

Unified Deep Supervised Domain Adaptation and Generalization

Saeid Motiian Marco Piccirilli Donald A. Adjeroh Gianfranco Doretto
West Virginia University
Morgantown, WV 26508

{samotiian, mpiccir1, daadjero, gidoretto}@mix.wvu.edu

Abstract

This work provides a unified framework for addressing the problem of visual supervised domain adaptation and generalization with deep models. The main idea is to exploit the Siamese architecture to learn an embedding subspace that is discriminative, and where mapped visual domains are semantically aligned and yet maximally separated. The supervised setting becomes attractive especially when only few target data samples need to be labeled. In this scenario, alignment and separation of semantic probability distributions is difficult because of the lack of data. We found that by reverting to point-wise surrogates of distribution distances and similarities provides an effective solution. In addition, the approach has a high “speed” of adaptation, which requires an extremely low number of labeled target training samples, even one per category can be effective. The approach is extended to domain generalization. For both applications the experiments show very promising results.

1. Introduction

Many computer vision applications require enough labeled data (*target data*) for training visual classifiers to address a specific task at hand. Whenever target data is either not available, or it is expensive to collect and/or label it, the typical approach is to use available datasets (*source data*), representative of a closely related task. Since this practice is known for leading to suboptimal performance, techniques such as *domain adaptation* [6] and/or *domain generalization* [5] have been developed to address the issue. Domain adaptation methods require target data, whereas domain generalization methods do not. Domain adaptation can be either *supervised* [60, 33], *unsupervised* [38, 38, 61], or *semi-supervised* [26, 29, 67]. Unsupervised domain adaptation (UDA) is attractive because it does not require target data to be labeled. Conversely, supervised domain adaptation (SDA) requires labeled target data.

UDA expects large amounts of target data in order to be effective, and this is emphasized even more when using deep models. Moreover, given the same amount of target

data, SDA typically outperforms UDA, as we will later explain. Therefore, especially when target data is *scarce*, it is more attractive to use SDA, also because limited amounts of target data are likely to not be very expensive to label.

In the absence of target data, domain generalization (DG) exploits several cheaply available datasets (sources), representing different specific but closely related tasks. It then attempts to learn by combining data sources in a way that produces visual classifiers that are less sensitive to the specific target data that will need to be processed.

In this work, we introduce a supervised approach for visual recognition that can be used for both SDA and DG. The SDA approach requires very few labeled target samples per category in training. Indeed, even one sample can significantly increase performance, and a few others bring it closer to a peak, showing a remarkable “speed” of adaptation. Moreover, the approach is also robust to adapting to categories that have no target labeled samples. Although domain adaptation and generalization are closely related, adaptation techniques are not directly applied to DG, and viceversa. However, we show that by making simple changes to our proposed training loss function, and by maintaining the same architecture, our SDA approach very effectively extends to DG.

Using basic principles, we analyze how visual classification is extended to handle UDA by aligning a *source domain* distribution to a *target domain* distribution to make the classifier domain invariant. This leads to observing that SDA approaches improve upon UDA by making the alignment *semantic*, because they can ensure the alignment of semantically equivalent distributions from different domains. However, we go one step ahead by suggesting that semantic distribution separation should further increase performance, and this leads to the introduction of a *classification and contrastive semantic alignment* (CCSA) loss.

We deal with the limited size of target domain samples by observing that the CCSA loss relies on computing distances and similarities between distributions (as typically done in adaptation and generalization approaches). Those are difficult to represent with limited data. Thus, we revert

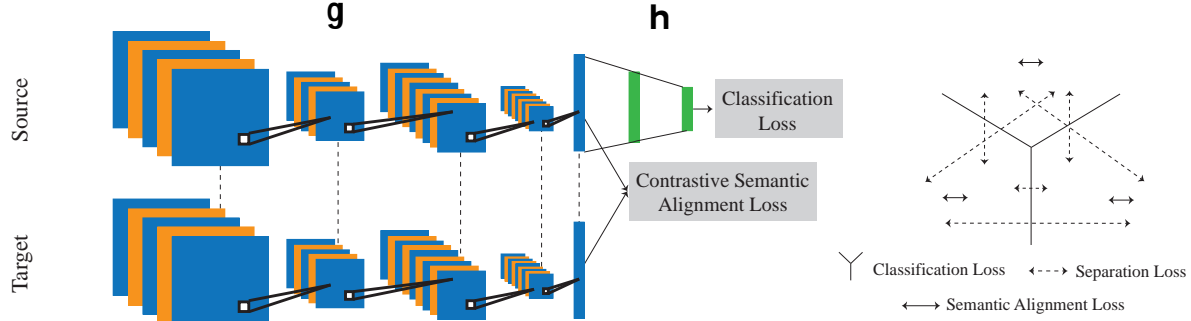


Figure 1. **Deep supervised domain adaptation.** In training, the semantic alignment loss minimizes the distance between samples from different domains but the same class label and the separation loss maximizes the distance between samples from different domains and class labels. At the same time, the classification loss guarantees high classification accuracy.

to point-wise surrogates. The resulting approach turns out to be very effective as shown in the experimental section.

2. Related work

Domain adaptation. Visual recognition algorithms are trained with data from a *source domain*, and when they are tested on a *target domain* with marginal distribution that differs from the one of the sources, we experience the visual domain adaptation (DA) problem (also known as dataset bias [48, 59, 58], or covariate shift [54]), and observe a performance decrease.

Traditional DA methods attempt to directly minimize the shift between source and target distributions. We divide them in three categories. The first one includes those that try to find a mapping between source and target distributions [53, 34, 27, 26, 19, 57]. The second one seeks to find a shared latent space for source and target distributions [40, 2, 44, 21, 22, 47, 43]. The third one regularizes a classifier trained on a source distribution to work well on a target distribution [4, 1, 66, 15, 3, 12]. UDA approaches fall in the first and second categories, while SDA methods could fall either in the second or third category or sometimes both. Recently, [7, 42] have addressed UDA when an auxiliary data view [36, 43], is available during training, which is beyond the scope of this work.

