Tackling unsupervised multi-source domain adaptation with optimism and consistency

Diogo Pernes and Jaime S. Cardoso
INESC TEC and University of Porto
Porto, Portugal
diogo.pernes@inesctec.pt, jaime.cardoso@inesctec.pt

Abstract

It has been known for a while that the problem of multi-source domain adaptation can be regarded as a single source domain adaptation task where the source domain corresponds to a mixture of the original source domains. Nonetheless, how to adjust the mixture distribution weights remains an open question. Moreover, most existing work on this topic focuses only on minimizing the error on the source domains and achieving domain-invariant representations, which is insufficient to ensure low error on the target domain. In this work, we present a novel framework that addresses both problems and beats the current state of the art by using a mildly optimistic objective function and consistency regularization on the target samples.

1 Introduction

Supervised training of deep neural networks has achieved outstanding results on multiple learning tasks greatly due to the availability of large and rich annotated datasets. Unfortunately, annotating such large-scale datasets is often prohibitively time-consuming and expensive. Furthermore, in many practical cases, it is not possible to collect annotated data with the same characteristics as the test data, and, as a result, training and test data are drawn from distinct underlying distributions (or *domains*). As a consequence, the model performance tends to decrease significantly on the test data. The goal of *domain adaptation* (DA) algorithms is to minimize this gap by finding transferable knowledge from the source to the target domain. Sometimes, it is assumed that a small portion of labeled target data are available at training time – a setting that is known as *semi-supervised* DA [12, 13, 26, 37, 46]. In this work, we focus on the more challenging scenario, where no labeled target data are available for training – known as *unsupervised* DA [2, 16, 24, 31, 51]. The DA problem, in its semi-supervised and unsupervised variants, has received increased attention in recent years, both from theoretical [4, 5, 6, 10, 18, 21, 50] and algorithmic perspectives [1, 3, 14, 23, 29, 32, 44, 45].

In many situations, the annotated training data may consist of a combination of distinct datasets, some of which may be closer or further away from the target data. Finding nontrivial ways of combining multiple datasets to approximate the target distribution and extracting relevant knowledge from such combination is the purpose of multi-source DA algorithms [25, 19, 21, 33, 38, 47, 51] and is also the focus of this work. Hoffman et al. [21] focus on learning a convex combination rule to combine the classifiers of each source domain into a single classifier for the target domain. Zhao et al. [51] optimize the worst-case adaptation scenario among all source domains on each training iteration. Kim et al. [25] and Guo et al. [19] learn example-to-domain relations to weigh each source domain differently for each target example. Our approach is comparable to those of Hoffman et al. [21] and Zhao et al. [51] in the sense that we learn a global alignment from target to source domains, but we follow a different principle: we treat the weights for each source domain as one further parameter that can be exploited to minimize the loss function and learned jointly with the remaining model parameters.

Preprint.

To make the DA task possible, it is often assumed that some properties are invariant across domains [17, 22, 28, 30, 42, 47, 49, 48]. The scenario where the marginal distribution of features changes but the conditional of labels given features is the same across domains is known as the *covariate shift* setting. It has been explored extensively either by reweighting samples from the source domain to match the target distribution [9, 39, 43] or, more recently, by learning domain-invariant feature representations [1, 15, 16, 19, 35, 38]. The ability of deep neural networks to learn rich feature representations and the recent surge of adversarial learning techniques have led to numerous approaches that resort to an adversarial objective to learn those representations. The adversary is usually implemented with a domain discriminator network [16], which aims to discriminate samples between source and target domains, and is jointly trained with the feature extractor and task classifier through a minimax game. Other models resort to the minimization of distribution dissimilarity metrics between the target and source feature distributions [15, 19]. It is known, however, that if the learned features violate the covariate shift assumption, domain-invariant features shall deteriorate the generalization performance of the model on the target domain [50] – a problem that we refer to as *the curse of domain-invariant representations*. Addressing this issue in the unsupervised setting is challenging, although some strategies have been proposed. Pei et al. [35] use a domain discriminator per class and the probability output of the task classifier as attention weights for the discriminator. Sebag et al. [38] introduce a loss term that repulses unlabeled examples from the labeled ones whenever the classifier has low confidence on classifying the unlabeled examples. Thus, both methods depend on the classifier to make correct predictions on the target samples early in the training. Here, we avoid this issue by employing a consistency loss, together with a minimum confidence threshold, that enforces agreement on the class predictions for original and augmented target samples.

Our contributions are summarized as follows: i) we present a corollary of the theoretical results from [4] which motivates our methodology; ii) we present our multi-source adversarial model which learns the distribution weights for each source domain jointly with all remaining parameters and following a theoretically-grounded mildly optimistic approach; iii) we discuss how a simple consistency regularization technique may help avoid the curse of domain-invariant representations; iv) we conduct extensive experiments on benchmark datasets that confirm the effectiveness of the proposed methodologies.

