MODIFIED DOMAIN-AWARE CURRICULUM GRAPH CO-TEACHING

Ayoub Benabbou

Télécom Paris

ABSTRACT

Deep learning models often face challenges in generalizing effectively to real-world scenarios due to domain shifts between training (source) and test (target) data distributions. Unsupervised Domain Adaptation (UDA) aims to mitigate this issue by leveraging both source and target data to train robust predictors[1]. While traditional UDA methods focus on single-source to single-target adaptation (STDA), practical applications increasingly require adaptation to multiple target domains, leading to the development of Multi-target Domain Adaptation [2]. This study examines the impact of mini-batch size on the performance of the D-CGCT method, observing that optimal classification by Graph Convolutional Networks (GCNs)[3] depends on the batch size relative to the number of classes. We propose an enhancement to Domain-Aware Curriculum Graph Co-teaching for the Office-Home[4] dataset, addressing memory limitations by concatenating multiple batches in CPU memory. This approach simulates a larger batch size, improving GCN classifier performance.

Index Terms— Domain Adaptation, Graph Neural Networks, Office Home, DomainNet

1. INTRODUCTION

1.1. Context

Deep learning models often struggle with limited ability to generalize effectively in real-world deployments. This limitation arises due to the disparity between the distributions of the training data (also known as the source domain) and the test data (also known as the target domain), commonly referred to as domain- shift. Collecting labeled data for every new operating environment can be impractical, leading to the development of a field of research known as Unsupervised Domain Adaptation (UDA) [1]. UDA aims to address this challenge by leveraging the source data as well as the target data to train a robust predictor that can perform well on the desired target domain.

The objective of MTDA [2]is to learn more concise representations using a single predictor that performs well across all target domains. We also add the assumption that the distributions are different between the source domain and the individual target domains, but also among multiple target domains .

1.2. Motivation

UDA methods have primarily focused on adapting from a single source domain to a single target domain (STDA). However, with the increasing availability of unlabeled data, the relevance of adapting to just a single target domain has diminished in real-world scenarios [2]. As the number of target domains grows, the need to train a corresponding number of models scales linearly. Consequently, recent research has shifted towards addressing a more practical scenario: adapting simultaneously to multiple target domains from a single source domain, formally known as Multi-target Domain Adaptation (MTDA)[5, 6].

1.3. What we did

In this study, we observe the effect the number of images in a mini-batch has over the performance of the method D-CGCT. We make the observation that for the GCN to make good classification, the number of samples in each mini-batch should depend on the number of classes in the dataset. Namely, the size of the batch fed to the GCN should grow with the number of classes.

1.4. Contribution

In this paper, we propose an improvement of the method called Domain-Aware Curriculum Graph Co-teaching on the Office-Home dataset, that overcomes the limitation of memory: by concatenating multiple batches in the cpu memory, we are able to simulate doubling the batch size, improving the performance of the GCN classifier.

2. RELATED WORK

2.1. STDA

Single Target Domain Adaptation (STDA) addresses the challenge of adapting a model trained on a labeled source domain to perform well on an unlabeled target domain. This approach is crucial when labeled data is abundant in the source domain but scarce or unavailable in the target domain. STDA techniques often employ methods such as domain adversarial training, feature alignment, and self-training to bridge the gap between domains. Notable advancements in STDA include adversarial methods like Domain-Adversarial Neural Networks (DANN), which minimize the domain discrepancy by learning domain-invariant features, and self-training methods that iteratively refine the model by generating pseudo-labels for the target domain. These methods have demonstrated significant improvements in various tasks, such as image classification and object detection, by effectively leveraging the underlying structure of the target domain data without requiring explicit labels. However, practical applications go beyond the single-source and single-target setting and often involve multiple source or target domains, necessitating more sophisticated adaptation techniques to handle the increased complexity and variability.