Here, we are interested in finding a shared subspace for source and target distributions. Among algorithms for subspace learning, Siamese networks [11] work well for different tasks [14, 55, 35, 63, 9]. Recently, Siamese networks have been used for domain adaptation. In [60], which is an SDA approach, unlabeled and sparsely labeled target domain data are used to optimize for domain invariance to facilitate domain transfer while using a soft label distribution matching loss. In [56], which is a UDA approach, unlabeled target data are used to learn a nonlinear transformation that aligns correlations of layer activations in deep neural networks. Some approaches went beyond the Siamese weight-

sharing and used couple networks for DA. [33] uses two CNN streams, for source and target, fused at the classifier level. [50] uses a two-streams architecture, for source and target, with related but not shared weights. Here we use a Siamese network to learn an embedding such that samples from the same class are mapped as close as possible to each other. This semantic alignment objective is similar to other deep approaches, but unlike them, we explicitly model and introduce cross-domain class separation forces. Moreover, we do so with very few training samples, which makes the problem of characterizing distributions challenging, and this is why we propose to use point-wise surrogates.

Domain generalization. Domain generalization (DG) is a less investigated problem and is addressed in two ways. In the first one, all information from the training domains or datasets is aggregated to learn a shared invariant representation. Specifically, [5] pulls all of the training data together in one dataset, and learns a single SVM classifier. [44] learns an invariant transformation by minimizing the dissimilarity across domains. [23], which can be used for SDA too, finds a representation that minimizes the mismatch between domains and maximizes the separability of data. [24] learns features that are robust to variations across domains.

The second approach to DG is to exploit all information from the training domains to train a classifier or regulate its weights [32, 17, 65, 45, 46]. Specifically, [32] adjusts the weights of the classifier to work well on an unseen dataset, and [65] fuses the score of exemplar classifiers given any test sample. While most works use the shallow models, here we approach DG as in the first way, and extend the proposed SDA approach by training a deep Siamese network to find a shared invariant representation where semantic alignment as well as separation are explicitly accounted for. To the best of our knowledge, [24] is the only DG approach using deep models, and our method is the first deep method that solves both adaptation and generalization.

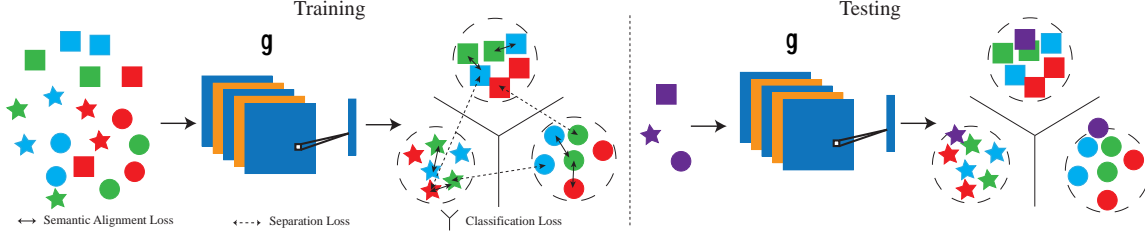


Figure 2. **Deep domain generalization.** In training, the semantic alignment loss minimizes the distance between samples from different domains but the same class label and the separation loss maximizes the distance between samples from different domains and class labels. At the same time, the classification loss guarantees high classification accuracy. In testing, the embedding function embeds samples from unseen distributions to the domain invariant space and the prediction function classifies them (right). In this figure, different colors represent different domain distributions and different shapes represent different classes.

3. Supervised DA with Scarce Target Data

In this section we describe the model we propose to address supervised domain adaptation (SDA), and in the following Section 4 we extend it to address the domain generalization problem. We are given a training dataset made of pairs $D_s = \{(x_i^s, y_i^s)\}_{i=1}^N$. The feature $x_i^s \in X$ is a realization from a random variable X^s , and the label $y_i^s \in Y$ is a realization from a random variable Y . In addition, we are also given the training data $D_t = \{(x_i^t, y_i^t)\}_{i=1}^M$, where $x_i^t \in X$ is a realization from a random variable X^t , and the labels $y_i^t \in Y$. We assume that there is a *covariate shift* [54] between X^s and X^t , i.e., there is a difference between the probability distributions $p(X^s)$ and $p(X^t)$. We say that X^s represents the *source domain* and that X^t represents the *target domain*. Under this settings the goal is to learn a prediction function $f : X \rightarrow Y$ that during testing is going to perform well on data from the target domain.

The problem formulated thus far is typically referred to as *supervised domain adaptation*. In this work we are especially concerned with the version of this problem where only very few target labeled samples per class are available. We aim at handling cases where there is only one target labeled sample, and there can even be some classes with no target samples at all.

3.1. Deep SDA

In the absence of covariate shift a visual classifier f is trained by minimizing a *classification loss*

$$L_C(f) = E[(f(X^s), Y)], \quad (1)$$

where $E[\cdot]$ denotes statistical expectation and \cdot could be any appropriate loss function (for example categorical cross-entropy for multi-class classification). When the distributions of X^s and X^t are different, a deep model f_s trained with D_s will have reduced performance on the target domain. Increasing it would be trivial by simply training a new model f_t with data D_t . However, D_t is small and deep models require large amounts of labeled data.

In general, f could be modeled by the composition of two functions, i.e., $f = h \circ g$. Here $g : X \rightarrow Z$ would be an embedding from the input space X to a feature or embedding space Z , and $h : Z \rightarrow Y$ would be a function for predicting from the feature space. With this notation we would have $f_s = h_s \circ g_s$ and $f_t = h_t \circ g_t$, and the SDA problem would be about finding the best approximation for g_t and h_t , given the constraints on the available data.

The unsupervised DA paradigm (UDA) assumes that D_t does not have labels. In that case the typical approach assumes that $g_t = g_s = g$, and f minimizes (1), while g also minimizes

$$L_{CA}(g) = d(p(g(X^s)), p(g(X^t))) . \quad (2)$$

The purpose of (2) is to align the distributions of the features in the embedding space, mapped from the source and the target domains. d is meant to be a metric between distributions that once aligned, they will no longer allow to tell whether a feature is coming from the source or the target domain. For that reason, we refer to (2) as the *confusion alignment loss*. A popular choice for d is the Maximum Mean Discrepancy [28]. In the embedding space Z , features are assumed to be domain invariant. Therefore, UDA methods say that from the feature to the label space it is safe to assume that $h_t = h_s = h$.

