Unsupervised Domain Adaptation via Regularized Conditional Alignment

Safa Cicek, Stefano Soatto
UCLA Vision Lab
University of California, Los Angeles, CA 90095
{safacicek,soatto}@ucla.edu

Abstract

We propose a method for unsupervised domain adaptation that trains a shared embedding to align the joint distributions of inputs (domain) and outputs (classes), making any classifier agnostic to the domain. Joint alignment ensures that not only the marginal distributions of the domains are aligned, but the labels as well. We propose a novel objective function that encourages the class-conditional distributions to have disjoint support in feature space. We further exploit adversarial regularization to improve the performance of the classifier on the domain for which no annotated data is available.

1. Introduction

In the context of classification, unsupervised domain adaptation (UDA) consists of modifying a classifier trained on a labeled dataset, called the "source," so it can function on data from a different "target" domain, for which no annotations are available. More in general, we want to train a model to operate on input data from both the source and target domains, despite absence of annotated data for the latter. For instance, one may have a synthetic dataset, where annotation comes for free, but wish for the resulting model to work well on real data, where manual annotation is scarce or absent [30].

The most successful methods learn the parameters of a deep neural network using adversarial (min-max) criteria. The idea is to simultaneously recognize the class (output) as well as the domain (*e.g.*, "real vs. synthetic") by training the classifier to work as well as possible on both while encoder is fooling the discriminator for the latter. In a sense, the classifier becomes agnostic to the domain. This can be understood as aligning the marginal distribution of the inputs from the two domains. Unfortunately, this does not guarantee successful transfer, for it is possible that the source (say synthetic images) be perfectly aligned with the target (say natural images), and yet a natural image of a cat map to a synthetic image of a dog. It would be desirable, therefore, for the adaptation to align the outputs, along with the inputs. This prompted other methods to align, instead of the marginal distributions, the joint or conditional distribution of domain and class. This creates two problems: First, the target class labels are unknown; second, since there is a shared representation of the inputs, aligning the joint distributions may cause them to collapse thus losing the discriminative power of the model.

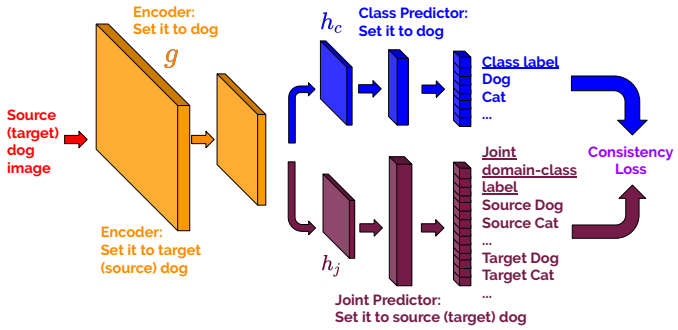

Figure 1. The network structure of the proposed approach. We propose to learn a joint distribution $P(d, y)$ over domain label d and class label y by a joint predictor (purple). The encoder (orange) is trained to confuse this joint predictor by matching the features corresponding to the same category samples of both domains. Since labels for the target data is not known, predictions of the class predictor (blue) on the target data is used with the help of consistency loss. Unlabeled data is further exploited with input smoothing algorithm VAT [26] from the SSL literature.

To address these problems, we propose a method to perform the alignment of the joint distribution (Sect. 2.2). We employ ideas from semi-supervised learning (SSL) to improve generalization performance (Sect. 2.3). We propose an optimization scheme that uses a two-folded label space. The resulting method performs at the state of the art without pushing the limits of hyperparameter optimization (Sect. 3). We analyze the proposed objective function in the supervised setting and prove that the optimal solution conditionally aligns the distributions while keeping them discriminative (Sect. 4). Finally, we discuss our contribution in

relation to the vast and growing literature on UDA (Sect. 5).

Formalization

We are given N^s labeled source samples $x^s \in X^s$ with corresponding labels $y^s \in Y^s$ and N^t unlabeled target samples, $x^t \in X^t$. The entire training dataset X has cardinality $N = N^s + N^t$. Labeled source data and unlabeled target data are drawn from two different distributions (domain shift): $(x^s, y^s) \sim P^s$, $(x^t, y^t) \sim P^t$ where their discrepancy, measured by Kullbach-Liebler's (KL) divergence, is $KL(P^s||P^t) > 0$ (covariate shift). Both distributions are defined on $X \times Y$ where $Y = \{1, ..., K\}$. Marginal distributions are defined on X and samples are drawn from them as $x^s \sim P^s_x$, $x^t \sim P^t_x$. Given finite samples $\{(x^s_i, y^s_i)\}_{i=1}^{N^s} := \{(x^s_1, y^s_1), (x^s_2, y^s_2), ..., (x^s_{N^s}, y^s_{N^s})\}$ from P^s and $\{(x^t_i)\}_{i=1}^{N^t} := \{x^t_1, x^t_2, ..., x^t_{N^t}\}$ from P^t_x, the goal is to learn a classifier $f : X \to Y$ with a small risk in the target domain. This risk can be measured with cross-entropy:

$$\min_f E_{(x,y)\sim P^t} \ell_{CE}(f(x); y) \tag{1}$$

where

$$\ell_{CE}(f(x); y) := -\langle y, \log f(x) \rangle \tag{2}$$

is the cross-entropy loss calculated for one-hot, ground-truth labels $y \in \{0, 1\}^K$ and label estimates $f(x) \in \mathbb{R}^K$, which is the output of a deep neural network with input x, and K is the number of classes.