2.2. MTDA

Applying Single Target Domain Adaptation (STDA) methods to MTDA can be suboptimal due to multiple domain shifts. MTDA requires aligning multiple data distributions, which is more challenging but essential for successful adaptation. One notable approach is Multi-Teacher MTDA through knowledge distillation, where expert models trained on each target domain collaborate to teach a single domain-generic student model [7] . The student learns from specialized teacher models using STDA techniques, capturing performance through knowledge distillation. This method was also applied to image segmentation in [5], highlighting its effectiveness in unsupervised settings with limited labeled data. Another strategy is Curriculum Graph Co-Teaching, which incrementally improves generalization by incorporating target samples progressively, ordered by difficulty [2, 6]. This approach uses Graph Neural Networks (GNNs) to propagate labels from the source to target samples, enhancing label refinement and model training through a co-teaching framework involving MLP classifiers and GNNs.

2.3. GNNs

Graph Neural Networks (GNNs) are designed for processing data with an underlying graph structure. One fundamental architecture is Graph Convolutional Networks (GCNs), introduced by Kipf and Welling [3], which adapt convolutional operations to graph data using layer-wise propagation rules based on matrix multiplication to aggregate features from neighboring nodes. Another structure is Graph Attention Networks (GATs) [8], which use attention coefficients instead of adjacency matrices for feature aggregation, allowing nodes to attend to each other dynamically. Lastly, Graph Learning-Convolutional Networks (GLCNs) [9] address cases where input data is not graph-structured or contains noise by first learning a better graph structure before applying GCNs. This can be done through an end-to-end framework that iteratively refines both the graph structure and GNN parameters [10].

2.4. How to position this work regarding the existing literature

When dealing with large datasets with many classes, traditional Unsupervised Domain Adaptation (UDA) methods that focus on single-source to single-target adaptation (STDA) become impractical due to the need for multiple models. Our work addresses this by focusing on Multi-Target Domain Adaptation (MTDA), which aims to create a single predictor that performs well across multiple target domains.

We enhance the Domain-Aware Curriculum Graph Co-teaching (D-CGCT) method by concatenating multiple mini-batches in CPU memory, simulating a larger batch size. This improves the probability of including samples from the same class within a mini-batch, enhancing graph construction and feature propagation in the GCN classifier. Our approach demonstrates improved performance on the Office-Home dataset and provides a scalable solution for complex, multi-domain environments, advancing the state-of-the-art in MTDA.

3. METHODOLOGY

3.1. Problem Statement

In MTDA scenario, we have a source dataset with N_s labeled samples $S = \{x_i^{(s)}, y_i^{(s)}\}_{i=1}^{N_s}$, and multiple datasets, called target datasets $T = \{T_j\}_{j=1}^{N}$, where $T_j = \{x_i^{(t_j)}\}_{i=1}^{N_j}$ containing N_j unlabeled samples. We assume that the label space is the same for the source domain and the target domains. And the goal is to train a single classifier for all the target domains using the data in $S \bigcup \{T_j\}_{j=1}^{N}$. Finally we denote by n_c, the number of classes.

3.2. Overview of the method

The Domain-Aware Curriculum graph co-teaching method relies on two key concepts: feature aggregation and curriculum learning. In order to lea

Our method addresses the challenge of Multi-Target Domain Adaptation (MTDA) by enhancing the Domain-Aware Curriculum Graph Co-teaching (D-CGCT) approach. The goal is to improve generalization across multiple target domains with a single predictor. The key components of our method are as follows:

1. Incremental Learning Approach: The model first adapts to easier target domains, progressively incorporating their samples as new source samples. This reduces domain shifts incrementally, facilitating better adaptation to more challenging target domains.

2. Graph Construction: During the adaptation stage, we construct a graph using both source and target samples. We sample mini-batches of size BB and compute the probability that two samples in a batch belong to the same class, ensuring effective feature propagation.

3. Batch Concatenation: To address memory limitations and enhance performance, we concatenate multiple mini-batches in CPU memory. This effectively simulates a larger batch size, improving the GCN classifier's ability to propagate accurate information between samples of the same class.

4. GCN and Feature Propagation: The GCN classifier utilizes a refined affinity matrix, encoded by f_{edge}, to capture relationships between nodes. These coefficients are used by f_{node} to refine node labels, leveraging each node's neighborhood for better classification.

3.3. Detailed Explanation

3.3.1. Preliminaries

In this subsection we will present, the different architectures that compose the framework for this method.