Since we are interested in visual recognition, the embedding function g would be modeled by a convolutional neural network (CNN) with some initial convolutional layers, followed by some fully connected layers. In addition, the training architecture would have two streams, one for source and the other for target samples. Since $g_s = g_t = g$, the CNN parameters would be shared as in a Siamese architecture. In addition, the source stream would continue with additional fully connected layers for modeling h . See Figure 1.

From the above discussion it is clear that in order to perform well, UDA needs to align effectively. This can happen only if distributions are represented by a sufficiently large dataset. Therefore, UDA approaches are in a position of weakness because we assume D_t to be small. More-

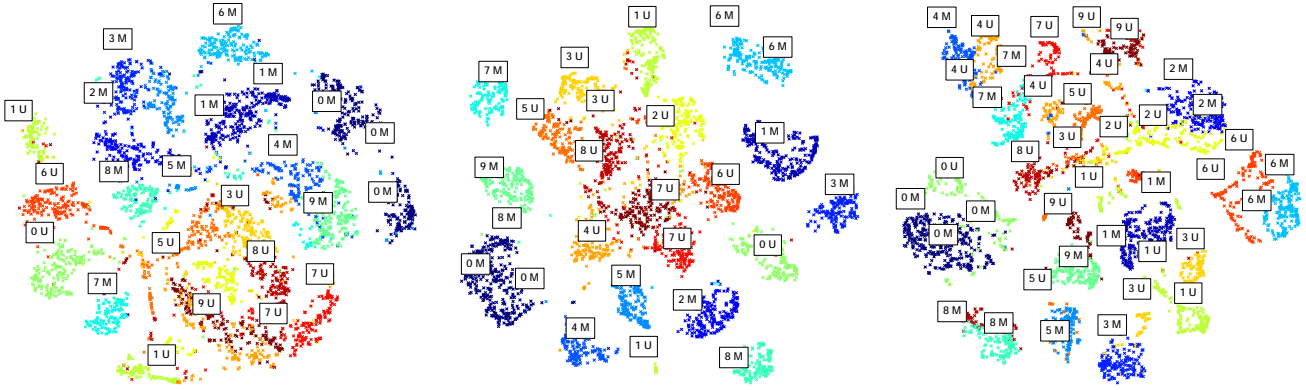


Figure 3. **Visualization of the MNIST-USPS datasets.** **Left:** 2D visualization of the row images of the MNIST-USPS datasets. The samples from the same class and different domains lie far from each other on the 2D subspace. **Middle:** 2D visualization of the embedded images using our base model (without domain adaptation). The samples from the same class and different domains still lie far from each other on the 2D subspace. **Right:** 2D visualization of the embedded images using our SDA model. The samples from the same class and different domains lie very close to each other on the 2D subspace.

over, UDA approaches have also another intrinsic limitation, which is that even with perfect confusion alignment, there is no guarantee that samples from different domains but the same class label, would map nearby in the embedding space. This lack of *semantic alignment* is a major source of performance reduction.

SDA approaches easily address the semantic alignment problem by replacing (2) with

$$L_{SA}(g) = \sum_{a=1}^C d(p(g(X_a^s)), p(g(X_a^t))) , \quad (3)$$

where C is the number of class labels, and $X_a^s = X^s|Y = a$ and $X_a^t = X^t|Y = a$ are conditional random variables. d instead is a suitable distance measure between the distributions of X_a^s and X_a^t in the embedding space. We refer to (3) as the *semantic alignment* loss, which clearly encourages samples from different domains but the same label, to map nearby in the embedding space.

While the analysis above clearly indicates why SDA provides superior performance than UDA, it also suggests that deep SDA approaches have not considered that greater performance could be achieved by encouraging class separation, meaning that samples from different domains and with different labels, should be mapped as far apart as possible in the embedding space. This idea means that, in principle, a semantic alignment less prone to errors should be achieved by adding to (3) the following term

$$L_S(g) = \sum_{a,b|a \neq b} k(p(g(X_a^s)), p(g(X_b^t))) , \quad (4)$$

where k is a suitable similarity measure between the distributions of X_a^s and X_b^t in the embedding space, which adds

a penalty when the distributions $p(g(X_a^s))$ and $p(g(X_b^t))$ come close, since they would lead to lower classification accuracy. We refer to (4) as the *separation* loss.

Finally, we suggest that SDA could be approached by learning a deep model $f = h \circ g$ such that

$$L_{CCSA}(f) = L_C(h \circ g) + L_{SA}(g) + L_S(g) . \quad (5)$$

We refer to (5) as the *classification and contrastive semantic alignment* loss. This would allow to set $g_s = g_t = g$. The classification network h is trained only with source data, so $h_s = h$. In addition, to improve performance on the target domain, h_t could be obtained via fine-tuning based on the few samples in D_t , i.e.,

$$h_t = \text{fine-tuning}(h|D_t) . \quad (6)$$

Note that the network architecture remains the one in Figure 1, only with a different loss, and training procedure.

3.2. Handling Scarce Target Data

When the size of the labeled target training dataset D_t is very small, minimizing the loss (5) becomes a challenge. The problem is that the semantic alignment loss as well as the separation loss rely on computing distances and similarities between distributions, and those are very difficult to represent with as few as one data sample.

