Contradistinguisher: A Vapnik's Imperative to Unsupervised Domain Adaptation

Sourabh Balgi and Ambedkar Dukkipati

Abstract—Recent domain adaptation works rely on an indirect way of first aligning the source and target domain distributions and then train a classifier on the labeled source domain to classify the target domain. However, the main drawback of this approach is that obtaining a near-perfect domain alignment in itself might be difficult/impossible (e.g., language domains). To address this, inspired by how humans use supervised-unsupervised learning to perform tasks seamlessly across multiple domains or tasks, we follow Vapnik's imperative of statistical learning that states any desired problem should be solved in the most direct way rather than solving a more general intermediate task and propose a direct approach to domain adaptation that does not require domain alignment. We propose a model referred to as Contradistinguisher that learns contrastive features and whose objective is to jointly learn to contradistinguish the unlabeled target domain in an unsupervised way and classify in a supervised way on the source domain. We achieve the state-of-the-art on Office-31, Digits and VisDA-2017 datasets in both single-source and multi-source settings. We demonstrate that performing data augmentation results in an improvement in the performance over vanilla approach. We also notice that the contradistinguish-loss enhances performance by increasing the shape bias.

Index Terms—Contrastive Feature Learning, Deep Learning, Domain Adaptation, Transfer Learning, Unsupervised Learning.

1 Introduction

The recent success of deep neural networks for supervised learning tasks in several areas like computer vision, speech, and natural language processing can be attributed to the models trained on large amounts of labeled data. However, acquiring massive amounts of labeled data in some domains can be very expensive or not possible at all. Additionally, the amount of time required for labeling the data to use existing deep learning techniques can be very high initially for a new domain. This is known as *cold-start*. On the contrary, cost-effective unlabeled data can be easily obtained in large amounts for most new domains. So, one can aim to transfer the knowledge from a labeled source domain to perform tasks on an unlabeled target domain. To study this, under the purview of transductive transfer learning, several approaches like domain adaptation, sample selection bias, co-variance shift have been explored in recent times.

Existing domain adaptation approaches mostly rely on domain alignment, i.e., align both domains so that they are superimposed and indistinguishable in the latent space. This domain alignment can be achieved in three main ways: (a) discrepancy-based methods [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], (b) reconstruction-based methods [19], [20] and (c) adversarial adaptation methods [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40].

These domain alignment strategies of indirectly addressing the task of unlabeled target domain classification have three main drawbacks. (i) The sub-task of obtaining a perfect alignment of the domain in itself might be impossible or very difficult due to large domain shifts (e.g., language domains). (ii) The use of multiple classifiers and/or GANs to align the distributions unnecessarily increases the complexity of the neural networks leading to over-fitting in many cases. (iii) Due to distribution alignment, the domain-specific information is lost as the domains get morphed.

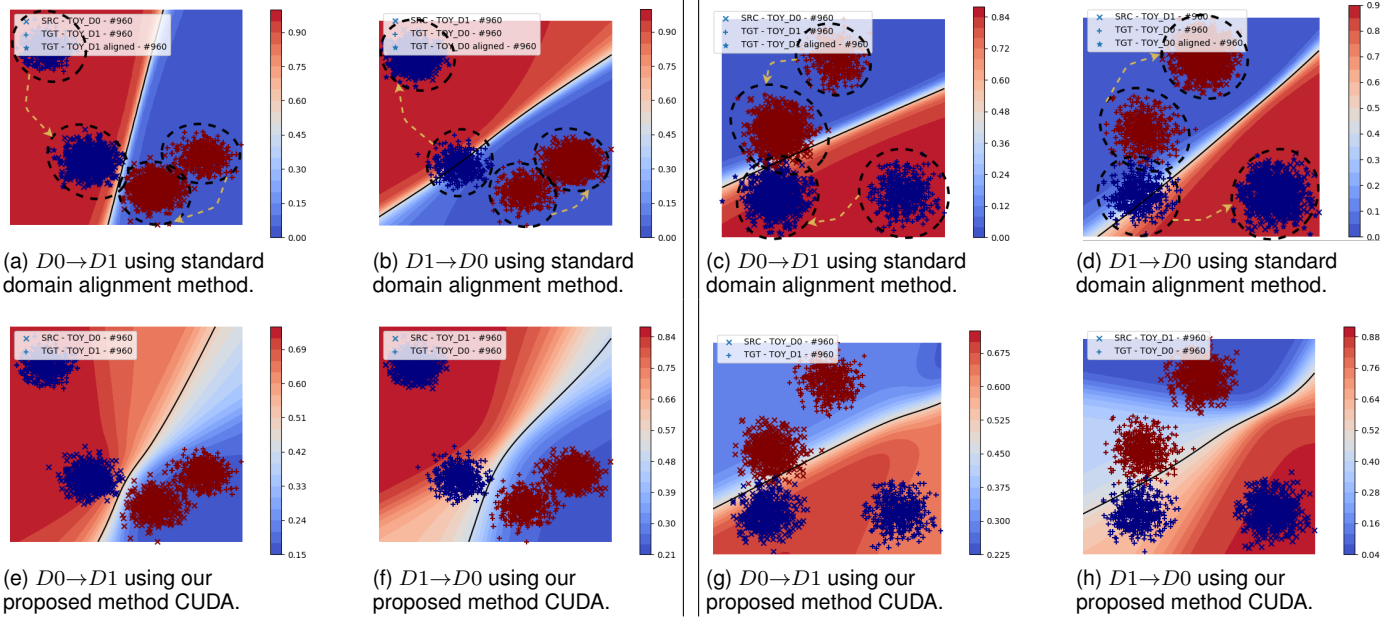
A particular case where the domain alignment and the classifier trained on the source domain might fail is that the target domain is more suited to classification tasks than the source domain with lower classification performance. In this case, it is advised to perform the classification directly on the unlabeled target domain in an unsupervised manner as domain alignment onto a less suited source domain only leads to loss of information. It is reasonable to assume that for the main objective of unlabeled target domain classification, one can use all the information in the target domain and optionally incorporate any useful information from the labeled source domain and not the other way around. These drawbacks push us to challenge the idea of solving domain adaptation problems without solving the general problem of domain alignment.

In this work, we study unsupervised domain adaptation by learning contrastive features in the unlabeled target domain in a fully unsupervised manner with the help of a classifier simultaneously trained on the labeled source domain. More importantly, we derive our motivation from the **Vapnik's imperative** that motivated the statistical learning theory [41], [42].

> *"When solving a given problem, try to avoid solving a more general problem as an intermediate step."* [42]

In the context of domain adaptation, the desired problem is classification on the unlabeled target domain, and domain

• *S. Balgi and A. Dukkipati (Contact Author) are with the Department of Computer Science and Automation, Indian Institute of Science, Bengaluru, Karnataka, India, 560012.*
E-mail: {sourabhbalgi, ambedkar}@iisc.ac.in



(a) $D0 \rightarrow D1$ using standard domain alignment method.

(b) $D1 \rightarrow D0$ using standard domain alignment method.

(c) $D0 \rightarrow D1$ using standard domain alignment method.

(d) $D1 \rightarrow D0$ using standard domain alignment method.

(e) $D0 \rightarrow D1$ using our proposed method CUDA.

(f) $D1 \rightarrow D0$ using our proposed method CUDA.

(g) $D0 \rightarrow D1$ using our proposed method CUDA.

(h) $D1 \rightarrow D0$ using our proposed method CUDA.

Fig. 1. Demonstration of difference in domain alignment and proposed method CUDA on the 2-dimensional blobs synthetic toy-dataset for domain distributions from popular $scikit-learn$ [46]. We provide two example settings of different data distributions for domains $D0$ and $D1$ in both the directions of domain adaptation $D0 \leftrightarrow D1$ separated by the vertical lines. In each setting, the top row corresponds to the domain alignment approach. The bottom row corresponds to the proposed method CUDA compared to their respective domain alignment in the top row in both $D0 \leftrightarrow D1$ domain adaptation tasks. The yellow dotted lines indicate the domain alignment process to superimpose the target domain onto the source domain, thereby morphing both the domains. similarly, the two sub-columns indicate the experiments with swapped source and target domains. Unlike the domain alignment approach, where the classifier is learnt only on source domain, CUDA demonstrates the contradistinguisher jointly learnt to classify on both the domains. As seen above, swapping domains affect the classifier learnt in domain alignment because the classifier depends on the source domain. However, because of joint learning on both the domains simultaneously, *contradistinguisher* shows almost the same decision boundary irrespective of the source domain, i.e., irrespective of the direction of the domain adaptation, i.e., $D0 \rightarrow D1$ or $D1 \rightarrow D0$. (Best viewed in color.)

alignment followed by most standard methods is the general intermediate. Considering the various drawback of the domain alignment approach, in this paper, we propose a method for domain adaptation that does not require domain alignment and approach the problem directly.

This work extends our earlier conference paper [43] in the following way. (i) We provide additional experimental results on more complex domain adaptation dataset Office-31 [44] which includes images from three different sources, AMAZON ($\mathcal{A}$), DSLR ($\mathcal{D}$) and WEBCAM ($\mathcal{W}$) categorized into three domains respectively with only a few labeled high-resolution images. (ii) We provide additional experimental results on the benchmark VisDA-2017 [45] dataset for unsupervised domain adaptation that includes synthetic and real-world images from 12 different classes. (iii) We provide several ablation studies and demonstrations that will provide insights into the working of our proposed method CUDA [43]. Additionally, we observed that the proposed contradistinguish loss helps to learn high-level features related to the shapes of the objects in the image. (iv) We extend our algorithm to the case of multi-source domain adaptation and establish benchmark results on Office-31 [44] dataset and Digits datasets.

A summary of our contributions in this paper is as follows.

1) We propose a simple method Contradistinguisher for Unsupervised Domain Adaptation (CUDA) that directly addresses the problem of domain adaptation by learning a single classifier, which we refer to as Contradistinguisher, jointly in an unsupervised manner over the unlabeled target domain and in a supervised manner over the labeled source domain. Hence, overcoming the drawbacks of distribution alignment-based techniques.
2) We formulate a 'contradistinguish loss' to directly utilize unlabeled target domain and address the classification task using unsupervised feature learning. Note that a similar approach called DisCoder [47] was used for a much simpler task of semi-supervised feature learning on a single domain with no domain distribution shift.
3) We extend our experiments to more complex domain adaptation datasets Office-31 [44] and VisDA-2017 [45]. From our experiments, we show that by jointly training contradistinguisher on the source domain and the target domain distributions, we can achieve above/on-par results over several recently proposed domain adaptation methods. We also observed an improvement in the classification performance on VisDA-2017 [45] over the vanilla CUDA with the data augmentation.
4) We further demonstrate our proposed method's simplicity and effectiveness by easily extending single-source domain adaptation to a more complex and general multi-source domain adaptation. We demonstrate the effectiveness of the multi-source domain adaptation extension by performing experiments on Office-31 [44] dataset and Digits datasets (USPS (us) [48], MNIST (mn) [49], SVHN (sv) [50], MNIST-M (mm) [21], SYN-NUMBERS (sn) [21]) in a multi-source setting.

5) Apart from these real-world benchmark datasets, we also validate the proposed method using the synthetically created toy-datasets (Fig. 1). From our toy-dataset experiments, we provide two main insights.
(i) CUDA does indeed address the classification directly on the target domain in a fully unsupervised way without the domain alignment.
(ii) Since the classification is done directly on the unlabeled target domain in a fully unsupervised manner, the target domain classification performance is not limited by the source domain classification performance, i.e., the irrespective of the domain is used as the labeled source domain and the unlabeled target domain, the performance is the respective domains are similar. In other words, swapping of the domains or the direction of the domain adaptation has little effect on the classification performance on each individual domain.

The rest of this paper is structured as follows. Section 2 discusses related works in domain adaptation. In Section 3, we elaborate on the problem formulation, neural network architecture used by us, loss functions, model training, and inference algorithms of our proposed method. Section 4 deals with the discussion of the experimental setup, results and analysis on visual datasets. Finally, in Section 5, we conclude by highlighting the key contributions of CUDA.

2 Related Work

As mentioned earlier, almost all domain adaptation approaches rely on domain alignment techniques. Here we briefly outline three main techniques of domain alignment.

(a) *Discrepancy-based methods*: Deep Adaptation Network (DAN) [1] proposes mean-embedding matching of multi-layer representations across domain by minimizing Maximum Mean Discrepancy (MMD) [51], [52], [53] in a reproducing kernel Hilbert space (RKHS). Residual Transfer Network (RTN) [2] introduces separate source and target domain classifiers differing by a small residual function along with fusing the features of multiple layers in a reproducing kernel Hilbert space (RKHS) to match the domain distributions. Joint Adaptation Network (JAN) [3] proposes to optimize Joint Maximum Mean Discrepancy (JMMD), which measures the Hilbert-Schmidt norm between kernel mean embedding of empirical joint distributions of source and target domain. Associative Domain Adaptation (ADA) [4] learns statistically domain invariant embeddings by associating the embeddings of the final fully-connected layer before applying softmax as an alternative to MMD loss.Maximum Classifier Discrepancy (MCD) [5] aligns source and target distributions by maximizing the discrepancy between two separate classifiers. Self Ensembling (SE) [6] uses mean teacher variant [54] of temporal ensembling [55] with heavy reliance on data augmentation to minimize the discrepancy between student and teacher network predictions. Variational Fair Autoencoder (VFAE) [7] uses Variational Autoencoder (VAE) [56] with MMD to obtain domain invariant features. Central Moment Discrepancy (CMD) [8] proposes to match higher-order moments of source and target domain distributions. Rozantsev et al. [9] propose to explicitly model the domain shift using two-stream architecture, one for each domain along with MMD to align the source and target representations. A more recent approach multi-domain Domain Adaptation layer (mDA-layer) [10], [11] proposes a novel idea of replacing standard Batch-Norm layers [57] with specialized Domain Alignment layers [12], [13] thereby reducing the domain shift by discovering and handling multiple latent domains. Geodesic Flow Subspaces (GFS/SGF) [14] performs domain adaptation by first generating two subspaces of the source and the target domains by performing PCA, followed by learning a finite number of the interpolated subspaces between source and target subspaces based on the geometric properties of the Grassmann manifold. In the presence of multi-source domains, this method is very effective as this identifies the optimal subspace for domain adaptation.sFRAME (sparse Filters, Random fields and Maximum Entropy) [15] models are defined as Markov random field model that model data distributions based on maximum entropy distribution to fit the observed data by identifying the patterns in the observed data. Transferrable Prototypical Networks (TPN) [16] propose to identify prototypes for each class in source and target domains that are close in the embedding space and minimize the distance between these prototypes for domain adaptation. Contrastive Adaptation Network (CAN) [17] uses MMD-loss for the feature encodings along with the heuristic clustering schema to selectively pick a subset of the high confidence image samples from the target domain. These samples are then utilized in training the classifier on the target domain instead of the entire target domain. Moment Matching for Multi-Source Domain Adaptation (M3SDA) [18] proposes to dynamically align multiple labeled source domains and the unlabeled target domain by matching the moments of the feature distributions.