2 Background and motivation

We first introduce the problem of unsupervised multi-source DA and review the theoretical model for DA from Ben-David et al. [4, 5].

2.1 Problem statement

In the unsupervised multi-source DA setting, the learning algorithm has access to M annotated datasets $S_j = \{(x_i, y_i)\}_{i=1}^{n_j} \in (\mathcal{X} \times \mathcal{Y})^{n_j}$, $j \in \{1, 2, ..., M\}$, sampled i.i.d. from M joint distributions of samples and labels, each one corresponding to a source domain $\mathcal{S}_j$. We further assume to have access to a set of unlabeled examples $T = \{x_i\}_{i=1}^{n} \in \mathcal{X}^n$, sampled i.i.d. from the marginal distribution of samples in the target domain $\mathcal{T}$. In this work, we focus on classification problems, so the set $\mathcal{Y}$ is discrete. The ground-truth labels of the examples in $\mathcal{T}$ are known to be in $\mathcal{Y}$, i.e. every class in the target domain is represented in the source domains. Loosely speaking, the goal of unsupervised multi-source DA is to exploit the data $S_1, ..., S_M$, and T to learn a classifier with good generalization performance on the target domain.

2.2 Notation and useful definitions

Formally, a domain is a triplet $(\mathcal{X}, \mathcal{Y}, \mathcal{D}_{\mathrm{X,Y}})$, where $\mathcal{X}$ is the input space, $\mathcal{Y}$ is the set of labels and $\mathcal{D}_{\mathrm{X,Y}}$ is a joint distribution over $\mathcal{X} \times \mathcal{Y}$ with marginals $\mathcal{D}_{\mathrm{X}} = \int_{\mathcal{Y}} \mathcal{D}_{\mathrm{X,Y}}$ and $\mathcal{D}_{\mathrm{Y}} = \int_{\mathcal{X}} \mathcal{D}_{\mathrm{X,Y}}$. In the DA setting, $\mathcal{X}$ and $\mathcal{Y}$ are shared across all source and target domains. With some abuse of notation, from now on we denote a domain and its marginal $\mathcal{D}_{\mathrm{X}}$ using the same symbol $\mathcal{D}$. The distribution of samples in $\mathcal{D}$ after a feature transformation $g : \mathcal{X} \mapsto \mathcal{Z}$ is denoted by $\mathcal{D}^g$. For a given α in the simplex Δ, we define $\mathcal{S}_\alpha$ as the α-weighted mixture of source domains, i.e. $\mathcal{S}_\alpha = \sum_{j=1}^{M} \alpha_j \mathcal{S}_j$.

As is common in the DA literature, we shall restrain the theoretical analysis to binary classification. Under this setting, each domain $\mathcal{D}$ is associated with a labeling function $f_{\mathcal{D}} : \mathcal{X} \mapsto [0, 1]$ that

corresponds to the Bayes optimal classifier in $\mathcal{D}$, $f_{\mathcal{D}}(x) = \Pr_{\mathcal{D}}(y = 1 \,|\, x)$. A *hypothesis* is any function $h : \mathcal{X} \mapsto \{0, 1\}$ and a set $\mathcal{H}$ of hypotheses is called a *hypothesis class*. The *risk* of a hypothesis h w.r.t. the domain $\mathcal{D}$ is defined as $\varepsilon_{\mathcal{D}}(h) = \mathbb{E}_{\mathcal{D}}\left(|h(x) - f_{\mathcal{D}}(x)|\right)$. Its empirical estimation is denoted by $\widehat{\varepsilon}_{\mathcal{D}}(h)$. We further restrain to the deterministic case, where $f_{\mathcal{D}} : \mathcal{X} \mapsto \{0, 1\}$ and the label $y \in \{0, 1\}$ associated with a sample $x \in \mathcal{X}$ is uniquely determined by the rule $y = f_{\mathcal{D}}(x)$, and thus $\varepsilon_{\mathcal{D}}(h) = \Pr_{\mathcal{D}}\left(h(x) \neq f_{\mathcal{D}}(x)\right)$.

Given two distributions $\mathcal{D}$ and $\mathcal{D}'$ over $\mathcal{X}$, the *$\mathcal{H}$-divergence* [4], defined below, provides a measure of the distance between the distributions according to a fixed hypothesis class $\mathcal{H}$:

$$d_{\mathcal{H}}(\mathcal{D}, \mathcal{D}') = 2 \sup_{h \in \mathcal{H}} |\Pr_{\mathcal{D}}(\mathbf{1}(h)) - \Pr_{\mathcal{D}'}(\mathbf{1}(h))|, \tag{1}$$

where $\mathbf{1}(h) = \{x \in \mathcal{X} : h(x) = 1\}$. The $\mathcal{H}$-divergence is a particularly convenient metric since it can be estimated empirically from a finite number of samples whenever $\mathcal{H}$ has a finite VC-dimension. This empirical estimation is denoted by $\widehat{d}_{\mathcal{H}}(\mathcal{D}, \mathcal{D}')$. A very useful insight is the fact that, under a weak assumption on $\mathcal{H}$, computing $\widehat{d}_{\mathcal{H}}(\mathcal{D}, \mathcal{D}')$ is equivalent to finding the optimal classifier in $\mathcal{H}$ on the task of discriminating between samples of $\mathcal{D}$ and $\mathcal{D}'$ [4, 5].