Rather than attempting to characterize distributions with statistics that require enough data, because of the reduced size of D_t , we compute the distance in the semantic alignment loss (3) by computing average pairwise distances between points in the embedding space, i.e., we compute

$$d(p(g(X_a^s)), p(g(X_a^t))) = \sum_{i,j} d(g(x_i^s), g(x_j^t)) , \quad (7)$$

Figure 4. (a), (b), (c): Improvement of CCSA over the base model. (d): Average classification accuracy for the $M \rightarrow U$ task for different number of labeled target samples per category (n). It shows that our model provides significant improvement over baselines.

where it is assumed $y_i^s = y_j^t = a$. The strength of this approach is that it allows even a single labeled target sample to be paired with all the source samples, effectively trying to semantically align the entire source data with the few target data. Similarly, we compute the similarities in the separation loss (4) by computing average pairwise similarities between points in the embedding space, i.e., we compute

$$k(p(g(X_a^s)), p(g(X_b^t))) = \frac{1}{i,j} k(g(x_i^s), g(x_j^t)), \quad (8)$$

where it is assumed that $y_i^s = a = y_j^t = b$.

Moreover, our implementation further assumes that

$$d(g(x_i^s), g(x_j^t)) = \frac{1}{2} \|g(x_i^s) - g(x_j^t)\|^2, \quad (9)$$

$$k(g(x_i^s), g(x_j^t)) = \frac{1}{2} \max(0, m - \|g(x_i^s) - g(x_j^t)\|)^2 \quad (10)$$

where $\|\cdot\|$ denotes the Frobenius norm, and m is the margin that specifies the separability in the embedding space. Note that with the choices outlined in (9) and (10), the loss $L_{SA}(g) + L_S(g)$ becomes the well known contrastive loss as defined in [30]. Finally, to balance the classification versus the contrastive semantic alignment portion of the loss (5), (7) and (8) are normalized and weighted by $1 - \alpha$ and (1) by α .

4. Extension to Domain Generalization

In visual domain generalization (DG), D labeled datasets D_{s_1}, \dots, D_{s_D} , representative of D distinct source domains are given. The goal is to learn from them a visual classifier f that during testing is going to perform well on data D_t , not

available during training, thus representative of an unknown target domain.

The SDA method in Section 3 treats source and target datasets D_s and D_t almost symmetrically. In particular, the embedding g aims at achieving semantic alignment, while favoring class separation. The only asymmetry is in the prediction function h that is trained only on the source, to be then fine-tuned on the target.

In domain generalization, we are not interested in adapting the classifier to the target domain, because it is unknown. Instead, we want to make sure that the embedding g maps to a domain invariant space. To do so we consider every distinct unordered pair of source domains (u, v) , represented by D_{s_u} and D_{s_v} , and, like in SDA, impose the semantic alignment loss (3) as well as the separation loss (4). Moreover, the losses are summed over every pair in order to make the map g as domain invariant as possible. Similarly, the classifier h should be as accurate as possible for any of the mapped samples, to maximize performance on an unseen target. This calls for having a fully symmetric learning for h by training it on all the source domains, meaning that the classification loss (1) is summed over every domain s_u . See Figure 2.

The network architecture is still the one in Figure 1, and we have implemented it with the same choices for distances and similarities as those made in Section 3.2. However, since we are summing the losses (3) and (4) over every unordered pair of source domains, there is a quadratic growth of paired training samples. So, if necessary, rather than processing every paired sample, we select them randomly.

5. Experiments

We divide the experiments into two parts, domain adaptation and domain generalization. In both sections, we use benchmark datasets and compare our domain adaptation model and our domain generalization model, both indicated as CCSA, with the state-of-the-art.

5.1. Domain Adaptation

We present results using the Office dataset [53], the MNIST dataset [37], and the USPS dataset [31].

5.1.1 Office Dataset

The office dataset is a standard benchmark dataset for visual domain adaptation. It contains 31 object classes for three domains: Amazon, Webcam, and DSLR, indicated as A , W , and D , for a total of 4,652 images. We consider six domain shifts using the three domains ($A \rightarrow W$, $A \rightarrow D$, $W \rightarrow A$, $W \rightarrow D$, $D \rightarrow A$, and $D \rightarrow W$). We performed several experiments using this dataset.

First experiment. We followed the setting described in [60]. All classes of the office dataset and 5 train-test splits

Table 1. **Office dataset.** Classification accuracy for domain adaptation over the 31 categories of the Office dataset. A, W, and D stand for Amazon, Webcam, and DSLR domain. Lower Bound is our base model without adaptation.

	Lower Bound	Unsupervised			Supervised		
		[62]	[39]	[25]	[60]	[33]	CCSA
A W	61.2 ± 0.9	61.8 ± 0.4	68.5 ± 0.4	68.7 ± 0.3	82.7 ± 0.8	84.5 ± 1.7	88.2 ± 1.0
A D	62.3 ± 0.8	64.4 ± 0.3	67.0 ± 0.4	67.1 ± 0.3	86.1 ± 1.2	86.3 ± 0.8	89.0 ± 1.2
W A	51.6 ± 0.9	52.2 ± 0.4	53.1 ± 0.3	54.09 ± 0.5	65.0 ± 0.5	65.7 ± 1.7	72.1 ± 1.0
W D	95.6 ± 0.7	98.5 ± 0.4	99.0 ± 0.2	99.0 ± 0.2	97.6 ± 0.2	97.5 ± 0.7	97.6 ± 0.4
D A	58.5 ± 0.8	52.1 ± 0.8	54.0 ± 0.4	56.0 ± 0.5	66.2 ± 0.3	66.5 ± 1.0	71.8 ± 0.5
D W	80.1 ± 0.6	95.0 ± 0.5	96.0 ± 0.3	96.4 ± 0.3	95.7 ± 0.5	95.5 ± 0.6	96.4 ± 0.8
Average	68.2	70.6	72.9	73.6	82.21	82.68	85.8

Table 2. **Office dataset.** Classification accuracy for domain adaptation over the Office dataset when only the labeled target samples of 15 classes are available during training. Testing is done on all 31 classes. A, W, and D stand for Amazon, Webcam, and DSLR domain. Lower Bound is our base model without adaptation.