Domain Adversarial Network(DANN): used for feature distribution alignment by using adversarial training comprises of three networks:

- Feature extractor: $F_\theta : R^{3x\omega xh} \rightarrow R^d$, with parameter θ, that for each image x, outputs a feature vector, $f = F_\theta(x)$.
- Classifier network: $G_\phi : R^d \rightarrow R^{n_c}$, with parameter ϕ, outputs class logits $g = G_\phi(f)$.
- Domain Discriminator network: $D_\psi : R^d \rightarrow R$, with parameter ψ, takes the feature vector and outputs a single logit.

We combine all the target domains as one target domain and write the overall loss for DANN in MTDA:

$max_\psi min_{\phi,\theta} l_{ce} - \lambda_{adv} l_{adv}$,

where:

$$l_{ce} = -\mathbf{E}_{(x_i^{(s)}, y_i^{(s)}) \sim \mathcal{S}}[y_i^{(s)} log(G_\phi(F_\theta(x_i^{(s)}))],$$

is the cross-entropy loss,

and: $l_{adv} = -\mathbf{E}_{(x_i^{(s)}) \sim \mathcal{S}}[log(D_\psi(F_\theta(x_i^{(s)})) - E_{(x_i^{(t)}) \sim \mathcal{T}}[log(1 - D_\psi(F_\theta(x_i^{(t)}))]$ is the adversarial loss.

Graph Convolutional Networks: In this framework two different GCNs are used each of them represent a different aspect of the information: f_{node} ans f_{edge}. The former will encodes the nodes features by feature aggregation using the affinity matrix provided by the latter. Indeed, f_{edge} is the GCN that encodes the relationship between nodes, and provides a more refined affinity matrix for the f_{node} to give more robust node features.

3.3.2. Modified D-CGCT

The framework,as described above, consists of a feature extractor F_θ, a domain discriminator D, an mlp classifier G_ϕ, and a GNN classifier G_{GCN}, which consists of two gcn networks f_{edge} and f_{node}, parameterized respectively by, Φ and Φ'. The idea of the method called Domain-Aware Curriculum Graph Co-Teaching is to progressively improve the model's generalization ability to the different target domains by incorporating more target samples in each step, in an incremental learning approach, processing one target domain at a time.

Domain Selection:

After being pre-trained on the source domain, at each step, we choose the target domain that we adapt the model on by an easy-to-hard strategy. In other words, processing target domains in an increasing order of difficulty. The difficulty is measured by the dissimilarity to the source domain. This approach is justified by the following observation: different target domains may present different domain shifts from the source domain, some domain shifts larger than others, the classifier will find it easier then to adapt to those target domains that are closest in this sense.

Explicitly, the dissimilarity of a target domain T_j to the source domain S is measured by the uncertainty in the target predictions with a source trained model

$$H(T_j) = -E_{x_{tj,k} \sim T_j} \left[\sum_{c=1}^{n_c} p(\hat{y}_{tj,k,c} | x_{tj,k}) \log p(\hat{y}_{tj,k,c} | x_{tj,k}) \right].$$

Thanks to this incremental approach, the intermediate target domains help reduce the largest domain shifts. When the more difficult target domains are treated, the model has already reduced the domain shift by adapting to the less difficult domains. Additionally, it incorporates some of their samples as new source samples.

Denote: $S^0 = \{S\}$ and $T^0 = \{T_j\}_{j=1}^N$. So the first step is choosing the target domain to be processed as :

$$D_0 = \arg\min_j \left\{ H_j(T_j) : T_j \in T^0 \right\}.$$

Adaptation Stage:

Once the target domain has been chosen, we can start adapting the network on that particular domain.

We choose batches $(\hat{\mathcal{B}}_s)$ and $(\hat{\mathcal{B}}_t)$ respectively from S and D_0 each of size B. These batches are first fed to the feature extractor F, to obtain the features.

We represent the data in $\mathcal{B} = (\hat{\mathcal{B}}_s) \bigcup (\hat{\mathcal{B}}_t)$ by a graph $G = (N, E, A)$, where N is the set of feature vectors of images represented by nodes ν_i for $f_i = F(x_i)$, E is the set of edges, and A is the affinity matrix. This graph is the basis for feature aggregation and label propagation, that will allow us to propagate information from the labeled samples that come from the source batch, to the not yet labeled samples coming from the target batch.

