Graph Matching and Pseudo-Label Guided Deep Unsupervised Domain Adaptation

Debasmit Das(✉) and C. S. George Lee

School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA
{das35,csglee}@purdue.edu

Abstract. The goal of domain adaptation is to train a high-performance predictive model on the target domain data by using knowledge from the source domain data, which has different but related data distribution. In this paper, we consider unsupervised domain adaptation where we have labelled source domain data but unlabelled target domain data. Our solution to unsupervised domain adaptation is to learn a domain-invariant representation that is also category discriminative. Domain-invariant representations are realized by minimizing the domain discrepancy. To minimize the domain discrepancy, we propose a novel graph-matching metric between the source and target domain representations. Minimizing this metric allows the source and target representations to be in support of each other. We further exploit confident unlabelled target domain samples and their pseudo-labels to refine our proposed model. We expect the refining step to improve the performance further. This is validated by performing experiments on standard image classification adaptation datasets. Results showed our proposed approach out-perform previous domain-invariant representation learning approaches.

Keywords: Unsupervised domain adaptation · Transfer learning
Graph matching · Pseudo-labels

1 Introduction

Unsupervised Domain Adaptation (UDA) defines the problem when the target domain is unlabelled and the source domain is fully labelled and these domains have different marginal distributions [15]. UDA tries to transfer knowledge from a source domain to help learning in a target domain. The assumption in UDA for the classification problem is that the source and target categories are the same. Because of shifting distributions and the lack of annotations, machine learning models trained in the source domain will fail to perform well in the target domain and hence UDA is necessary.

Most popular domain-adaptation methods involve feature transformation. Among these methods, asymmetric feature-based methods transform the features of one domain to more closely match another domain [3,10]. Symmetric

© Springer Nature Switzerland AG 2018
V. Kůrková et al. (Eds.): ICANN 2018, LNCS 11141, pp. 342–352, 2018.
https://doi.org/10.1007/978-3-030-01424-7_34

methods on the other hand transform the source and target domains to a common latent space where the distribution discrepancy is minimized. Deep-learning-based domain adaptation methods allow symmetric feature-based methods to be included in the form of learning a domain-invariant representation [7,13]. Among these methods, minimizing the maximum mean discrepancy (MMD) [9] is common. MMD is a non-parametric metric that measures the distribution divergence between the mean embeddings of two distributions in reproducing kernel Hilbert space (RKHS), and MMD has been used as a domain discrepancy metric between the deep activations of the source and target domains [13,20]. On the other hand, the correlation alignment (CORAL) method [17] aligns the covariances of the source and target distributions. They also extended their work to learn representations that align correlations of features extracted from the deep neural network [18]. A different class of symmetric feature-based methods uses an adversarial objective to reduce domain discrepancy. Domain adversarial neural network (DANN) [7] was proposed for learning domain-invariant representations by forcing a minimax game between the domain discriminator and the feature extractor. Tzeng et al. [19] generalized the idea of adversarial adaptation by choosing adversarial loss for the domain classifier and also proposed a weight sharing strategy. Shen et al. [16] also considers an adversarial adaptation method where it minimizes the empirical Wasserstein distance between source and target features. Previous work on using graph-matching on hand-crafted features for unsupervised domain adaptation was also proposed [4,5].

Our proposed method is a symmetric feature transformation method where both the source and target samples are transformed to a common space using the feature extractor of a deep neural network. This is done by carrying out domain-invariant representation learning that uses graph-matching (GM) loss as the domain discrepancy metric. The graph-matching loss considers the cost of matching the source and target graphs constructed from the corresponding representations. The matching consists of both node-to-node matching and edge-to-edge matching between the source and target representation graphs. This second-order matching of edges provides additional structural and geometric information about the representations that are absent on just using the first-order information [16]. The feature extraction network is iteratively optimized to minimize this graph matching loss along with minimizing the mis-classification loss using the source domain labelled data. Our proposed method adopts an iterative adversarial training scheme where the adversarial loss is a combination of first-order and second-order graph-based matchings between the source and target domain features. It is important to note that our matching approach is local and it considers matching between each instance of the source and target domain representations. On the other hand, methods like CORAL [18] and those based on MMD [13,20] are global moment-matching methods that match statistics of the source and target feature distributions.