	Lower Bound	[60]	CCSA
A W	52.1 ± 0.6	59.3 ± 0.6	63.3 ± 0.9
A D	61.6 ± 0.8	68.0 ± 0.5	70.5 ± 0.6
W A	34.5 ± 0.9	40.5 ± 0.2	43.6 ± 1.0
W D	95.1 ± 0.2	97.5 ± 0.1	96.2 ± 0.3
D A	40.1 ± 0.3	43.1 ± 0.2	42.6 ± 0.6
D W	89.7 ± 0.8	90.0 ± 0.2	90.0 ± 0.2
Average	62.26	66.4	67.83

are considered. For the source domain, 20 examples per category for the Amazon domain, and 8 examples per category for the DSLR and Webcam domains are randomly selected for training for each split. Also, 3 labeled examples are randomly selected for each category in the target domain for training for each split. The rest of the target samples are used for testing. Note that we used the same splits generated by [60]. We also report the classification results of the SDA algorithm presented in [39] and [33]. In addition to the SDA algorithms, we report the results of some recent UDA algorithms. They follow a different experimental protocol compared to the SDA algorithms, and use all samples of the target domain in training as unlabeled data together with all samples of the source domain.

For the embedding function g , we used the convolutional layers of the VGG-16 architecture [55] followed by 2 fully connected layers with output size of 1024 and 128, respectively. For the prediction function h , we used a fully connected layer with softmax activation. Similar to [60], we used the weights pre-trained on the ImageNet dataset [51] for the convolutional layers, and initialized the fully connected layers using all the source domain data. We then fine-tuned all the weights using the train-test splits.

Table 1 reports the classification accuracy over 31 classes for the Office dataset and shows that CCSA has better performance compared to [60]. Since the difference between W domain and D domain is not considerable, unsupervised algorithms work well on D → W and W → D. However, in the cases when target and source domains are very different (A → W, W → A, A → D, and D → A), CCSA shows larger margins compared to the second best. This

suggests that CCSA will provide greater alignment gains when there are bigger domain shifts. Figure 4(a) instead, shows how much improvement can be obtained with respect to the base model. This is simply obtained by training g and h with only the classification loss and source training data, so no adaptation is performed.

Second experiment. We followed the setting described in [60] when only 10 target labeled samples of 15 classes of the Office dataset are available during training. Similar to [60], we compute the accuracy on the remaining 16 categories for which no target data was available during training. We used the same network structure as in the first experiment and the same splits generated by [60].

Table 2 shows that CCSA is effective at transferring information from the labeled classes to the unlabeled target classes. Similar to the first experiment, CCSA works well when shifts between domains are larger.

Third experiment. We used the original train-test splits of the Office dataset [53]. The splits are generated in a similar manner to the first experiment but here instead, only 10 classes are considered (backpack, bike, calculator, headphones, keyboard, laptop-computer, monitor, mouse, mug, and projector). In order to compare our results with the state-of-the-art, we used DeCaF-fc6 features [14] and 800-dimension SURF features as input. For DeCaF-fc6 features (SURF features) we used 2 fully connected layers with output size of 1024 (512) and 128 (32) with ReLU activation as the embedding function, and one fully connected layer with softmax activation as the prediction function. The features and splits are available on the Office dataset webpage¹.

We compared our results with three UDA (GFK [26], mSDA [8], and RTML [13]) and one SDA (CDML [64]) algorithms under the same settings. Table 3 shows that CCSA provides an improved accuracy with respect to the others. Again, greater domain shifts are better compensated by CCSA. Figure 4(b) shows the improvement of CCSA over the base model using DeCaF-fc6 features.

5.1.2 MNIST-USPS Datasets

The MNIST (M) and USPS (U) datasets have recently been used for domain adaptation [20, 50]. They contain images

¹<https://cs.stanford.edu/~jhoffman/domainadapt/>

Table 3. **Office dataset.** Classification accuracy for domain adaptation over the 10 categories of the Office dataset. A, W, and D stand for Amazon, Webcam, and DSLR domain. Lower Bound is our base model with no adaptation.

		Lower Bound	GFK [26]	mSDA [8]	CDML [64]	RTML [13]	CCSA
SURF							
A	W	26.5 ± 3.1	39.9 ± 0.9	35.5 ± 0.5	37.3 ± 0.7	43.4 ± 0.9	71.2 ± 1.3
A	D	17.5 ± 1.2	36.2 ± 0.7	29.7 ± 0.7	35.3 ± 0.5	43.3 ± 0.6	74.2 ± 1.3
W	A	25.9 ± 1.0	29.8 ± 0.6	32.1 ± 0.8	32.4 ± 0.5	37.5 ± 0.7	42.9 ± 0.9
W	D	46.9 ± 1.1	80.9 ± 0.4	56.6 ± 0.4	77.9 ± 0.9	91.7 ± 1.1	85.1 ± 1.0
D	A	19.3 ± 1.9	33.2 ± 0.6	33.6 ± 0.8	29.4 ± 0.8	36.3 ± 0.3	28.9 ± 1.3
D	W	48.0 ± 2.1	79.4 ± 0.6	68.6 ± 0.7	79.4 ± 0.6	90.5 ± 0.7	77.3 ± 1.6
Average		30.6	43.5	38.4	43.5	49.8	63.2
DeCaF-fc6							
A	W	78.9 ± 1.8	73.1 ± 2.8	64.6 ± 4.2	75.9 ± 2.1	79.5 ± 2.6	94.5 ± 1.9
A	D	79.2 ± 2.1	82.6 ± 2.1	72.6 ± 3.5	81.4 ± 2.6	83.8 ± 1.7	97.2 ± 1.0
W	A	77.3 ± 1.1	82.6 ± 1.3	71.4 ± 1.7	86.3 ± 1.6	90.8 ± 1.6	91.2 ± 0.8
W	D	96.6 ± 1.0	98.8 ± 0.9	99.5 ± 0.6	99.4 ± 0.4	100 ± 0.0	99.6 ± 0.5
D	A	84.0 ± 1.3	85.4 ± 0.7	78.8 ± 0.5	88.4 ± 0.5	90.6 ± 0.5	91.7 ± 1.0
D	W	96.7 ± 0.9	91.3 ± 0.4	97.5 ± 0.4	95.1 ± 0.5	98.6 ± 0.3	98.7 ± 0.6
Average		85.4	85.63	80.73	87.75	90.55	95.4

Table 4. **MNIST-USPS datasets.** Classification accuracy for domain adaptation over the MNIST and USPS datasets. M and U stand for MNIST and USPS domain. Lower Bound is our base model without adaptation. CCSA - n stands for our method when we use n labeled target samples per category in training.