2. Proposed Method

In this section, we show how to formalize the criterion for aligning both inputs and outputs, despite the latter being unknown for the target classes in the absence of supervision.

Alignment of the marginal distributions can be done using Domain Adversarial Neural Networks (DANN) [10], that add to the standard classification loss for the source data a binary classification loss for the domain: Source vs. target. If all goes well, the class predictor classifies the *source* data correctly, and the binary-domain predictor is unable to tell the difference between the source and the target data. Therefore, the class predictor might also classify the target data correctly. Unfortunately, this is not guaranteed as there can be a misalignment of the output spaces that cause some class in the source to map to a different class in the target, *e.g.*, a natural cat to a synthetic dog.

The *key idea* of our approach is to impose *not* a binary adversarial loss on the domain alignment, but a $2K$-way adversarial loss, as if we had $2K$ possible classes: The first K are the known *source classes*, and the second K are the unknown *target classes*. We call the result a *joint domain-class predictor* or *joint predictor* in short, since it learns a distribution over domain and class variables. The encoder will try to fool the predictor by minimizing the classification loss between a dog sample in the source and a sample in the target domain whose predicted label in the aligned domain is also dog.

During training, the probabilities assigned to the first K labels of the joint predictor are very small for the target samples, and they eventually converge to zero. Therefore, we need a separate mechanism to provide pseudo-labels to the target samples to be aligned by the joint predictor. For this, we train another predictor, that we call *class predictor* outputting only class labels. The class predictor is trained on both the source data using ground-truth labels and the target data using semi-supervised learning (SSL) regularizers.

Both the joint predictor and the class predictor can be used for inference. However, we find that the class predictor performs slightly better. We conjecture this is because joint predictor is trained on a harder task of domain and class prediction while only the latter one is needed at inference time.

We consider UDA as a two-fold problem. The first step deals with domain shift by aligning distributions in feature space. Given a successful alignment, one can use a source-only trained model for inference. But, once the domains are matched, it is possible to further improve generalization by acting on the label space. Ideas from SSL can help to that end [26].

The overall architecture of the model is described in Fig. 1.

2.1. Network structure

We denote the shared encoder with g, the class predictor with h_c, the joint predictor with h_j and the overall networks as $f_c = h_c \circ g$ and $f_j = h_j \circ g$. Then, the class-predictor output for an input x can be written as,

$$f_c(x) = h_c(g(x)) \in \mathbb{R}^K. \tag{3}$$

Similarly, the joint-predictor output can be written as,

$$f_j(x) = h_j(g(x)) \in \mathbb{R}^{2K}. \tag{4}$$

2.2. Loss functions

The class predictor is the main component of the network which is used for inference. Its marginal features are aligned by the loss provided by the joint predictor. The class predictor is trained with the labeled source samples using the cross-entropy loss. This source classification loss can be written as,

$$L_{sc}(f_c) = E_{(x,y)\sim P^s} \ell_{CE}(f_c(x), y). \tag{5}$$

Both the encoder (g) and the class predictor (h_c) are updated while minimizing this loss.

We also update the joint predictor with the same classification loss for the labeled source samples. This time, only the joint-predictor (h_j) is updated. The joint-source classification loss is

$$L_{jsc}(h_j) = E_{(x,y)\sim P^s} \ell_{CE}(h_j(g(x)), [y,\, \mathbf{0}]) \qquad (6)$$

where $\mathbf{0}$ is the zero vector of size K, chosen to make the last K joint probabilities zero for the source samples.

Similarly, the joint predictor is trained with target samples. As ground-truth labels for the target samples are not given, label estimates from the class predictors are used as pseudo-labels. The joint target classification loss is

$$L_{jtc}(h_j) = E_{x\sim P_x^t} \ell_{CE}(h_j(g(x)), [\mathbf{0},\, \hat{y}]) \qquad (7)$$

where $\hat{y} = e_k$ and $k = \arg\max_k f_c(x)[k] = \arg\max_k h_c(g(x))[k]$, e_k is the identity of size K whose kth element is 1.[1] Here, we assume that the source-only model achieves reasonable performance on the target domain (e.g. better than a chance). For experiments where the source-only trained model has poor performance initially, we apply this loss after the class predictor is trained for some time. Since the joint predictor is trained with the estimates of the class predictor on the target data, it can also be interpreted as a *student* of the class predictor.

The goal of introducing a joint predictor was to align label-conditioned feature distributions. For this, encoders are trained to fool the joint predictor as in [10]. Here, we apply conditional fooling. The joint source alignment loss is

$$L_{jsa}(g) = E_{(x,y)\sim P^s} \ell_{CE}(h_j(g(x)), [\mathbf{0},\, y]). \qquad (8)$$

The encoder is trained to fool by changing the joint label from $[y,\, \mathbf{0}]$ to $[\mathbf{0},\, y]$. Similarly, the joint-target alignment loss is defined by changing the pseudo-labels from $[\mathbf{0},\, \hat{y}]$ to $[\hat{y},\, \mathbf{0}]$,

$$L_{jta}(g) = E_{x\sim P_x^t} \ell_{CE}(h_j(g(x)), [\hat{y},\, \mathbf{0}]). \qquad (9)$$

The last two losses are minimized only by the encoder g.

