Increasing Model Generalizability for Unsupervised Domain Adaptation

Mohammad Rostami
Information Sciences Institute
University of Southern California
Los Angeles, CA, USA
rostamim@usc.edu

Abstract

A dominant approach for addressing unsupervised domain adaptation is to map data points for the source and the target domains into an embedding space which is modeled as the output-space of a shared deep encoder. The encoder is trained to make the embedding space domain-agnostic to make a source-trained classifier generalizable on the target domain. A secondary mechanism to improve UDA performance further is to make the source domain distribution more compact to improve model generalizability. We demonstrate that increasing the interclass margins in the embedding space can help to develop a UDA algorithm with improved performance. We estimate the internally learned multi-modal distribution for the source domain, learned as a result of pretraining, and use it to increase the interclass class separation in the source domain to reduce the effect of domain shift. We demonstrate that using our approach leads to improved model generalizability on four standard benchmark UDA image classification datasets and compares favorably against exiting methods.

1 Introduction

Despite remarkable progress in deep learning, deep neural networks suffer from poor generalization when distributional gaps emerge during model execution due to the existence of *domain shift* Gretton et al. (2009). Existence of distributional discrepancies between a source, i.e., training, and a target, i.e., testing, domains necessitates retraining the neural network in the target domain so the model generalizes well again. However, this process is costly and time-consuming due to requiring persistent manual data annotation Rostami et al. (2018). Unsupervised Domain Adaptation (UDA) is a learning framework for mitigating the effect of domain shift when only unlabeled data is accessible in the target domain. The major approach to address UDA is to map the source domain labeled data and the target domain unlabeled data into a shared latent embedding space and then minimize the distance between the distributions in this space to mitigate domain shift Daumé III (2007); Long et al. (2015); He et al. (2016); Gabourie et al. (2019). When the embedding space becomes domain-agnostic, a classifier network that receives its input from the data representations in the shared embedding space and is trained in the source domain will generalize well in the target domain.

Recent domain adaptation methods model the shared embedding space as the output-space of a deep encoder network and enforce domain alignment by training the encoder accordingly. In the UDA literature, domain alignment has been been explored extensively with most methods enforcing domain alignment either using adversarial learning Ganin & Lempitsky (2015b); Ganin et al. (2016); Tzeng et al. (2017); Hoffman et al. (2018) or through direct cross-domain distribution alignment Bhushan Damodaran et al. (2018a); Pan et al. (2019); Rostami (2021). Generative adversarial networks (GANs) Goodfellow et al. (2014) can be adapted to align two distributions indirectly in an embedding space for UDA. The shared encoder is modeled as a cross-domain feature generator network. The source domain and the target domain features become indistinguishable by a competing discriminator network which is trained jointly using the adversarial min-max training procedure Goodfellow et al. (2014). This procedure aligns the two distributions effectively, but adversarial learning is known to require delicate engineering, including setting the optimization initial point, the architecture of the auxiliary networks, and selection of hyper-parameters to remain stable Roth et al. (2017) as well as mode collapse vulnerability Srivastava et al. (2017). Direct probability matching is based on directly minimizing a probability distribution distance between two domain-specific distributions in the embedding space. It requires less data engineering if the right distribution metric is selected. However, performance of methods based on probability matching on average is less than methods based on adversarial learning. The reason is partially because measuring distances on higher dimensions is a challenging task, i.e., curse of dimensionality, and small distances between two distributions in an embedding space does not necessarily translate into semantic similarities in the input.

Despite significant prior explorations, UDA still is an active research area. The general strategy of domain alignment for UDA can be improved by augmenting secondary mechanisms that increase the target domain data separability in the embedding space. A diverse set of approaches has been proposed to improve UDA performance and recent advances in UDA are primarily result of designing new secondary mechanisms. For example, Motiian et al. Motiian et al. (2017) enforce semantic class-consistent alignment of the distributions by assuming that a few labeled target domain data is accessible. Contrastive Adaptation Network Kang et al. (2019) uses pseudo-labels to enforce class-conditional alignment. Class-conditional alignment is particularly helpful to avoid matching wrong classes in the embedding space. Xu et al. Xu et al. (2020) propose domain mixup on pixel and feature levels to have a continuous latent shared distribution to mitigate the oscillation of target data distribution. Li et al. Li et al. (2020) regularize the loss function with entropy criterion to benefit from the intrinsic structure of the target domain classes. When we train a deep neural network in a supervised classification settings, data representations, i.e., network responses, form class-specific separate clusters in the final layer of the network. A helpful strategy to improve domain alignment is to induce larger margins between these class-specific clusters in the embedding space such that the class-specific clusters become unconfined. An approach to this end using adversarial min-max optimization to increase model generalizability Kim et al. (2019). In addition to be intuitive, recent theoretical results demonstrate that large margin-separation on source domain leads to improved model generalizability that is helpful for UDA Dhouib et al. (2020).

Contributions: we develop a new secondary mechanism to improve model generalizability to address UDA. We use the internal data distribution that is formed in a shared embedding space, as a result of a pretraining stage on the source domain, to increase margins between different visual class clusters to mitigate the effect of domain shift in a target domain. Since the internally learned distribution in the embedding space is a multimodal distribution, we use a Gaussian mixture modal (GMM) to parametrize and estimate the internal distribution. To increase the interclass margins, we build a pseudo-dataset by drawing random samples with confident labels from the estimated GMM and regularize the model to repulse the target domain samples away from class boundaries by minimizing the distance between the target domain and the pseudo-dataset distribution. We demonstrate that our algorithm is effective by validating it on four benchmark UDA datasets and observe that it is competitive when compared with existing methods.

2 Related Work

We follow the direct probability matching approach for UDA. The primary challenge is to select a suitable probability discrepancy measure. A suitable metric should be computable easily and should possess nice properties for numerical minimization. Various probability metrics have been used for probability matching. To name a few works, the Maximum Mean Discrepancy (MMD) has been used for probability matching by aligning the means of the two distributions Long et al. (2015; 2017). Sun et al. Sun & Saenko (2016) improve upon this baseline via aligning distribution correlations to take advantage of second-order statistics. Other improvements include using the Central moment discrepancy Zellinger et al. (2017) and Adaptive batch normalization Li et al. (2018). Domain alignment using lower order probability moments is simple, yet it overlooks mismathces in higher moments. The Wasserstein distance (WD) Courty et al. (2016); Bhushan Damodaran et al. (2018b) has been used to include encoded information in higher-order statistics. Damodaran et al. Bhushan Damodaran et al. (2018b) demonstrated that using WD improves UDA performance compared to using MMD or correlation alignment Long et al. (2015); Sun & Saenko (2016). In contrast to more common probability metrics such as KL-divergence or JS-divergence, WD possesses non-vanishing gradients even when the two distributions do not have overlapping supports. Hence, WD can be minimized effectively using the first-order optimization methods. This property makes WD a suitable choice for deep learning because deep learning objective functions are usually optimized using gradient-based methods. Although using WD leads to improved UDA performance, a downside of using WD is heavy computational load in the general case compared to simpler probability metrics. The high computational load is because WD is defined as a linear programming optimization and does not have a closed-form solution for dimensions more than one. To account for this constraint, we use the sliced Wasserstein distance (SWD) Lee et al. (2019) which we have previously been used for addressing UDA in different settings Rostami et al. (2019); Gabourie et al. (2019); Stan & Rostami (2021a;b). SWD is defined in terms of a closed-form solution of WD in 2D. It can also be computed fast from empirical samples. Compared to these works that use SWD for distributional alignment, we develop a secondary mechanism that improves model generalizability.