After the learning has converged and the source and target representations lie in support of each other, we perform an additional refinement of the model. The pseudo-labels (PL) of the confident unlabelled target domain data are used to make sure that target samples lie further from the softmax decision boundary.

This allows better generalization to unseen target samples. Finally, to validate our approach, we perform experiments on standard domain adaptation datasets for image classification.

2 Proposed Approach

2.1 Problem Definition

For the unsupervised domain adaptation problem, we have n^s labelled samples, $\mathbf{X}^s = \{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^{n^s}$ from the source domain $\mathcal{D}^s$. We also have n^t unlabelled samples $\mathbf{X}^t = \{\mathbf{x}_i^t\}_{i=1}^{n^t}$ from the target domain $\mathcal{D}^t$. We assume that the domains share the same feature and label space but follow different marginal data distributions; that is, $P(\mathbf{X}^s) \neq P(\mathbf{X}^t)$. The goal is to learn a transferable classifier $\mathbf{K}(\cdot)$ and a representation $\phi(\cdot)$ to minimize the target risk $\epsilon_t = P_{(\mathbf{x},y)\sim D_t}[\mathbf{K}(\phi(\mathbf{x})) \neq y]$.

2.2 Minimizing Domain Discrepancy with Graph Matching

Our goal is to learn domain-invariant representations by minimizing a graph matching loss between the source and target representations. In our case, we realize feature extraction using a neural network. We force the feature extractor to learn domain-invariant representations. Given an input sample $\mathbf{x} \in \mathbb{R}^n$ from a domain, the feature extractor learns a function $\phi : \mathbb{R}^n \rightarrow \mathbb{R}^d$ that maps an instance to a d-dimensional feature space. The parameters of the feature extractor can be represented by Θ_F. In order to minimize the discrepancy between the source and target domains, we minimize the graph matching loss between the source and target representations. To encounter excess discrepancy between the source and target domains, we allow an additional affine transformation on the source domain representations. Thus, we have a modified source domain representation $\phi'(\cdot)$ such that $\phi'^T(\mathbf{x}^s) = \phi^T(\mathbf{x}^s)\mathbf{W}_{map} + \mathbf{b}_{map}^T$, where $\mathbf{W}_{map} \in \mathbb{R}^{d\times d}$ and $\mathbf{b}_{map} \in \mathbb{R}^d$ are scaling matrix and bias, respectively. Superscript T indicates the transpose operation. The graph-matching loss considers minimizing a combination of first and second-order matching cost between graphs constructed from the source and target domains. So, if a mini-batch contains n_b^s, n_b^t source and target samples respectively, we represent the matching between the source and target representations through a matching matrix $\mathbf{C} \in \mathbb{R}^{n_b^s \times n_b^t}$. An element $[\mathbf{C}]_{ij}$ is a measure of matching between mini-batch source sample i and mini-batch target sample j. The source mini-batch features can be stacked to form a matrix $\boldsymbol{\Phi}^s \in \mathbb{R}^{n_b^s \times d}$. Similarly, the target mini-batch features are stacked to form $\boldsymbol{\Phi}^t \in \mathbb{R}^{n_b^t \times d}$. Accordingly, for the first-order matching we want the corresponding target representation to be close to the corresponding mapped source representation. Mathematically, this implies minimizing $||\mathbf{C}\boldsymbol{\Phi}^t - \boldsymbol{\Phi}'^s||_F^2$ where $\boldsymbol{\Phi}'^s$ is the modified source domain feature matrix after affine transformation on $\boldsymbol{\Phi}^s$ and $||\cdot||_F$ is the Frobenius norm. For the second-order matching, we try to minimize the discrepancy between the adjacency matrix of graphs constructed using the source and target mini-batches. Mathematically, this implies minimizing $||\mathbf{C}\mathbf{D}^t - r\mathbf{D}^s\mathbf{C}||_F^2$, where $\mathbf{D}^t \in \mathbb{R}^{n_b^t \times n_b^t}$ and $\mathbf{D}^s \in \mathbb{R}^{n_b^s \times n_b^s}$ are adjacency

