary networks, they consistently show performance decreases when evaluated in a domain-adaptive setting, in which our method achieves a substantial performance advantage due to techniques specifically designed to tackle this problem.

4.6. Node Recommendation

Node recommendation is a widely employed task to evaluate retrieval performance for social and commercial networks, for which discrete embeddings can save much time [20]. Given a query code, node recommendation aims to returning a list of nodes, ranked by their structural similarity. Following the settings [20], we sample 90% neighbors of each node to train the model while the remaining 10% neighbors are reserved for evaluation, and use NDCG@50 as the evaluation metric.

Table 6 presents the performance of node recommendation on six cross-domain tasks. From the table, we could observe that TD-DNE outperforms all the baseline methods in terms of the average performance, outperforming the second best method INH by approx. 2 points. INH in turn outperforms

Table 6: Node recommendation results on six cross-domain tasks compared with the state-of-the-art discrete embedding methods in terms of NDCG@50 (%).

Methods	A→D	A→C	C→D	C→A	D→A	D→C	Avg.
SH	9.64	12.74	14.04	13.16	10.72	9.12	11.57
DCF	11.61	14.93	17.38	15.47	13.58	10.36	13.89
NetHash	14.29	13.07	16.62	14.70	12.93	12.03	13.94
DNE	13.84	15.39	18.85	14.52	14.02	14.28	15.15
BANE	19.30	16.83	19.40	18.31	15.19	17.95	17.83
INH	20.51	22.34	26.31	21.38	18.30	**20.76**	21.60
DELN	19.73	20.15	24.25	20.17	17.84	19.31	20.24
TD-DNE	**23.12**	**24.53**	**28.10**	**24.28**	**21.15**	20.23	**23.56**

the third best DELN by 1.26 points. Although both of them are based on matrix factorization, INH's advantage comes from its capability to learn high-order proximity. Noticeably, even if TD-DNE only explores the first-order neighborhood structure preservation, our other techniques aiming to reduce the domain discrepancy play an important role in network structure and semantics preservation. In contrast, the other methods, such as DCF, NetHash and DNRE, perform much more poorly in this task because of missing the transferability for cross-domain tasks.

4.7. Ablation Study

In this section, we study the effectiveness of each component in TD-DNE. Specifically, we first examine three main techniques: groupwise contrastive loss, Gumbel-softmax based discretization, hierarchical knowledge distillation. Besides, we also test another mainstream encoder network, i.e., graph neural network [24], which has been demonstrated to be effective in learning

representations for structural data. Thus, we ablate TD-DNE into four variants:

- $-\mathcal{L}_1$ uses a pointwise contrastive loss Eq.(2) instead of our groupwise constraint Eq.(3).
- $-\mathcal{L}_2$ uses the regression strategy instead of the Gumbel-Sofmax strategy based discretization in Sec. 3.2.
- $-\mathcal{L}_{hkd}$ discards hierarchical knowledge distillation described in Sec. 3.3.
- TD-DNE$_{gnn}$ denotes that we replace MLP-based encoders in Eq.(1) with GNN.

We thoroughly conduct experiments on node classification and link prediction, as shown in Tables 7 and 8, respectively. It is worth noting that all variants are under the same experimental configurations expect for their corresponding ablated module(s). From the two tables, we could make the following observations:

(1) From the comparison of $-\mathcal{L}_1$ and TD-DNE, a moderate decrease can be seen in the variant of $-\mathcal{L}_1$, which confirms that our groupwise contrastive loss is more effective than the pairwise version. We think the main reason is that Eq. (2) is prone to falling into sub-optimality, because Eq. (2) can only select $|\mathcal{P}_i^+|$ negative samples each time, but due to the large scale of $|\mathcal{P}_i^-|$, we hardly guarantee that all anchor nodes' positive pairs have smaller distances than its negative ones. By contrast, in Eq. (3), more negative

Table 7: The impact of each component on the task of node classification (%).