When a deep neural network is trained in a supervised classification setting, learning often can be interpreted as geometric separability of the input data points representations in an embedding space (see Figure 1). The embedding space can be modeled by network responses in a final higher layer. This interpretation implies that the input distribution is transformed into a multi-modal internal distribution, where each class is represented by a single distributional mode (see Figure 1). This is necessary for class separation because final layer of a classifier network is usually a softmax layers which implies classes should become linearly separable. Properties for this internally learned distribution can be used to improve UDA performance. For example, a recent approach for UDA is to match the internal distributions

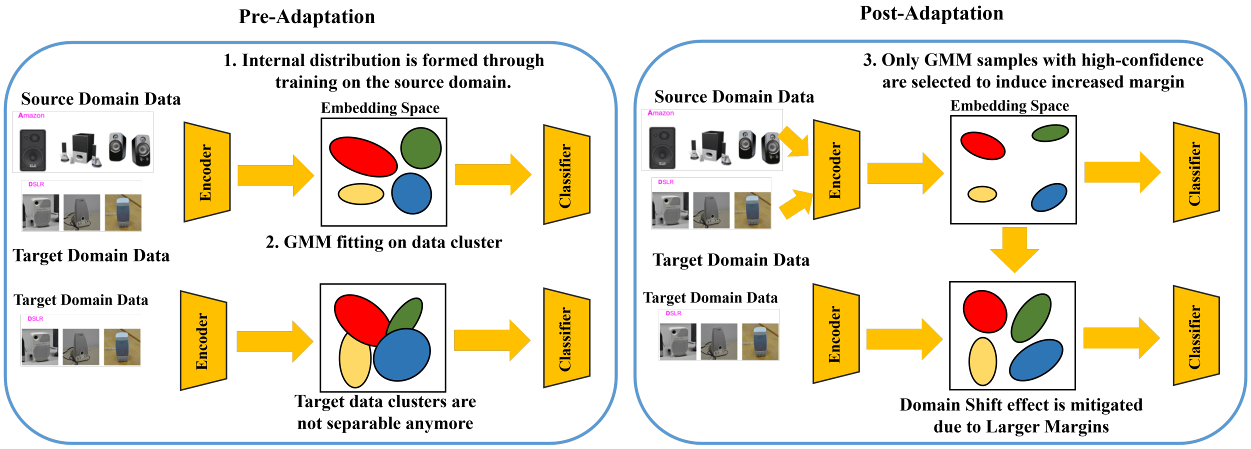

Pre-Adaptation
1. Internal distribution is formed through training on the source domain.
Source Domain Data
Amazon
DSLR
Target Domain Data
Embedding Space
Encoder
Classifier
2. GMM fitting on data cluster
Target Domain Data
DSLR
Encoder
Classifier
Target data clusters are not separable anymore
Post-Adaptation
3. Only GMM samples with high-confidence are selected to induce increased margin
Source Domain Data
Amazon
DSLR
Target Domain Data
Embedding Space
Encoder
Classifier
Target Domain Data
DSLR
Encoder
Classifier
Domain Shift effect is mitigated due to Larger Margins

Figure 1: High-level description of the proposed unsupervised domain adaptation algorithm: a pretrained model on the source domain learns separable source clusters in the embedding space (left-top), but the model generalizes poorly on the target domain due to domain shift which decimates class separability in the embedding space due to violating margins between data clusters (left-bottom). The generated pseudo-dataset using confident samples helps to induce larger margins between the class clusters in the embedding space by making each class cluster more compact. As a result, we can improve generalizability (right-top) which helps to mitigate domains shift in the target domain by allowing more room for domain-shift before violating the margins between the classes (right-bottom)

based on aligning the cluster-specific means for each class across both domains Pan et al. (2019); Chen et al. (2019b). This class-aware domain alignment procedure helps to avoid class mismatch problem. Our goal is to regularize the shared encoder such that the internal distribution remains robust with respect to input distributional perturbations by making class clusters more compact in the embedding space Zhang et al. (2020). Representation compactness will introduce large interclass margins, leading to stability with respect to domain-shift. Theoretical exploration in few-shot learning settings has demonstrated that increasing the inter-class margin reduces misclassification rate due to the existence of larger margins Cao et al. (2019). Inspired by these works, our idea is to estimate the formed internal distribution as a parametric GMM and use it to induce larger margins between class-specific distributional modes.

3 Problem Statement

Consider that we are given a target domain $\mathcal{T}$ with the unlabeled dataset $D_{\mathcal{T}} = (\boldsymbol{X}_{\mathcal{T}})$, where $\boldsymbol{X}_{\mathcal{T}} = [\boldsymbol{x}_1^t, \ldots, \boldsymbol{x}_M^t] \in \mathcal{X} \subset \mathbb{R}^{d\times M}$, denotes the input data points. The goal is to train a generalizable model for the target domain. In the absence of data annotations, this task is intractable in most cases. However, we are also given a source domain $\mathcal{S}$ with the labeled dataset $D_{\mathcal{S}} = (\boldsymbol{X}_{\mathcal{S}}, \boldsymbol{Y}_{\mathcal{S}})$, where $\boldsymbol{X}_{\mathcal{S}} = [\boldsymbol{x}_1^s, \ldots, \boldsymbol{x}_N^s] \in \mathcal{X} \subset \mathbb{R}^{d\times N}$ and $\boldsymbol{Y}_{\mathcal{S}} = [\boldsymbol{y}_1^s, ..., \boldsymbol{y}_N^s] \in \mathcal{Y} \subset \mathbb{R}^{k\times N}$ denotes the corresponding one-hot labels. The two domain are related and both share the same k semantic classes. The source and the target input data points are drawn i.i.d from two domain-specific distributions $\boldsymbol{x}_i^s \sim p_S(\boldsymbol{x})$ and $\boldsymbol{x}_i^t \sim p_T(\boldsymbol{x})$. Despite cross-domain similarities, distribution discrepancy exists between the two domains, i.e, $p_T(\boldsymbol{x}) \neq p_S(\boldsymbol{x})$. In the absence of a target domain annotated dataset, a naive approach is to select a family of parameterized functions $f_\theta : \mathbb{R}^d \to \mathcal{Y}$, e.g., deep neural networks with learnable weight parameters θ, and search for the Bayes-optimal model $f_{\theta^*}(\cdot)$ in this family using the standard empirical risk minimization (ERM) using the source domain annotated data: $\hat{\theta} = \arg\min_\theta\{\hat{e}_\theta(\boldsymbol{X}_{\mathcal{S}}, \boldsymbol{Y}_{\mathcal{S}}, \mathcal{L})\} = \arg\min_\theta\{\frac{1}{N}\sum_i \mathcal{L}(f_\theta(\boldsymbol{x}_i^s), \boldsymbol{y}_i^s)\}$, where $\mathcal{L}(\cdot,\cdot)$ is a proper point-wise loss function, e.g., the cross-entropy loss. In an ideal situation, the ERM-trained model is generalizable on the source domain and due to cross-domain knowledge transfer, this source-trained model likely would perform better than chance in the target domain. However, the source-trained model will still suffer from generalization degradation compared to the performance on the source domain due to the existence of domain gap, i.e., $p_T(\boldsymbol{x}) \neq p_S(\boldsymbol{x})$.

The above naive solution does not benefit from the target domain unannotated dataset. The goal in UDA is to take advantage of the encoded information in the unlabeled target data points and improve the source-trained model generalization in the target domain. We use the common approach of reducing the domain gap across the two domains by mapping data points from both domain into a shared embedding space $\mathcal{Z}$ using a shared encoder such that the distribution discrepancy becomes minimal in the embedding space. In other words, domain gap between the two domains becomes negligible after this transformation. The critical challenge is to find such an embedding space. Diversity among the many existing UDA algorithms stems from how to exactly find such a shared embedding space.

To model the above shared embedding, the end-to-end deep neural network $f_\theta(\cdot)$ can be decomposed into an encoder subnetwork $\phi_{\boldsymbol{v}}(\cdot) : \mathcal{X} \to \mathcal{Z} \subset \mathbb{R}^p$ and a classifier subnetwork $h_{\boldsymbol{w}}(\cdot) : \mathcal{Z} \to \mathcal{Y}$ with learnable parameters $\boldsymbol{v}$ and $\boldsymbol{w}$, respectively. In this decomposition, we have: $f_\theta = h_{\boldsymbol{w}} \circ \phi_{\boldsymbol{v}}$ and $\theta = (\boldsymbol{w}, \boldsymbol{v})$. Following the class separability condition for a good model generalization on source domain, we assume that the classes are separable in $\mathcal{Z}$ for the source domain as a result of supervised pretraining in the source domain (see Figure 1, left). Benefiting from the target domain data, most UDA framework adapt the source-trained encoder such that the distributions of both domains are matched in the embedding space $\mathcal{Z}$, i.e., we have $\phi(p_{\mathcal{S}}(\cdot)) \approx \phi(p_{\mathcal{T}}(\cdot))$. Hence, the classifier subnetwork will generalize well in the target domain, despite having been trained using solely the source domain data. A major class of UDA methods match the distributions $\phi(p_{\mathcal{S}}(\boldsymbol{x}^s))$ and $\phi(p_{\mathcal{T}}(\boldsymbol{x}^t))$ by training the encoder $\phi(\cdot)$ such that the distance between these two distribution, in terms of a probability distribution metric, is minimized in the embedding space:

$$\hat{\boldsymbol{v}}, \hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{v},\boldsymbol{w}} \frac{\lambda}{N} \sum_{i=1}^{N} \mathcal{L}\big(h_{\boldsymbol{w}}(\phi_{\boldsymbol{v}}(\boldsymbol{x}_i^s)), \boldsymbol{y}_i^s\big) + D\big(\phi_{\boldsymbol{v}}(p_{\mathcal{S}}(\boldsymbol{X}_{\mathcal{S}})), \phi_{\boldsymbol{v}}(p_{\mathcal{T}}(\boldsymbol{X}_{\mathcal{T}}))\big), \tag{1}$$

where $D(\cdot,\cdot)$ denotes a probability discrepancy metric and λ is a trade-off parameter between the empirical risk and the domain alignment terms. There are various choices for $D(\cdot,\cdot)$ and many UDA methods are developed by selecting difference probablity discrepancy metrics. In this work, we select SWD for the metric $D(\cdot,\cdot)$.