matrices constructed from Φ^t and Φ^s, respectively. We use the dot product for the similarity measure of the adjacency matrices and consequently $\mathbf{D}^t = \Phi^t \Phi^{tT}$ and $\mathbf{D}^s = \Phi^s \Phi^{sT}$, with diagonals set to 0. $r = \frac{n_b^t}{n_b^s}$ is a correction factor to account for the difference in the size of the source and target mini-batches. In addition, the constraints on $\mathbf{C}$ are as follows: $\mathbf{C} \geq \mathbf{0}, \quad \mathbf{C}\mathbf{1}_{n_b^t} = \mathbf{1}_{n_b^s}$ and $\mathbf{C}^T\mathbf{1}_{n_b^s} = (\frac{n_b^s}{n_b^t})\mathbf{1}_{n_b^t}$. The equality constraint $\mathbf{C}\mathbf{1}_{n_b^t} = \mathbf{1}_{n_b^s}$ implies that the sum of the correspondences of all target samples to each source sample is one. The second equality constraint $\mathbf{C}^T\mathbf{1}_{n_b^s} = (\frac{n_b^s}{n_b^t})\mathbf{1}_{n_b^t}$ implies that the sum of correspondences of all source samples to each target sample should increase proportionately by $\frac{n_b^s}{n_b^t}$ to allow for multiple correspondences. Accordingly the optimization problem becomes

$$\min_{\mathbf{C},\mathbf{W}_{map},\mathbf{b}_{map}} \mathcal{L}_{0GM} = \frac{1}{(n_s d)}||\mathbf{C}\Phi^t - \Phi'^s||_F^2 + \lambda_s||\mathbf{C}\mathbf{D}^t - r\mathbf{D}^s\mathbf{C}||_F^2$$
$$s.t. \quad \mathbf{C} \geq \mathbf{0}, \quad \mathbf{C}\mathbf{1}_{n_b^t} = \mathbf{1}_{n_b^s}, \quad \mathbf{C}^T\mathbf{1}_{n_b^s} = (\frac{1}{r})\mathbf{1}_{n_b^t} \tag{1}$$

In the context of training neural networks, the above optimization problem can be solved using the projected gradient descent, where each iterate is projected onto the constraint set. Training neural networks generally requires a lot of time and further projection might increase the time complexity. As a result, we propose to reformulate the equality constraints as penalties in addition to the cost function. Thus our optimization problem becomes

$$\min_{\mathbf{C},\mathbf{W}_{map},\mathbf{b}_{map}} \mathcal{L}_{GM} = \frac{1}{(n_s d)}||\mathbf{C}\Phi^t - \Phi'^s||_F^2 + \lambda_s||\mathbf{C}\mathbf{D}^t - r\mathbf{D}^s\mathbf{C}||_F^2$$
$$+ \lambda_p(||\mathbf{C}\mathbf{1}_{n_b^t} - \mathbf{1}_{n_b^s}||_2^2 + ||\mathbf{C}^T\mathbf{1}_{n_b^s} - (\frac{1}{r})\mathbf{1}_{n_b^t}||_2^2) \quad s.t. \quad \mathbf{C} \geq \mathbf{0}, \tag{2}$$

where λ_p weighs the penalty terms. As a result, we can carry out gradient descent on $\mathcal{L}_{GM}$ and project it onto the set of positive matrices after each iteration.

In addition, we can exploit the labels of the source domain data to build a classifier on top of the feature extractor. We can add several layers as the classifier on top of the feature extraction network. Since the graph-matching loss ensures transferability of the learned representations, the shared classifier can be directly applied to the target domain. The objective of the classifier $\mathbf{K}(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^l$ is to compute softmax prediction for the l classes. Let us denote the parameters of the classifier as Θ_K. The classifier loss function is the cross-entropy between the predicted probabilistic distribution and one-hot encoding of the class labels:

$$\mathcal{L}_c(\mathbf{x}^s, y^s) = -\frac{1}{n_b^s}\sum_{i=1}^{n_b^s}\sum_{k=1}^{l} 1(y_i^s = k)\log(\mathbf{K}(\phi'(\mathbf{x}_i^s))_k) \tag{3}$$

where $1(y_i^s = k)$ is a 0-1 indicator function and $\mathbf{K}(\phi'(\mathbf{x}_i^s))_k$ corresponds to the k^{th} dimension value of the softmax output. Thus, the classification loss is combined with the graph matching loss to obtain the following objective function

$$\min_{\Theta_F,\Theta_K} \{\mathcal{L}_c + \lambda \min_{\mathbf{C}\geq\mathbf{0},\mathbf{W}_{map},\mathbf{b}_{map}} [\mathcal{L}_{GM}]\} \tag{4}$$