Methods	A→D	A→C	C→D	C→A	D→A	D→C	Avg.
$-\mathcal{L}_1$	23.91	27.49	29.03	25.17	28.11	29.84	27.24
$-\mathcal{L}_2$	22.53	25.20	27.42	26.20	27.26	28.01	26.10
$-\mathcal{L}_{hkd}$	22.28	25.48	26.37	22.09	24.12	26.12	24.41
$\mathcal{L}_{hkd}-p$	23.03	27.83	29.25	24.72	26.47	28.52	26.57
$\mathcal{L}_{hkd}-r$	24.22	28.90	30.41	26.03	28.31	29.54	27.90
$\mathcal{L}_{hkd}-c$	23.85	28.12	29.70	25.39	27.83	29.06	27.32
TD-DNE$_{gnn}$	25.85	30.27	32.24	26.63	29.74	31.35	29.27
TD-DNE	25.41	29.82	31.37	26.79	29.25	30.14	28.90

Table 8: The impact of each component on the task of link prediction (%).

Methods	A→D	A→C	C→D	C→A	D→A	D→C	Avg.
$-\mathcal{L}_1$	74.34	76.11	70.51	67.25	71.30	72.04	71.95
$-\mathcal{L}_2$	76.82	78.03	73.25	70.10	74.24	74.18	74.43
$-\mathcal{L}_{hkd}$	75.11	77.50	73.03	71.20	74.62	73.25	74.12
TD-DNE$_{gnn}$	77.81	79.58	74.91	72.14	75.43	75.20	75.68
TD-DNE	77.02	78.52	74.27	71.79	75.37	74.36	75.17

pairs are considered during the distance calculation, i.e., we only select the negative points with the largest distance to optimize the contrastive loss. This benefits the model to find the optimal solution. Besides, we could observe that the groupwise contrastive loss plays a more important role in the link prediction than in node classification, because $\mathcal{L}_1$ is to preserve network structure instead of semantics.

(2) Comparing $-\mathcal{L}_2$ with TD-DNE, we could notice that the Gumbel-Softmax reparameterization trick brings about 2 and 1 points lift on node classification and link prediction, respectively. We conjecture the reason is

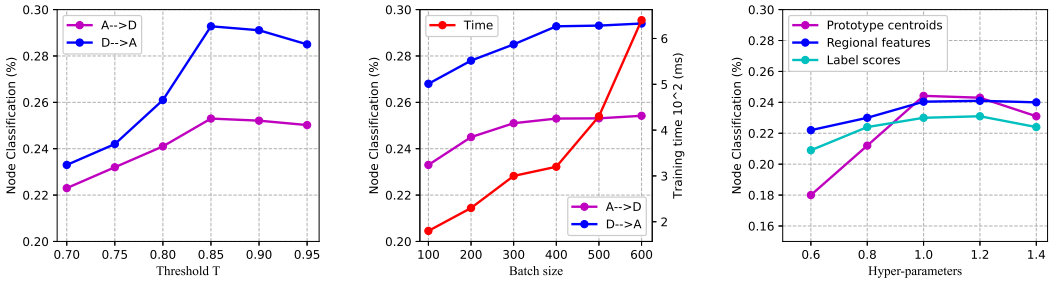
that we directly treat the generation of discrete codes as a classification problem, and intrinsically avoid the quantization errors of the linear regression.

(3) When removing the hierarchical knowledge distillation strategy, it is noticeable that the node classification suffers a large performance degradation, on average with more than 4 points. This confirms that knowledge distillation is essential for the semantical knowledge transferring. On the other hand, the knowledge distillation does not show much effectiveness on the task of link prediction, possibly because our distillation only focuses on the semantical knowledge but not the network structure. To further study impacts of the three types of knowledge in the distillation on the task of node classification, we conduct the other ablation experiments, i.e., removing each knowledge type in turn, and the bottom block in the Table 7 shows the results, where $\mathcal{L}_{hkd}$-p, $\mathcal{L}_{hkd}$-r and $\mathcal{L}_{hkd}$-c denote the model removes the prototype-based semantic centroids, regional embedding distribution and cross-domain prediction score, respectively. Generally, we could witness all of the three knowledge types can positively contribute to the node classification, especially the prototype centroids.