SWD is defined by applying the idea of slicing Le et al. (2019) on the Wasserstein distance (WD), defined as:

$$W(p_{\mathcal{S}}, p_{\mathcal{T}}) = \inf\nolimits_{\gamma \in \Gamma(p_{\mathcal{S}}, p_{\mathcal{T}})} \int_{X \times Y} c(x,y) d\gamma(x,y) \tag{2}$$

where $\Gamma(p_{\mathcal{S}}, p_{\mathcal{T}})$ denotes the set of joint distributions $p_{\mathcal{S},\mathcal{T}}$ with such that distributions $p_{\mathcal{S}}$ and $p_{\mathcal{T}}$ are its two marginal distribution and $c : X \times Y \to \mathbb{R}^+$ is a cost function, e.g., ℓ_2-norm Euclidean distance. In general, WD does not have a closed form solution. However, in the case of $1-$dimensional distributions, it has a closed-form solution as follows:

$$W(p_{\mathcal{S}}, p_{\mathcal{T}}) = \int_0^1 c(P_{\mathcal{S}}^{-1}(\tau), P_{\mathcal{T}}^{-1}(\tau)) d\tau, \tag{3}$$

where $P_{\mathcal{S}}$ and $P_{\mathcal{T}}$ are the cumulative distributions of the $1-$dimensional distributions $p_{\mathcal{S}}$ and $p_{\mathcal{T}}$. This closed-form solution has motivated the definition of SWD using the slice sampling technique Neal (2003). The idea is to select a 1-dimensional subspace and project two d-dimensional distributions on the subspace to generate their marginal $1-$dimensional distributions and define SWD by integrating the $1-$dimensional WD between the marginal distributions the over all possible $1-$dimensional subspaces. A one-dimensional slice for a distribution $p_{\mathcal{S}}$ is defined as:

$$\mathcal{R}p_{\mathcal{S}}(t;\boldsymbol{\gamma}) = \int_{\mathcal{S}^{d-1}} p_{\mathcal{S}}(\boldsymbol{x}) \boldsymbol{\delta}(t - \langle \boldsymbol{\gamma}, \boldsymbol{x} \rangle) d\boldsymbol{x}, \tag{4}$$

where $\boldsymbol{\delta}(\cdot)$ denotes the Kronecker delta function, $\langle \cdot, \cdot \rangle$ denotes the inner dot product, $\mathbb{S}^{d-1}$ is the d-dimensional unit sphere, and $\boldsymbol{\gamma}$ is the one-dimensional projection direction. The SWD then is defined as the following integral:

$$SW(p_{\mathcal{S}}, p_{\mathcal{T}}) = \int_{\mathbb{S}^{d-1}} W(\mathcal{R}p_{\mathcal{S}}(\cdot;\gamma), \mathcal{R}p_{\mathcal{T}}(\cdot;\gamma)) d\gamma. \tag{5}$$

The main advantage of using the SWD for UDS is that unlike WD,the integrand of equation 5 has a closed form solutio for a known γ. To computer the integral in Eq. equation 5, we can use a Monte Carlo style integration. First, we sample the projection subspace $\boldsymbol{\gamma}$ from a uniform distribution, defined over the unit sphere, compute the $1-$dimensional WD, and then approximate Eq. equation 5 by computing the arithmetic mean over a sufficient number of random projection:

$$\hat{D}(p_{\mathcal{S}}, p_{\mathcal{T}}) \approx \frac{1}{L} \sum_{l=1}^{L} \sum_{i=1}^{M} |\langle \gamma_l, \phi(\boldsymbol{x}^{\mathcal{S}}_{s_l[i]}) \rangle - \langle \gamma_l, \phi(\boldsymbol{x}^{\mathcal{T}}_{t_l[i]}) \rangle|^2 \tag{6}$$

where $\gamma_l \in \mathbb{S}^{f-1}$ is a uniformly drawn random sample from the unit f-dimensional ball $\mathbb{S}^{d-1}$, and $s_l[i]$ and $t_l[i]$ are the sorted indices of $\{\gamma_l \cdot \phi(\boldsymbol{x}_i)\}_{i=1}^{M}$ for source and target domains, respectively. Following Gabourie et al. (2019), we rely on Eq. equation 6 to compute SWD empirically. We use SWD in our work because: i) as discussed, is a suitable metric for solving optimization problems of deep learning, ii) it can be computed efficiently due to its closed form solution, and iii) its empirical version can be computed efficiently according to equation 6. Many UDA methods are developed based on variations of equation 1 using different probability metrics and additional regularization to enforce a helpful property. In our work, we aim for improving upon the previous work that obtain a model by solving equation 1 through inducing larger margins between the learned class clusters in the embedding space to mitigate the effect of domain shift in the target domain. In Figure 1, we have visualized conceptually the positive effect of large interclass margins between classes which makes feature representations more compact.

Algorithm 1 IMUDA

```
**Input:** The datasets $\mathcal{D}_{\mathcal{S}} = (\boldsymbol{X}_{\mathcal{S}}, \boldsymbol{Y}_{\mathcal{T}})$, $\mathcal{D}_{\mathcal{T}} = (\boldsymbol{X}_{\mathcal{S}})$
    **Pretraining on the Source Domain:**
      $\hat{\theta}_0 = (\hat{\boldsymbol{w}}_0, \hat{\boldsymbol{v}}_0) = \arg\min_{\theta} \sum_i \mathcal{L}(f_{\theta}(\boldsymbol{x}_i^s), \boldsymbol{y}_i^s)$
    **GMM Estimation:**
      Use equation 8 and estimate GMM paramters $\alpha_j, \boldsymbol{\mu}_j, \Sigma_j$
**Domain Adaptation**:
    **Pseudo-Dataset Generation:**
      Generate $\hat{\mathcal{D}}_{\mathcal{P}}$ based on equation 9
**for** $itr = 1, \ldots, ITR$ **do**
    draw random data batches from $\mathcal{D}_{\mathcal{S}}$, $\mathcal{D}_{\mathcal{T}}$, and $\mathcal{D}_{\mathcal{P}}$ and update the model based on equation 10
**end for**
```

4 Proposed Algorithm for Increased Interclass Margins

We rely on the internally learned distribution to increase interclass margins. Following the explained rationale above, this distribution is a multi-modal distribution $p_J(\cdot) = \phi_{\boldsymbol{v}}(p_{\mathcal{S}}(\cdot))$ with k modes in $\mathcal{Z}$ (see Figure 1, left). Note that formation of a multimodal distribution is not guaranteed but if we model the embedding space using network responses just prior to the eventual softmax layer, it is a prerequisite for our model to learn the source domain. In Figure 1 (left-top), we can see the margins between classes corresponds to the geometric distances between the boundaries of modes of the source-learned internal distribution $\phi_{\boldsymbol{v}}(p_{\mathcal{S}}(\cdot))$. As seen in Figure 1 (left-bottom), domain shift is a result of deviations of the internal distribution for the target domain $\phi_{\boldsymbol{v}}(p_{\mathcal{T}}(\cdot))$ from the source-learned internal distribution $\phi_{\boldsymbol{v}}(p_{\mathcal{S}}(\cdot))$ which leads to more overlap between the class clusters in the target domain. In other words, the effect of domain shift in the input space translates into cross the boundary between some of the class clusters. Our idea is to develop a mechanism to repulse the target domain data samples from interclass margins towards the class means, as visualized in Figure 1 (right-top). As a result, we intuitively expect more model robustness with respect to domain shift in the input space, as visualized in Figure 1 (right-bottom). In other words, our goal is to make the class clusters more compact in the embedding space for the source domain to allow more variability in the target domain. To implement this idea, we first estimate the internally learned distribution in $\mathcal{Z}$ as a parametric GMM distribution as follows:

$$p_J(\boldsymbol{z}) = \sum_{j=1}^{k} \alpha_j \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j), \tag{7}$$