2.3. Exploiting unlabeled data with SSL regularizers

Once features of the source and the target domains are matched, our formulation of UDA turns into a semi-supervised learning problem. In a way, adversarial domain adaptation deals with the large domain shift between source and target datasets while adversarial input smoothing removes the shift in predictions within a small neighborhood of a domain (See Fig. 2).

[1] We use the notation $x[k]$ for indexing the value at the kth index of the vector x.

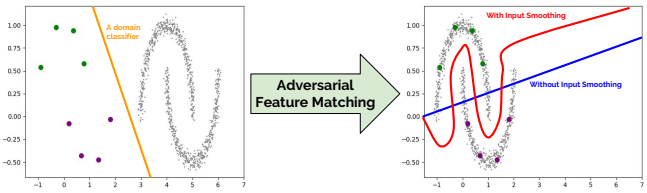

Figure 2. **Left.** In the UDA setting, there exist domain classifiers (e.g. orange line segment) being able to distinguish the source samples (green and purple dots) from the target samples (gray dots). Conditional feature matching is applied until there is no such classifier in the finite-capacity classifier space. As a result, the label-conditioned feature distributions of the source and the target data are matched. **Right.** Once the features are matched, exploiting unlabeled data using SSL regularizers like VAT [26] becomes trivial. Only using labeled samples (green and purple dots) gives a poor decision boundary (blue line segment). When input adversarial training is applied using unlabeled samples (gray dots), the desired decision boundary is achieved (red curve). Best viewed in color.

For a discriminative model to exploit unlabeled data, there has to be some prior on the model parameters or on the unknown labels [5]. Applying entropy minimization for the predictions on the unlabeled data is a well-known regularizer in the SSL literature [13, 15, 7]. This regularization forces decision boundaries to be in the low-density region, a desired property under the cluster assumption [5]. Our class predictor is trained to minimize this target entropy loss,

$$L_{te}(f_c) = E_{x\sim P_x^t} \ell_E(h_c(g(x))) \qquad (10)$$

where $\ell_E(f(x)) := -\langle f(x), \log f(x)\rangle$. Since the joint predictor is already trained on the low-entropy estimates of the class predictor, it is enough to apply it to the class predictor. Minimizing entropy satisfies the cluster assumption only for Lipschitz classifiers [13]. The Lipschitz condition can be realized by applying adversarial training as suggested by [27, 26]. VAT [26] makes a second-order approximation for adversarial input perturbations Δx and proposes the following approximation to the adversarial noise for each input x:

$$\Delta x \approx \epsilon_x \frac{r}{||r||_2}$$
$$\text{subject to } r = \nabla_{\Delta x} \ell_{CE}(f(x), f(x+\Delta x))\Big|_{\Delta x = \xi d} \qquad (11)$$

where $d \sim N(0,1)$. Therefore, the regularization loss of [27, 26] is

$$\ell_{VAT}(f(x)) := \ell_{CE}(f(x), f(x + \epsilon_x \frac{r}{||r||_2}))$$
$$\text{subject to } r = \nabla_{\Delta x} \ell_{CE}(f(x), f(x+\Delta x))\Big|_{\Delta x = \xi d} \qquad (12)$$

for one input sample x. We will apply this regularizer both on the source and the target training data as in [35, 17]. So, the source and target losses are given as follows:

$$L_{svat}(f_c) = E_{(x,y)\sim P^s} \ell_{VAT}(f_c(x)) \tag{13}$$

and

$$L_{tvat}(f_c) = E_{x\sim P_x^t} \ell_{VAT}(f_c(x)). \tag{14}$$

SSL regularizations can be applied in a later stage once feature matching is achieved [35]. But, we find that in most tasks, applying SSL regularizers from the beginning of the training also works well. More details are given in the Supp. Mat.

We combine the objective functions introduced in this section and the previous section. The overall adversarial loss functions for the source and the target samples can be written as follows,

$$L_{adv}(g) = \lambda_{jsa} L_{jsa}(g) + \lambda_{jta} L_{jta}(g) \tag{15}$$

The remaining objective functions are

$$L(g, h_j, h_c) = L_s(g, h_j, h_c) + \lambda_t L_t(g, h_j, h_c). \tag{16}$$

where

$$L_s(g, h_j, h_c) = L_{sc}(f_c) + \lambda_{svat} L_{svat}(f_c) + \lambda_{jsc} L_{jsc}(h_j) \tag{17}$$

$$L_t(g, h_j, h_c) = L_{te}(f_c) + \lambda_{tvat} L_{tvat}(f_c) + \lambda_{jtc} L_{jtc}(h_j). \tag{18}$$

The proposed method minimizes Eq. 15 and Eq. 16 in an alternating fashion.