(4) For TD-DNE$_{gnn}$, we also use a three layers GNN as our encode network. From the results, we could see that when replacing our MLP-based encoder with GNN, there is ~ 1 point improvement on average on the task of node classification. Similarly, the same situation can been found on the link prediction. This further demonstrates that graph convolution network is more effective than MLP in embedding structural data.

4.8. Parameters Analysis

This section empirically studies the hyperparameters in our method: the threshold T for the pseudo label selection, the training batch size for K-mean in the prototype-based centroids and the weights for the three hierarchical knowledge.



(a) (b) (c)

Figure 2: Parameter sensitive analysis on the transfers of $A \rightarrow D$ and $D \rightarrow A$. (a) shows the results under different threshold T; (b) presents the results and time cost with different batchsize values; and (c) shows shows impacts of three different weights for the three hierarchical knowledge on node classification.

We particularly examine them on the two domain-transfer schemes: $A \rightarrow D$ and $D \rightarrow A$ on the task of node classification and the results are shown in Figure 2. From the Figure 2a, we could see that with the increase of threshold T value, both results on the two settings gains a relative growth when the threshold is less than 0.85. However, with further enlarging T over it, we could observe a slight performance degrease. We deem the main reason is that smaller T could incur lower-quality pseudo labels for the target domain, degrading the effectiveness of prototype centroids and cross-domain prediction scores, but a larger T could result in less number of pseudo labels

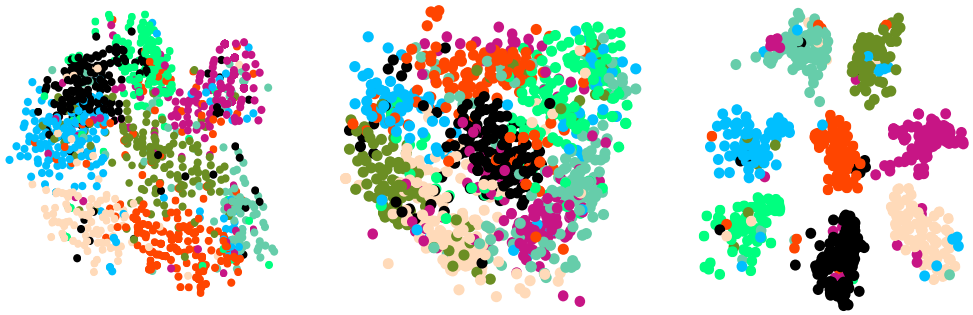
and inefficient training on the student network.

In Figure 2b, we can find that larger batch size could benefit the node classification because of the more accurate K-means clustering. However, the model also needs more training time. To trade off both of them, we choose 400 as our training batch size.

Figure 3c shows impacts of three different weights for the three hierarchical knowledge on node classification. From the weights, we could see that when setting a small weight for the prototype knowledge, the performance of node classification suffers from the most significant degradation, compared to the other two knowledge. To some extent, we could deem that the pivot prototype-based knowledge plays the most import role in the knowledge transferring. However, on the other hand, when we enlarge the weight of prototype-based knowledge, i.e., the red curve in Figure 3c, we could find the performance improves but then decreases. This is possibly because increasing the weight of the prototype negatively influences the other two types of knowledge. Regarding the regional features and prediction scores, both of them suffers from less impacts from their weights, but both of them tend to increase and then decrease, around the weight equal to 1. Therefore, based on the above analysis, we equally set the weights of the three knowledge as 1.

4.9. Embedding Space Visualization

We use t-SNE to reduce the embeddings to 2-dimensional vectors and randomly sample $1,000$ points. Figure 3a, 3b and 3c visualizes the embeddings of DELN, TD-DNE without hierarchical knowledge distillation and TD-DNE, respectively. Note that the same color dots denote the same category. Generally, we can make the observation that nodes of the same labels tend to cluster together in Figure 3c. However, the points in Figure 3a and 3b are much more tightly tangled, especially removing the hierarchical knowledge distillation strategy. This confirms that single-domain network embeddings do not perform well in domain-adaptive settings and our distillation technique can transfer the supervised semantic information to the new unlabeled domain.



(a) DELN

(b) TD-DNE without hierarchical knowledge distillation

(c) TD-DNE

Figure 3: A visualization of target domain's embeddings learned from $D \rightarrow A$.