where $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$ denote the mean and co-variance matrices for each mode and α_j denotes the mixture weights. This probability distribution is a suitable model for the internal probability distribution because we know k. Additionally, as opposed to the general GMM estimation problem in which we need to rely on iterative and time-consuming procedures, e.g., expectation maximization (EM) Moon (1996), estimating the GMM parameters is simple our UDA setting. Because the source data points are labeled and also k is known. Hence, we can estimate $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$ specifically as parameters of k independent Gaussian distributions. The mixture weights α_j can be computed easily by a Maximum a Posteriori (MAP) estimate. Let $\boldsymbol{S}_j$ denotes the training data points that belong to the class j, i.e., $\boldsymbol{S}_j = \{(\boldsymbol{x}_i^s, \boldsymbol{y}_i^s) \in \mathcal{D}_{\mathcal{S}} | \arg\max \boldsymbol{y}_i^s = j\}$. Then, the MAP estimation for the GMM parameters can be computed as:

$$\hat{\alpha}_j = \frac{|\boldsymbol{S}_j|}{N}, \quad \hat{\boldsymbol{\mu}}_j = \sum_{(\boldsymbol{x}_i^s, \boldsymbol{y}_i^s) \in \boldsymbol{S}_j} \frac{1}{|\boldsymbol{S}_j|} \phi_v(\boldsymbol{x}_i^s), \quad \hat{\boldsymbol{\Sigma}}_j = \sum_{(\boldsymbol{x}_i^s, \boldsymbol{y}_i^s) \in \boldsymbol{S}_j} \frac{1}{|\boldsymbol{S}_j|} \big(\phi_v(\boldsymbol{x}_i^s) - \hat{\boldsymbol{\mu}}_j\big)^{\top} \big(\phi_v(\boldsymbol{x}_i^s) - \hat{\boldsymbol{\mu}}_j\big). \tag{8}$$

As opposed to the high computational complexity of EM Roweis (1998), equation 8 does not add a significant computational overload to perform UDA. If the source domain dataset is balanced, we only need to check whether data points $\boldsymbol{x}_i^s$ belong to $\boldsymbol{S}_j$ to compute α_j which has the computational complexity of $O(N)$. If we denote the dimension of the embedding space with F, the computational Complexity of computing $\boldsymbol{\mu}_j$ would be $O(NF/k)$. The computational complexity for computing the co-variance matrices $\boldsymbol{\Sigma}_j$ would $O(F(\frac{N}{k})^2)$. Finally, since there are k classes, the total computational complexity for estimating the GMM distribution would be $O(\frac{FN^2}{k})$. If $O(F) \approx O(k)$, i.e., a reasonable assumption when modeling the embedding space as network responses in the final layer, then the total computational complexity would be $O(N^2)$. This is a small computational overload compared to computational complexity of one epoch of back-propagation which additionally needs to be performed several times. Because deep neural networks generally have significantly larger number of weights than the number of data points N.

We use the estimated GMM distribution induce larger interclass margins to improve the solution by equation 1. To induce repulsive biases from the class margins (see Figure 1, top-right), we first generate a pseudo-dataset in the embedding space with confident labels $\mathcal{D}_{\mathcal{P}} = (\mathbf{Z}_{\mathcal{P}}, \mathbf{Y}_{\mathcal{P}})$ using randomly drawn samples from the estimated GMM, where $\boldsymbol{Z}_{\mathcal{P}} = [\boldsymbol{z}_1^p, \ldots, \boldsymbol{z}_{N_p}^p] \in \mathbb{R}^{p\times N_p}, \boldsymbol{Y}_{\mathcal{P}} = [\boldsymbol{y}_1^p, ..., \boldsymbol{y}_{N_p}^p] \in \mathbb{R}^{k\times N_p}$, and $\boldsymbol{z}_i$ is drawn randomly $\boldsymbol{z}_i^p \sim \hat{p}_J(\boldsymbol{z})$. To ensure that these samples lie away from the margins and within the proximity of the corresponding class means in the embedding space, we feed all the initially drawn samples into the classifier subnetwork and include only the samples for which the classifier confidence about the predicted label is more than a predetermined confidence threshold $0 < \tau < 1$. This procedure helps us to avoid including samples close to the margins. More specifically, we follow:

$$\mathcal{D}_{\mathcal{P}} = \Big\{(\boldsymbol{z}_i^p, \boldsymbol{y}_i^p) | \boldsymbol{z}_i^p \sim \hat{p}_J(\boldsymbol{z}), \max\{h(\boldsymbol{z}_i^p)\} > \tau, \boldsymbol{y}_i^p = \arg\max_i \{h(\boldsymbol{z}_i^p)\}\Big\}. \tag{9}$$

Sample selection based on the threshold values close to one, $\tau \approx 1$, implies that the pseudo-dataset samples are in the proximity of the class means (see Figure 1 (right-top)). This means that the margins between the empirical data clusters are larger in the generated pseudo-dataset compared to the source domain data empirical clusters in the embedding space. We benefit from this property of the pseudo-dataset and update equation 1 to induce larger margins:

$$\hat{\boldsymbol{v}}, \hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{v},\boldsymbol{w}} \Big\{ \frac{\lambda}{N} \sum_{i=1}^{N} \mathcal{L}\big(h_{\boldsymbol{w}}(\phi_{\boldsymbol{v}}(\boldsymbol{x}_i^s)), \boldsymbol{y}_i^s\big) + \frac{\lambda}{N_p} \sum_{i=1}^{N_p} \mathcal{L}\big(h_{\boldsymbol{w}}(\boldsymbol{z}_i^p), \boldsymbol{y}_i^p\big) + \hat{D}\big(\phi_{\boldsymbol{v}}(\boldsymbol{X}_{\mathcal{T}}), \boldsymbol{X}_{\mathcal{P}}\big) + \hat{D}\big(\phi_{\boldsymbol{v}}(\boldsymbol{X}_{\mathcal{S}}), \boldsymbol{X}_{\mathcal{P}}\big) \Big\}, \tag{10}$$

where $\hat{D}(\cdot, \cdot)$ denotes the empirical SWD probability distribution metric. The first and the second terms in equation 10 are ERM terms for the source dataset and the pseudo-dataset to keep the embedding space discriminative. The third term is an alignment term that matches the source domain distribution with the pseudo-dataset empirical distribution. The fourth term is an alignment term that matches the target domain distribution with the pseudo-dataset empirical distribution which as we describe possesses larger margins. These terms help to increase the interclass margins which as we described would increase the model generalizability. Our proposed algorithm, called Increased Margins for Unsupervised Domain Adaptation (IMUDA), is presented and visualized in Algorithm 1 and Figure 1, respectively.

5 Empirical Validation

5.1 Datasets and Tasks

We validate our method on four standard UDA benchmarks.

Digit recognition tasks: the three MNIST ($\mathcal{M}$), USPS ($\mathcal{U}$), and SVHN ($\mathcal{S}$) are used as domains. Following the literature, the UDA tasks include three digit recognition tasks: $\mathcal{M} \rightarrow \mathcal{U}$, $\mathcal{U} \rightarrow \mathcal{M}$, and $\mathcal{S} \rightarrow \mathcal{M}$. we resized the images of SVHN dataset to 28×28 images to have the same size of the MNIST and the USPS datasets.

Office-31 Detest: this dataset is probably the most common UDA dataset that consists of 31 visual classes with a total of $4,652$ images. There are three domains: Amazon ($\mathcal{A}$), Webcam ($\mathcal{W}$) and DSLR ($\mathcal{D}$) with six definable UDA tasks, defined in a pair-wise manner among the three domains.

ImageCLEF-DA Dataset: it consists of the 12 shared classes between the Caltech-256 ($\mathcal{C}$), the ILSVRC 2012 ($\mathcal{I}$), and the Pascal VOC 2012 ($\mathcal{P}$) visual recognition datasets as domains. This dataset is completely balanced as each class has 50 images or 600 images per domain. Since the domains and the classes have the same number of images, this dataset is a complements the Office-31 dataset which has varying domain and class sizes. Similarly, there six possible UDA tasks can be defined.

VisDA-2017: the goal is to train a model for natural images by pretraining the model on samples of a synthetic domain and then adapt it to generalize on the real image domain. The synthetic images are generated using 3D models of objects based on applying different lightning conditions across 12 classes. The dataset is larger with 280K images.