(b) *Reconstruction-based methods*: Deep Reconstruction-Classification Networks (DRCN) [19] and Domain Separation Networks (DSN) [20] approaches learn shared encodings of source and target domains using reconstruction networks.

(c) *Adversarial adaptation methods*: Reverse Gradient (RevGrad) [21] or Domain Adversarial Neural Network (DANN) [22] uses domain discriminator to learn domain invariant representations of both the domains. Coupled Generative Adversarial Network (CoGAN) [23] uses Generative Adversarial Network (GAN) [58] to obtain domain invariant features used for classification. Adversarial Discriminative Domain Adaptation (ADDA) [24] uses GANs along with weight sharing to learn domain invariant features. Generate to Adapt (GTA) [25] learns to generate an equivalent image in the other domain for a given image, thereby learning common domain invariant embeddings. Cross-Domain Representation Disentangler (CDRD) [26] learns cross-domain disentangled features for domain adaptation. Symmetric Bi-Directional Adaptive GAN (SBADA-GAN) [27] aims to learn symmetric bidirectional mappings among the domains by trying to mimic a target image given a source image. Cycle-Consistent Adversarial Domain Adaptation (CyCADA) [28] adapts representations at both the pixel-level and feature-level over the domains. Moving Semantic Transfer Network (MSTN) [29] proposes a moving semantic transfer network that learns semantic representations for the unlabeled target samples by aligning labeled source centroids and pseudo-labeled target centroids. Con-

ditional Domain Adversarial Network (CDAN) [30] conditions the adversarial adaptation models on discriminative information conveyed in the classifier predictions. Decision-boundary Iterative Refinement Training with a Teacher (DIRT-T) [32] and Augmented Cyclic Adversarial Learning (ACAL) [33] learn by using a domain discriminator along with data augmentation for domain adaptation. Deep Cocktail Network (DCTN) [35] proposes a k-way domain discriminator and category classifier for digit classification and real-world object recognition in a multi-source domain adaptation setting. Batch Spectral Penalization (BSP) [36] investigates the transferability and the discriminability of the features learnt using the standard adversarial domain adaptation techniques. Also, BSP proposes an additional batch spectral loss as it is observed that the transferable features learnt using adversarial domain adaptation result in the loss of the discriminability of the classifier. Transferable Normalization (TransNorm) [37] proposes a further improvement in transferability by replacing the normal batch-normalization layer with separate normalization layers for source and target domain input batches. Adversarial Tight Match (ATM) [38] proposes to combine the adversarial training with discrepancy metric by introducing a novel discrepancy metric Maximum Density Divergence (MDD) to minimize inter-domain divergence and maximize the intra-class density. Certainty based Attention for Domain Adaptation (CADA) [39] propose to identify features that increase the certainty of the domain discriminator prediction to improve the classifier. Progressive Feature Alignment Network (PFAN) [40] proposes to align the discriminative features across domains progressively and effectively, via exploiting the intra-class variation in the target domain.

Apart from these approaches, a slightly different method that has been recently proposed is called *Tri-Training*. Tri-Training algorithms use three classifiers trained on the labeled source domain and refine them for the unlabeled target domain. To be precise, in each round of tri-training, a target sample is pseudo-labeled if the other two classifiers agree on the labeling, under certain conditions such as confidence thresholding. Asymmetric Tri-Training (ATT) [59] uses three classifiers to bootstrap high confidence target domain samples by confidence thresholding. This way of bootstrapping works only if the source classifier has very high accuracy. In the case of low source classifier accuracy, target samples are never obtained to bootstrap, resulting in a bad model. Multi-Task Tri-training (MT-Tri) [60] explores the tri-training technique on the language domain adaptation tasks in a multi-task setting.

All the domain adaptation approaches mentioned earlier have a common unifying theme: they attempt to morph the target and source distributions so as to make them indistinguishable. However, aligning domains is a complex task than the classification task. In this paper, we propose a completely different approach: instead of focusing on aligning the source and target distributions, we learn a single classifier referred to as *Contradistinguisher*, jointly on both the domain distributions using contradistinguish loss for the unlabeled target domain data and supervised loss for the labeled source domain data.

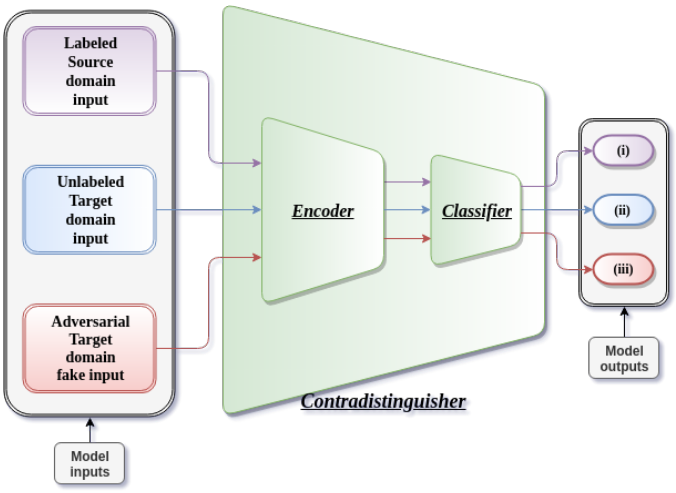


Fig. 2. Architecture of the proposed method CUDA with *Contradistinguisher* (*Encoder* and *Classifier*). Three optimization objectives with their respective inputs involved in training of CUDA: (i) Source supervised (2), (ii) Target unsupervised (5) and Adversarial regularization (9).

3 Proposed Method: CUDA

A domain $\mathcal{D}_d$ is specified by its input feature space $\mathcal{X}_d$, the label space $\mathcal{Y}_d$ and the joint probability distribution $p(\mathbf{x}_d, \mathbf{y}_d)$, where $\mathbf{x}_d \in \mathcal{X}_d$ and $\mathbf{y}_d \in \mathcal{Y}_d$. Let $|\mathcal{Y}_d| = K$ be the number of class labels such that $\mathbf{y}_d \in \{0, \ldots, K-1\}$ for any instance $\mathbf{x}_d$. Domain adaptation, in particular, consists of two domains $\mathcal{D}_s$ and $\mathcal{D}_t$ that are referred as the source and target domains respectively. We define $(\mathbf{x}_s, \mathbf{y}_s)$ as the random variables that denote the source domain input features and the corresponding source domain label. Similarly, we define $(\mathbf{x}_t, \mathbf{y}_t)$ as the random variables that denote the target domain input features and the corresponding target domain label. A common assumption in domain adaptation is that the input feature space as well as the label space remains unchanged across the source and the target domain, i.e., $\mathcal{X}_s = \mathcal{X}_t = \mathcal{X}_d$ and $\mathcal{Y}_s = \mathcal{Y}_t = \mathcal{Y}_d$. Hence, the only difference between the source and target domain is input-label space distributions, i.e., $p(\mathbf{x}_s, \mathbf{y}_s) \neq p(\mathbf{x}_t, \mathbf{y}_t)$. This is referred to as *domain shift* in the domain adaptation literature.

In particular, in an unsupervised domain adaptation, the training data consists of the labeled source domain instances $\{(\mathbf{x}_s^i, \mathbf{y}_s^i)\}_{i=1}^{n_s}$ corresponding to the random variables $(\mathbf{x}_s, \mathbf{y}_s)$ and the unlabeled target domain instances $\{\mathbf{x}_t^j\}_{j=1}^{n_t}$ corresponding to the random variables $(\mathbf{x}_t)$. Observe that in the unsupervised domain adaptation setting, the target domain labels $\{\mathbf{y}_t^j\}_{j=1}^{n_t}$ corresponding to the random variable $(\mathbf{y}_t)$ are unobserved/missing. Given a labeled data in the source domain, it is straightforward to learn a classifier by maximizing the conditional probability $p(\mathbf{y}_s|\mathbf{x}_s)$ over the labeled samples. However, the task at hand is to learn a classifier on the unlabeled target domain by transferring the knowledge from the labeled source domain to the unlabeled target domain.

3.1 Overview

The outline of the proposed method CUDA that involves *contradistinguisher* and the respective losses involved in

training are depicted in Fig. 2. The objective of contradistinguisher is to find a clustering scheme using the most contrastive features on unlabeled target in such a way that it also satisfies the target domain prior over the labels, i.e., *target domain prior enforcing*. We achieve this by jointly training on labeled source samples in a supervised manner and unlabeled target samples in an unsupervised end-to-end manner by using a contradistinguish loss same as [47].

This fine-tunes the classifier learnt from the source domain also to the target domain, as demonstrated in Fig. 1. The crux of our approach is the contradistinguish loss (5) which is discussed in detail in Section 3.3. Hence, the apt name *contradistinguisher* for our neural network architecture.

Note that the objective of contradistinguisher is not the same as a classifier, i.e., distinguishing is not the same as classifying. Suppose there are two contrastive entities $e_1 \in C_1$ and $e_2 \in C_2$, where C_1, C_2 are two classes. The aim of a classifier is to classify $e_1 \in C_1$ and $e_2 \in C_2$, where to train a classifier one requires labeled data. On the contrary, the job of contradistinguisher is to just identify $e_1 \neq e_2$, i.e., contradistinguisher can classify $e_1 \in C_1$ (or C_2) and $e_2 \in C_2$ (or C_1) indifferently. To train contradistinguisher, we do not need any class information but only need unlabeled entities e_1 and e_2. Using unlabeled target data, contradistinguisher is able to find a clustering scheme by distinguishing the unlabeled target domain samples in an unsupervised way. However, since the final task is classification, one would require a selective incorporation of the pre-existing informative knowledge required for the task of classification. This knowledge of assigning the label to the clusters is obtained by jointly training, thus classifying $e_1 \in C_1$ and $e_2 \in C_2$.

3.2 Supervised Source Classification

For the labeled source domain instances $\{(\mathbf{x}_s^i, \mathbf{y}_s^i)\}_{i=1}^{n_s}$ corresponding to the random variables $(\mathbf{x}_s, \mathbf{y}_s)$ of the labeled source domain, we define the conditional-likelihood of observing $\mathbf{y}_s$ given $\mathbf{x}_s$ as, $p_\theta(\mathbf{y}_s|\mathbf{x}_s)$, where θ denotes the parameters of contradistinguisher.

We estimate θ by maximizing the conditional log-likelihood of observing the labels given the labeled source domain samples. Therefore, the source domain supervised objective to maximize is given as

$$\mathcal{L}_s(\theta) = \sum_{i=1}^{n_s} \log(p_\theta(\mathbf{y}_s^i|\mathbf{x}_s^i)) \ . \tag{1}$$

Alternatively, one can minimize the cross-entropy loss, as used in practical implementation, instead of maximizing (1), i.e.,

$$\mathcal{L}_{ce}(\theta) = -\sum_{i=1}^{n_s} \sum_{k=0}^{K-1} \mathbb{1}[\mathbf{y}_s^i{=}k] \log(\hat{\mathbf{y}}_s^{ik}) \ , \tag{2}$$

where $\hat{\mathbf{y}}_s^{ik}$ is the softmax output of contradistinguisher that represents the probability of class k for the given sample $\mathbf{x}_s^i$.

3.3 Unsupervised Target Classification

For the unlabeled target domain instances $\{\mathbf{x}_t^j\}_{j=1}^{n_t}$ corresponding to the random variable $\mathbf{x}_t$ of the unlabeled target domain, the corresponding labels $\{\mathbf{y}_t^j\}_{j=1}^{n_t}$ corresponding to the random variable $\mathbf{x}_t$ are unknown/missing. Hence, a naive way of predicting the target labels is to directly use the classifier trained only with a supervised loss given in (2). While this approach may perform reasonably well in certain cases, it fails to deliver state-of-the-art performance. This may be attributed to the following reason: the support for the distribution p_θ is defined only over the source domain instances $\mathbf{x}_s$ and not the target domain instances $\mathbf{x}_t$. Hence, we model a non-trivial joint distribution $\hat{q}_\theta(\mathbf{x}_t, \mathbf{y}_t)$ parameterized by the same θ over target domain with only the target domain instances as the support as,

$$\hat{q}_\theta(\mathbf{x}_t, \mathbf{y}_t) = \frac{p_\theta(\mathbf{y}_t|\mathbf{x}_t)}{\sum_{\ell=1}^{n_t} p_\theta(\mathbf{y}_t|\mathbf{x}_t^\ell)} \ . \tag{3}$$

However (3) is not a joint distribution yet because $\sum_{\ell=1}^{n_t} \hat{q}_\theta(\mathbf{x}_t^\ell, \mathbf{y}_t) \neq p(\mathbf{y}_t)$, i.e., marginalizing over all $\{\mathbf{x}_t^j\}_{j=1}^{n_t}$ does not yield the *target prior distribution*, i.e., $p(\mathbf{y}_t)$. We modify (3) so as to include the marginalization condition. Hence, we refer to this as *target domain prior enforcing*.

$$q_\theta(\mathbf{x}_t, \mathbf{y}_t) = \frac{p_\theta(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{y}_t)}{\sum_{\ell=1}^{n_t} p_\theta(\mathbf{y}_t|\mathbf{x}_t^\ell)} \ , \tag{4}$$

where $p(\mathbf{y}_t)$ denotes the target domain prior probability over the labels.