where λ is the coefficient controlling the balance between classification and graph matching loss. Note that the minimization is carried out using mini-batch gradient descent. As described in Algorithm 1, using a mini-batch containing labelled source data and unlabelled target data, $\mathcal{L}_{GM}$ is optimized with respect to $\mathbf{C}$ and after that iteratively projecting onto positive matrices. After the optimized matching matrix $\mathbf{C}^*$ is obtained, we solve for $\mathbf{W}_{map}, \mathbf{b}_{map}$, for which a closed form solution exists. The solution for $\mathbf{W}_{map}$, $\mathbf{b}_{map}$ can be obtained as follows:

$$\begin{bmatrix} \mathbf{W}_{map} \\ \mathbf{b}_{map}^T \end{bmatrix} = \left[\frac{1}{n_b^s d} \begin{bmatrix} \Phi^{sT} \\ \mathbf{1}^T \end{bmatrix} [\Phi^s \ \mathbf{1}] + \frac{\lambda_w}{d^2} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} \right]^{-1} \left[\frac{1}{n_b^s d} \begin{bmatrix} \Phi^{sT} \\ \mathbf{1}^T \end{bmatrix} \mathbf{C}^* \Phi^t + \frac{\lambda_w}{d^2} \begin{bmatrix} \mathbf{I} \\ \mathbf{0}^T \end{bmatrix} \right] \quad (5)$$

Here λ_w regularizer is introduced to allow for a smooth mapping transformation. Subsequently, we optimize for the total loss as in Eq. (4) with respect to the parameters of the feature extractor and the classifier. The learned representations are domain invariant as well as target discriminative since the feature extractor parameter Θ_F receives gradients from both the graph matching and classification loss. The overall framework of our method is given in Fig. 1(a). The detailed algorithm of the training procedure is illustrated in Algorithm 1.

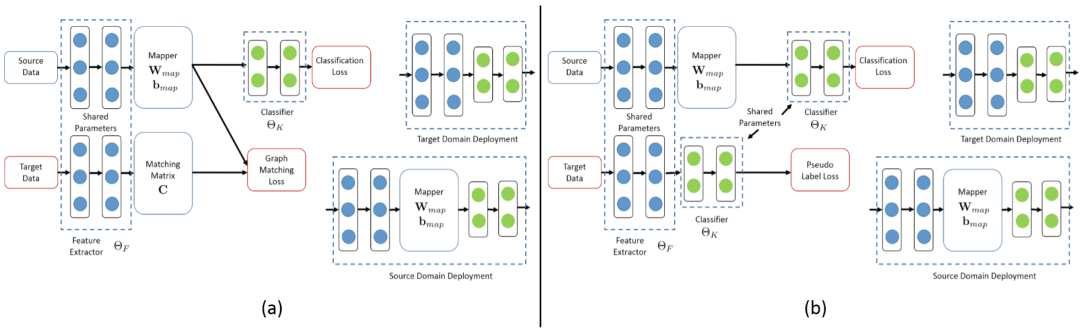

Fig. 1. The overall neural network framework for training using (a) Graph Matching (GM) Loss and (b) Pseudo-label (PL) Loss. On the right of (a) and (b), we see the model we should use for inference.

Algorithm 1. Graph-Matching-Guided Deep Domain Adaptation

Given : Source Labelled Data $\mathbf{X}^s$, $\mathbf{Y}^s$, Target Unlabelled Data $\mathbf{X}^t$
Parameters : $\lambda_s, \lambda_p, \lambda, m, T_i$ and learning rates
Randomly Initialize $\Theta_F, \Theta_K, \mathbf{C}, \mathbf{W}_{map}, \mathbf{b}_{map}$
Repeat
  Sample mini-batch $\{\mathbf{x}_i^s, y_i^s\}_{i=1}^m$, $\{\mathbf{x}_i^t\}_{i=1}^m$ from $\mathbf{X}^s$ and $\mathbf{X}^t$
  Use mini-batch to form Φ^t and Φ^s
  for $t_i = 1, 2, ...T_i$
    $\mathbf{C} \leftarrow \mathbf{C} - \alpha_1 \nabla_{\mathbf{C}} \mathcal{L}_{GM}(\Phi^t, \Phi^s)$
    $\mathbf{C} \leftarrow \max(\mathbf{C}, \mathbf{0})$
  end for
  Use Eq. (5) to obtain $\mathbf{W}_{map}$, $\mathbf{b}_{map}$
  $\Theta_K \leftarrow \Theta_K - \alpha_2 \nabla_{\Theta_K} \mathcal{L}_c(\mathbf{x}^s, y^s)$
  $\Theta_F \leftarrow \Theta_F - \alpha_3 \nabla_{\Theta_F} [\mathcal{L}_c(\mathbf{x}^s, y^s) + \lambda \mathcal{L}_{GM}(\Phi^t, \Phi^s)]$
Until Convergence