5.2 Backbone Structure and Evaluation Protocol:

We follow the literature for the network structures for each dataset to make fair evaluation of our work against existing works possible. The VGG16 network is used as the backbone model for the digit recognition tasks. For the Office-31, the ImageCLEF-DA datasets, we use the 50 network as the backbone. For the VisDa2017 dataset, we use the ReNet-101 network as the backbone. The backbone models are pretrained on the ImageNet dataset and their output is fed into a hidden layer with size 128, followed by the last year according to the number of classes in each dataset. The last layer is set to be a softmax layer and the embedding space $\mathcal{Z}$ is set to be the features before softmax. . We use the cross-entropy loss as the discrimination loss. At each training epoch, we computed the combined loss function on the training split of datasets of both domains and stopped training when the training loss function became stable. We used Keras for implementation and ADAM optimizer to solve the UDA optimization problems. We have used 100 projection to compute the SWD loss. We tune the learning rate for each dataset such that the training loss function reduces smoothly. We have run our code on a cluster node equipped with 4 Nvidia Tesla P100-SXM2 GPU's. We used the classification rate on the testing set to measure performance of the algorithms. We performed 5 training trials and reported the average performance and the standard deviation on the testing sets for these trials. Our code is available as a supplement.

For each task, we report the source-trained model performance (Source Only) on the target domain as a baseline. Performance improvements over this baseline serves as a simple ablative study and demonstrates the positive effect of model adaptation. We then adapt the model using IMUDA algorithm and report the performance on the target domain. In our results, we report the average classification rate and the standard deviation on the target domain, computed over ten randomly initialized runs for all datasets, except VisDA2017 for which we have reported the best result. Following our theorem, we set $\tau = 0.95$. We also set $\lambda = 10^{-2}$. The selection process for these values will be explored further.

There are many existing UDA methods in the literature which makes extensive comparison challenging. For our purpose, we need to select a subset of the these works. We included both pioneer and recent works to be representative of the recent progress in the field and also to present the improvements made over the pioneer works. We selected the methods that reported results on the majority of the benchmarks we used. These methods include those based on adversarial learning: GtA Sankaranarayanan et al. (2018), DANN Ganin et al. (2016), SymNets Zhang et al. (2019) ADDA Tzeng et al. (2017), MADA Pei et al. (2018), CDAN Long et al. (2018), DMRL Wu et al. (2020), DWL Xiao & Zhang (2021), HCL Huang et al. (2021), and CGDM Du et al. (2021) and the methods which are based on direct distribution matching: DAN Long et al. (2015), DRCN Ghifary et al. (2016), RevGrad Ganin & Lempitsky (2015a), JAN Long et al. (2017), JDDA Chen et al. (2019a), CADA-P Kurmi et al. (2019), ETD Li et al. (2020), MetaAlign Wei et al. (2021), and FixBi Na et al. (2021). For each benchmark dataset that we report our results, we included results of the above works if the original paper has reported performance on that dataset. In our Tables, bold font denotes the best performance among all methods. We report the pre-adaptation baseline performance in the first row of each table, followed by UDA methods based on adversarial learning, then followed by UDA methods based on direct matching. We report our result in the last rows.

5.3 Results

We have reported performances on the three digit recognition tasks in Table 1. We observe that IMUDA is quite competitive in these tasks. We also note that ETD, JDDA, and ETD, DWL which use secondary alignment mechanisms perform competitively. This observation suggests that using secondary alignment mechanisms is essential to improve current UDA methods given the performance levels of existing UDA methods.

We have provided the comparison result for the UDA tasks of the Office-31 dataset in Table 2. We observe that IMUDA on average leads to the best performance result. IMUDA also leads to the best results on two of the tasks and is quite competitive on the remaining tasks. This is likely because the "source only performance" for both of the $\mathcal{D} \rightarrow \mathcal{W}$ and $\mathcal{W} \rightarrow \mathcal{D}$ tasks are quite high, almost 100%. This means that the domain gap between these two tasks is relatively small to begin with, i.e., the two distributions are highly matched prior to model adaptation. As a result, this observation intuitively concludes that inducing larger margins is not going to be very helpful because the interclass margins are already relatively large in the target domain and are similar to the source domain. Hence, the margins are not violated prior to adaptation and we cannot benefit much from our algorithm to increase the margins further.

Results for the ImageCLEF-DA dataset are presented in Table 3. We see that IMUDA leads to a significant performance boost on this dataset, compared to prior work. We observe that we have about 7% boost on average compared to the next best performance. This performance is likely because the ImageCLEF-DA dataset is fully balanced in terms of the number of data points across both the domains and the classes. As a result, matching the internal distributions with a GMM distribution is more accurate. Because we have used the empirical distributions for domain alignment, a balanced source dataset makes the empirical source distribution more representative of the true source distribution in ImageCLEF-DA. The empirical interclass margin also has the same meaning across the domains because of data balance. We conclude that having a balanced training dataset in the source domain can boost our performance. One area for improving our algorithm is to replicate similar amount of improvement when the training dataset is imbalanced.

We have included the performance results for the single task of the VisDA2017 dataset in Table 4. We observe a competitive performance on this dataset. The large size of this dataset, makes the empirical probability distribution a more accurate representation of the true distribution. For this reason, empirical SWD loss can enforce domain alignment better. Quite intuitively, we conclude that having larger training datasets helps boosting our performance.

Method	$\mathcal{M} \rightarrow \mathcal{U}$	$\mathcal{U} \rightarrow \mathcal{M}$	$\mathcal{S} \rightarrow \mathcal{M}$	Method	$\mathcal{M} \rightarrow \mathcal{U}$	$\mathcal{U} \rightarrow \mathcal{M}$	$\mathcal{S} \rightarrow \mathcal{M}$
GtA Sankaranarayanan et al. (2018)	92.8 ± 0.9	90.8 ± 1.3	92.4 ± 0.9	CDAN Long et al. (2018)	93.9	96.9	88.5
ADDA Tzeng et al. (2017)	89.4 ± 0.2	90.1 ± 0.8	76.0 ± 1.8	ETD Li et al. (2020)	96.4 ± 0.3	96.3 ± 0.1	97.9 ± 0.4
DWL Xiao & Zhang (2021)	**97.3**	97.4	**98.1**				
RevGrad Ganin & Lempitsky (2015a)	77.1 ± 1.8	73.0 ± 2.0	73.9	JDDA Chen et al. (2019a)	-	97.0 ±0.2	93.1±0.2
DRCN Ghifary et al. (2016)	91.8 ± 0.1	73.7 ± 0.4	82.0 ± 0.2	DMRL Wu et al. (2020)	96.1	**99.0**	96.2
Source Only	90.1±2.6	80.2±5.7	67.3±2.6	Ours	96.6 ± 0.4	98.3 ± 0.3	96.6 ± 0.9

Table 1: Performance comparison for UDA tasks between MINIST, USPS, and SVHN datasets.

Method	$\mathcal{A} \rightarrow \mathcal{W}$	$\mathcal{D} \rightarrow \mathcal{W}$	$\mathcal{W} \rightarrow \mathcal{D}$	$\mathcal{A} \rightarrow \mathcal{D}$	$\mathcal{D} \rightarrow \mathcal{A}$	$\mathcal{W} \rightarrow \mathcal{A}$	Average
Source Only He et al. (2016)	68.4 ± 0.2	96.7 ± 0.1	99.3 ± 0.1	68.9 ± 0.2	62.5 ± 0.3	60.7 ± 0.3	76.1
GtA Sankaranarayanan et al. (2018)	89.5 ± 0.5	97.9 ± 0.3	99.8 ± 0.4	87.7 ± 0.5	72.8 ± 0.3	71.4 ± 0.4	86.5
DANN Ganin et al. (2016)	82.0 ± 0.4	96.9 ± 0.2	99.1 ± 0.1	79.7 ± 0.4	68.2 ± 0.4	67.4 ± 0.5	82.2
ADDA Tzeng et al. (2017)	86.2 ± 0.5	96.2 ± 0.3	98.4 ± 0.3	77.8 ± 0.3	69.5 ± 0.4	68.9 ± 0.5	82.8
SymNets Zhang et al. (2019)	90.8 ± 0.1	98.8 ± 0.3	**100.0** ± .0	93.9 ± 0.5	74.6 ± 0.6	72.5 ± 0.5	88.4
MADA Pei et al. (2018)	82.0 ± 0.4	96.9 ± 0.2	99.1 ± 0.1	79.7 ± 0.4	68.2 ± 0.4	67.4 ± 0.5	82.2
CDAN Long et al. (2018)	93.1 ± 0.2	98.2 ± 0.2	**100.0** ± 0.0	89.8 ± 0.3	70.1 ± 0.4	68.0 ± 0.4	86.6
DMRL Wu et al. (2020)	90.8±0.3	99.0±0.2	**100**.0±0.0	93.4±0.5	73.0±0.3	71.2±0.3	87.9
DWL Xiao & Zhang (2021)	89.2	99.2	**100.0**	91.2	73.1	69.8	87.1
DAN Long et al. (2015)	80.5 ± 0.4	97.1 ± 0.2	99.6 ± 0.1	78.6 ± 0.2	63.6 ± 0.3	62.8 ± 0.2	80.4
DRCN Ghifary et al. (2016)	72.6 ± 0.3	96.4 ± 0.1	99.2 ± 0.3	67.1 ± 0.3	56.0 ± 0.5	72.6 ± 0.3	77.7
RevGrad Ganin & Lempitsky (2015a)	82.0 ± 0.4	96.9 ± 0.2	99.1 ± 0.1	79.7 ± 0.4	68.2 ± 0.4	67.4 ± 0.5	82.2
CADA-P Kurmi et al. (2019)	83.4±0.2	99.8±0.1	**100**.0±0	80.1±0.1	59.8±0.2	59.5±0.3	80.4
JAN Long et al. (2017)	85.4 ± 0.3	97.4 ± 0.2	99.8 ± 0.2	84.7 ± 0.3	68.6 ± 0.3	70.0 ± 0.4	84.3
JDDA Chen et al. (2019a)	82.6 ± 0.4	95.2 ± 0.2	99.7 ± 0.0	79.8 ± 0.1	57.4 ± 0.0	66.7 ± 0.2	80.2
ETD Li et al. (2020)	92.1	**100.0**	**100.0**	88.0	71.0	67.8 86.2	86.5
MetaAlign Wei et al. (2021)	93.0 ± 0.5	98.6 ± 0.0	**100** ± 0.0	94.5 ± 0.3	75.0 ± 0.3	73.6 ± 0.0	89.2
HCL Huang et al. (2021)	92.5	98.2	**100.0**	94.7	75.9	77.7	89.8
FixBi Na et al. (2021)	96.1±0.2	99.3±0.2	**100.0**±0.0	95.0±0.4	**78.7**±0.5	**79.4**±0.3	91.4
IMUDA	**99.6** ± 0.2	98.1 ± 0.2	99.6 ± 0.1	**99.0** ± 0.4	74.9 ± 0.4	78.7 ± 1.1	**91.7**