Method	M	U	Average
ADDA [61]	89.4	90.1	89.7
CoGAN [38]	91.2	89.1	90.1
Lower Bound	65.4	58.6	62.0
CCSA-1	85.0	78.4	81.7
CCSA-2	89.0	82.0	85.5
CCSA-3	90.1	85.8	87.9
CCSA-4	91.4	86.1	88.7
CCSA-5	92.4	88.8	90.1
CCSA-6	93.0	89.6	91.3
CCSA-7	92.9	89.4	91.1
CCSA-8	92.8	90.0	91.4

Table 5. **VLCS dataset.** Classification accuracy for domain generalization over the 5 categories of the VLCS dataset. LB (Lower Bound) is our base model trained without the contrastive semantic alignment loss. 1NN stands for first nearest neighbor.

		Lower Bound			Domain Generalization			
		1NN	SVM	LB	UML [17]	LRE-SVM [65]	SCA [23]	CCSA
L, C, S	V	57.2	58.4	59.1	56.2	60.5	64.3	67.1
V, C, S	L	52.4	55.2	55.6	58.5	59.7	59.6	62.1
V, L, S	C	90.5	85.1	86.1	91.1	88.1	88.9	92.3
V, L, C	S	56.9	55.2	54.6	58.4	54.8	59.2	59.1
C, S, V	L	55.0	55.5	55.3	56.4	55.0	59.5	59.3
C, L, V	S	52.6	51.8	50.9	57.4	52.8	55.9	56.5
V, C, L	S	56.6	59.9	60.1	55.4	58.8	60.7	60.2
Average		60.1	60.1	60.2	61.5	61.4	64.0	65.0

of digits from 0 to 9. We considered two cross-domain tasks, M → U and U → M, and followed the experimental setting in [20, 50], which involves randomly selecting 2000 images from MNIST and 1800 images from USPS. Here, we randomly selected n labeled samples per class from target domain data and used them in training. We evaluated our approach for n ranging from 1 to 8 and repeated each experiment 10 times (we only show the mean of the accuracies because the standard deviation is very small).

Similar to [37], we used 2 convolutional layers with 6 and 16 filters of 5 × 5 kernels followed by max-pooling layers and 2 fully connected layers with size 120 and 84

as the embedding function g, and one fully connected layer with softmax activation as the prediction function h. We compare our method with 2 recent UDA methods. Those methods use all target samples in their training stage, while we only use very few labeled target samples per category in training.

Table 4 shows the average classification accuracy of the MNIST-USPS datasets. CCSA works well even when only one target sample per category (n = 1) is available in training. Also, we can see that by increasing n, the accuracy quickly converges to the top.

Ablation study. We consider three baselines to compare with CCSA for the M → U task. First, we train the network with source data and then fine-tune it with available target data. Second, we train the network using the classification and semantic alignment losses ($L_{CSA}(f) = L_C(h \circ g) + L_{SA}(g)$). Third, we train the network using the classification and separation losses ($L_{CS}(f) = L_C(h \circ g) + L_S(g)$). Figure 4(d) shows the average accuracies over 10 repetitions. It shows that CSA and CS improve the accuracy over fine-tuning, while using the proposed CCSA loss shows the best performance.

Visualization. We show how samples lie on the embedding space using CCSA. First, we considered the row images of the MNIST and USPS datasets and plotted 2D visualization of them using t-SNE [41]. As Figure 3(Left) shows the row images of the same class and different domains lie far away from each other in the 2D subspace. For example, the samples of the class zero of the USPS dataset (0 U) are far from the class zero of the MNIST dataset (0 M). Second, we trained our base model with no adaptation on the MNIST dataset. We then plotted the 2D visualization of the MNIST and USPS samples in the embedding space (output of g, the last fully connected layer). As Figure 3(Middle) shows, the samples from the same class and different domains still lie far away from each other in the 2D subspace. Finally, we trained our SDA model on the MNIST dataset and 3 la-

beled samples per class of the USPS dataset. We then plotted the 2D visualization of the MNIST and USPS samples in the embedding space (output of g). As Figure 3(Right) shows, the samples from the same class and different domains now lie very close to each other in the 2D subspace. Note however, that this is only a 2D visualization of high-dimensional data, and Figure 3(Right) may not perfectly reflect how close is the data from the same class, and how classes are separated.

Weight sharing: There is no restriction on whether or not g_t and g_s should share weights. Not sharing weights likely leads to overfitting, given the reduced amount of target training data, and weight-sharing acts as a regularizer. For instance, we repeated the experiment for the $M \rightarrow U$ task with $n = 4$. Not sharing weights provides an average accuracy of 88.6 over 10 repetitions, which is less than the average accuracy with weight-sharing (see Table 4). A similar behavior is observable in other experiments.

5.2. Domain Generalization

We evaluate CCSA on different datasets. The goal is to show that CCSA is able to learn a domain invariant embedding subspace for visual recognition tasks.

5.3. VLCS Dataset

In this section, we use images of 5 shared object categories (bird, car, chair, dog, and person), of the PASCAL VOC2007 (V) [16], LabelMe (L) [52], Caltech-101 (C) [18], and SUN09 (S) [10] datasets, which is known as VLCS dataset [17].

[24, 23] have shown that there are covariate shifts between the above 4 domains and have developed a DG method to minimize them. We followed their experimental setting, and randomly divided each domain into a training set (70%) and a test set (30%) and conducted a leave-one-domain-out evaluation (4 cross-domain cases) and a leave-two-domain-out evaluation (3 cross-domain cases). In order to compare our results with the state-of-the-art, we used DeCaF-fc6 features which are publicly available ², and repeated each cross-domain case 20 times and reported the average classification accuracy.