Note that $q_\theta(\mathbf{x}_t, \mathbf{y}_t)$ defines a non-trivial approximate of joint distribution over the target domain as a function of p_θ learnt over source domain. The resultant unsupervised maximization objective for the target domain is given by maximizing the log-probability of the joint distribution $q_\theta(\mathbf{x}_t, \mathbf{y}_t)$ which is

$$\mathcal{L}_t(\theta, \{\mathbf{y}_t^j\}_{j=1}^{n_t}) = \sum_{j=1}^{n_t} \log(q_\theta(\mathbf{x}_t^j, \mathbf{y}_t^j)) \ , \tag{5}$$

Next, we discuss how the objective given in (5) is solved, and the reason why (5) is referred to as contradistinguish loss. Since the target labels $\{\mathbf{y}_t^j\}_{j=1}^{n_t}$ are unknown, one needs to maximize (5) over the parameters θ as well as the unknown target labels $\mathbf{y}_t$. As there are two unknown variables for maximization, we follow a two-step approach to maximize (5) as analogous to Expectation-Maximization (EM) algorithm [61]. The two optimization steps are as follows.

(i) *Pseudo-label selection*: We maximize (5) only with respect to the label $\mathbf{y}_t$ for every $\mathbf{x}_t$ by fixing θ as

$$\hat{\mathbf{y}}_t^j = \underset{\mathbf{y}^j \in \mathcal{Y}_t}{\arg\max} \frac{p_\theta(\mathbf{y}^j|\mathbf{x}_t^j)p(\mathbf{y}_t)}{\sum_{\ell=1}^{n_t} p_\theta(\mathbf{y}^\ell|\mathbf{x}_t^\ell)} \ , \tag{6}$$

Pseudo-labeling approach under semi-supervised representation learning setting has been well studied in [62] and shown equivalent to *entropy regularization* [63]. As previously mentioned, the pseudo-label selection is analogous to E-step in the EM algorithm. Moreover, we derive the motivation from [47] that also uses pseudo-labeling in the context of semi-supervised representation learning. However, the proposed method addresses a more complex problem of domain adaptation in the presence of a domain shift. The pseudo-labeling essentially tries to cluster by assigning labels using source domain features of the classifier trained

on the source domain. This is effectively is similar to the E-step in EM algorithm in spirit.

(ii) *Maximization*: By fixing the pseudo-labels $\{\hat{\mathbf{y}}_t^j\}_{j=1}^{n_t}$ from (6), we train contradistinguisher to maximize (5) with respect to the parameter θ.

$$\mathcal{L}_t(\theta) = \sum_{j=1}^{n_t} \log(p_\theta(\hat{\mathbf{y}}_t^j|\mathbf{x}_t^j)) + \sum_{j=1}^{n_t} \log(p(\mathbf{y}_t)) - \sum_{j=1}^{n_t} \log(\sum_{\ell=1}^{n_t} p_\theta(\hat{\mathbf{y}}_t^j|\mathbf{x}_t^\ell)) \ . \quad (7)$$

Since the pseudo-labels from (6) are used for the maximization, this constrains the model to learn the features to further improve the current pseudo-labeling for the next iteration. This step is similar to the M-step in the EM algorithm in spirit.

The first term, i.e., log-probability for a label $\hat{\mathbf{y}}_t^j$ given $\mathbf{x}_t^j$ forces contradistinguisher to choose features to classify $\mathbf{x}_t^j$ to $\hat{\mathbf{y}}_t^j$. The second term is a constant, hence it has no effect on the optimization with respect to θ. The third term is the negative of log-probability for the pseudo-label $\hat{\mathbf{y}}_t^j$ given all the samples $\mathbf{x}_t^\ell$ in the entire domain. Maximization of this term forces contradistinguisher to choose features to not classify all the other $\mathbf{x}_t^{\ell \neq j}$ to selected pseudo-label $\hat{\mathbf{y}}_t^j$ except the given sample $\mathbf{x}_t^j$. This forces contradistinguisher to extract the most unique features of a given sample $\mathbf{x}_t^j$ against all the other samples $\mathbf{x}_t^{\ell \neq j}$, i.e., most unique contrastive feature of the selected sample $\mathbf{x}_t^j$ over all the other samples $\mathbf{x}_t^{\ell \neq j}$ to distinguish a given sample from all others.

The first and third term together in (7) enforce that contradistinguisher learns the most contradistinguishing features among the samples $\mathbf{x}_t \in \mathcal{X}_t$, thus performing unlabeled target domain classification in a fully unsupervised way. We refer to the unsupervised target domain objective (5) as contradistinguish loss because of this contradistinguishing feature learning.

Ideally, one would like to compute the third term in (7) using the complete target training data for each input sample. Since it is expensive to compute the third term over the entire $\mathbf{x}_t$ for each individual sample during training, one evaluates the third term in (7) over a mini-batch. In our experiments, we have observed that the mini-batch strategy does not cause any problem during training as far as it includes at least one sample from each class, which is a fair assumption for a reasonably large mini-batch size of 128. For numerical stability, we use $\log \sum \exp$ trick to optimize third term in (7).

3.4 Adversarial Regularization

To prevent contradistinguisher from over-fitting to the chosen pseudo labels during the training, we use adversarial regularization. In particular, we train contradistinguisher to be confused about the set of fake negative samples $\{\hat{\mathbf{x}}_t^j\}_{j=1}^{n_f}$ by maximizing the conditional log-probability over the given fake sample such that the sample belongs to all $K(|\mathcal{Y}_d|)$ classes simultaneously. The adversarial regularization objective is to multi-label the fake sample (e.g., a noisy image that looks like a cat and a dog) equally to all K classes as labeling to any unique class introduces more noise in pseudo labels. This strategy is similar to entropy regularization [63] in the sense that instead of minimizing the entropy for the real target samples, we maximize the conditional log-probability over the fake negative samples. Therefore, we add the following maximization objective to the total contradistinguisher objective as a regularizer.

$$\mathcal{L}_{adv}(\theta) = \sum_{j=1}^{n_f} \log(p_\theta(\hat{\mathbf{y}}_t^j|\hat{\mathbf{x}}_t^j)) \ , \quad (8)$$

for all $\hat{\mathbf{y}}_t^j \in \mathcal{Y}_t$. As maximization of (8) is analogous to minimizing the binary cross-entropy loss (9) of a multi-class multi-label classification task, in our practical implementation, we minimize (9) for assigning labels to all the classes for every sample.

$$\mathcal{L}_{bce}(\theta) = -\sum_{j=1}^{n_f} \sum_{k=0}^{K-1} \log(\hat{\mathbf{y}}_t^{jk}) \ , \quad (9)$$

where $\hat{\mathbf{y}}_t^{jk}$ is the softmax output of contradistinguisher which represents the probability of class k for the given sample $\hat{\mathbf{x}}_t^j$.

The fake negative samples $\hat{\mathbf{x}}_t$ can be directly sampled from, say, a Gaussian distribution in the input feature space $\mathcal{X}_t$ with the mean and standard deviation of the samples $\mathbf{x}_t \in \mathcal{X}_t$. For the language domain, fake samples are generated randomly, as mentioned above, because the input feature is the form of embeddings extracted from denoising auto-encoder with bag-of-words as the input. In case of visual datasets, as the feature space is high dimensional, the fake images $\hat{\mathbf{x}}_t$ are generated using a generator network G_ϕ with parameter ϕ that takes Gaussian noise vector η_t as input to produce a fake sample $\hat{\mathbf{x}}_t$, i.e., $\hat{\mathbf{x}}_t = G_\phi(\eta_t)$. Generator G_ϕ is trained by minimizing kernel-MMD loss [64], i.e., a modified version of MMD loss between the encoder output $\rho_{enc}(\hat{\mathbf{x}}_t)$ and $\rho_{enc}(\mathbf{x}_t)$ of n_f fake images $\hat{\mathbf{x}}_t$ and n_t real target domain images $\mathbf{x}_t$ respectively.

$$\mathcal{L}_{gen}(\phi) = \frac{1}{n_f^2} \sum_{i=1}^{n_f} \sum_{j=1}^{n_f} k(\rho_{enc}(\hat{\mathbf{x}}_t^i), \rho_{enc}(\hat{\mathbf{x}}_t^j)) + \frac{1}{n_t^2} \sum_{i=1}^{n_t} \sum_{j=1}^{n_t} k(\rho_{enc}(\mathbf{x}_t^i), \rho_{enc}(\mathbf{x}_t^j)) - \frac{2}{n_t n_f} \sum_{i=1}^{n_f} \sum_{j=1}^{n_t} k(\rho_{enc}(\hat{\mathbf{x}}_t^i), \rho_{enc}(\mathbf{x}_t^j)), \quad (10)$$

where $k(x, x') = e^{-\gamma \|x - x'\|^2}$ is the Gaussian kernel.

Note that the generator's objective is not to generate realistic images but to generate fake noisy images with mixed image attributes from the target domain. This reduces the effort of training powerful generators, which is the focus in adversarial based domain adaptation approaches [25], [26], [27], [28], [29] used for domain alignment. Algorithm 1 and 2 list steps involved in CUDA training and inference, respectively.

3.5 Extension to Multi-Source Domain Adaptation

Here, we argue that our proposed method CUDA has an implicit advantage in dealing with multi-source domain adaption problems over the techniques based on domain

Algorithm 1: CUDA Training

Input: b=$batch\_size$, $epochs$=max_epoch, n_{batch}=$number\ of\ batches$
Output: θ /* parameter of contradistinguisher */
Data: $\{(\mathbf{x}_s^i, \mathbf{y}_s^i)\}_{i=1}^{n_s}$, $\{\mathbf{x}_t^j\}_{j=1}^{n_t}$

1 **if** *target domain prior* $p(\mathbf{y}_t)$ *is known* **then**
2 use $p(\mathbf{y}_t)$ for the contradistinguish loss (5) /* target domain prior enforcing */
3 **else**
4 compute $p(\mathbf{y}_t)$ assuming $p(\mathbf{y}_t) = p(\mathbf{y}_s)$ /* fair assumption as most datasets are well balanced */
5 **for** $epoch = 1$ *to* $epochs$ **do**
6 **for** $batch = 1$ *to* n_{batch} **do**
7 sample a mini-batch $\{(\mathbf{x}_s^i, \mathbf{y}_s^i)\}_{i=1}^{b}$, $\{\mathbf{x}_t^j\}_{j=1}^{b}$
8 compute $\mathcal{L}_s(\theta)$ (1) using $\{(\mathbf{x}_s^i, \mathbf{y}_s^i)\}_{i=1}^{b}$ /* source supervised loss */
9 compute $\{\hat{\mathbf{y}}_t^j\}_{j=1}^{b}$ (6) using $\{\mathbf{x}_t^j\}_{j=1}^{b}$ /* *pseudo label selection* step */
10 compute $\mathcal{L}_t(\theta)$ (7) fixing $\{\hat{\mathbf{y}}_t^j\}_{j=1}^{b}$ /* *maximization* step */ /* steps 9 and 10 together optimize unsupervised contradistinguish loss (5) */
11 **if** *adversarial regularization is enabled* **then**
12 **if** *Generator* G_ϕ *is used* **then**
13 get fake samples $\{\hat{\mathbf{x}}_t^j\}_{j=1}^{b}$ from Gaussian noise vectors $\{\eta_t^j\}_{j=1}^{b}$ using G_ϕ, compute $\mathcal{L}_{gen}(\phi)$(10) /* generator training */
14 **else**
15 get fake samples $\{\hat{\mathbf{x}}_t^j\}_{j=1}^{b}$ by random sampling in the input feature space $\mathcal{X}_t$
16 compute $\mathcal{L}_{adv}(\theta)$ (9) using $\{\hat{\mathbf{x}}_t^j\}_{j=1}^{b}$ /* fake samples are assigned to all classes equally */
17 combine losses in steps 8,10,13 and 16 to compute gradients using back-propagation
18 update θ using gradient descent /* and ϕ if G_ϕ is used */

Algorithm 2: CUDA Inference

Input: $\{\mathbf{x}_{test}^i\}_{i=1}^{n_{test}}$ /* input test samples */
Output: $\{\hat{\mathbf{y}}_{test}^i\}_{i=1}^{n_{test}}$ /* predicted labels */

1 **for** $i = 1$ *to* n_{test} **do**
2 predict label as $\hat{\mathbf{y}}_{test}^i = \arg\max_{\mathbf{y} \in \mathcal{Y}_t} p_\theta(\mathbf{y}|\mathbf{x}_{test}^i)$

alignment. In a multi-source adaption setting, domain alignment methods need to consider the domain-shift between a source and a target domain and consider the domain-shift between the multiple source domains. Therefore, domain alignment methods are required to solve the even more complex intermediate problem of aligning multiple source and target domain distributions in addition to the complex intermediate problem of source and target domain alignment to deal with the multi-source domain adaptation problems. However, as the proposed method does not depend on the domain alignment, the extension to multi-source in the proposed method is very simple. As our main focus is to perform unsupervised learning directly on the target domain, the model obtained using is better generalized to the target domain and reduces overfitting on the source, which usually results in a negative transfer. We believe that this is one of the main advantages of addressing the domain adaptation by performing the primary task of target domain classification rather than the intermediate task of domain alignment.