2.4. Connection to domain adaptation theory

The work of [2] provides an upper bound on the target risk: $\ell_t(h, y) = E_{(x,y)\sim P^t}[|h(x) - y|]$ where h is the classifier. One component in the upper bound is a divergence term between two domain distributions. In UDA, we are interested in the difference of the measures between subsets of two domains on which a hypothesis in the finite-capacity hypothesis space $\mathcal{H}$ can commit errors. Instead of employing traditional metrics (e.g. the total variation distance), they use the $\mathcal{H}$-divergence. Given a domain X with P and Q probability distributions over X, and $\mathcal{H}$ a hypothesis class on X, the $\mathcal{H}$-divergence is

$$d_{\mathcal{H}\Delta\mathcal{H}}(P, Q) := 2 \sup_{h,h'\in\mathcal{H}} |Pr_{x\sim P}(h(x) \neq h'(x)) - Pr_{x\sim Q}(h(x) \neq h'(x))|. \tag{19}$$

Now, we can recall the main Theorem of [2]. *Let $\mathcal{H}$ be an hypothesis space of VC dimension d. If X^s, X^t are unlabeled samples of size m' each, drawn from P_x^s and P_x^t respectively, then for any $\delta \in (0, 1)$, with probability at least $1 - \delta$ (over the choice of the samples), for every $h \in \mathcal{H}$:*

$$\ell_t(h) \leq \ell_s(h) + \frac{1}{2}\hat{d}_{H\Delta H}(X^s, X^t) + 4\sqrt{\frac{2d\log(2m') + \log(\frac{2}{\delta})}{m'}} + \lambda \tag{20}$$

where $\lambda = \ell_s(h^*) + \ell_t(h^*)$, $h^* = \arg\min_{h\in\mathcal{H}} \ell_s(h) + \ell_t(h)$ and $\hat{d}_{H\Delta H}(X^s, X^t)$ is empirical $\mathcal{H}$ divergence. In words, the target risk is upper bounded by the source risk, empirical $\mathcal{H}$-divergence and combined risk of ideal joint hypothesis λ.

If there is no classifier which can discriminate source samples from target samples then the empirical $\mathcal{H}$-divergence is zero from Lemma 2 of [2]. DANN of [10] minimizes $\hat{d}_{H\Delta H}(X^s, X^t)$ by matching the marginal distributions (i.e. by aligning marginal push-forwards $g\#P_x^s$ and $g\#P_x^t$). But, if the joint push-forward distributions ($g\#P^s$ and $g\#P^t$) are not matched accurately, there may not be a classifier in the hypothesis space with low risk in both domains. Hence, λ has to be large for any hypothesis space $\mathcal{H}$.

Our proposed method tackles this problem, by making sure that the label-conditioned push-forwards are aligned disjointly. With disjoint alignment, we mean that no two samples with different labels can be assigned to the same feature point. Moreover, the third term in the upper bound decreases with the number of samples drawn from both domains. This number can increase with data augmentation. VAT has the same effect of augmenting the data with adversarially perturbed images where the small perturbations are nuisances for the task.

3. Empirical Evaluation

3.1. Implementation details

We evaluate the proposed method on the standard digit and object image classification benchmarks in UDA. Namely, CIFAR $\rightarrow$ STL, STL $\rightarrow$ CIFAR, MNIST $\rightarrow$ SVHN, SVHN $\rightarrow$ MNIST, SYN-DIGITS $\rightarrow$ SVHN and MNIST $\rightarrow$ MNIST-M. The first three settings are the most challenging ones where state-of-the-art (SOA) methods accuracies are still below 90%. Our method achieves SOA accuracy in all these tasks.

CIFAR $\leftrightarrow$ STL. Similar to CIFAR, STL images are acquired from labeled examples on ImageNet. However, images are 96×96 instead of the 32×32 images in CIFAR. All images are converted to 32×32 RGB in pretraining. We down-sampled images by local averaging. Note that we only used the labeled part of STL in all the experiments. CIFAR and STL both have 10 classes, 9 of which are common

Dataset	Number of training samples	Number of test samples	Number of classes	Resolution	Channels
MNIST [18]	$60,000$	$10,000$	10	28×28	Mono
SVHN [28]	$73,257$	$26,032$	10	32×32	RGB
CIFAR10 [16]	$50,000$	$10,000$	10	32×32	RGB
STL [8]	$5,000$	$8,000$	10	96×96	RGB
SYN-DIGITS [10]	$479,400$	$9,553$	10	32×32	RGB

Table 1. Specs of the datasets used in the experiments.

for both datasets. Like previous works [17, 9] we removed the non-overlapping classes (class frog and class monkey) reducing the problem into a 9-class prediction. See Table 1 for the specs of the datasets.

MNIST $\leftrightarrow$ SVHN. We convert MNIST images to RGB images by repeating the gray image for each color channel and we resize them to 32×32 by padding zeros. Following previous works [17, 9], we used Instance Normalization (IN) for MNIST $\leftrightarrow$ SVHN, which is introduced by [40] for image style transfer. We preprocess images both at training and test time with IN.

SYN-DIGITS $\rightarrow$ SVHN. SYN-DIGITS [10] is a dataset of synthetic digits generated from Windows fonts by varying position, orientation and background. In each image, one, two or three digits exist. The degrees of variation were chosen to match SVHN.

MNIST $\rightarrow$ MNIST-M. MNIST-M [10] is a difference-blend of MNIST over patches randomly extracted from color photos from BSDS500 [1]. I.e. $I^{out}_{ijk} = |I^1_{ijk} - I^2_{ijk}|$, where i, j are the coordinates of a pixel and k is a channel index. MNIST-M images are RGB and 28 by 28. MNIST images are replicated for each channel during preprocessing.