5. Conclusion

In this paper, we focus on the discrete network embedding and propose a transferable and differentiable discrete network embedding (TD-DNE) method for multiple-domain networks. We mainly solve two problems in our work: how to transfer the supervision knowledge in the source domain to novel domains and how to reduce the performance degradation caused by the discretization. To this end, we propose the hierarchical knowledge distillation strategy to mitigate the knowledge gap of multiple domains and deploy multiple linear classifiers to generate discrete codes based on Gumbel-softmax. Evaluation on three benchmark datasets against a number of state-of-the-art discrete embedding methods demonstrates the superiority of TD-DNE on three tasks: node classification, link prediction and node recommendation. **Limitations**: we only investigate the closed-set multi-domains currently from paper citation networks. However, in practice, an network embedding method has to face versatile domain's data, such as Protein-Protein Interaction (PPI) dataset and Wikipedia, a co-occurrence network of words. It would be a challenging task for the model to support more diverse adaption scenarios, i.e., the open-set domain adaptation discrete network embedding. Hence, for the future work, we will focus more on the research question how to develop a generic model which can handle multiple open-set domains with large distribution shifts.

Acknowledgement

This material is based on research sponsored by DARPA under agreement number HR001122C0029. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

References

[1] A. Bojchevski and S. Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *ICLR*, 2018.

[2] Z. Cao, M. Long, J. Wang, and P. S. Yu. Hashnet: Deep learning to hash by continuation. In *Proceedings of the IEEE international conference on computer vision*, pages 5608–5617, 2017.

[3] H. Chen, Y. Wang, C. Xu, C. Xu, and D. Tao. Learning student networks via feature embedding. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):25–35, 2020.

[4] T. Chen, L. Li, and Y. Sun. Differentiable product quantization for end-to-end embedding compression. In *International Conference on Machine Learning*, pages 1617–1626. PMLR, 2020.

[5] Q. Dai, X. Shen, X.-M. Wu, and D. Wang. Network transfer learning via adversarial domain adaptation with graph convolution. *arXiv preprint arXiv:1909.01541*, 2019.

[6] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *ASCG*, pages 253–262. ACM, 2004.

[7] G. Fang, J. Song, C. Shen, X. Wang, D. Chen, and M. Song. Data-free adversarial distillation. *arXiv preprint arXiv:1912.11006*, 2019.

[8] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *ICCV*, pages 2960–2967, 2013.

[9] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. S. Lempitsky. Domain-adversarial training of neural networks. *JMLR*, 17:59:1–59:35, 2016.

[10] J. Gou, B. Yu, S. J. Maybank, and D. Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021.

[11] P. Goyal and E. Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018.

[12] T. He, L. Gao, J. Song, and Y.-F. Li. Semisupervised network embedding with differentiable deep quantization. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[13] T. He, L. Gao, J. Song, X. Wang, K. Huang, and Y. Li. Sneq: Semi-supervised attributed network embedding with attention-based quanti-

sation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4091–4098, 2020.

[14] T. He, Y.-F. Li, L. Gao, D. Zhang, and J. Song. One network for multi-domains: Domain adaptive hashing with intersectant generative adversarial network. *IJCAI*, 2019.

[15] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[16] L. Hu, M. Kan, S. Shan, and X. Chen. Duplex generative adversarial network for unsupervised domain adaptation. In *CVPR*, June 2018.

[17] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *ICLR*, 2017.

[18] J. Kim, S. Park, and N. Kwak. Paraphrasing complex network: Network compression via factor transfer. In *NIPS*, 2018.

[19] J. N. Kundu, N. Venkat, A. Revanur, R. V. Babu, et al. Towards inheritable models for open-set domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12376–12385, 2020.

[20] D. Lian, K. Zheng, V. W. Zheng, Y. Ge, L. Cao, I. W. Tsang, and X. Xie. High-order proximity preserving information network hashing. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1744–1753, 2018.

[21] Y. Lu, X. Wang, C. Shi, P. S. Yu, and Y. Ye. Temporal network embedding with micro-and macro-dynamics. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 469–478, 2019.