We propose a simple extension our proposed method to perform multi-source domain adaptation in the following manner. Let us suppose we are given with R source domains $\{s_1, \ldots, s_R\}$, consisting of labeled training data $(\{(\mathbf{x}_{s_1}^i, \mathbf{y}_{s_1}^i)\}_{i=1}^{n_{s_1}}, \ldots, \{(\mathbf{x}_{s_R}^i, \mathbf{y}_{s_R}^i)\}_{i=1}^{n_{s_R}})$ and unlabeled target domain instances $\{\mathbf{x}_t^j\}_{j=1}^{n_t}$. We compute the source supervised loss for the r^{th} source domain using (2), i.e., $\mathcal{L}_{s_r}(\theta)$ (1) with $\{(\mathbf{x}_{s_r}^i, \mathbf{y}_{s_r}^i)\}_{i=1}^{n_{sr}}$ training data. We further compute the total multi-source supervised loss as

$$\mathcal{L}_{s_{total}}(\theta) = \sum_{r=1}^{R} \mathcal{L}_{s_r}(\theta). \tag{11}$$

We replace $\mathcal{L}_s(\theta)$ (1) in the total optimization objective with $\mathcal{L}_{s_{total}}(\theta)$ (11) in step 17 of Algorithm 1. It should be noted that the unsupervised loss for the target domain is still unmodified irrespective of the number of source domains. We experimentally demonstrate the efficacy of the proposed multi-source domain adaptation extension on Office-31 [44] and Digits datasets [21], [48], [49], [50].

4 Experiments

For our domain adaptation experiments, we consider both synthetic and real-world datasets. Under synthetic datasets, we experiment using 2-dimensional blobs with different source and target domain probability distributions to demonstrate the effectiveness of the proposed method under different domain shifts. Under real-world datasets, we consider only the complex, high-resolution Office-31 [44] and VisDA-2017 [45] object classification datasets for our experiment as the low-resolution datasets are already addressed in our conference paper CUDA: Contradistinguisher for Unsupervised Domain Adaptation (CUDA) [43]. We have published our python code for all the experiments at https://github.com/sobalgi/cuda.

Table 1 provides details on the visual datasets used in our experiments. We also experiment and report the results of our ablation study carried out with different combinations of the three optimization objectives with their respective domains as the inputs involved in CUDA training:
(i) source supervised loss: ss described in (2),
(ii) source/target unsupervised loss: su/tu described in (5),
(iii) source/target adversarial regularization loss: sa/ta described in (9).

ss indicates the minimum target domain test accuracy that can be attained with a chosen contradistinguisher

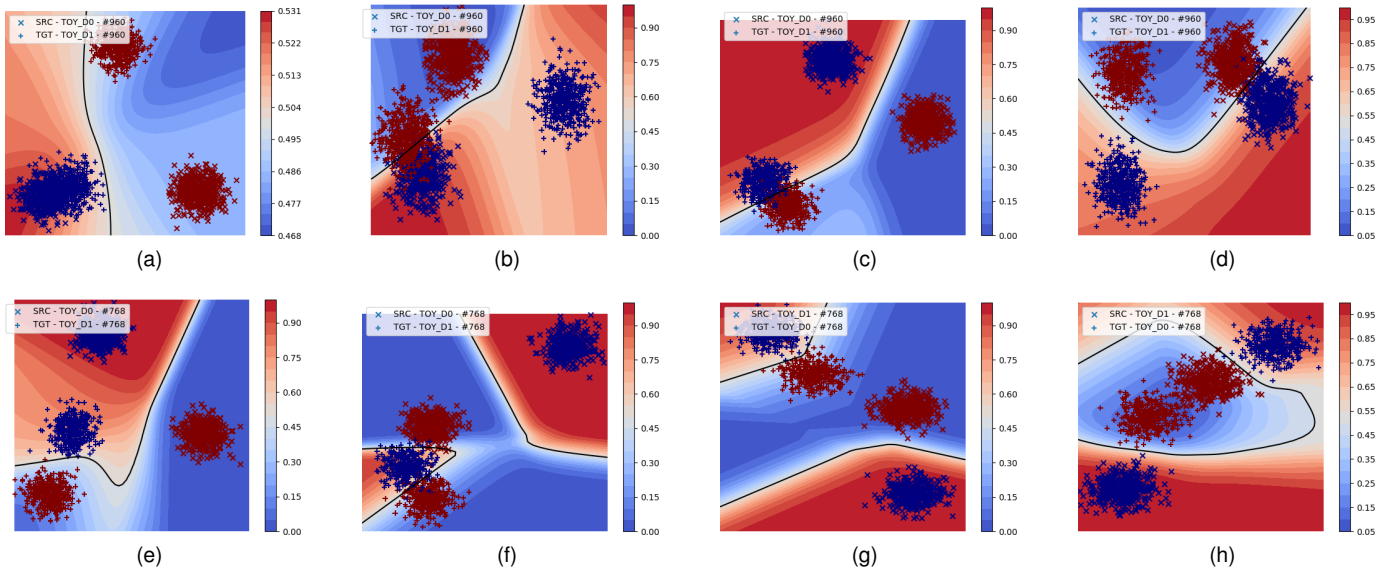


Fig. 3. Contour plots show the probability contours along with clear decision boundaries on different toy-dataset settings trained using CUDA. (source domain: $\times$, target domain: $+$, class 0: blue, class 1: red.) (Best viewed in color.)

TABLE 1
Details of visual domain adaptation datasets.

Dataset	Domain	# Train	# Test	# Classes
Office-31	AMAZON ($\mathcal{A}$) [44]	2,817	-	31
	DSLR ($\mathcal{D}$) [44]	498	-	
	WEBCAM ($\mathcal{W}$) [44]	795	-	
VisDA-2017	SYNTHETIC ($\mathcal{V}_{syn}$) [45]	152,397	-	12
	REAL ($\mathcal{V}_{real}$) [45]	55,388	72,372	
Digits	USPS (us) [48]	7,291	2,007	10
	MNIST (mn) [49]	60,000	10,000	
	MNIST-M (mm) [21]	25,000	9,000	
	SVHN (sv) [50]	73,257	26,032	
	SYNNUMBERS (sn) [21]	479,400	9,553	

neural network by training only using the labeled source domain. Any improvement over ss using CUDA (i.e., combination of $su/tu/sa/ta$) indicates effectiveness of CUDA as the chosen contradistinguisher neural network is fixed.

4.1 Experiments on Synthetic Toy-dataset

We validate our proposed method by performing experiments on synthetically created simple datasets that model different source and target domain distributions in a 2-dimensional input feature space using different blobs of source-target domain orientations and offsets (i.e., domain shift). We create blobs for source and target domains with 4000 samples using standard $scikit{-}learn$ [46] as indicated in Fig. 1 and 3. We further evenly split these 4000 data-points into equal train and test sets. Each of the splits consists of the same number of samples corresponding to both the class labels.

The main motivation of the experiments on toy-dataset is to understand and visualize the behavior of the proposed method under some typical domain distribution scenarios and analyze the performance of CUDA. $Blobs$ toy-dataset plots in Fig. 1 shows clear comparisons of the classifier decision boundaries learnt using CUDA over domain alignment approaches. The top row in Fig. 1 corresponds to the domain alignment classifier trained only on the labeled source domain, i.e., ss. However, the bottom row in Fig. 1 corresponds to contradistinguisher trained using the proposed method CUDA with labeled source and unlabeled target domain, i.e., $ss{+}tu{+}ta$.

Fig. 3 demonstrates the classifier learnt using CUDA on the synthetic datasets with different complex shapes and orientations of the source and target domain distributions for the input data. Fig. 1e and 1g indicate the simplest form of the domain adaptation tasks with similar orientations in source and target domain distributions.It is important to note that the prior enforcing used in pseudo-label selection is the reason such fine classifier boundaries are observed, especially in Fig. 1f,1h and 3a-3e. Fig. 3f and 3g represent more complex configurations of source and target domain distributions that indicate the hyperbolic decision boundaries jointly learnt on both the domains simultaneously using a single classifier without explicit domain alignment. Similarly, Fig. 3h represents a complex configuration of source and target domain distributions that indicates an elliptical decision boundary.

These simulated experiments points to some significant inner workings of our approach CUDA. These are the two main takeaways from the toy-dataset experiments. (i) The non-necessity of the domain alignment in the form of distribution distance metric minimization or data augmentation. In the case of these toy-datasets, it is not possible to perform any form of data augmentation, unlike some of the visual domain adaptation tasks, because the data is directly available in the form of encoded features that cannot be easily data augmented through standard heuristics. In such a case, it is necessary to realize a generic approach applicable to multiple modalities of the input, e.g., similar to the toy-dataset in language domain adaptation tasks. The features

are presented in the form of word2vec/doc2vec, and no data augmentation is possible. (ii) Fig. 1c and 3d provide some interesting observations. Here, we can observe that the classes in the source domain are overlapping, resulting in less than 100% classification on the source domain, which in turn results in less than 100% classification on the target domain when considering domain alignment approaches. However, CUDA does not try to morph the target domain onto the source domain by directly classifying on the target domain resulting in a perfect classification. Since the classification is done directly on the unlabeled target domain in a fully unsupervised manner, the target domain classification performance is not limited by the source domain classification performance, i.e., the irrespective of the domain is used as the labeled source domain and the unlabeled target domain, the performance is the respective domains are similar. In other words, swapping of the domains or the direction of the domain adaptation has little effect on the classification performance on each individual domain.

4.2 Experiments on Real-world Datasets

In our previous work [43], we have demonstrated the effectiveness of CUDA in real-world domain adaptation on low-resolution visual datasets and language datasets. In contrast to low-resolution visual datasets, we consider the complex, high-resolution Office-31 [44] and VisDA-2017 [45] object classification datasets for domain adaptation. In addition to the single-source domain adaptation experiments, we also extend CUDA to Office-31 [44] and Digits datasets [21], [48], [49], [50].

4.2.1 Office-31 Dataset

In high-resolution visual datasets, we consider Office-31 [44] dataset for our experiments. Unlike low-resolution visual datasets, here, we have only a few hundreds of training samples that make this an even more challenging task.

Office objects: Office-31 [44] dataset consists of high resolution images of objects belonging to 31 classes obtained from three different domains AMAZON ($\mathcal{A}$), DSLR ($\mathcal{D}$) and WEBCAM ($\mathcal{W}$). Fig. 4 shows illustrations of the images from all the three above mentioned domains of the Office-31 [44] dataset. AMAZON ($\mathcal{A}$) domain consists of synthetic images with clear white background. DSLR ($\mathcal{D}$) and WEBCAM ($\mathcal{W}$) domains consist of real-world images with noisy background and surroundings. We consider all possible six combinatorial tasks of domain adaptation involving all the three domains, i.e., $\mathcal{A}\leftrightarrow\mathcal{D}$, $\mathcal{A}\leftrightarrow\mathcal{W}$ and $\mathcal{D}\leftrightarrow\mathcal{W}$. Compared to low-resolution visual datasets, Office-31 [44] dataset domain adaptation tasks have increased complexity due to the small number of training images. Unlike low-resolution visual datasets, the high-resolution Office-31 [44] dataset does not have separate pre-defined train and test splits. Since we do not use any labels from the target domain during training, we report ten-crop test accuracy on the target domain by summing the softmax values of all the ten crops of the image and assign the label with maximum aggregate softmax value for the given image as in CDAN [30] in Table 2.

To further alleviate the lack of a large number of training samples, pre-trained networks such as ResNet-50 [65] and ResNet-152 [65] were used to extract 2048 dimensional features from high-resolution images similar to CDAN [30]. Since the images are not well centered and have a high resolution, we use the standard ten-crop of the image to extract features from the same images during training and testing, also similar to CDAN [30].

The use of pre-trained models leads to two choices of training, (i) Fine-tune the pre-trained model used as feature extractor along with the final classifier layer: This requires careful selection of several hyper-parameters such as learning rate, learning rate decay, batch size, etc., to fine-tune the network to the current dataset while preserving the ability of the pre-trained network. We observed that fine-tuning also depends on the loss function used for training [66], which in our case, the use of contradistinguish loss greatly affected the changes in the pre-trained model as it is trained only using cross-entropy loss. Fine-tuning is also computationally expensive and time-consuming as each iteration requires computing gradients of all the pre-trained model parameters. (ii) Fix the pre-trained model and only train the final classifier layer: Alternative to fine-tuning is to fix the pre-trained model and use it only as a feature extractor. This approach has multiple practical benefits such as, (a) The computational time and cost of fine-tuning the parameters of the pre-trained model are alleviated. (b) Since the feature extractor is fixed, it requires only once to extract and store the features locally instead of extracting the same features every iteration. Hence reducing the training time and the GPU memory as it is only required to train the final classifier.

4.2.2 VisDA-2017 Dataset

The VisDA-2017 dataset consists of two domains, (i) synthetic and (ii) real, with three predefined data splits. Fig. 5 indicates the samples from all the 12 classes of the three data splits. The three predefined data splits in the VisDA-2017 dataset are as follows.

(i) Training set: This split includes 152,397 labeled synthetic images obtained using 2D renderings of 3D models from different angles and different lighting conditions. This split is considered as a labeled source domain for training.

(ii) Validation set: This split includes 55,388 real-world images obtained from a curated subset of MS COCO [67] dataset. This split is considered an unlabeled target domain training set, and this is used during the training without labels.

(iii) Testing set: This split includes 72,372 real-world images obtained from YouTube Bounding Boxes [68] dataset. This split is considered as the target domain testing set used for evaluation and to report the results.

4.3 Analysis of Experimental Results on Real-world Datasets

4.3.1 Office-31 Single-Source Domain Adaptation Results

We report the standard ten-crop accuracy on the target domain images as reported by several state-of-the-art domain adaptation methods [3], [25], [30]. Since there are no explicit test split specified in the dataset and no labels are used from the target domain during training, it is common to report ten-crop accuracy considering the whole target domain.