No data augmentation is used in any of the experiments to allow for a fair comparison with SOA methods [17, 35]. Again, to allow fair comparison with the previous works [9, 35], we have not used sophisticated architectures like ResNet [14]. Networks used in the experiments are given in the Supp. Mat. We report inference performance of the class-predictor.

We feed source and training samples into two different mini-batches at each iteration of training. As we are using the same batch layers for both source and target datasets, mean and variance learned – to be used at inference time – are the running average over both source and target data statistics.

Office UDA experiments (Amazon$\rightarrow$Webcam, Webcam$\rightarrow$DSLR, DSLR$\rightarrow$Webcam) were used as the standard benchmark in early UDA works [23, 34, 19, 38]. However, recent SOA methods [31, 17, 35, 9] did not report on these datasets, as labels are noisy [3]. Moreover, this is a small dataset with $4,652$ images from 31 classes necessitating the use of Imagenet-pretrained networks. Hence, we also choose not to report experiments on this dataset.

3.2. Results

We report the performance of the proposed method in Table 2. In all the experiments, the proposed method achieves the best or the second-best results after Co-DA [17]. Especially in the most challenging tasks, for which SOA accuracies are below 90%, our method outperforms all the previous methods. Numbers reported in the corresponding papers are used except DANN for which reported scores from [35] are used.

Works we compare to include [11] which proposed Deep Reconstruction Classification network (DRCN). The cross-entropy loss on the source data and reconstruction loss on the target data are minimized. [35] applied the SSL method VAT to UDA which they call VADA (Virtual Adversarial Domain Adaptation). After training with domain-adversarial loss of [10] and VAT, they further fine-tune only on the target data with the entropy and VAT objectives. [17] suggested having two hypotheses in a way that they learn diverse feature embeddings while class predictions are encouraged to be consistent. They build this method on VADA of [35]. The proposed method can be further improved by combining with Co-DA even though we ignore it to highlight the effectiveness of the clean method. Compared to Co-DA, our method has the memory and computational time advantage of not training multiple encoders. [31] introduced ATT where two networks are trained on the source data and predictions of the networks are used as pseudo labels on the target data. Another network is trained on the target data with pseudo labels. A pseudo label is assigned if two networks agree and at least one of them is confident.

Source-only models are also reported as baselines. These models are trained without exploiting the target training data in standard supervised learning setting using the same learning procedure (e.g. network, number of iterations etc.) as UDA methods. Since CIFAR has a large labeled set (45000 after removing samples of class frog), CIFAR $\rightarrow$ STL has a high accuracy even without exploiting the unlabeled data. Still, the proposed method outperforms the source-only baseline by 2.24%. The target-only models are trained only on the target domain with class labels revealed. The target-only performance is considered as the empirical upper bound in some papers, but it is not necessarily the

Source dataset Target dataset	MNIST SVHN	SVHN MNIST	CIFAR STL	STL CIFAR	SYN-DIGITS SVHN	MNIST MNIST-M
[10] DANN*	60.6	68.3	78.1	62.7	90.1	94.6
[11] DRCN	40.05	82.0	66.37	58.86	NR	NR
[33] kNN-Ad	40.3	78.8	NR	NR	NR	86.7
[31] ATT	52.8	86.2	NR	NR	92.9	94.2
[9] Π-model**	33.87	93.33	77.53	71.65	96.01	NR
[35] VADA	47.5	97.9	80.0	73.5	94.8	97.7
[35] DIRT-T	54.5	99.4	NR	75.3	96.1	98.9
[35] VADA + IN	73.3	94.5	78.3	71.4	94.9	95.7
[35] DIRT-T +IN	76.5	99.4	NR	73.3	96.2	98.7
[17] Co-DA	81.7	99.0	81.4	76.4	96.4	99.0
[17] Co-DA + DIRT-T	88.0	**99.4**	NR	77.6	**96.4**	99.1
Ours	**89.19**	99.33	**81.65**	**77.76**	96.22	**99.47**
Source-only (baseline)	44.21	70.58	79.41	65.44	85.83	70.28
Target-only	94.82	99.28	77.02	92.04	96.56	99.87

Table 2. **Comparison to SOA UDA algorithms on the UDA image classification tasks.** Accuracies on the target test data are reported. Algorithms are trained on entire labeled source training data and unlabeled target training data. NR stands for not reported. * DANN results are implementation of [35] with instance normalized input. ** Results of [9] with minimal augmentations are reported. The proposed method achieves the best or second highest score after Co-DA. The proposed method can be combined with Co-DA, but we report the naked results to illustrate the effectiveness of the idea.

case, as seen in the CIFAR $\rightarrow$ STL setting where the target data is scarce; thus the target-only model is even worse than the source-only model.

The advantage of the proposed method is more apparent in the converse direction STL $\rightarrow$ CIFAR where the accuracy increases from 65.44% to 77.76%. STL contains a very small (4500 after removing samples of class monkey) labeled training set. That is why DIRT-T which fine-tunes on the target data, gave unreliable results for CIFAR $\rightarrow$ STL so they only report VADA result.