Table 2: Performance comparison for UDA tasks for Office-31 dataset.

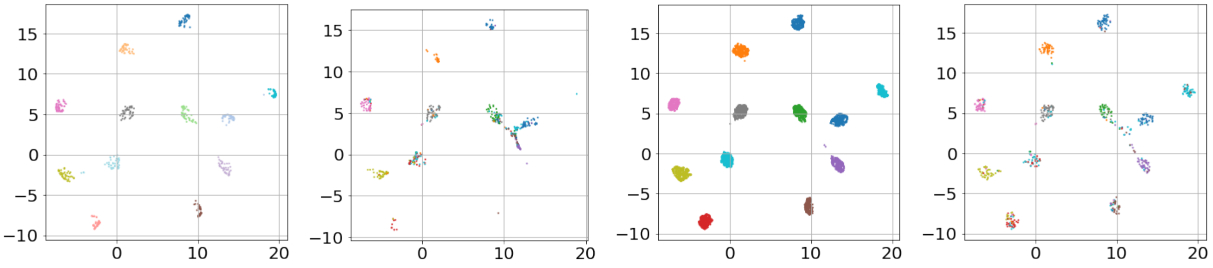
Method	$\mathcal{I} \rightarrow \mathcal{P}$	$\mathcal{P} \rightarrow \mathcal{I}$	$\mathcal{I} \rightarrow \mathcal{C}$	$\mathcal{C} \rightarrow \mathcal{I}$	$\mathcal{C} \rightarrow \mathcal{P}$	$\mathcal{P} \rightarrow \mathcal{C}$	Average
Source Only He et al. (2016)	74.8 ± 0.3	83.9 ± 0.1	91.5 ± 0.3	78.0 ± 0.2	65.5 ± 0.3	91.2 ± 0.3	80.8
DANN Ganin et al. (2016)	82.0 ± 0.4	96.9 ± 0.2	99.1 ± 0.1	79.7 ± 0.4	68.2 ± 0.4	67.4 ± 0.5	82.2
SymNets Zhang et al. (2019)	80.2 ± 0.3	93.6 ± 0.2	97.0 ± 0.3	93.4 ± 0.3	78.7 ± 0.3	96.4 ± 0.1	89.9
MADA Pei et al. (2018)	75.0 ± 0.3	87.9 ± 0.2	96.0 ± 0.3	88.8 ± 0.3	75.2 ± 0.2	92.2 ± 0.3	85.9
CDAN Long et al. (2018)	76.7 ± 0.3	90.6 ± 0.3	97.0 ± 0.4	90.5 ± 0.4	74.5 ± 0.3	93.5 ± 0.4	87.1
DMRL Wu et al. (2020)	77.3±0.4	90.7±0.3	97.4±0.3	91.8±0.3	76.0±0.5	94.8±0.3	88.0
DWL Xiao & Zhang (2021)	82.3	94.8	98.1	92.8	77.9	97.2	90.5
DAN Long et al. (2015)	74.5 ± 0.4	82.2 ± 0.2	92.8 ± 0.2	86.3 ± 0.4	69.2 ± 0.4	89.8 ± 0.4	82.4
RevGrad Ganin & Lempitsky (2015a)	75.0 ± 0.6	86.0 ± 0.3	96.2 ± 0.4	87.0 ± 0.5	74.3 ± 0.5	91.5 ± 0.6	85.0
JAN Long et al. (2017)	76.8 ± 0.4	88.0 ± 0.2	94.7 ± 0.2	89.5 ± 0.3	74.2 ± 0.3	91.7 ± 0.3	85.8
CADA-P Kurmi et al. (2019)	78.0	90.5	96.7	92.0	77.2	95.5	88.3
ETD Li et al. (2020)	81.0	91.7	97.9	93.3	79.5	95.0	89.7
CGDM Du et al. (2021)	78.7 ± 0.2	93.3 ± 0.1	97.5 ± 0.3	92.7 ± 0.2	79.2 ± 0.1	95.7 ± 0.2	89.5
IMUDA	**89.5** ± 1.2	**99.8** ± 0.2	**100** ± 0.0	**99.9** ± 0.1	**92.6** ± 0.9	**99.8** ± 0.2	**96.9**

Table 3: Performance comparison for UDA tasks for ImageCLEF-DA dataset.

From Tables 1–4, we can conclude that despite simplicity, IMUDA is a competitive UDA method when compared against the existing works given that it either outperforms the existing methods or its performance is close to the best method. Note that due to diversity of the benchmark UDA tasks, no single UDA method outperforms all the other existing UDA methods on all the major UDA tasks in the literature. Moreover, it is highly likely that performances for a given algorithm can change to some extend by finetuning the hyper-parameters/parameters and selecting different optimization techniques. Hence, those algorithms with close performances should be treated to have equally competitive performances. We can see that each algorithms leads to the best performance only on a subset of the tasks. The diversity of experiments helped identifying circumstances under which, our algorithm likely would work better.

Method	Plane	Bike	Bus	Car	Horse	Knife	Motor	Person	Plant	Skateboard	Train	Truck	Average
Source Only	70.6	51.8	55.8	68.9	77.9	7.6	93.3	34.5	81.1	27.9	88.6	5.6	55.3
DANN	81.9	77.7	82.8	44.3	81.2	29.5	65.1	28.6	51.9	54.6	82.8	7.8	57.4
RevGrad	75.9	70.5	65.3	17.3	72.8	38.6	58.0	77.2	72.5	40.4	70.4	44.7	58.6
JAN	92.1	66.4	81.4	39.6	72.5	70.5	81.5	70.5	79.7	44.6	74.2	24.6	66.5
GtA	-	-	-	-	-	-	-	-	-	-	-	-	77.1
CDAN	85.2	66.9	83.0	50.8	84.2	74.9	88.1	74.5	83.4	76.0	81.9	38.0	73.7
MCD	87.0	60.9	83.7	64.0	88.9	79.6	84.7	76.9	88.6	40.3	83.0	25.8	71.9
DMRL	-	-	-	-	-	-	-	-	-	-	-	-	75.5
DAN	68.1	15.4	76.5	87	71.1	48.9	82.3	51.5	88.7	33.2	88.9	42.2	61.1
DWL	90.7	80.2	86.1	67.6	92.4	81.5	86.8	78.0	90.6	57.1	85.6	28.7	77.1
CGDM	93.4	82.7	73.2	68.4	92.9	94.5	88.7	82.1	93.4	82.5	86.8	49.2	82.3
HCL	93.3	85.4	80.7	68.5	91.0	88.1	86.0	78.6	86.6	88.8	80.0	74.7	83.5
BiFix	96.1	**87.8**	90.5	**90.3**	**96.8**	95.3	92.8	**88.7**	**97.2**	94.2	90.9	25.7	**87.2**
IMUDA	**98.5**	63.9	**92.8**	74.9	84.4	**98.8**	**93.9**	86.1	92.7	**95.5**	**94.2**	**45.3**	85.1

Table 4: Performance for the VisDA UDA task.