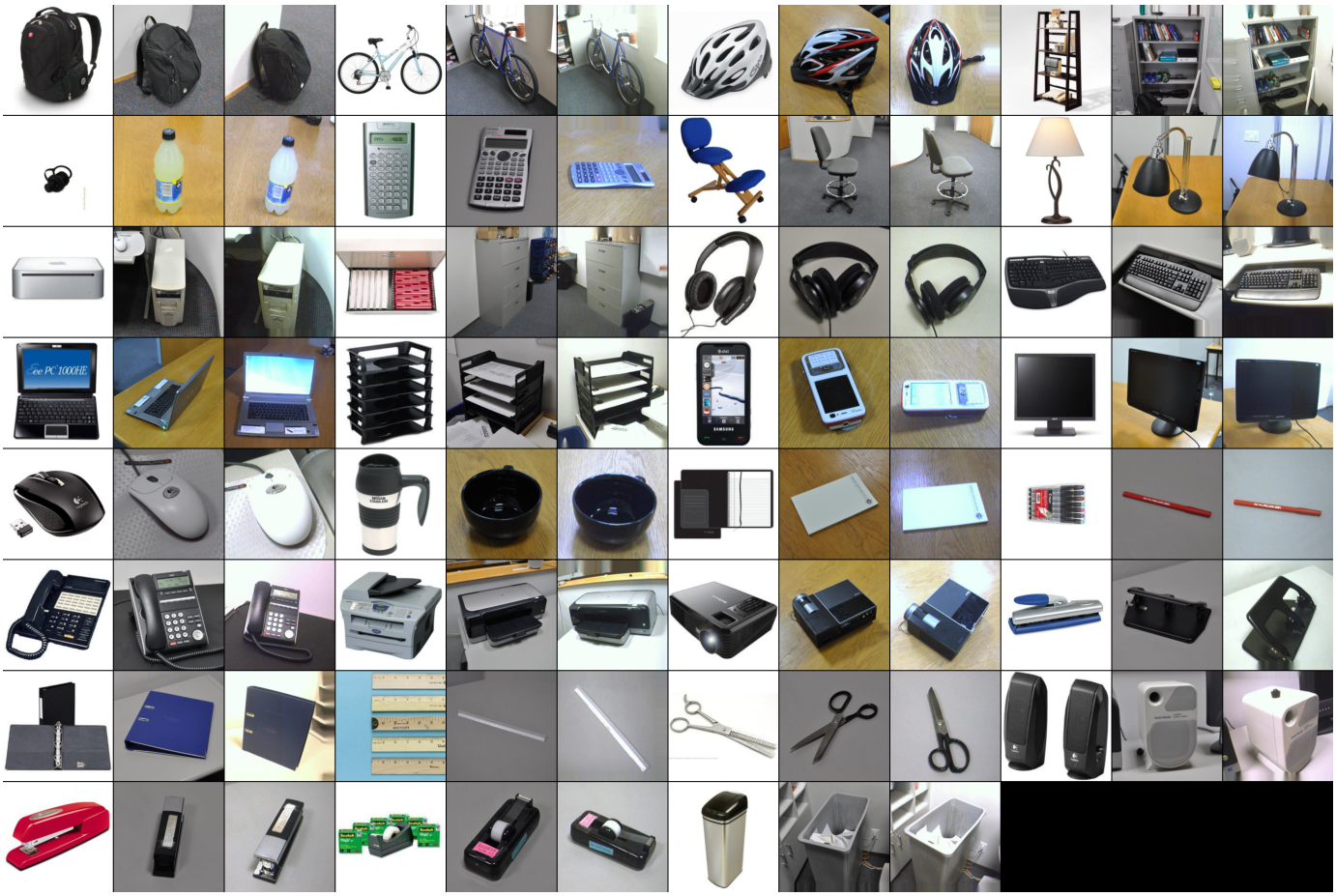


Fig. 4. Illustrations of samples from all the three domains of high resolution Office-31 [44] dataset with one instance per each class from every domain (column {1,4,7,10}: $\mathcal{A}$, {2,5,8,11}: $\mathcal{D}$, {3,6,9,12}: $\mathcal{W}$). (Best viewed in color.)

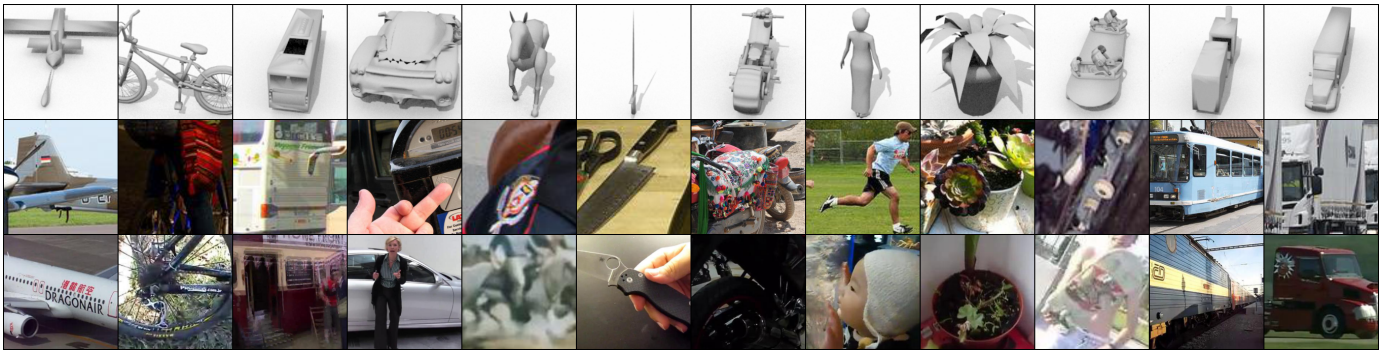


Fig. 5. Illustrations of samples from all the three data-splits of VisDA-2017 [45] dataset with one instance per each class from every domain ({row 1}: $\mathcal{V}_{syn}$ source domain synthetic images (training set), {row 2}: $\mathcal{V}_{real}$ target domain real-world images (validation set), {row 3}: $\mathcal{V}_{real}$ target domain real-world images (testing set)). It should be noted that unlike the Office-31 dataset and other standard benchmark domain adaptation datasets discussed in [43], most of the real-world images in the target domain of the VisDA-2017 dataset contains multiple true labels, which are only annotated with only one of the multiple labels. (Best viewed in color.)

In Table 2, we report accuracies obtained by fine-tuning ResNet-50 [65] using the learning rate schedule followed in CDAN [30] and also without fine-tuning ResNet-50 [65]. Apart from fixed ResNet-50 [65], we also report accuracies with fixed ResNet-152 [65] in Table 2 for comparison. Fig. 6a-6f indicate the t-SNE [69] plots of the softmax output after aggregating the ten-crop of each image corresponding to training configuration $ss{+}tu{+}su{+}ta$ reported in Table 2. Fig. 6 reports the t-SNE [69] plots of the training setting using ResNet-152 [65] encoder with the highest mean accuracy of all the six domain adaptation tasks. We clearly observe that CUDA outperforms several state-of-the-art methods that also use ResNet-50 [65] and even further surpasses by using ResNet-152 [65] encoder with CUDA.

Among the three domains in Office-31 [44] dataset, $\mathcal{A}$ can be considered as a well-curated synthetic dataset with clear

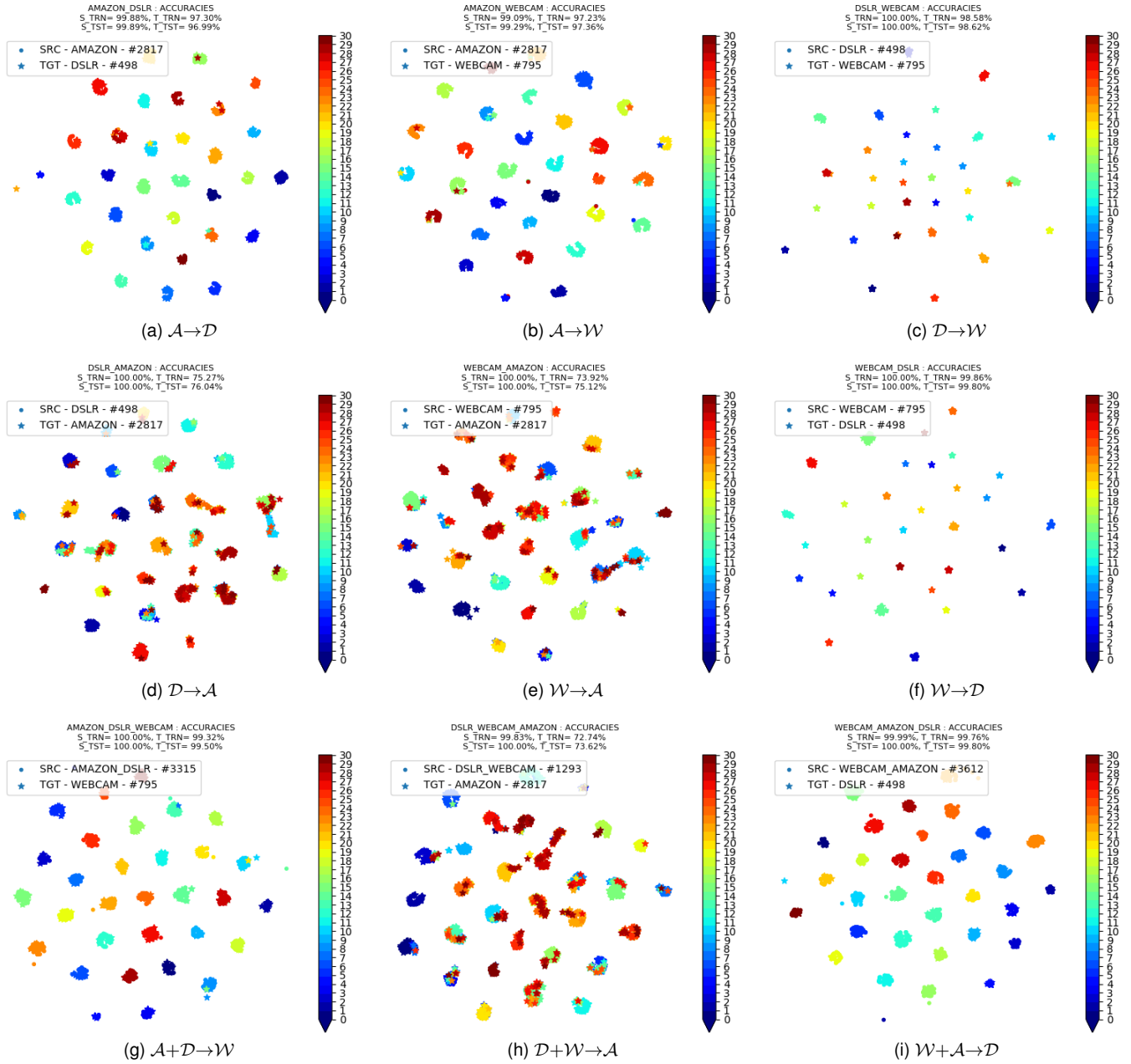


Fig. 6. Row 1 and 2: t-SNE [69] plots for embeddings from the output of contradistinguisher with samples from Office-31 [44] dataset as input corresponding to the highest mean accuracy setting $ss+tu+su+ta$ indicated in Table 2 for single-source domain adaptation using ResNet-152 [65] as the fixed encoder. Row 3: t-SNE [69] plots for embeddings from the output of contradistinguisher corresponding to the samples from Office-31 [44] dataset in high-resolution visual tasks after applying softmax trained with CUDA with ResNet-50 [65] as the encoder in a **multi-source domain adaptation** setting as indicated in Table 3. We can observe the clear class-wise clustering among all the 31 classes in the Office-31 [44] datasets. We achieve high accuracies in spite of having only a few hundred training samples in each domain. (Best viewed in color.)

background and $\{\mathcal{D},\mathcal{W}\}$ as an uncurated real-world dataset with noisy background and surroundings. We report the six domain adaptation tasks in the order of their complexity from low to high as, (i) Fig. 6c and 6f indicate highest accuracies because of similar real-world to real-world domain adaptation task, (ii) Fig. 6a and 6b indicate moderately high accuracies because of synthetic to real-world domain adaptation task and (iii) Fig. 6d and 6e indicate the lowest accuracies among all the six tasks because of real-world to synthetic domain adaptation task. Comparing CUDA with ss in Tables 2 and 3, we can see significant improvements in the target domain test accuracies due to the use of contradistinguish loss (5) demonstrating the effectiveness of contradistinguisher. As our method is mainly dependent on the contradistinguish loss (5), we observed further improved results by experimenting with better neural networks, e.g., using ResNet-152 over ResNet-50 along with our contradistinguish loss (5). From our ablations study in Table 2, we observe the effect of selection of ImageNet pre-trained ResNet-50 and ResNet-152 models on the domain adaptation with similar implications with the work [70]. In general, one can always obtain better results irrespective

TABLE 2
Target domain accuracy (%) on high resolution Office-31 [44] dataset containing three domains. CUDA corresponds to our best results obtained with the best hyper-parameter settings. ss: source supervised (2), tu: target unsupervised (5), su: source unsupervised (5), sa: source adversarial regularization (9) and ta: target adversarial regularization (9) represents different training configurations.