The source-only baseline has its lowest score in the MNIST $\rightarrow$ SVHN setting. This is a challenging task as MNIST is greyscale, in contrast to color digits in SVHN. Moreover, SVHN contains multiple digits within an image while MNIST pictures contain single, centered digits. SVHN $\rightarrow$ MNIST is a much simpler experimental setting where SOA accuracies are above 99%. We achieve SOA in MNIST $\rightarrow$ SVHN while being second best in SVHN $\rightarrow$ MNIST after Co-DA. Note that our accuracy in SVHN $\rightarrow$ MNIST is 99.33%. MNIST $\rightarrow$ MNIST-M and SYN-DIGITS $\rightarrow$ SVHN are other saturated tasks where our method beats SOA in the former one while being second best in the latter. At these levels of saturation of the dataset, top-rated performance is not as informative.

In MNIST $\rightarrow$ SVHN, our method (89.19%) is substantially better than VADA+IN (73.3%) which also uses input smoothing but with DANN (marginal alignment). Similarly, in STL $\rightarrow$ CIFAR, VADA achieves 73.5% while our method is SOA with 77.76% accuracy. This shows the effectiveness of our joint-alignment method.

To demonstrate the effectiveness of the proposed approach in aligning the samples of the same class, we visualize the t-Distributed Stochastic Neighbor Embedding (t-SNE) [24] of the source-only baseline and the proposed approach in Fig. 3. t-SNE is performed on the encoder output for 1000 randomly drawn samples from both source and target domains for STL $\rightarrow$ CIFAR setting. As one can see, samples of the same classes are better aligned for the proposed approach compared to the source-only method.

4. Analysis

The main result of our analysis is that the objective introduced in Sect. 2.2 is minimized only for matching conditional push-forwards given the optimal joint predictor (Theorem 1). For that, we first find the optimal joint predictor in Proposition 1. We operate under the supervised setting, assuming the target labels are revealed. So, we replace $\hat{y}$ in the objective functions with ground-truth labels y for the target samples. Proofs follow similar steps to Proposition 1 and Theorem 1 in [12].

Proposition 1. *The optimal joint predictor h_j minimizing $L_{jsc}(h_j) + L_{jtc}(h_j)$ given in the Eq. 6,7 for any feature z*

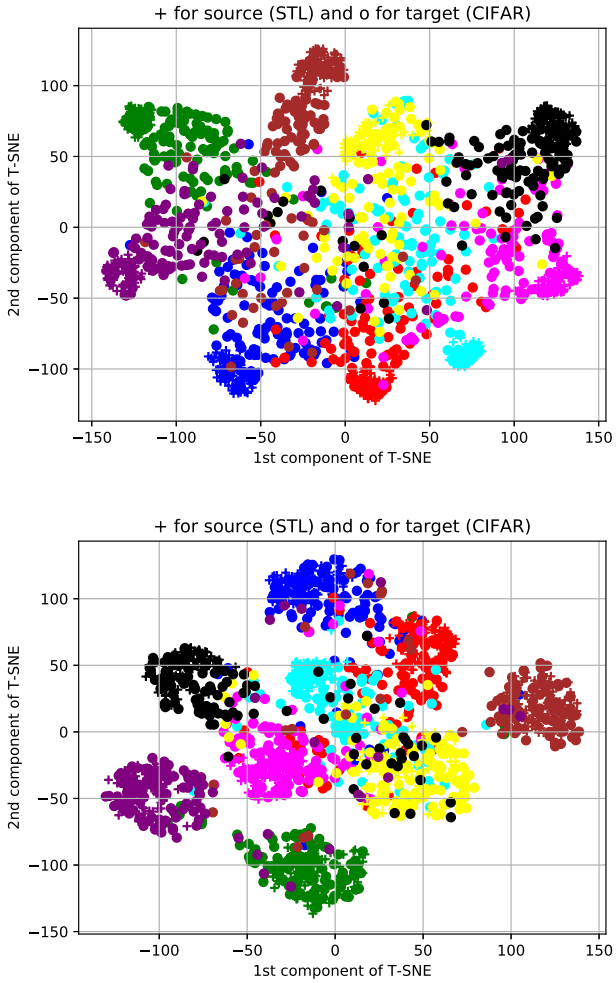

Figure 3. **t-SNE plots for STL → CIFAR.** t-SNE plots of the source-only trained (top panel) and the proposed method model (bottom panel). Encoder outputs are projected to two-dimensional space with t-SNE. Samples corresponding to the same class are visualized with the same color. The symbol "+" is used for the source samples and "o" is for the target samples. Best viewed in color.

with non-zero measure either on $g\#P_x^s(z)$ *or* $g\#P_x^t(z)$ *is*[2]

$$h_j(z)[i] = \frac{g\#P^s(z, y = e_i)}{g\#P_x^s(z) + g\#P_x^t(z)}$$

$$h_j(z)[i+K] = \frac{g\#P^t(z, y = e_i)}{g\#P_x^s(z) + g\#P_x^t(z)} \text{ for } i \in \{1, ..., K\}.$$

Theorem 1. *The objective* $L_{jsa}(g) + L_{jta}(g)$ *given in the Eq. 8-9 is minimized for the given optimal joint predictor if and only if* $g\#P^s(z|y = e_k) = g\#P^t(z|y = e_k)$ *and* $g\#P^s(z|y = e_k) > 0 \Rightarrow g\#P^s(z|y = e_i) = 0$ *for* $i \neq k$ *for any* $y = e_k$ *and* z.