$\hat{a_{i,j}}^{(l)} = g_{edge}^{(l)}(\nu_i^{(l-1)}, \nu_j^{(l-1)})$, with $\nu_i^{(0)} = f_i$.

The obtained matrix $\hat{A}^{(l)}$ is then normalized after adding self-connections, as follows:

$$A^{(l)} = \mathcal{M}^{-1/2}(\hat{A}^{(l)} + I)\mathcal{M}^{-1/2},$$

where $\mathcal{M}$ is the degree matrix.

The affinity matrix $A^{(l)}$ is then used by f_{node} for feature aggregation as follows:

$$\nu_i^{(l)} = g_{node}^{(l)}([\nu_i^{(l-1)}, \sum_{j\in\mathcal{B}} a_{i,j}^{(l)}\nu_j^{(l-1)}])$$

where $g_{node}^{(l)}$ and $g_{node}^{(l)}$ are non linearity function of the l-th layers of the networks f_{node} and f_{node} respectively, and $[.,.]$ is feature concatenation function. The final layer $g_{node}^{(L)}$ is the output layer for G_{GNN} with n_c output logits.

We can gain intuition of the G_{GCN} network as follows: f_{edge} encodes the relationship between nodes and provides a refined affinity matrix. The coefficients of the affinity matrix are used by f_{node} to refine the node labels. This process takes into account each node's neighborhood.

The G_{GCN} gives a global scale prediction based on the entire batch $\mathcal{B}$, whereas G_{mlp} gives prediction based solely on individual feature vector of an image. The two classifiers capture different aspects of information, and that's why we have them co-operate in Co-Teaching framework.

we denote:

$$\hat{y} = softmax(G_{mlp}(F(x))),$$

the prediction given by the mlp classifier

$$\bar{y} = softmax(G_{GCN}(F(x))),$$

the prediction given by the gcn classifier

There are two information flows:

1. MLP$\longrightarrow$GCN: the goal of f_{edge} is to build an affinity matrix which is going to be used by f_{node} for feature aggregation. To do that, we take the predictions given by MLP, and consider them as pseudo-labels for the unlabeled target samples. And we construct the following $\hat{\mathcal{A}}^{tar}$matrix: $\hat{a}_{i,j}^{tar} = 1$ if $y_i = y_j$, and 0 otherwise. And learn the affinity matrix $\hat{\mathcal{A}}$ by minimizing the binary-cross entropy loss l_{bce}^{edge} between elements of the current affinity matrix and those of $\hat{\mathcal{A}}^{tar}$.

$$l_{bce}^{edge} = \hat{a}_{i,j}^{tar}\log p(\hat{a}_{i,j}) + (1 - \hat{a}_{i,j}^{tar})\log(1 - p(\hat{a}_{i,j})).$$

2. GCN$\longrightarrow$MLP : denote $\omega_j = maxp(\bar{y}_{t,j} = c|x_{t,j})$. If $\omega_j \geq \tau$ where τ is a certain threshold, the target sample $x_{t,j}$, as of now labeled by $y_{t,j} = c$, is added to the source domain, and in the next iteration, the mlp will include it in its training set, in the loss:

$$l_{bce}^{mlp} = -\frac{1}{|B_s^0|}\sum_{i=1}^{|B_s^0|}\tilde{y}_i \log p(\hat{y}_{s,i}|\hat{g}_{s,i}^0),$$

with $\tilde{y}_i$ being the real labels. and finally f_{node} is trained by the loss:

$$l_{bce}^{node} = -\frac{1}{|B_s^0|}\sum_{i=1}^{|B_s^0|}\tilde{y}_i \log p(\bar{y}_{s,i}|\bar{g}_{s,i}^0).$$

The overall loss is then :

$$max_{\psi}min_{\phi,\theta,\Phi,\Phi'}l_{bce}^{mlp} - \lambda_{adv}l_{adv} + \lambda_{edge}l_{bce}^{edge} + \lambda_{node}l_{bce}^{node}.$$

Furthermore, we make the observation that when the number n_c is high, the number of samples in each batch should be sufficient so that each sample can find other samples in the graph that are from the same class. For the Office-Home dataset, we consider a batch size of $B = 64$, since the number of classes is 65. We give an explanation for this choice in the section 4.3.