Method	$\mathcal{A}\rightarrow\mathcal{D}$	$\mathcal{A}\rightarrow\mathcal{W}$	$\mathcal{D}\rightarrow\mathcal{A}$	$\mathcal{D}\rightarrow\mathcal{W}$	$\mathcal{W}\rightarrow\mathcal{A}$	$\mathcal{W}\rightarrow\mathcal{D}$	$Mean$
DAN [1]	78.6	80.5	63.6	97.1	62.8	99.6	80.3
RTN [2]	77.5	84.5	66.2	96.8	64.8	99.4	81.5
JAN [3]	84.7	85.4	68.6	97.4	70.0	99.8	84.3
Rozantsev et. al. [9]	75.5	75.8	55.7	96.7	57.6	99.6	76.8
CAN [17]	95.0	94.5	**78.0**	99.1	**77.0**	99.8	**90.6**
RevGrad [21]	79.7	82.0	68.2	96.9	67.4	99.1	82.2
ADDA [24]	77.8	86.2	69.5	96.2	68.9	98.4	82.8
GTA [25]	87.7	89.5	72.8	97.9	71.4	99.8	86.5
CDAN [30]	92.9	94.1	71.0	98.6	69.3	**100.0**	87.6
DICE [34]	68.5	72.5	58.1	97.2	60.3	**100.0**	76.1
ATM [38]	**96.4**	95.7	74.1	**99.3**	73.5	**100.0**	89.8
CADA-P [39]	95.6	**97.0**	71.5	**99.3**	73.1	**100.0**	89.5
PFAN [40]	76.3	83.0	63.3	99.0	60.8	99.9	80.4
CUDA (Ours) (with ResNet-50)	96.0	95.6	73.2	99.1	74.7	**100.0**	89.8
CUDA (Ours) (with ResNet-152)	**97.0**	**98.5**	76.0	99.0	76.0	**100.0**	**91.1**
ss (Ours) (fine-tune ResNet-50)	41.0	38.7	23.2	80.6	25.6	94.2	50.6
ss (Ours) (fixed ResNet-50)	82.0	77.9	68.4	97.2	67.1	**100.0**	82.1
ss+tu (Ours) (fixed ResNet-50)	95.0	93.8	71.5	98.9	73.3	99.4	88.7
ss+tu+su (Ours) (fixed ResNet-50)	**96.0**	**95.6**	69.5	**99.1**	70.7	**100.0**	88.5
ss+tu+su+ta (Ours) (fixed ResNet-50)	92.8	91.6	72.5	98.4	72.8	99.8	88.0
ss+tu+su+ta+sa (Ours) (fixed ResNet-50)	91.8	**95.6**	**73.2**	98.0	**74.7**	**100.0**	**88.9**
ss (Ours) (fixed ResNet-152)	84.9	82.8	70.3	98.2	71.1	**100.0**	84.6
ss+tu (Ours) (fixed ResNet-152)	**97.0**	94.3	73.9	**99.0**	75.5	**100.0**	90.0
ss+tu+su (Ours) (fixed ResNet-152)	95.6	95.6	73.8	98.7	74.3	**100.0**	89.7
ss+tu+su+ta (Ours) (fixed ResNet-152)	**97.0**	97.4	**76.0**	98.6	75.1	99.8	**90.7**
ss+tu+su+ta+sa (Ours) (fixed ResNet-152)	95.4	**98.5**	75.0	98.9	**76.0**	**100.0**	90.6

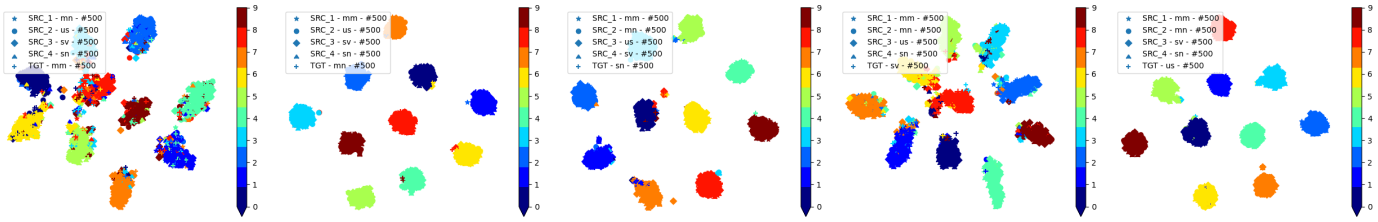
TABLE 3
Target domain accuracy (%) on high resolution Office-31 [44] dataset under **multi-source domain adaptation** setting by combining two domains into a single source domain and the remaining domain as the target domain with ResNet-50 [65] as the encoder. CUDA corresponds to our best results obtained with the best hyper-parameter settings. ss: source supervised (2), tu: target unsupervised (5), su: source unsupervised (5), sa: source adversarial regularization (9) and ta: target adversarial regularization (9) represents different training configurations.

Setting	Method	$\mathcal{A}$+$\mathcal{D}\rightarrow\mathcal{W}$	$\mathcal{D}$+$\mathcal{W}\rightarrow\mathcal{A}$	$\mathcal{W}$+$\mathcal{A}\rightarrow\mathcal{D}$	$Mean$
Best single source	DAN [1]	97.1	63.6	99.6	86.8
	RTN [2]	96.8	66.2	99.4	87.5
	JAN [3]	97.4	70.0	99.8	89.1
	Rozantsev et. al. [9]	96.7	57.6	99.6	84.6
	mDA-layer [10]	94.5	64.9	94.9	84.8
	RevGrad [21]	96.9	68.2	99.1	88.1
	ADDA [24]	96.2	69.5	98.4	88.0
	GTA [25]	97.9	72.8	99.8	90.2
	CDAN [30]	98.6	71.0	**100.0**	89.8
	DICE [34]	97.2	60.3	**100.0**	85.8
	CUDA (Ours)	**99.1**	**74.7**	**100.0**	**91.3**
Multi-source	DAN [1]	95.2	53.4	98.8	82.5
	mDA-layer [10]	94.6	62.6	93.7	83.6
	DIAL [13]	94.3	62.5	93.8	83.5
	SGF [14]	52.0	28.0	39.0	39.7
	sFRAME [15]	52.2	32.1	54.5	46.3
	RevGrad [21]	96.2	54.6	98.8	83.2
	DCTN [35]	96.9	54.9	99.6	83.8
	CUDA (Ours)	**99.5**	**73.6**	**99.8**	**91.0**
	ss (Ours)	95.6	68.1	99.2	87.6
	ss+tu (Ours)	99.4	72.1	**99.8**	90.4
	ss+tu+su (Ours)	98.9	70.3	99.4	89.5
	ss+tu+su+ta (Ours)	**99.5**	73.3	99.6	90.8
	ss+tu+su+ta+sa (Ours)	99.4	**73.6**	99.2	90.7

of the approach by using better/deeper pre-trained models and/or data augmentation. However, since our main aim is to isolate, observe and benchmark only the true effect of different benchmark approaches, in our experiments, we

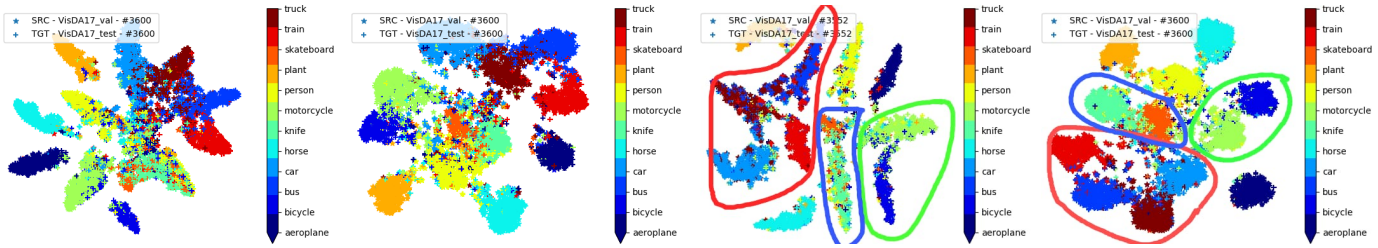
TABLE 4
Target domain accuracy reported on the test set (%) on all 5 combinations of Digits datasets under **multi-source domain adaptation** setting.

Method	$mn, sn, sv, us \rightarrow mm$	$mm, sn, sv, us \rightarrow mn$	$mm, mn, sv, us \rightarrow sn$	$mm, mn, sn, us \rightarrow sv$	$mm, mn, sn, sv \rightarrow us$	$Mean$
DAN [1]	63.78	96.31	85.43	62.45	94.24	80.44
M3SDA [18]	72.82	98.43	89.58	81.32	96.14	87.66
RevGrad/DANN [21]	71.30	97.60	85.34	63.48	92.33	82.01
ADDA [24]	71.57	97.89	86.45	75.48	92.83	84.84
DCTN [35]	70.53	96.23	86.77	77.61	82.81	82.79
CUDA (Ours)	**87.96**	**99.51**	**97.96**	**89.03**	**99.30**	**94.75**



(a) $mn, sn, sv, us \rightarrow mm$ (b) $mm, sn, sv, us \rightarrow mn$ (c) $mm, mn, sv, us \rightarrow sn$ (d) $mm, mn, sn, us \rightarrow sv$ (e) $mm, mn, sn, sv \rightarrow us$

Fig. 7. The t-SNE plots of the unseen test set samples corresponding to the CUDA result in Table 4. The t-SNE plots show clear clustering of all the 10 classes in Digits datasets distinctively. (Best viewed in color.)



(a) BSP (b) CAN (c) CUDA (d) CUDA*

Fig. 8. The t-SNE plots of CUDA/CUDA* shows the clear clustering of all the twelve classes of VisDA-2017 distinctively compared to the t-SNE plots of BSP/CAN. The t-SNE plots of CUDA/CUDA* represent some important visual semantics of the image embeddings obtained from contradistinguisher in the following manner. (i) The vehicular classes such as 'bus', 'car', 'train', and 'truck' can be seen clustered closely as semantically these classes are similar to each other (region bounded in red). (ii) The two-wheeler classes such as 'bicycle' and 'motorcycle' are clustered closely as these are semantically similar to each other compared to vehicular classes that are clustered exactly opposite (region bounded in green). (iii) Irrespective of the approach used, there is always confusion between 'knife' and 'skateboard' classes. This confusion between 'knife' and 'skateboard' classes represented in the confusion matrices, which is also clearly seen in the t-SNE plots as well, can be attributed to the nature of images of these classes in the dataset on close observation (region bounded in blue). (iv) The remaining classes such as 'aeroplane', 'horse', 'person' and 'plant' can be seen clustered independently and distinctively as these classes have almost no visual semantic similarities to one another. (Best viewed in color.)

maintain all the other parameters such as pre-trained neural network/data augmentation similar except for the core idea of the approaches and report our results both on Office-31 and VisDA-2017 datasets.

4.3.2 *Multi-Source Domain Adaptation Results*

We also extend the experiments to multi-source domain adaptation on the Office-31 [44] and Digits datasets [21], [48], [49], [50]. In Table 3, we can clearly observe that in $\mathcal{A}+\mathcal{D}\rightarrow\mathcal{W}$ task, multi-source domain adaptation provides better results than their respective best single source domain adaptation experiments. However in case of $\mathcal{D}+\mathcal{W}\rightarrow\mathcal{A}$ and $\mathcal{W}+\mathcal{A}\rightarrow\mathcal{D}$, the multi-source domain adaptation improves over ss, it underperforms compared to best single source domain adaptation task. This can be attributed to the fact that the model tends to overfit on the source domains resulting in a negative transfer.This negative transfer behavior is also prevalent in other multi-source domain adaptation approaches since all the other multi-source domain adaptation methods also underperform compared to their best single source domain adaptation results, as reported in Table 3. Fig. 6g-6i indicates t-SNE [69] plots for embeddings from the output of contradistinguisher corresponding to the samples from Office-31 [44] dataset after applying softmax trained with CUDA with ResNet-50 [65] as the encoder in a **multi-source domain adaptation** setting. We can observe the best results when the target domain is one of the real-world domain, i.e., $\mathcal{D}$ and $\mathcal{W}$. It was consistently observed that domain adaptation tasks with synthetic domain $\mathcal{A}$ as the target domain to be the most complex tasks of all the domain adaptation tasks across all the domain adaptation methods.

Similarly, Table 4 presents the results of multi-source

TABLE 5
Results on VisDA-2017 dataset reproduced from the current state-of-the-art method BSP, CAN and our proposed method CUDA reported on both the validation set and test set. We report all the evaluation metrics such as precision, recall, and accuracy, unlike BSP/CAN, where the recall scores are mistakenly reported as accuracy. CUDA* represents the results reproduced using vanilla CUDA with the data augmentation and target domain clustering similar to CAN for a fair comparison of the effect of the CUDA over CAN. The results reported for BSP, CAN, and CUDA/CUDA* are from our own best reproduction from the original source code.

Data-split	Metric (%)	Method	Aeroplane	Bicycle	Bus	Car	Horse	Knife	Motorcycle	Person	Plant	Skateboard	Train	Truck	*Mean*
Validation	Precision	BSP [36]	93.79	91.40	71.03	78.36	94.98	35.25	78.90	63.12	91.37	55.28	82.16	58.15	74.48
		CAN [17]	96.92	85.87	83.45	85.06	92.18	70.73	91.25	61.38	91.77	63.45	89.91	74.94	82.24
		CUDA (Ours)	96.33	87.04	79.20	80.68	92.25	67.66	86.74	70.28	91.57	67.37	87.36	50.18	79.72
		CUDA* (Ours)	97.08	86.27	83.71	86.03	91.56	77.53	91.59	67.23	95.47	64.66	90.89	72.89	**83.74**
	Recall	BSP [36]	90.35	71.54	76.42	59.11	87.91	80.58	89.41	73.75	84.70	76.41	81.35	45.84	76.45
		CAN [17]	96.60	88.63	81.90	68.47	96.25	95.95	87.80	80.45	96.88	94.74	85.01	60.04	86.06
		CUDA (Ours)	91.47	83.11	75.59	69.16	94.65	93.35	88.13	82.52	90.79	81.02	82.22	51.69	81.97
		CUDA* (Ours)	96.76	90.04	83.94	70.82	96.18	96.63	88.53	86.28	95.43	94.48	85.98	62.09	**87.26**
	Accuracy	TPN [16]	93.70	85.10	69.20	81.60	93.50	61.90	89.30	81.40	93.50	81.60	84.50	49.90	80.40
		BSP [36]	98.97	97.79	95.36	89.26	98.58	93.73	96.39	94.99	98.09	96.48	97.22	91.27	95.68
		CAN [17]	99.57	98.37	97.09	91.82	98.99	98.36	97.84	94.93	99.03	97.54	98.12	93.99	97.14
		CUDA (Ours)	99.21	98.16	96.25	91.10	98.87	98.08	97.35	96.22	98.56	97.60	97.73	90.02	96.60
		CUDA* (Ours)	99.60	98.48	97.26	92.36	98.93	98.82	97.95	95.97	99.25	97.65	98.27	93.89	**97.37**
Test	Precision	BSP [36]	91.76	95.83	71.50	83.08	94.87	33.08	76.45	72.25	87.38	34.54	83.36	80.12	75.35
		CAN [17]	95.72	89.20	86.38	93.31	95.28	74.33	91.12	76.74	86.90	45.56	93.92	87.24	84.64
		CUDA (Ours)	93.31	94.44	81.28	82.26	91.77	66.12	80.43	77.37	88.48	45.19	89.43	60.74	79.24
		CUDA* (Ours)	95.18	87.61	86.79	92.01	96.16	77.36	90.60	81.28	93.49	47.33	94.15	85.83	**85.65**
	Recall	BSP [36]	77.98	44.64	79.44	91.11	72.75	70.26	73.99	54.85	88.69	44.50	73.84	67.84	69.99
		CAN [17]	92.01	74.79	85.60	91.84	87.33	89.26	68.09	80.89	97.34	84.79	81.94	89.24	85.26
		CUDA (Ours)	82.41	58.45	71.41	92.16	83.70	79.58	72.46	75.55	93.03	70.78	73.95	79.58	77.75
		CUDA* (Ours)	92.44	80.97	86.41	92.18	87.85	89.42	69.35	84.20	96.41	85.34	83.48	89.27	**86.44**
	Accuracy	BSP [36]	97.92	96.62	94.97	97.24	97.27	86.96	94.55	92.98	98.57	94.66	95.89	95.42	95.25
		CAN [17]	99.13	97.98	97.31	98.52	98.51	96.85	95.70	95.38	98.97	95.55	97.66	97.77	97.44
		CUDA (Ours)	98.31	97.34	95.66	97.21	97.92	95.36	94.96	95.06	98.87	95.61	96.51	93.28	96.34
		CUDA* (Ours)	99.12	98.20	97.42	98.41	98.63	97.21	95.78	96.27	99.39	95.82	97.82	97.62	**97.64**

TABLE 6
Total classification accuracy (%) on VisDA-2017 dataset reported on both the validation set and test set. The results from JAN, GTA, CDAN and TransNorm are reported from TransNorm [37]. The results reported for BSP, CAN and CUDA/CUDA* are from our own best reproduction from the original source code. CUDA* represents the results reproduced using vanilla CUDA with the data augmentation and target domain clustering similar to CAN for a fair comparison of the effect of the CUDA over CAN.