We used 2 fully connected layers with output size of 1024 and 128 with ReLU activation as the embedding function g , and one fully connected layer with softmax activation as the prediction function h . To create positive and negative pairs for training our network, for each sample of a source domain we randomly selected 5 samples from each remaining source domain, and help in this way to avoid overfitting. However, to train a deeper network together with convolutional layers, it is enough to create a large amount of positive and negative pairs.

²http://www.cs.dartmouth.edu/~chenfang/proj_page/FXR_i_ccv13/index.php

Table 6. **MNIST dataset.** Classification accuracy for domain generalization over the MNIST dataset and its rotated domains.

	CAE [49]	MTAE [24]	CCSA
$M_{15}, M_{30}, M_{45}, M_{60}, M_{75}$ M	72.1	82.5	84.6
$M, M_{30}, M_{45}, M_{60}, M_{75}$ M_{15}	95.3	96.3	95.6
$M, M_{15}, M_{45}, M_{60}, M_{75}$ M_{30}	92.6	93.4	94.6
$M, M_{15}, M_{30}, M_{60}, M_{75}$ M_{45}	81.5	78.6	82.9
$M, M_{15}, M_{30}, M_{45}, M_{75}$ M_{60}	92.7	94.2	94.8
$M, M_{15}, M_{30}, M_{45}, M_{60}$ M_{75}	79.3	80.5	82.1
Average	85.5	87.5	89.1

We report comparative results in Table 5, where all DG methods work better than the base model, emphasizing the need for domain generalization. Our DG method has higher average performance. Also, note that in order to compare with the state-of-the-art DG methods, we only used 2 fully connected layers for our network and precomputed features as input. However, when using convolutional layers on row images, we expect our DG model to provide better performance. Figure 4(c) shows the improvement of our DG model over the base model using DeCaF-fc6 features.

5.4. MNIST Dataset

We followed the setting in [24], and randomly selected a set M of 100 images per category from the MNIST dataset (1000 in total). We then rotated each image in M five times with 15 degrees intervals, creating five new domains $M_{15}, M_{30}, M_{45}, M_{60}$, and M_{75} . We conducted a leave-one-domain-out evaluation (6 cross-domain cases in total). We used the same network of Section 5.1.2, and we repeated the experiments 10 times. To create positive and negative pairs for training our network, for each sample of a source domain we randomly selected 2 samples from each remaining source domain. We report comparative average accuracies for CCSA and others in Table 6, showing again a performance improvement.

6. Conclusions

We have introduced a deep model in combination with the classification and contrastive semantic alignment (CCSA) loss to address SDA. We have shown that the CCSA loss can be augmented to address the DG problem without the need to change the basic model architecture. However, the approach is general in the sense that the architecture sub-components can be changed. We found that addressing the semantic distribution alignments with point-wise surrogates of distribution distances and similarities for SDA and DG works very effectively, even when labeled target samples are very few. In addition, we found the SDA accuracy to converge very quickly as more labeled target samples per category are available.

Acknowledgments

This material is based upon work supported, in part, by the Center for Identification Technology Research and the National Science Foundation under Grant No. 1066197.

References

- [1] Y. Aytar and A. Zisserman. Tabula rasa: Model transfer for object category detection. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2252–2259. IEEE, 2011.
- [2] M. Baktashmotlagh, M. T. Harandi, B. C. Lovell, and M. Salzmann. Unsupervised domain adaptation by domain invariant projection. In *IEEE ICCV*, pages 769–776, 2013.
- [3] C. J. Becker, C. M. Christoudias, and P. Fua. Non-linear domain adaptation with boosting. In *Advances in Neural Information Processing Systems*, pages 485–493, 2013.
- [4] A. Bergamo and L. Torresani. Exploiting weakly-labeled web images to improve object classification: a domain adaptation approach. In *Advances in Neural Information Processing Systems*, pages 181–189, 2010.
- [5] G. Blanchard, G. Lee, and C. Scott. Generalizing from several related classification tasks to a new unlabeled sample. In *Advances in neural information processing systems*, pages 2178–2186, 2011.
- [6] J. Blitzer, R. McDonald, and F. Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128. Association for Computational Linguistics, 2006.
- [7] L. Chen, W. Li, and D. Xu. Recognizing RGB images by learning from RGB-D data. In *CVPR*, pages 1418–1425, June 2014.
- [8] M. Chen, Z. Xu, K. Weinberger, and F. Sha. Marginalized denoising autoencoders for domain adaptation. *arXiv preprint arXiv:1206.4683*, 2012.
- [9] Q. Chen, J. Huang, R. Feris, L. M. Brown, J. Dong, and S. Yan. Deep domain adaptation for describing people based on fine-grained clothing attributes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5315–5324, 2015.
- [10] M. J. Choi, J. J. Lim, A. Torralba, and A. S. Willsky. Exploiting hierarchical context on a large database of object categories. In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*, pages 129–136. IEEE, 2010.
- [11] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 539–546. IEEE, 2005.
- [12] H. Daume III and D. Marcu. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26:101–126, 2006.
- [13] Z. Ding and Y. Fu. Robust transfer metric learning for image classification. *IEEE Transactions on Image Processing*, 26(2):660–670, 2017.
- [14] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. DeCAF: a deep convolutional activation feature for generic visual recognition. In *arXiv:1310.1531*, 2013.
- [15] L. Duan, I. W. Tsang, D. Xu, and S. J. Maybank. Domain transfer svm for video concept detection. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1375–1381. IEEE, 2009.
- [16] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.
- [17] C. Fang, Y. Xu, and D. N. Rockmore. Unbiased metric learning: On the utilization of multiple datasets and web images for softening bias. In *International Conference on Computer Vision*, 2013.
- [18] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer vision and Image understanding*, 106(1):59–70, 2007.
- [19] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *IEEE ICCV*, pages 2960–2967, 2013.
- [20] B. Fernando, T. Tommasi, and T. Tuytelaars. Joint cross-domain classification and subspace learning for unsupervised adaptation. *Pattern Recognition Letters*, 2015.
- [21] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495*, 2014.
- [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*, 17(59):1–35, 2016.
- [23] M. Ghifary, D. Balduzzi, W. B. Kleijn, and M. Zhang. Scatter component analysis: A unified framework for domain adaptation and domain generalization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.
- [24] M. Ghifary, W. Bastiaan Kleijn, M. Zhang, and D. Balduzzi. Domain generalization for object recognition with multi-task autoencoders. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2551–2559, 2015.
- [25] M. Ghifary, W. B. Kleijn, M. Zhang, D. Balduzzi, and W. Li. Deep reconstruction-classification networks for unsupervised domain adaptation. In *European Conference on Computer Vision*, pages 597–613. Springer, 2016.
- [26] B. Gong, Y. Shi, F. Sha, and K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2066–2073. IEEE, 2012.
- [27] R. Gopalan, R. Li, and R. Chellappa. Domain adaptation for object recognition: An unsupervised approach. In *IEEE ICCV*, pages 999–1006, 2011.
- [28] A. Gretton, K. M. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola. A kernel method for the two-sample-problem. In *NIPS*, 2006.
- [29] Y. Guo and M. Xiao. Cross language text classification via subspace co-regularized multi-view learning. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*, 2012.
- [30] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 2, pages 1735–1742. IEEE, 2006.