[2]We used $P(z)$ both to denote the distribution of a random variable and the corresponding density function, but the meaning should be clear from the context.

Theorem 1 states that no two samples with different labels can be assigned to the same feature point for the encoder to minimize its loss given the optimal joint predictor. Moreover, the measure assigned to each feature is the same for the source and the target push-forward distributions to maximally fool the optimal joint predictor.

This result indicates that the global minimum of the proposed objective function is achieved when conditional feature distributions are aligned. But, this analysis does not necessarily give a guarantee that the converged solution is optimal in practice as we do not have access to the target labels in UDA. But, we demonstrated empirically in Fig. 3 that with reasonably good pseudo-labels provided by a separate class predictor, the objective gives better alignment than the source-only model.

The second issue is that finding the optimal predictor or generator with finite samples may not be possible, as optimal solutions are derived as functions of true measures instead of the network parameters trained on finite samples. Lastly, the joint predictor is not trained until convergence; instead, a gradient step is taken in alternating fashion for computational efficiency. So, the predictor is also not necessarily optimal in practice. Even though there are still gaps to be filled between this theory and practice, this analysis shows us that the proposed objective function is doing a sensible job given pseudo-labels for the target data are reasonably good.

5. Discussion and Related Work

In this section, we will summarize the most relevant works from the UDA literature. For more in-depth coverage of the literature see the recent survey of [41] on deep domain adaptation for various vision tasks. Many of the domain adaptation works can be categorized into two: (1) the ones learning a shared feature space (symmetric-feature based) and the ones transferring features of one domain to another (asymmetric-feature based).

Shared feature (symmetric-feature based). Feature transferability drops in the higher layers of a network and there may not exist an optimal classifier for both the source and the target data. Hence many works use two separate classifiers for the source and the target domains while the encoder parameters are shared. In these works, the source classifier is trained with the labeled source data and the target classifier is regularized by minimizing a distance metric between the source classifier using all the data.

One common such metric is the (Maximum Mean Discrepancy) MMD which is a measurement of the divergence between two probability distributions from their

samples by computing the distance of mean embeddings: $||\frac{1}{N^s}\sum_{i=1}^{N^s} g(x_i^s) - \frac{1}{N^t}\sum_{i=1}^{N^t} g(x_i^t)||$. DDC of [39] applies MMD to the last layer while Deep Adaptation Network (DAN) of [21] applies to the last 3 FC layers. CoGAN of [20] shares early layer parameters of the generator and later layer parameters of the discriminators instead of minimizing the MMD. [23] models target classifier predictions as the sum of source classifier predictions and a learned residual function. Central Moment Discrepancy (CMD) of [42] extends MMD by matching higher moment statistics of the source and the target features.

Adversarial domain adaption methods described in the early sections [10] are another way of learning a shared feature space without needing separate classifiers for the source and the target data. DANN [10] proposed a shared encoder and two discriminator branches for domain and class predictions. This makes *marginal* feature distributions similar for the domain classifier. Upcoming works [35, 17] applied the same idea but instead of multiplying the gradient with a negative value, they optimize the discriminator and generator losses in an alternating fashion. [34] suggested replacing the domain discrepancy loss with the Wasserstein distance to tackle gradient vanishing problems. The work of [22] resembles ours where they also condition the domain alignment loss to labels. Unlike us, their domain discriminator takes the outer product of the features and the class predictions as input. Similarly, [6] applies conditional domain alignment using K different class-conditioned binary predictors instead of one predictor with $2K$-way adversarial loss. Our approach allows to not only align the conditional push-forward distributions, but also encourage them to be disjoint. If our sole goal was to align the conditional distributions, a constant encoder function would be a trivial solution. Furthermore, these methods do not exploit SSL regularizers like VAT.

Multiple hypotheses. Another line of work trains multiple encoders and/or classifiers with some consistency loss connecting them. Other than aforementioned methods of [31, 17], [4] proposed domain separation network (DSN). They have two private encoders and a shared encoder for the source and the target samples. The classifier is trained with the summed representations of the shared and the private features. Similarly, [38] trained two encoders for the source and the target data. At test time, they use the encoder learned for the target data and the classifier trained with the source data. [32] had one encoder and two classifiers. Both classifiers are trained on the labeled source samples. The distance between predictions of two classifiers on the same target sample is minimized by the encoder and maximized by classifiers. With the adversarial training of the encoder, they make sure that no two classifiers can have different predictions on the same target sample. Our model also has two predictors but unlike these methods, the purpose of the second predictor (the joint predictor) is to provide conditional alignment for the encoder.

Mapping representations (asymmetric-feature based). These methods apply a transformation from the source domain to the target domain or vice-versa [3]. Adaptive Batch Normalization (Ad-aBN) of [19] proposed to map domain representations with first-order statistics. Before inference time, they pass all target samples through the network to learn the mean and the variance for each activation and apply these learned statistics to normalize the test instances. [36] proposed Correlation Alignment (CORAL). They match the second-order statistics of the source data to target by recoloring whitened source data with target statistics.