Method	Data-split	
	Validation	Test
JAN [3]	61.6	-
GTA [25]	69.5	-
CDAN [30]	70.0	-
TransNorm [37]	71.4	-
BSP [36]	74.1	71.5
CAN [17]	82.8	84.7
CUDA (Ours)	79.6	78.1
CUDA* (Ours)	**84.2**	**85.8**

domain adaptation on Digits datasets against benchmark approaches. In Fig. 7, we see the t-SNE plots on the test set depicting clear class-wise clustering that indicates the efficacy of CUDA single-source to multi-source extension.

4.3.3 VisDA-2017 Single-Source Domain Adaptation Results

For experiments on the VisDA-2017 dataset, we consider the most recent state-of-the-art benchmark domain adaptation approaches BSP [36] and CAN [17]. Like BSP and CAN, we use the same neural network architecture with Imagenet pre-trained ResNet-101 with contradistinguish loss for training. BSP and CAN report the evaluation metric of accuracy in their papers. However, on reproducing the results with BSP[1] and CAN[2] to set the baseline for comparison, we noticed that the results reported had the following inconsistencies.

(i) The results reported in the paper as accuracy in actual were the class-wise *recall* scores.

(ii) The most standard procedure in machine learning is to report experimental performances on the test split, which is unseen during the training. However, in BSP and CAN, the results are reported on the validation set, which is used as the unlabeled target domain training set. Since the VisDA-2017 dataset has pre-defined splits for evaluation, reporting the results on the validation set used during the training does not indicate these models' generalizing capability, which is the most important aspect one would base an evaluation. We correct the above misreporting by reproducing the results from BSP and CAN and report all the relevant metrics for both validation and test splits of the VisDA-2017 dataset. Apart from these, we also validate our results on the official challenge evaluation portal[3] by submitting the results of our approach CUDA on the VisDA-2017 dataset.

In Fig. 8, we report the t-SNE plots reproduced from the current state-of-the-art unsupervised domain adaptation approach BSP [36] and CAN [17], in comparison with our approach CUDA/CUDA* on both the pre-defined validation split (seen target domain training set) and testing split (unseen target domain testing set) of the VisDA-2017 dataset. The results reported as CUDA corresponds to the CUDA experiments without any data augmentation using the BSP source code as the baseline to keep all the param-

1. https://github.com/thuml/Batch-Spectral-Penalization
2. https://github.com/kgl-prml/Contrastive-Adaptation-Network\-for-Unsupervised-Domain-Adaptation
3. https://competitions.codalab.org/competitions/17052#results

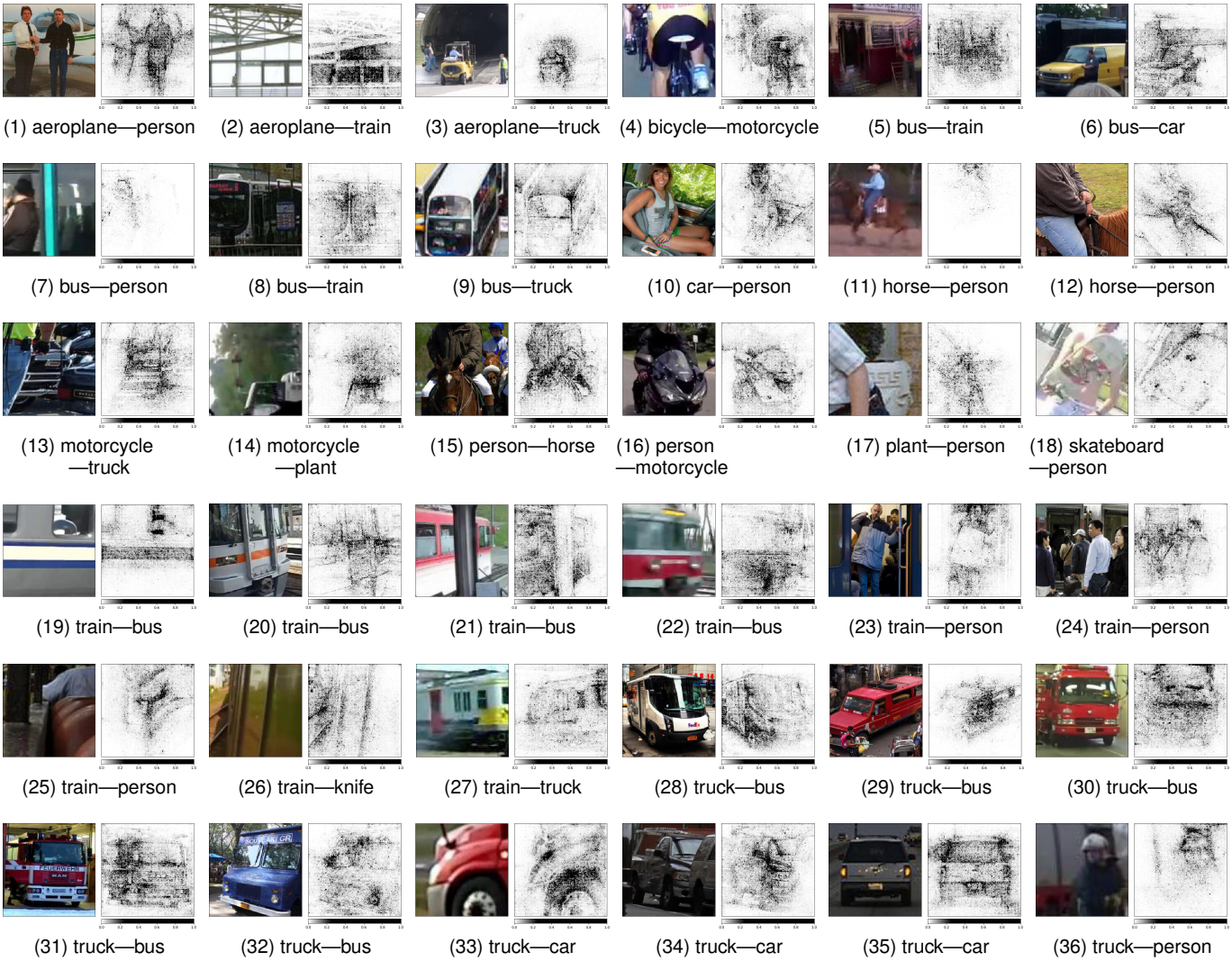


Fig. 9. We indicate few samples that are misclassified by the contradistinguisher in the following subcaption format 'original_label—predicted_label'. In most cases, the original ground truth labels are dubious, and the predicted labels make more sense realistically. Subplots (6), (7), (11), (12), (14), (18), (23) and (25) shows that the object is identified based on the shape and not if the object is present only in the foreground. This indicates that the contradistinguisher makes the predictions based on the clearly visible shapes and not the presence of the object in the foreground/background. The visualization of the features responsible for the respective predicted outcome indicates the shape bias as mostly the features are detected as edges corresponding to the shape of the object in the image. This shows the importance of shape bias to achieve high performance in transfer learning and domain adaptation tasks.

eters the same for a fair comparison with BSP. Similarly, the results reported as CUDA* corresponds to the CUDA experiments with data augmentation and clustering of high confidence target domain samples using the CAN source code as the baseline to keep all the parameters same for a fair comparison with CAN. We can see the classwise clusters in BSP/CUDA are narrower compared to CAN/CUDA*, which are broader due to the use of data augmentation. Data augmentation helps modify/broaden the data distribution aiding in an improvement over the vanilla approaches without data augmentation. We further validate the results best results from our approach, i.e., CUDA* by submitting to the official challenge evaluation leaderboard. In Table 5 we compare the per-class precision, recall, and accuracies on both the pre-defined validation set and test set of the VisDA-2017 against the results reproduced from the BSP/CAN. In Table 6 we compare the total classification accuracies on both the pre-defined validation set and test set of the VisDA-2017 dataset against different benchmark methods. The results in Tables 5 and 6 indicate the superior performance of the proposed method CUDA/CUDA* over the current state-of-the-art domain alignment approaches BSP/CAN on both the pre-defined validation set and a test set of VisDA-2017 dataset. Even though the validation set and test set belong to the real-world domain, there is an inherent domain shift between them as both the data splits are collected from two different datasets, i.e., MS COCO [67] and YouTube Bounding Boxes , [68] respectively. Results from CUDA indicate a better generalization to real-world domain as the scores in validation and test sets are closer compared to other approaches on the VisDA-2017 dataset. We can also observe that the t-SNE plot of CUDA in Fig. 8c and 8d clearly shows the visual semantics captured between the classes of images in the VisDA-2017 dataset.

Apart from setting CUDA as the solid baseline for VisDA-2017, we further put a conscious effort to carefully investigate the reasons for the misclassification using the contradistinguisher to check if we can further improve the results. In most misclassified cases, we have observed that the labels predicted by CUDA appeared to be correct in comparison to the ground truth label of the dataset. In Fig. 9, we present some of these instances where the predicted label is more close to the real label than the ground truth. We can explain this misclassification as a limitation of the VisDA-2017 dataset in the following way. The misclassification observed in Fig. 9 is due to the fact that the images in the VisDA-2017 dataset consist of objects belonging to more than one of the twelve classes, i.e., the images in the dataset consists of multiple labels for a single image, but the dataset only records one of these several true labels. As the assumed task for domain adaptation is single-label multi-class We see this as a limitation of the VisDA-2017 dataset compared other benchmark domain adaptation datasets such as Office-31 or the low-resolution visual datasets demonstrated in our conference paper [43]. It is necessary that the datasets be consistent in the sense that each image has a unique label corresponding to it so that during the evaluation, there is no ambiguity between the original label and the predicted label from the trained model. The presence of this ambiguity in the dataset classification would then lead to observing the true evaluation metrices resulting in improper benchmarking for any given approach. However, in the case of the VisDA-2017 dataset, the predicted label from the model cannot be considered as a wrong label as it contains the object of the predicted label in the image. We believe that this is one of the reasons for the overall low performance apart from the complexity of the VisDA-2017 dataset compared to other visual datasets. It should also be observed that CUDA identifies the most distinguishing/prominent and assigns the label irrespective of the position (foreground/background) of the object as indicated in some of the subplots in Fig. 9. These misclassified cases indicates one of the strong drawback/limitation of VisDA-2017 dataset compared to other visual datasets, i.e., VisDA-2017 dataset has image samples with multiple true labels instead of a unique label for each image sample. Since the images might contain multiple true classes for an image, ideally all these true labels are to be associated with the image in the dataset to rightly evaluate any trained model for its efficacy. Because we perform single-label multi-class classification, predicting any one of the true labels of the image should be considered as right for the evaluation metric. However, this is not the case as the dataset does not record all the true labels for the images. So, if one plans to rightly use this dataset, all the true labels are to be annotated for each of the images in the dataset or use other benchmark datasets such as DomainNet/LSDAC (Large Scale Domain Adaptation Challenge) dataset [18] that alleviates the problem of multi-labels of VisDA-2017 dataset as DomainNet dataset only consist of single true label per each image in the dataset, resulting in correct evaluation without the issue of misclassification we have indicated above for the VisDA-2017 dataset.

Apart from analyzing the limitation of the VisDA-2017 dataset, we also analyze the nature of the feature representations learnt by contradistinguisher. In order to visualize the features that prompted the predicted label, we use Captum[4], an open-source, extensible library for model interpretability built on PyTorch[5] [71]. We use gradient-based attribution to compute the integrated gradients for a given image using the predicted label. We obtain the high-level features or the saliency maps [72] for the given image. In terms of high-level features in a given image, one can imagine features such as shape, color, texture, size, etc. to be the features that help in predicting the classifier outcome. Out of all these features, the most natural and basic feature influencing the outcome is observed to be the shapes of the objects. Extensive research materials in psychology such as [73], [74], [75], [76], [77] have indicated that human babies and adults tend to utilize shapes than color/material/texture to assign a word label to the given object. This particular phenomenon is widely termed as 'shape bias' in the literature. However, recently, it was shown that the ImageNet pre-trained models possess a texture bias over shape bias [78]. To improve the shape bias, [78] propose a new modified dataset called 'Stylized-ImageNet' to overcome the texture bias. By increasing the shape bias, [78] demonstrated improved performance and robustness. Since we use the ImageNet pre-trained ResNet-101 as a feature extractor, it is necessary to understand the nature of extracted features from the input images.