(a) (b) (c) (d)

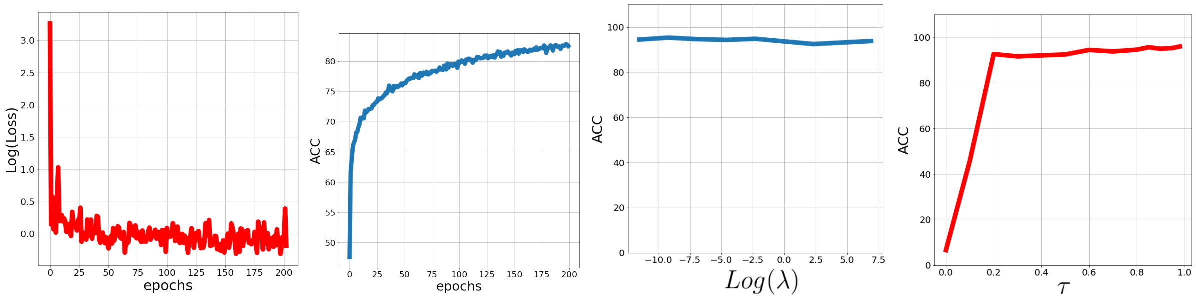
Figure 2: UMAP visualization for the representations of the dataset testing split for the $\mathcal{C} \to \mathcal{P}$ task: (a) the source domain (b) the target domain prior to adaptation, (c) samples drawn from the learned GMM, (d) the target domain after adaptation. (Best viewed enlarged on screen and in color).

5.4 Analytic and Ablative Analysis

We empirically analyzed our algorithm for further exploration and providing better understanding of its effect on data representation. We first checked the effect of the algorithm on the alignment of the distributions in the embedding space. We used the $\mathcal{C} \to \mathcal{P}$ task of the ImageClef-DA dataset for this purpose. After passing the testing split of the data for both domains into the embedding space, we used the UMAP McInnes et al. (2018) visualization tool to reduce the dimension of the data representations to two for 2D visualization. We have visualized the result in Figure 2. We have visualized the source domain testing split representations, samples that are drawn from the estimated GMM distribution, and the target domains testing split representations, prior and after performing unsupervised domain adaptation. In this figure, each point represents a single data point and each color represents one of the twelve classes. Comparing Figures 2a and 2c, we observe that high-confidence GMM samples match the source domain distribution quite well for this task which suggests GMM is a good parametric model for the source distribution. Comparing Figure 2b with Figure 2a, we observe that domain shift has led to overlapping clusters in the target domain, similar to the qualitative visualization in Figure 1. Finally, Figure 2d demonstrate that the IMUDA algorithm successfully has aligned the distribution of the target domain with the distribution of the source domain,mitigating domain shift. This empirical observations supports the intuition behind our approach for developing the algorithm, visualized in Figure 1.

For further investigation, we have plotted the training loss function value in equation 1 as well as the classification performance on the testing split of the VisDA task versus the number of optimization epochs in Figures 3a and 3b, respectively. Comparing the two curves, we observe that as the optimization objective loss function decreases, i.e., the distance between the two empirical distributions decreases, model generalization on the target domain increases. This observation confirms that IMUDA algorithm implements the desired domain alignment effect for UDA and accords nicely with our reasoning that more domain alignment can lead to better performance on the target domain data.

We have studied the effect of the values of the primary hyper-parameters of the algorithm on UDA performance in Figure 3c and Figure 3d. An advantage of our algorithm over some of the existing UDA methods is that IMUDA only has two primary hyper-parameters: λ and τ. We have plotted the classification accuracy versus varying values



(a) (b) (c) (d)

Figure 3: Empirical analysis based on the VisDA task: (a) loss function on the training split versus #epochs and (b) learning curve for the testing split versus #epochs; Effect of parameter values for the $\mathcal{C} \rightarrow \mathcal{P}$ task (c) performance versus the trade-off parameter λ and (d) classification accuracy versus the confidence parameter τ. (Best viewed enlarged on screen and in color).

Method	$\mathcal{A} \rightarrow \mathcal{W}$	$\mathcal{D} \rightarrow \mathcal{W}$	$\mathcal{W} \rightarrow \mathcal{D}$	$\mathcal{A} \rightarrow \mathcal{D}$	$\mathcal{D} \rightarrow \mathcal{A}$	$\mathcal{W} \rightarrow \mathcal{A}$	Average
3^{rd} Term	70.2 ± 0.9	91.3 ± 1.9	96.8 ± 0.5	77.4 ± 0.4	58.1±0.3	57.7 ± 0.4	75.3
4^{th} Term	99.3 ± 0.3	95.1 ± 0.5	98.9 ± 0.1	99.0± 0.1	77.0 ±0.5	77.8 ±0.1	91.2
IMUDA	99.6 ± 0.2	98.1 ± 0.2	99.6 ± 0.1	99.0 ± 0.4	74.9 ± 0.4	78.7 ± 1.1	91.7

Table 5: Ablative Experiments on Optimization Terms Using the Office-31 dataset.

of the hyper-parameter λ in Figure 3c. We can see that the performance is relatively constant with respect to this parameter. This is expected because the ERM terms in equation 10 are already small prior to the UDA optimization due to the pretraining step. Hence, when the UDA optimization is performed, it mainly minimizes the alignment loss terms. Figure 3d presents the classification accuracy versus varying values of the confidence hyper-parameter τ. As predictable from prior discussion, we observe that larger values for τ improve the performance. Very Small values for τ can degrade the performance because the low-confidence GMM samples can potentially behave as outliers, making domain alignment even more challenging compared to simple UDA based on distribution matching. Quite importantly, this experiments also serves as an ablation study to confirm that using high-confidence samples is indeed critical for our algorithm to improve model generalization. We can see as we use larger τ, i.e., a larger interclass margin in the source domain, the UDA performance also increases. This observation conforms that our secondary mechanism to induce larger margin is indeed effective. We also observe that after $\tau \approx 0.2$, increasing τ is not as helpful. As we can see, this observation is likely because the performance is already high which means the margins are already high. Note, however, the performance still is increasing with a smaller slope for $\tau > 0.2$. Finally, we have performed ablative experiments on the optimization terms in equation 1. Note that our experiments on the parameter λ, already inlcuded the effect of dropping the 1^{st} and the 2^{nd} terms in equation 1. Hence, we have studied effect of dropping the 3^{rd} and the 4^{th} terms to study the effect of each term on performance using the Office-31 dataset. The results are presented in Table 5. Comparing the performances with IMUDA, we observe that both terms contribute to the optimal performance, but the 3^{rd} term is more crucial. This observation is expected because the 3^{rd} term enforces alignment of the target domain with the pseudo-dataset directly. However, the 4^{th} term is helpful to induce larger margins. We conclude that the 4^{th} term helps to implement a complementary mechanism to improve the UDA results.

6 Conclusions

In this paper, we developed a UDA algorithm by developing a mechanism that increases the interclass margins between the formed class clusters in an embedding space to reduce the effect of domain shift. The embedding space is modeled by responses of a neural network in its final layer. Increasing the interclass margins mitigates the effect of domain shift on the model generalization in the target domain. Our algorithm is based on learning the internal distribution of the source domain and use it to repulse the target domain data representations in the embedding space away from the class boundaries. We estimate this distribution as a parametric Gaussian mixture model (GMM). We then draw random samples with confident labels from the GMM and use them to induce larger interclass margins. Our empirical results suggest that our approach is effective and compares favorably against existing methods. We also conclude that secondary mechanisms are necessary to improve domain alignment for UDA given the current performance level of UDA algorithms. Future works includes performance improvement when the source dataset is imbalanced.