Exploiting unlabeled data with SSL regularizers. Given the features of the source and the target domains are aligned, standard SSL methods can be applied. [9] employed the Mean Teacher [37] for UDA where the consistency loss on the target data between student and teacher networks is minimized. Even with extra tricks like confidence-thresholding and some data augmentation, the accuracy they achieved for MNIST→SVHN was 34%. This shows that, especially when domain discrepancy is high, SSL regularizers are not sufficient without first reducing the discrepancy.

Conditional GAN. [25] proposed conditional GAN where generation and discrimination are conditioned onto labels by inputting labels. [29], instead, augmented the discriminator with an auxiliary task of predicting the class labels. The generator also generates samples respecting the correct class label. Our approach differs from these works as we are not generating fake sample in the input space.

6. Conclusion

We proposed a novel method for UDA with the motivation of conditionally aligning the features. We achieved this goal by introducing an additional joint predictor which learns a distribution over class and domain labels. The encoder is trained to fool this predictor within the same-class samples of each domain. We also employed recent tools from SSL to improve the generalization. The proposed idea achieved state-of-the-art accuracy in most challenging image classification tasks for which accuracy are still below 90%. The code will be made available after the review process. Implementation details and proofs are provided in the Supp. Mat.

Acknowledgment

Research supported by ONR - N00014-17-1-2072 and ARO MURI - W911NF-17-1-0304.

References

[1] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916, 2011.

[2] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

[3] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 7, 2017.

[4] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks. In *Advances in Neural Information Processing Systems*, pages 343–351, 2016.

[5] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[6] Yi-Hsin Chen, Wei-Yu Chen, Yu-Ting Chen, Bo-Cheng Tsai, Yu-Chiang Frank Wang, and Min Sun. No more discrimination: Cross city adaptation of road scene segmenters. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1992–2001, 2017.

[7] Safa Cicek, Alhussein Fawzi, and Stefano Soatto. Saas: Speed as a supervisor for semi-supervised learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 149–163, 2018.

[8] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223, 2011.

[9] Geoff French, Michal Mackiewicz, and Mark Fisher. Self-ensembling for visual domain adaptation. 2018.

[10] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495*, 2014.

[11] Muhammad Ghifary, W Bastiaan Kleijn, Mengjie Zhang, David Balduzzi, and Wen Li. Deep reconstruction-classification networks for unsupervised domain adaptation. In *European Conference on Computer Vision*, pages 597–613. Springer, 2016.

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[13] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *Advances in neural information processing systems*, pages 529–536, 2005.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[15] Andreas Krause, Pietro Perona, and Ryan G Gomes. Discriminative clustering by regularized information maximization. In *Advances in neural information processing systems*, pages 775–783, 2010.

[16] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[17] Abhishek Kumar, Prasanna Sattigeri, Kahini Wadhawan, Leonid Karlinsky, Rogerio Feris, Bill Freeman, and Gregory Wornell. Co-regularized alignment for unsupervised domain adaptation. In *Advances in Neural Information Processing Systems*, pages 9366–9377, 2018.

[18] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[19] Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. Revisiting batch normalization for practical domain adaptation. *arXiv preprint arXiv:1603.04779*, 2016.

[20] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In *Advances in neural information processing systems*, pages 469–477, 2016.

[21] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I Jordan. Learning transferable features with deep adaptation networks. *arXiv preprint arXiv:1502.02791*, 2015.

[22] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In *Advances in Neural Information Processing Systems*, pages 1647–1657, 2018.

[23] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Unsupervised domain adaptation with residual transfer networks. In *Advances in Neural Information Processing Systems*, pages 136–144, 2016.

[24] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[25] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[26] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *arXiv preprint arXiv:1704.03976*, 2017.

[27] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. Distributional smoothing with virtual adversarial training. *arXiv preprint arXiv:1507.00677*, 2015.

[28] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 5, 2011.

[29] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2642–2651. JMLR. org, 2017.

[30] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *European Conference on Computer Vision*, pages 102–118. Springer, 2016.

[31] Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. Asymmetric tri-training for unsupervised domain adaptation. *arXiv preprint arXiv:1702.08400*, 2017.

[32] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3723–3732, 2018.

[33] Ozan Sener, Hyun Oh Song, Ashutosh Saxena, and Silvio Savarese. Learning transferrable representations for unsupervised domain adaptation. In *Advances in Neural Information Processing Systems*, pages 2110–2118, 2016.

[34] Jian Shen, Yanru Qu, Weinan Zhang, and Yong Yu. Adversarial representation learning for domain adaptation. *arXiv preprint arXiv:1707.01217*, 2017.

[35] Rui Shu, Hung H Bui, Hirokazu Narui, and Stefano Ermon. A dirt-t approach to unsupervised domain adaptation. *arXiv preprint arXiv:1802.08735*, 2018.

[36] Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation. In *AAAI*, volume 6, page 8, 2016.

[37] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. 2017.

[38] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 4, 2017.

[39] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

[40] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, page 6, 2017.

[41] Mei Wang and Weihong Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 2018.

[42] Werner Zellinger, Thomas Grubinger, Edwin Lughofer, Thomas Natschläger, and Susanne Saminger-Platz. Central moment discrepancy (cmd) for domain-invariant representation learning. *arXiv preprint arXiv:1702.08811*, 2017.