Unlike 'Stylized-ImageNet', in domain adaptation tasks, one cannot always expect to get such a curated dataset with ground truth labels on the target domain for each task. Instead, it might be easy and desirable to change the loss function that enhances the shape features with the same training dataset.Surprisingly, in our observations, we find that the features learnt by the classifier indicate the high-level features or the saliency maps [72] representing the shape of the object in the image. The contradistinguish loss is formulated and optimized in such a way that the features extracted are most unique and contrastive for a given image in comparison to other images in the dataset. This consequently is observed as the features corresponding to shapes in the form of silhouette in the feature visualizations in Fig. 9 as each object posses a unique shape as it's most contrasdistinguishing character, i.e., the character which is most discriminative and unique to the given image.

5 Concluding Remarks

In this paper, we have proposed a direct approach to solve the problem of unsupervised domain adaptation that is different from the standard distribution alignment approaches. In our approach, we jointly learn a Contradistinguisher on the source and target domain distribution in the same input-label feature space using contradistinguish loss for unsupervised target domain to identify contrastive features. We have shown that contrastive learning overcomes the need and drawbacks of domain alignment, especially in tasks where domain shift is very high (e.g., language domains) and data augmentation techniques cannot be applied. Due to the inclusion of prior enforcing in the contradistinguish

4. https://captum.ai/tutorials/Resnet_TorchVision_Interpret
5. https://pytorch.org/

loss, the proposed unsupervised domain adaptation method CUDA could incorporate any known target domain prior to overcoming the drawbacks of skewness in the target domain, thereby resulting in a skew-robust model. We validated the efficacy of CUDA by experimenting on the synthetically created toy-dataset. We further demonstrated the simplicity and effectiveness of our proposed method by performing multi-source domain adaptation on Office-31 and Digits datasets to consistently outperform other multi-source domain adaptation approaches.

We have also tested the proposed method CUDA on the recent benchmark visual domain adaptation datasets such Office-31 and VisDA-2017 classification datasets and demonstrated above/on-par results with the state-of-the-art approaches. We further analyzed the nature of the feature representation learnt using contradistinguish loss to identify the features related to the shapes that influence the predicted outcome. As the features related to shapes are learnt, we observed that it helps improving the performance and robustness of the trained model as the model is not biased to colors/textures in the images. We concluded that learning and improving shape bias is one of the keys to achieve ideal transfer learning and domain adaptation.

ACKNOWLEDGMENTS

The authors would like to thank the Ministry of Human Resource Development (MHRD), Government of India, for their generous funding towards this work through the UAY Project: IISc 001. The authors thank Tejas Duseja for helping the authors with setting up some experiments. The authors would also like to thank anonymous reviewers for providing their valuable feedback that helped in improving the manuscript.

REFERENCES

[1] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks," in *ICML*, 2015.
[2] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Unsupervised domain adaptation with residual transfer networks," in *NIPS*, 2016.
[3] ——, "Deep transfer learning with joint adaptation networks," in *ICML*, 2017.
[4] P. Häusser, T. Frerix, A. Mordvintsev, and D. Cremers, "Associative domain adaptation," in *ICCV*, 2017.
[5] K. Saito, K. Watanabe, Y. Ushiku, and T. Harada, "Maximum classifier discrepancy for unsupervised domain adaptation," in *CVPR*, 2018.
[6] G. French, M. Mackiewicz, and M. Fisher, "Self-ensembling for visual domain adaptation," in *ICLR*, 2018.
[7] C. Louizos, K. Swersky, Y. Li, M. Welling, and R. S. Zemel, "The variational fair autoencoder," in *ICLR*, 2016.
[8] W. Zellinger, T. Grubinger, E. Lughofer, T. Natschläger, and S. Saminger-Platz, "Central moment discrepancy (CMD) for domain-invariant representation learning," in *ICLR*, 2017.
[9] A. Rozantsev, M. Salzmann, and P. Fua, "Beyond sharing weights for deep domain adaptation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2019.
[10] M. Mancini, L. Porzi, S. Rota Bulò, B. Caputo, and E. Ricci, "Inferring latent domains for unsupervised deep domain adaptation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2019.
[11] ——, "Boosting domain adaptation by discovering latent domains," in *CVPR*, 2018.
[12] F. M. Carlucci, L. Porzi, B. Caputo, E. Ricci, and S. R. Bulò, "Autodial: Automatic domain alignment layers," in *ICCV*, 2017.
[13] ——, "Just dial: Domain alignment layers for unsupervised domain adaptation," in *ICIAP*, 2017.
[14] R. Gopalan, R. Li, and R. Chellappa, "Domain adaptation for object recognition: An unsupervised approach," in *ICCV*, 2011.
[15] J. Xie, W. Hu, S.-C. Zhu, and Y. N. Wu, "Learning sparse frame models for natural image patterns," *International Journal of Computer Vision (IJCV)*, 2015.
[16] Y. Pan, T. Yao, Y. Li, Y. Wang, C. Ngo, and T. Mei, "Transferrable Prototypical Networks for Unsupervised Domain Adaptation," in *CVPR*, 2019.
[17] G. Kang, L. Jiang, Y. Yang, and A. G. Hauptmann, "Contrastive Adaptation Network for Unsupervised Domain Adaptation," in *CVPR*, 2019.
[18] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang, "Moment Matching for Multi-Source Domain Adaptation," in *ICCV*, 2019.
[19] M. Ghifary, W. B. Kleijn, M. Zhang, D. Balduzzi, and W. Li, "Deep reconstruction-classification networks for unsupervised domain adaptation," in *ECCV*, 2016.
[20] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan, "Domain separation networks," in *NIPS*, 2016.
[21] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *ICML*, 2015.
[22] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *Journal of Machine Learning Research (JMLR)*, 2016.
[23] M. Liu and O. Tuzel, "Coupled generative adversarial networks," in *NIPS*, 2016.
[24] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *CVPR*, 2017.
[25] S. Sankaranarayanan, Y. Balaji, C. D. Castillo, and R. Chellappa, "Generate to adapt: Aligning domains using generative adversarial networks," in *CVPR*, 2018.
[26] Y. C. Liu, Y. Y. Yeh, T. C. Fu, S. D. Wang, W. C. Chiu, and Y. C. F. Wang, "Detach and adapt: Learning cross-domain disentangled deep representation," in *CVPR*, 2018.
[27] P. Russo, F. M. Carlucci, T. Tommasi, and B. Caputo, "From source to target and back: Symmetric bi-directional adaptive gan," in *CVPR*, 2018.
[28] J. Hoffman, E. Tzeng, T. Park, J. Zhu, P. Isola, K. Saenko, A. Efros, and T. Darrell, "CyCADA: Cycle-consistent adversarial domain adaptation," in *ICML*, 2018.
[29] S. Xie, Z. Zheng, L. Chen, and C. Chen, "Learning semantic representations for unsupervised domain adaptation," in *ICML*, 2018.
[30] M. Long, Z. Cao, J. Wang, and M. I. Jordan, "Conditional adversarial domain adaptation," in *NIPS*, 2018.
[31] C. Chen, Z. Chen, B. Jiang, and X. Jin, "Joint domain alignment and discriminative feature learning for unsupervised deep domain adaptation," in *AAAI*, 2019.
[32] R. Shu, H. Bui, H. Narui, and S. Ermon, "A DIRT-t approach to unsupervised domain adaptation," in *ICLR*, 2018.
[33] E. Hosseini-Asl, Y. Zhou, C. Xiong, and R. Socher, "Augmented cyclic adversarial learning for low resource domain adaptation," in *ICLR*, 2019.
[34] J. Liang, R. He, Z. Sun, and T. Tan, "Aggregating randomized clustering-promoting invariant projections for domain adaptation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2018.
[35] R. Xu, Z. Chen, W. Zuo, J. Yan, and L. Lin, "Deep cocktail network: Multi-source unsupervised domain adaptation with category shift," in *CVPR*, 2018.
[36] X. Chen, S. Wang, M. Long, and J. Wang, "Transferability vs. discriminability: Batch spectral penalization for adversarial domain adaptation," in *ICML*, 2019.
[37] X. Wang, Y. Jin, M. Long, J. Wang, and M. I. Jordan, "Transferable Normalization: Towards Improving Transferability of Deep Neural Networks," in *NIPS*, 2019.
[38] J. Li, E. Chen, Z. Ding, L. Zhu, K. Lu, and H. T. Shen, "Maximum Density Divergence for Domain Adaptation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020.
[39] V. K. Kurmi, S. Kumar, and V. P. Namboodiri, "Attending to Discriminative Certainty for Domain Adaptation," in *CVPR*, 2019.
[40] C. Chen, W. Xie, W. Huang, Y. Rong, X. Ding, Y. Huang, T. Xu, and J. Huang, "Progressive Feature Alignment for Unsupervised Domain Adaptation," in *CVPR*, 2019.

[41] V. N. Vapnik, "An overview of statistical learning theory," *IEEE transactions on neural networks*, 1999.
[42] V. Vapnik, *The nature of statistical learning theory*. Springer Science & Business Media, 2013.
[43] S. Balgi and A. Dukkipati, "Cuda: Contradistinguisher for unsupervised domain adaptation," in *ICDM*, 2019.
[44] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *ECCV*, 2010.
[45] X. Peng, B. Usman, N. Kaushik, J. Hoffman, D. Wang, and K. Saenko, "VisDA: The Visual Domain Adaptation Challenge," *CoRR*, 2017.
[46] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research (JMLR)*, 2011.
[47] G. Pandey and A. Dukkipati, "Unsupervised feature learning with discriminative encoder," in *ICDM*, 2017.
[48] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, 1989.
[49] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *IEEE*, 1998.
[50] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading Digits in Natural Images with Unsupervised Feature Learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.
[51] A. Gretton, K. Fukumizu, Z. Harchaoui, and B. K. Sriperumbudur, "A fast, consistent kernel two-sample test," in *NIPS*, 2009.
[52] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *Journal of Machine Learning Research (JMLR)*, 2012.
[53] D. Sejdinovic, B. Sriperumbudur, A. Gretton, K. Fukumizu *et al.*, "Equivalence of distance-based and rkhs-based statistics in hypothesis testing," *The Annals of Statistics*, 2013.
[54] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *NIPS*, 2017.
[55] S. Laine and T. Aila, "Temporal ensembling for semi-supervised learning," in *ICLR*, 2017.
[56] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *ICLR*, 2014.
[57] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015.
[58] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014.
[59] K. Saito, Y. Ushiku, and T. Harada, "Asymmetric tri-training for unsupervised domain adaptation," in *ICML*, 2017.
[60] S. Ruder and B. Plank, "Strong baselines for neural semi-supervised learning under domain shift," in *ACL*, 2018.
[61] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society: Series B (Methodological)*, 1977.
[62] D.-H. Lee, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," *ICML Workshops (WREPL)*, 2013.
[63] Y. Grandvalet and Y. Bengio, "Semi-supervised learning by entropy minimization," in *NIPS*, 2005.
[64] C. L. Li, W. C. Chang, Y. Cheng, Y. Yang, and B. Póczos, "MMD-GAN: Towards deeper understanding of moment matching network," in *NIPS*, 2017.
[65] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.
[66] J. Jacobsen, J. Behrmann, R. S. Zemel, and M. Bethge, "Excessive invariance causes adversarial vulnerability," in *ICLR*, 2019.
[67] T. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *ECCV*, 2014.
[68] E. Real, J. Shlens, S. Mazzocchi, X. Pan, and V. Vanhoucke, "YouTube-BoundingBoxes: A Large High-Precision Human-Annotated Data Set for Object Detection in Video," in *CVPR*, 2017.
[69] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research (JMLR)*, 2008.
[70] Y. Zhang and B. D. Davison, "Impact of ImageNet Model Selection on Domain Adaptation," in *WACV Workshops*, 2020.
[71] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," *NIPS Workshops*, 2017.
[72] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps," in *ICLR Workshop*, 2014.
[73] B. Landau, L. B. Smith, and S. Jones, "Syntactic context and the shape bias in children's and adults' lexical learning," *Journal of Memory and Language*, 1992.
[74] G. Diesendruck and P. Bloom, "How specific is the shape bias?" *Child development*, 2003.
[75] E. R. Potrzeba, D. Fein, and L. Naigles, "Investigating the shape bias in typically developing children and children with autism spectrum disorders," *Frontiers in Psychology*, 2015.
[76] S. Ritter, D. G. Barrett, A. Santoro, and M. M. Botvinick, "Cognitive psychology for deep neural networks: A shape bias case study," in *ICML*, 2017.
[77] H. Hosseini, B. Xiao, M. Jaiswal, and R. Poovendran, "Assessing shape bias property of Convolutional Neural Networks," in *CVPR Workshops*, 2018.
[78] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, "ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness," in *ICLR*, 2019.

Sourabh Balgi is a doctoral candidate in causality and machine learning, advised by Prof. Jose M. Peña at the Statistics and Machine Learning (STIMA) division, Department of Computer and Information Science (IDA), Linköping University, Sweden. He received his B.E. degree in Electronics and Communication Engineering from Sri Jayachamarajendra College of Engineering, Mysore, India in 2013. He worked at Mercedes-Benz Research and Development India, Bangalore as a software engineer from 2013 to 2016. He received his M.Tech. degree in Artificial Intelligence from Indian Institute of Science (IISc), Bangalore, India in 2019. He worked as a Research Associate at the Statistics and Machine Learning Lab, Department of Computer Science and Automation (CSA), Indian Institute of Science (IISc), Bangalore, India. His research interests include unsupervised deep learning for computer vision and machine learning with an emphasis on transfer learning, specifically domain adaptation and disentangled representation learning for domain adaptation and causal inference.

Ambedkar Dukkipati is an Associate Professor at the Department of Computer Science and Automation, IISc. He received his Ph.D. degree from the Department of Computer Science and Automation, Indian Institute of Science (IISc), Bangalore, India and B.Tech from IIT Madras. He held a post-doctoral position at EURANDOM, Netherlands. Currently, he also heads the Statistics and Machine Learning group at the Department of Computer Science and Automation, IISc. His research interests include statistical network analysis, network representation learning, spectral graph methods, machine learning in low data regime, sequential decision-making under uncertainty and deep reinforcement learning.