Pseudo-labeling stage:

Once the adaptation on the target domain D_q is done, we update the source domain and the set of target domains as follows:

$$S^{q+1} = S^q \cup \mathbf{D}_t^{D_q}$$

where $\mathbf{D}_t^{D_q} = \{x_{t,j} \in D_q|\omega_j = maxp(\bar{y}_{t,j} = c|x_{t,j}) \geq \tau\}$, in other words the target samples that satisfied the condition to join the source domain, as labeled samples.

$$T^{q+1} = T^q \setminus T_{D_q}$$

These three stages are repeated until all the N target domains have been processed.

4. EXPERIMENTS AND RESULTS

4.1. Dataset

In this work we focus on the Office-Home dataset, which is a standard Domain Adaptation dataset. It contains 15500 images, 4 domains, and 65 classes.

For our experiments, we took the domain 'art' as the source domain, while the remaining domains, 'clipart', 'product' and 'real' were the target domains.

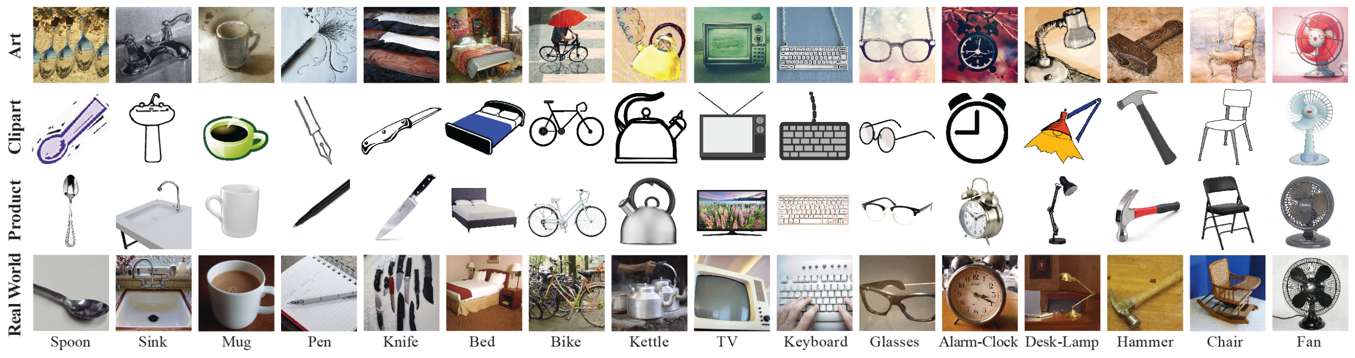

Fig. 1. Sample images from the Office-Home dataset. The dataset consists of images of everyday objects organized into 4 domains; Art: paintings, sketches andor artistic depictions, Clipart: clipart images, Product: images without background and Real-World: regular images captured with a camera. The figure displays examples from 16 of the 65 categories [4]

4.2. Experimental Framework

To evaluate the performance of the model, we use the classification accuracy, which is computed as the average accuracy of the G_{mlp} on all the target domains. We only consider G_{mlp} since G_{gcn} requires a whole batch .

Hyperparameter selection: In our final model we used only a single set of hyperparameters, which are λ_{edge} = 1, λ_{node} = 0.3, λ_{adv} = 1 and τ = 0.7.

4.3. Results and discussion(table)

Model	Office-home Accuracy
MT-MTDA	64.6
D-CGCT	70.5
Ours	70.7

Table 1. Performance comparison of different models on Office-home and DomainNet datasets.

Let's suppose the target domain being processed is T^q, its cardinal is $M = \sum_{k=1}^{n_c} M_k$ where M_k is the number of samples in T^q that are of class k. We suppose that all the classes have the same cardinality so $M = n_c M_1$. At the adaptation stage, we want to construct the graph from the source and target samples. We first sample a mini-batch $\hat{\mathcal{B}_t^n}$ of size B. We are interested in the probability that in this batch, two samples are from the same class.
$A = \{\exists i, j : \tilde{y}_i = \tilde{y}_j = 1\}$,
$\mathbf{P}(A) = \frac{\mathrm{C}_{M_1}^2 \mathrm{C}_{M-2}^{B-2}}{\mathrm{C}_M^B} = \frac{M_1(M_1-1)B(B-1)}{M(M-1)}$.