[22] P. Panareda Busto and J. Gall. Open set domain adaptation. In *Proceedings of the IEEE international conference on computer vision*, pages 754–763, 2017.

[23] W. Park, D. Kim, Y. Lu, and M. Cho. Relational knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3967–3976, 2019.

[24] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[25] X. Shen, S. Pan, W. Liu, Y.-S. Ong, and Q.-S. Sun. Discrete network embedding. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 3549–3555, 2018.

[26] R. Shu and H. Nakayama. Compressing word embeddings via deep compositional code learning. *ICLR*, 2018.

[27] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 4080–4090, 2017.

[28] J. Song, T. He, L. Gao, X. Xu, A. Hanjalic, and H. T. Shen. Binary generative adversarial networks for image retrieval. In *AAAI*, 2018.

[29] J. Song, H. Zhang, X. Wang, M. Xue, Y. Chen, L. Sun, D. Tao, and M. Song. Tree-like decision distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13488–13497, 2021.

[30] S. Sun, Y. Cheng, Z. Gan, and J. Liu. Patient knowledge distillation for bert model compression. In *ACL*, 2019.

[31] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su. Arnetminer: extraction and mining of academic social networks. In *ACM SIGKDD*, pages 990–998. ACM, 2008.

[32] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. In *CVPR*, pages 2962–2971, 2017.

[33] J. Wang, T. Zhang, N. Sebe, H. T. Shen, et al. A survey on learning to hash. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(4):769–790, 2018.

[34] X. Wang, Y. Shi, and K. M. Kitani. Deep supervised hashing with triplet labels. In *ACCV*, pages 70–84. Springer, 2016.

[35] M. Wu, S. Pan, C. Zhou, X. Chang, and X. Zhu. Unsupervised domain adaptive graph convolutional networks. In *WWW*, pages 1457–1467, 2020.

[36] M. Wu, S. Pan, C. Zhou, X. Chang, and X. Zhu. Unsupervised domain adaptive graph convolutional networks. In *Proceedings of The Web Conference 2020*, pages 1457–1467, 2020.

[37] W. Wu, B. Li, L. Chen, and C. Zhang. Efficient attributed network embedding via recursive randomized hashing. In *IJCAI*, volume 18, pages 2861–2867, 2018.

[38] H. Xuan, A. Stylianou, X. Liu, and R. Pless. Hard negative examples are hard, but useful. In *European Conference on Computer Vision*, pages 126–142. Springer, 2020.

[39] H. Yang, L. Chen, M. Lei, L. Niu, C. Zhou, and P. Zhang. Discrete embedding for latent networks. In *IJCAI*, pages 1223–1229, 2020.

[40] H. Yang, S. Pan, P. Zhang, L. Chen, D. Lian, and C. Zhang. Binarized attributed network embedding. In *ICDM*, pages 1476–1481. IEEE, 2018.

[41] Y. Yang, H. Chen, and J. Shao. Triplet enhanced autoencoder: model-free discriminative network embedding. In *AAAI*, pages 5363–5369, 2019.

[42] L. Yu, W. Zhang, J. Wang, and Y. Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

[43] D. Zhang, J. Yin, X. Zhu, and C. Zhang. Network representation learning: A survey. *IEEE transactions on Big Data*, 2018.

[44] H. Zhang, F. Shen, W. Liu, X. He, H. Luan, and T.-S. Chua. Discrete collaborative filtering. In *ACM SIGIR*, pages 325–334. ACM, 2016.

[45] M. Zhang and Y. Chen. Link prediction based on graph neural networks. In *NIPS*, pages 5165–5175, 2018.

[46] Y. Zhang, G. Song, L. Du, S. Yang, and Y. Jin. Dane: Domain adaptive network embedding. *IJCAI*, 2019.

[47] J. Zhao, J. Li, S. Tu, and J. Feng. Multi-prototype networks for unconstrained set-based face recognition. In *IJCAI*, 2019.

[48] D. Zhu, P. Cui, D. Wang, and W. Zhu. Deep variational network embedding in wasserstein space. In *ACM SIGKDD*, pages 2827–2836, 2018.