- [31] J. J. Hull. A database for handwritten text recognition research. *IEEE Transactions on pattern analysis and machine intelligence*, 16(5):550–554, 1994.
- [32] A. Khosla, T. Zhou, T. Malisiewicz, A. A. Efros, and A. Torralba. Undoing the damage of dataset bias. In *European Conference on Computer Vision*, pages 158–171. Springer, 2012.
- [33] P. Koniusz, Y. Tas, and F. Porikli. Domain adaptation by mixture of alignments of second- or higher-order scatter tensors. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
- [34] B. Kulis, K. Saenko, and T. Darrell. What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1785–1792. IEEE, 2011.
- [35] B. Kumar, G. Carneiro, I. Reid, et al. Learning local image descriptors with deep siamese and triplet convolutional networks by minimising global loss functions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5385–5394, 2016.
- [36] M. Lapin, M. Hein, and B. Schiele. Learning using privileged information: SVM+ and weighted SVM. *Neural Networks*, 53:95–108, 2014.
- [37] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [38] M.-Y. Liu and O. Tuzel. Coupled generative adversarial networks. In *Advances in Neural Information Processing Systems*, pages 469–477, 2016.
- [39] M. Long, Y. Cao, J. Wang, and M. I. Jordan. Learning transferable features with deep adaptation networks. In *ICML*, pages 97–105, 2015.
- [40] M. Long, G. Ding, J. Wang, J. Sun, Y. Guo, and P. S. Yu. Transfer sparse coding for robust image representation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 407–414, 2013.
- [41] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.
- [42] S. Motiian and G. Doretto. Information bottleneck domain adaptation with privileged information for visual recognition. In *European Conference on Computer Vision*, pages 630–647. Springer, 2016.
- [43] S. Motiian, M. Piccirilli, D. A. Adjeroh, and G. Doretto. Information bottleneck learning using privileged information for visual recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1496–1505, 2016.
- [44] K. Muandet, D. Balduzzi, and B. Schölkopf. Domain generalization via invariant feature representation. In *ICML (1)*, pages 10–18, 2013.
- [45] L. Niu, W. Li, and D. Xu. Multi-view domain generalization for visual recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4193–4201, 2015.
- [46] L. Niu, W. Li, D. Xu, and J. Cai. An exemplar-based multi-view domain generalization framework for visual recognition. *IEEE Transactions on Neural Networks and Learning Systems*, 2017.
- [47] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang. Domain adaptation via transfer component analysis. *IEEE TNN*, 22(2):199–210, 2011.
- [48] J. Ponce, T. L. Berg, M. Everingham, D. A. Forsyth, M. Hebert, S. Lazebnik, M. Marszalek, C. Schmid, B. C. Russell, A. Torralba, et al. Dataset issues in object recognition. In *Toward category-level object recognition*, pages 29–48. Springer, 2006.
- [49] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 833–840, 2011.
- [50] A. Rozantsev, M. Salzmann, and P. Fua. Beyond sharing weights for deep domain adaptation. *arXiv preprint arXiv:1603.06432*, 2016.
- [51] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015.
- [52] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. Labelme: a database and web-based tool for image annotation. *International journal of computer vision*, 77(1-3):157–173, 2008.
- [53] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *ECCV*, pages 213–226, 2010.
- [54] H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, 2000.
- [55] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.
- [56] B. Sun and K. Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *Computer Vision–ECCV 2016 Workshops*, pages 443–450. Springer, 2016.
- [57] T. Tommasi, M. Lanzi, P. Russo, and B. Caputo. Learning the roots of visual domain shift. In *Computer Vision–ECCV 2016 Workshops*, pages 475–482. Springer, 2016.
- [58] T. Tommasi, N. Patricia, B. Caputo, and T. Tuytelaars. A deeper look at dataset bias. In *German Conference on Pattern Recognition*, pages 504–516. Springer, 2015.
- [59] A. Torralba and A. A. Efros. Unbiased look at dataset bias. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1521–1528, 2011.
- [60] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko. Simultaneous deep transfer across domains and tasks. In *ICCV*, 2015.
- [61] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
- [62] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

- [63] R. R. Varior, B. Shuai, J. Lu, D. Xu, and G. Wang. A siamese long short-term memory architecture for human re-identification. In *European Conference on Computer Vision*, pages 135–153. Springer, 2016.
- [64] H. Wang, W. Wang, C. Zhang, and F. Xu. Cross-domain metric learning based on information theory. In *AAAI*, pages 2099–2105, 2014.
- [65] Z. Xu, W. Li, L. Niu, and D. Xu. Exploiting low-rank structure from latent domains for domain generalization. In *ECCV*, pages 628–643, 2014.
- [66] J. Yang, R. Yan, and A. G. Hauptmann. Adapting svm classifiers to data with shifted distributions. In *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*, pages 69–76. IEEE, 2007.
- [67] T. Yao, Y. Pan, C.-W. Ngo, H. Li, and T. Mei. Semi-supervised domain adaptation with subspace learning for visual recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.