2.3 Refinement with Pseudo-labels

This is the second stage of our proposed unsupervised domain adaptation approach. Till now, we have the mapped source domain representations in support of the target domain representations. Since the domain discrepancy has been minimized, we can think of all the source and target representations belonging to a single domain. This single domain consists of labelled and unlabelled data. This is a semi-supervised learning setting that has been explored from a low-density separation, manifold regularization point of view [2]. In this paper, we propose a novel approach to exploit the confident unlabelled target domain data to further refine the classification decision boundary.

Initially, we select a subset of a mini-batch of the unlabelled target data that provide highly-confident labels as output. In other words, we select those samples whose maximum softmax probability output is greater than a threshold (t_h). Mathematically, we select those $\mathbf{x}_i^t$ for which $\max\{\mathbf{K}(\phi(\mathbf{x}_i^t))_k\} \geq t_h$ over all classes $k \in \{1, 2, ...l\}$ and we repeat this for all unlabelled target domain samples in the mini-batch $i \in \{1, 2, ...m\}$. The pseudo-labels for those selected samples would be $\underset{k}{\text{argmax}}\{\mathbf{K}(\phi(\mathbf{x}_i^t))_k\}$. After that, we use the original labelled data $\{\mathbf{x}_i^s, y_i^s\}_{i=1}^m$ and the selected unlabelled samples as $\{\mathbf{x}_i^t\}_{i=1}^{m'}$, where $m' \leq m$ to further refine our model. The intuition for our method is that we want the unlabelled samples to be as far as possible from the decision boundaries. This would make it possible for unseen examples in the target domain to not be misclassified easily. As a result, we expect performance in the target domain to increase significantly.

In our model, we have a softmax classifier that returns probabilities of each class that the sample belongs to. Also pairwise relations between the probabilities give a measure of how far a sample is from a decision boundary between the corresponding pair of classes. For example, if the softmax classifier returns $(p_1, p_2, ...p_l)$ as outputs to input sample $\mathbf{x}$, $|p_i - p_j|$ is a measure of how far the sample $\mathbf{x}$ is from the decision boundary between class i and class j. If $p_i = p_j$, then the sample lies on the decision boundary between class i and class j. The general expression for maximizing the distance to the decision boundaries for all selected unlabelled samples and all classes is as follows:

$$\mathcal{L}_p(\mathbf{x}^t, \hat{y}^s) = \frac{1}{m'} \sum_{i=1}^{m'} \sum_{j,k} 1(\hat{y}_i^t = j \ \ OR \ \ \hat{y}_i^t = k)(p_j - p_k)^2. \tag{6}$$

Here, $p_j = \mathbf{K}(\phi(\mathbf{x}_i^t))_j$, and $\hat{y}_i^t$ is the pseudo-label corresponding to the input sample $\mathbf{x}_i^t$ as obtained using thresholding. When $\hat{y}_i^t = j$ or $\hat{y}_i^t = k$ is true, we have $1(\hat{y}_i^t = j \text{ OR } \hat{y}_i^t = k) = 1$, and 0 otherwise. We call $\mathcal{L}_p$ as the *Pseudo-Label* (PL) loss. We also use the classification loss $\mathcal{L}_c$ introduced in Eq. (3) to regularize $\mathcal{L}_p$. Hence, we need to solve the following optimization problem,

$$\min_{\Theta_F, \Theta_K} \{-\mathcal{L}_p + \gamma \mathcal{L}_c\}, \tag{7}$$

where γ weighs the classification cost term. In Fig. 1(b), we show the overall neural network framework for using the Pseudo-Label (PL) loss. Algorithm 2 outlines the detailed approach of the training procedure.