REFERENCES

Bharath Bhushan Damodaran, Benjamin Kellenberger, Rémi Flamary, Devis Tuia, and Nicolas Courty. Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation. In *ECCV*, pp. 447–463, 2018a. 1

Bharath Bhushan Damodaran, Benjamin Kellenberger, Rémi Flamary, Devis Tuia, and Nicolas Courty. Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation. In *Proceedings of ECCV*, pp. 447–463, 2018b. 2

Tianshi Cao, Marc T Law, and Sanja Fidler. A theoretical analysis of the number of shots in few-shot learning. In *International Conference on Learning Representations*, 2019. 3

Chao Chen, Zhihong Chen, Boyuan Jiang, and Xinyu Jin. Joint domain alignment and discriminative feature learning for unsupervised deep domain adaptation. In *Proceedings of the AAAI*, pp. 3296–3303, 2019a. 7, 8

Chaoqi Chen, Weiping Xie, Wenbing Huang, Yu Rong, Xinghao Ding, Yue Huang, Tingyang Xu, and Junzhou Huang. Progressive feature alignment for unsupervised domain adaptation. In *Proceedings of CVPR*, pp. 627–636, 2019b. 3

Nicolas Courty, Rémi Flamary, Devis Tuia, and Alain Rakotomamonjy. Optimal transport for domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(9):1853–1865, 2016. 2

Hal Daumé III. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pp. 256–263, 2007. 1

Sofien Dhouib, Ievgen Redko, and Carole Lartizien. Margin-aware adversarial domain adaptation with optimal transport. In *International Conference on Machine Learning*, pp. 2514–2524. PMLR, 2020. 2

Zhekai Du, Jingjing Li, Hongzu Su, Lei Zhu, and Ke Lu. Cross-domain gradient discrepancy minimization for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3937–3946, 2021. 7, 8

Alexander J Gabourie, Mohammad Rostami, Philip E Pope, Soheil Kolouri, and Kuyngnam Kim. Learning a domain-invariant embedding for unsupervised domain adaptation using class-conditioned distribution alignment. In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 352–359. IEEE, 2019. 1, 2, 4

Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016. 1, 7, 8

Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *Proceedings of Proceedings of ICML*, pp. 1180–1189, 2015a. 7, 8

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

Muhammad Ghifary, W Bastiaan Kleijn, Mengjie Zhang, David Balduzzi, and Wen Li. Deep reconstruction-classification networks for unsupervised domain adaptation. In *Proceedings of ECCV*, pp. 597–613. Springer, 2016. 7, 8

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of NeurIPS*, pp. 2672–2680, 2014. 1

A. Gretton, A. Smola, J. Huang, M. Schmittfull, K. Borgwardt, and B. Schölkopf. Covariate shift by kernel mean matching. 2009. 1

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of CVPR*, pp. 770–778, 2016. 1, 8

Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *Proceedings of ICML*, pp. 1989–1998. PMLR, 2018. 1

Jiaxing Huang, Dayan Guan, Aoran Xiao, and Shijian Lu. Model adaptation: Historical contrastive learning for unsupervised domain adaptation without source data. *Advances in Neural Information Processing Systems*, 34, 2021. 7, 8

Guoliang Kang, Lu Jiang, Yi Yang, and Alexander G Hauptmann. Contrastive adaptation network for unsupervised domain adaptation. In *Proceedings of CVPR*, pp. 4893–4902, 2019. 2

Minyoung Kim, Pritish Sahu, Behnam Gholami, and Vladimir Pavlovic. Unsupervised visual domain adaptation: A deep max-margin gaussian process approach. In *Proceedings of CVPR*, pp. 4380–4390, 2019. 2

Vinod Kumar Kurmi, Shanu Kumar, and Vinay P Namboodiri. Attending to discriminative certainty for domain adaptation. In *CVPR*, pp. 491–500, 2019. 7, 8

Tam Le, Makoto Yamada, Kenji Fukumizu, and Marco Cuturi. Tree-sliced variants of wasserstein distances. *Advances in neural information processing systems*, 32, 2019. 4

Chen-Yu Lee, Tanmay Batra, Mohammad Haris Baig, and Daniel Ulbricht. Sliced wasserstein discrepancy for unsupervised domain adaptation. In *Proceedings of CVPR*, pp. 10285–10295, 2019. 2

Mengxue Li, Yi-Ming Zhai, You-Wei Luo, Peng-Fei Ge, and Chuan-Xian Ren. Enhanced transport distance for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13936–13944, 2020. 2, 7, 8

Yanghao Li, Naiyan Wang, Jianping Shi, Xiaodi Hou, and Jiaying Liu. Adaptive batch normalization for practical domain adaptation. *Pattern Recognition*, 80:109–117, 2018. 2

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *Proceedings of Proceedings of ICML*, pp. 97–105, 2015. 1, 2, 7, 8

Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. In *Proceedings of the 34th Proceedings of ICML-Volume 70*, pp. 2208–2217. JMLR. org, 2017. 2, 7, 8

Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In *Proceedings of NeurIPS*, pp. 1640–1650, 2018. 7, 8

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. UMAP: Uniform manifold approximation and projection. *Jour. of Open Source Soft.*, 3(29):861, 2018. 9

Todd K Moon. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60, 1996. 5

Saeid Motiian, Marco Piccirilli, Donald A Adjeroh, and Gianfranco Doretto. Unified deep supervised domain adaptation and generalization. In *Proceedings of CVPR*, pp. 5715–5725, 2017. 2

Jaemin Na, Heechul Jung, Hyung Jin Chang, and Wonjun Hwang. Fixbi: Bridging domain spaces for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1094–1103, 2021. 7, 8

R. Neal. Slice sampling. *Annals of statistics*, pp. 705–741, 2003. 4

Yingwei Pan, Ting Yao, Yehao Li, Yu Wang, Chong-Wah Ngo, and Tao Mei. Transferrable prototypical networks for unsupervised domain adaptation. In *Proceedings of CVPR*, pp. 2239–2247, 2019. 1, 3

Zhongyi Pei, Zhangjie Cao, Mingsheng Long, and Jianmin Wang. Multi-adversarial domain adaptation. In *Proceedings Thirty-Second AAAI*, pp. 3934–3941, 2018. 7, 8

Mohammad Rostami. Lifelong domain adaptation via consolidated internal distribution. *Advances in Neural Information Processing Systems*, 34:11172–11183, 2021. 1

Mohammad Rostami, David Huber, and Tsai-Ching Lu. A crowdsourcing triage algorithm for geopolitical event forecasting. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pp. 377–381. ACM, 2018. 1

Mohammad Rostami, Soheil Kolouri, Eric Eaton, and Kyungnam Kim. Sar image classification using few-shot cross-domain transfer learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 0–0, 2019. 2

Kevin Roth, Aurelien Lucchi, Sebastian Nowozin, and Thomas Hofmann. Stabilizing training of generative adversarial networks through regularization. In *Proceedings of NeurIPS*, pp. 2018–2028, 2017. 1

Sam T Roweis. Em algorithms for pca and spca. In *Proceedings of NeurIPS*, pp. 626–632, 1998. 5

Swami Sankaranarayanan, Yogesh Balaji, Carlos D Castillo, and Rama Chellappa. Generate to adapt: Aligning domains using generative adversarial networks. In *Proceedings of CVPR*, pp. 8503–8512, 2018. 7, 8

Akash Srivastava, Lazar Valkov, Chris Russell, Michael U Gutmann, and Charles Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. In *Proceedings of NeurIPS*, pp. 3308–3318, 2017. 1

Serban Stan and Mohammad Rostami. Privacy preserving domain adaptation for semantic segmentation of medical images. *arXiv preprint arXiv:2101.00522*, 2021a. 2

Serban Stan and Mohammad Rostami. Unsupervised model adaptation for continual semantic segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 2593–2601, 2021b. 2

Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *Proceedings of ECCV*, pp. 443–450. Springer, 2016. 2

Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of CVPR*, pp. 7167–7176, 2017. 1, 7, 8

Guoqiang Wei, Cuiling Lan, Wenjun Zeng, and Zhibo Chen. Metaalign: Coordinating domain alignment and classification for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16643–16653, 2021. 7, 8

Yuan Wu, Diana Inkpen, and Ahmed El-Roby. Dual mixup regularized learning for adversarial domain adaptation. In *Proceedings of ECCV*, pp. 540–555. Springer, 2020. 7, 8

Ni Xiao and Lei Zhang. Dynamic weighted learning for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15242–15251, 2021. 7, 8

Minghao Xu, Jian Zhang, Bingbing Ni, Teng Li, Chengjie Wang, Qi Tian, and Wenjun Zhang. Adversarial domain adaptation with domain mixup. In *Proceedings of the AAAI*, volume 34, pp. 6502–6509, 2020. 2

Werner Zellinger, Thomas Grubinger, Edwin Lughofer, Thomas Natschläger, and Susanne Saminger-Platz. Central moment discrepancy (cmd) for domain-invariant representation learning. In *ICLR*, 2017. 2

Dejiao Zhang, Ramesh Nallapati, Henghui Zhu, Feng Nan, Cicero dos Santos, Kathleen McKeown, and Bing Xiang. Unsupervised domain adaptation for cross-lingual text labeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pp. 3527–3536, 2020. 3

Yabin Zhang, Hui Tang, Kui Jia, and Mingkui Tan. Domain-symmetric networks for adversarial domain adaptation. In *Proceedings of CVPR*, pp. 5031–5040, 2019. 7, 8