If we suppose the number of images is high, as it is the case in the standard UDA datasets, we get:

$$\mathbf{P}(A) \simeq \frac{M_1^2 B(B-1)}{M^2} = \frac{B(B-1)}{n_c^2} \simeq (\frac{B}{n_c})^2 .$$

We can consider that if $\mathbf{P}(A)$ is high enough, then the graph will contain at least two samples of the same class, and the feature propagation will result in the propagation of some accurate information between these two samples, which is likely to improve the overall training.

Here, we have two cases:

- If $B \equiv n_c$, $\mathbf{P}(A) \equiv 1$.
- and if $B << n_c$ we have $\mathbf{P}(A) << 1$.

In the case of the Office-Home dataset, in [CVPR paper], B was chosen 32, so having $n_c = 65$, we get $\mathbf{P}(A) \simeq 0.25$. Whereas for the DomainNet dataset, n_c being 345, we get $\mathbf{P}(A) \simeq 0.008$, which can explain the poor accuracy of D-CGCT on DomainNet, which an average of 34%.
Additionally, an improvement to the method involves concatenating multiple batches in CPU memory to simulate a larger batch size. In the case of Office-Home, we chose to concatenate 2 batches.

5. CONCLUSION AND FUTURE WORK

5.1. Summary

Our method improves Multi-Target Domain Adaptation (MTDA) by enhancing the Domain-Aware Curriculum Graph Co-teaching (D-CGCT) approach. We employ an incremental learning strategy where the model first adapts to easier target domains, progressively incorporating their samples as new source samples. This reduces domain shifts incrementally and aids in better adaptation to more challenging target domains.

To construct the graph for adaptation, we sample mini-batches and calculate the probability that two samples in a batch belong to the same class. We address memory limitations by concatenating multiple mini-batches in CPU memory, effectively simulating a larger batch size and improving the GCN classifier's performance.

The GCN classifier uses a refined affinity matrix, encoded by fedgefedge, to capture relationships between nodes. These coefficients are then used by fnodefnode to refine node labels by considering each node's neighborhood, enhancing feature propagation and classification accuracy.

5.2. Future directions

Given the results of the experiments, and the observation regarding the batch size elaborated in section 4.3, it seems natural to test the method on the DomainNet[11] dataset, which is a very large scale dataset with 0.6 million images, 6 domains, and 345 classes. It would be interesting to investigate the performance of the method, with a batch size $B = 320$ for example.

6. REFERENCES

[1] Yaroslav Ganin and Victor Lempitsky, "Unsupervised domain adaptation by backpropagation," in *International conference on machine learning*. PMLR, 2015, pp. 1180–1189.

[2] Subhankar Roy, Evgeny Krivosheev, Zhun Zhong, Nicu Sebe, and Elisa Ricci, "Curriculum graph co-teaching for multi-target domain adaptation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 5351–5360.

[3] Thomas N Kipf and Max Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[4] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan, "Deep hashing network for unsupervised domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5018–5027.

[5] Takashi Isobe, Xu Jia, Shuaijun Chen, Jianzhong He, Yongjie Shi, Jianzhuang Liu, Huchuan Lu, and Shengjin Wang, "Multi-target domain adaptation with collaborative consistency learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 8187–8196.

[6] Sudipan Saha, Shan Zhao, and Xiao Xiang Zhu, "Multitarget domain adaptation for remote sensing classification using graph neural network," *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1–5, 2022.

[7] Le Thanh Nguyen-Meidine, Atif Belal, Madhu Kiran, Jose Dolz, Louis-Antoine Blais-Morin, and Eric Granger, "Unsupervised multi-target domain adaptation through knowledge distillation," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 1339–1347.

[8] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[9] Bo Jiang, Ziyan Zhang, Doudou Lin, Jin Tang, and Bin Luo, "Semi-supervised learning with graph learning-convolutional networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 11313–11320.

[10] Yu Chen, Lingfei Wu, and Mohammed Zaki, "Iterative deep graph learning for graph neural networks: Better and robust node embeddings," *Advances in neural information processing systems*, vol. 33, pp. 19314–19326, 2020.

[11] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang, "Moment matching for multi-source domain adaptation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1406–1415.