Algorithm 2. Pseudo-label-guided Deep Domain Adaptation

Given : Source Labelled Data $\mathbf{X}^s$, $\mathbf{Y}^s$, Target Unlabelled Data $\mathbf{X}^t$
Parameters : γ, t_h, m and learning rates
Restart $\Theta_F, \Theta_K, \mathbf{W}_{map}, \mathbf{b}_{map}$ obtained from Algorithm 1
Repeat
Sample mini-batch $\{\mathbf{x}_i^s, y_i^s\}_{i=1}^m$, $\{\mathbf{x}_i^t\}_{i=1}^m$ from $\mathbf{X}^s$ and $\mathbf{X}^t$
Obtain high-confidence samples and pseudo-labels $\{\mathbf{x}_i^t, \hat{y}_i^t\}_{i=1}^{m'}$ using t_h criterion and use those samples for parameter update as follows
$\Theta_K \leftarrow \Theta_K - \alpha_2 \nabla_{\Theta_K}[-\mathcal{L}_p(\mathbf{x}^t, \hat{y}^t) + \gamma \mathcal{L}_c(\mathbf{x}^s, y^s)]$
$\Theta_F \leftarrow \Theta_F - \alpha_3 \nabla_{\Theta_F}[-\mathcal{L}_p(\mathbf{x}^t, \hat{y}^t) + \gamma \mathcal{L}_c(\mathbf{x}^s, y^s)]$
Until Convergence

3 Experiments and Results

To evaluate the effectiveness of our proposed approach on standard domain adaptation datasets for image classification, we utilized the Office-Caltech dataset, a small-scale domain adaptation benchmark dataset, initially released by [8]. The dataset is composed of 10 common categories across 4 domains - Amazon (A), Webcam (W), DSLR (D) and Caltech (C). Each of these domains varies in terms of image quality, viewpoints, presence/absence of backgrounds, etc. For domain adaptation, we would have 12 tasks, where each task consists of a source domain and a target domain picked from the 4 domains. For our experiments, we use *Decaf* features as the input. These deep features [6] are 4096-dimensional FC7 hidden activations of the deep convolutional neural network AlexNet [12].

We compared our method to recent approaches in learning domain-invariant representations. As a lower bound on recognition accuracy, we also compare against the no-adaptation (NA) baseline which includes training the model using only the source data and directly testing on the target data. The methods that we compared against include: (a) DANN [7], (b) MMD [9], (c) CORAL [18] and (d) WDGRL [16]. These approaches have been described in the Introduction section. We have implemented our approach in Tensorflow [1] and the training was carried out using *Adam* [11] optimizer. We followed the standard protocol used in previous method as in [16]. Since hyper-parameter selection is not possible using deep unsupervised domain-adaptation methods, we reported the best results of each approach after carrying out grid search on their respective hyper-parameters. For training, we have used a batch size of 64 samples with 32 samples from each domain. The feature extractor is a 2-layer neural network with 500 and 100 nodes and a ReLU activation. We used this same feature extractor in all the methods for fair comparisons. For our method, we used the following values of the penalty parameter $\lambda_p = 10$, threshold $t_h = 0.8$, and mapping regularization $\lambda_w = 0.1$. We set λ_s, λ and γ as the tunable hyper-parameters over which

we reported the best results averaged over 10 trials in Table 1. From Table 1, we see that in almost all cases our proposed graph-matching method (GM) is close to the previous best method. However, with the additional pseudo-labelling stage (PL), our proposed method produces better recognition accuracy in almost all the domain-adaptation tasks. Also, in almost all cases, the improvement of GM+PL over GM is 2–3%. This justifies the exploitation of labelled and unlabelled data after minimizing domain discrepancy, leading to an improvement in performance. For the task D→C, GM and eventually GM+PL do not produce the best result. This is possibly because the datasets D and C do not have enough structurally similar regions to be matched appropriately.

Table 1. Domain-adaptation results for object recognition using Office-Caltech datasets using Decaf features for a pair of source → target domain.

Task	NA	MMD	DANN	CORAL	WDGRL	GM	GM+PL
A→C	83.93	86.72	87.12	86.24	87.84	87.99	**89.48**
A→D	82.23	89.96	83.27	90.36	91.67	92.82	**95.73**
A→W	76.69	90.68	80.13	89.61	89.34	92.63	**94.23**
W→A	80.23	89.34	81.36	83.42	92.34	90.64	**94.68**
W→D	96.49	**100**	**100**	**100**	**100**	**100**	**100**
W→C	78.65	88.64	80.11	86.27	89.42	88.78	**91.31**
D→A	82.91	90.24	84.72	84.1	91.34	89.24	**92.34**
D→W	96.86	97.68	98.34	96.93	97.24	97.84	**99.83**
D→C	78.61	86.58	83.69	80.49	**90.24**	85.68	88.87
C→A	89.97	91.6	90.84	92.49	93.57	93.68	**95.83**
C→W	86.47	90.36	88.74	91.62	91.23	92.68	**94.21**
C→D	87.79	90.64	89.41	88.71	92.68	92.83	**94.51**

We chose a particular task A→W and studied the effect of varying hyper-parameters on recognition performance. In Fig. 2(a), we see that the performance reaches a peak at $\lambda_s = 10$. The red-dotted line is the base-line performance for $\lambda_s = 0$. So, the presence of the second-order matching term increases the performance over when it is not. Also, for $\lambda_s = 100$, the performance dips by a large amount, suggesting that putting excess weight on second-order term is not recommended. We saw a similar trend for the hyper-parameter λ in Fig. 2(b). λ weighs the graph-matching loss with respect to the classification loss. As expected, putting too much weight ($\lambda = 10$) ignores the classification loss in domain adaptation and produces a dip in performance. Recognition performance is comparatively less sensitive to γ as seen in Fig. 2(c). This is because domain discrepancy has already been minimized and the presence of classification loss on the source data does not affect target domain recognition rate much. Figure 2(d) shows the convergence of source and target error. We used GM stage for the first 2000 iterations followed by the PL stage in the next 2000 iterations. We noticed

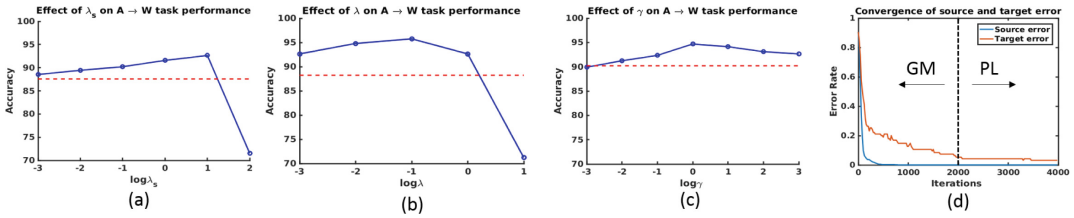

Fig. 2. Accuracy results on the A→W task due to change in (a) λ_s, (b) λ, (c) γ and (d) convergence results.

the drop in error rate when the PL stage was introduced after 2000 iterations. We also visualized the learned features using t-SNE [14] in Fig. 3. The clusters in the figure correspond to 10 classes. The blue and red points correspond to the source and target data respectively. For the un-adapted data in Fig. 3(a), the target domain classes do not form compact clusters. Also, there is a lot of discrepancy between the corresponding source and target clusters, causing a lot of mis-classification. For UDA, using only the GM procedure as in Fig. 3(b), the target domain classes form clusters but there are still some divergence between some of the corresponding source and target classes, which are reduced further using the PL stage as shown in Fig. 3(c).

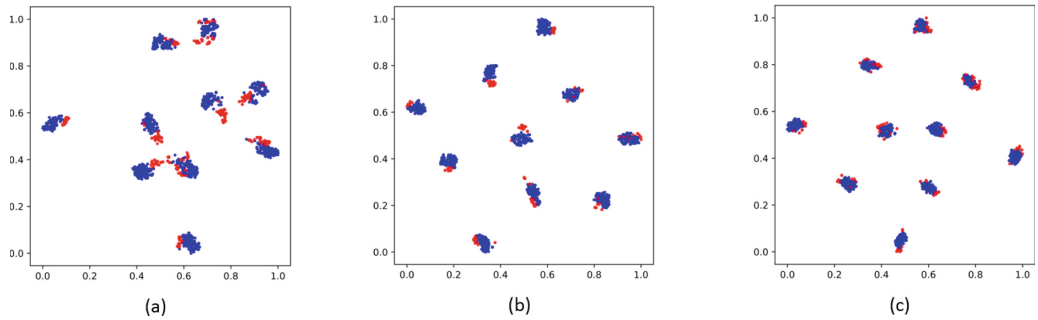

Fig. 3. Feature visualization for the A→W task for (a) no adaptation, (b) UDA with only Graph Matching and (c) UDA with Graph Matching and Pseudo-labelling. (Color figure online)

4 Conclusions

In this paper, we proposed a two-stage approach to learning domain-invariant-feature representations for unsupervised domain adaptation. In the first stage, we considered minimizing graph matching (GM) loss to minimize the discrepancy between source and target domains. The graph matching loss includes a second-order structural similarity term that allows us to consider structural similarity between two domains. For the second stage, we refined the feature/classifier using the confident pseudo-labels (PL) of the target domain data. Empirical results on image classification datasets demonstrated that our proposed GM+PL method outperforms previous domain-invariant representation learning approaches.

Acknowledgments. This work was supported in part by the National Science Foundation under Grant IIS-1813935. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

1. Abadi, M., et al.: TensorFlow: a system for large-scale machine learning. In: OSDI, pp. 265–283 (2016)
2. Chapelle, O., Schlkopf, B., Zien, A.: Semi-Supervised Learning, 1st edn. The MIT Press, Cambridge (2010)
3. Courty, N., Flamary, R., Tuia, D., Rakotomamonjy, A.: Optimal transport for domain adaptation. IEEE Trans. Pattern Anal. Mach. Intell. **39**(9), 1853–1865 (2017)
4. Das, D., Lee, C.S.G.: Sample-to-sample correspondence for unsupervised domain adaptation. Eng. Appl. Artif. Intell. **73**, 80–91 (2018)
5. Das, D., Lee, C.S.G.: Unsupervised domain adaptation using regularized hypergraph matching. In: Proceedings of IEEE International Conference on Image Processing (2018, to appear)
6. Donahue, J., et al.: DECAF: a deep convolutional activation feature for generic visual recognition. In: International Conference on Machine Learning, pp. 647–655 (2014)
7. Ganin, Y., et al.: Domain-adversarial training of neural networks. J. Mach. Learn. Res. **17**(59), 1–35 (2016)
8. Gong, B., Shi, Y., Sha, F., Grauman, K.: Geodesic flow kernel for unsupervised domain adaptation. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2066–2073 (2012)
9. Gretton, A., Smola, A., Huang, J., Schmittfull, M., Borgwardt, K., Schölkopf, B.: Covariate shift by kernel mean matching. Dataset Shift Mach. Learn. **3**(4), 5 (2009)
10. Hoffman, J., Rodner, E., Donahue, J., Kulis, B., Saenko, K.: Asymmetric and category invariant feature transformations for domain adaptation. Int. J. Comput. Vis. **109**(1–2), 28–41 (2014)
11. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: International Conference on Learning Representations (2015)
12. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
13. Long, M., Cao, Y., Wang, J., Jordan, M.I.: Learning transferable features with deep adaptation networks. In: International Conference on Machine Learning, pp. 97–105 (2015)
14. Maaten, L.V.D., Hinton, G.: Visualizing data using t-SNE. J. Mach. Learn. Res. **9**(Nov), 2579–2605 (2008)
15. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Trans. Knowl. Data Engg. **22**(10), 1345–1359 (2010)
16. Shen, J., Qu, Y., Zhang, W., Yong, Y.: Wasserstein distance guided representation learning for domain adaptation. In: AAAI, pp. 3–9 (2018)
17. Sun, B., Feng, J., Saenko, K.: Return of frustratingly easy domain adaptation. In: Thirtieth AAAI Conference on Artificial Intelligence (2016)

18. Sun, B., Saenko, K.: Deep CORAL: correlation alignment for deep domain adaptation. In: Hua, G., Jégou, H. (eds.) ECCV 2016. LNCS, vol. 9915, pp. 443–450. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49409-8_35
19. Tzeng, E., Hoffman, J., Saenko, K., Darrell, T.: Adversarial discriminative domain adaptation. arXiv preprint arXiv:1702.05464 (2017)
20. Tzeng, E., Hoffman, J., Zhang, N., Saenko, K., Darrell, T.: Deep domain confusion: maximizing for domain invariance. arXiv preprint arXiv:1412.3474 (2014)