For a hypothesis class $\mathcal{H}$, we may define the *symmetric difference hypothesis class* $\mathcal{H}\Delta\mathcal{H}$ as:

$$\mathcal{H}\Delta\mathcal{H} = \{l : l(x) = h(x) \oplus h'(x),\ h, h' \in \mathcal{H}\}, \tag{2}$$

where $\oplus$ denotes the "exclusive or" (xor) operation. Combining this definition with equation (1), the definition of $\mathcal{H}\Delta\mathcal{H}$-divergence, which plays a major role in the next subsection, follows immediately.

2.3 An upper bound on the target risk

Intuitively, if the source and target domains are sufficiently similar, a classifier trained to reach low empirical error on the source domain will likely achieve a low error on the target domain too. When multiple source domains are available, a combination of their respective data yields the optimal strategy. The following bound, which is a corollary of the results in [4], formalizes these ideas and enlightens some properties that will be exploited by our approach. For completeness, the proof is provided in the appendix (section A).

Theorem 1. *Let $\mathcal{H}$ be a hypothesis class with VC-dimension d. Consider an unlabeled set of n samples drawn from the target domain $\mathcal{T}$ and, for each $j \in \{1, 2, ..., M\}$, a labeled set of n/M samples drawn from the source domain $\mathcal{S}_j$. Then, for any $h \in \mathcal{H}$ and any $\alpha \in \Delta$, with probability at least $1 - \delta$ over the choice of samples,*

$$\varepsilon_{\mathcal{T}}(h) \leq \sum_{j=1}^{M} \alpha_j \widehat{\varepsilon}_{\mathcal{S}_j}(h) + \frac{1}{2} \widehat{d}_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S}_\alpha, \mathcal{T}) + \lambda_\alpha + B_\alpha(\delta) + V(\delta), \tag{3}$$

where

$$B_\alpha(\delta) = 2 \sqrt{\frac{M\left(2d \log(2(n+1)) + \log\left(\frac{8}{\delta}\right)\right)}{n} \sum_{j=1}^{M} \alpha_j^2}, \tag{4}$$

$$\lambda_\alpha = \min_{h \in \mathcal{H}} \sum_{j=1}^{M} \alpha_j \varepsilon_{\mathcal{S}_j}(h) + \varepsilon_{\mathcal{T}}(h), \tag{5}$$

$$V(\delta) = 2 \sqrt{\frac{2d \log(2n) + \log\left(\frac{4}{\delta}\right)}{n}}. \tag{6}$$

This bound is structurally very similar to the one presented in [51], with two slight differences: i) we work with the (empirical) $\mathcal{H}\Delta\mathcal{H}$-divergence between the target and the mixture of source domains directly, instead of upper-bounding it with the convex combination of (empirical) $\mathcal{H}\Delta\mathcal{H}$-divergences between the target and each source domain; and ii) we show the dependency of the bound on the quantity $B_\alpha(\delta)$, whose interpretation is given in the following discussion.

2.3.1 The curse of domain-invariant representations

If the optimal α was known, a learning algorithm for DA could, in principle, be trained to minimize the first two terms in the bound. For this purpose, it should find a function $g : \mathcal{X} \mapsto \mathcal{Z}$ in a set of feature transformations $\mathcal{F}$ and a classifier $h : \mathcal{Z} \mapsto \mathcal{Y}$ in $\mathcal{H}$. The first term in the bound is minimized

by training the classifier h on the desired task, using the labeled data from all source domains. The second term is minimized by finding a feature transformation g such that the induced distributions $\mathcal{T}^g$ and $\mathcal{S}^g_\alpha$ are similar. This is the main idea exploited by adversarial-based DA algorithms, which use an adversarial objective to match source and target distributions in a latent feature space. The problem with this approach is that it completely overlooks the role of the third term, λ_α, which corresponds to the minimum possible combined risk of a classifier in $\mathcal{H}$ on the source and target domains. A recent work [50] shows constructively that a low error on the source domain and domain-invariant features are insufficient to ensure low target risk and may actually have the opposite effect, by increasing λ_α. Sufficiency is established only under the covariate shift assumption, where the conditional distributions of labels $y \in \mathcal{Y}$ given features $z \in \mathcal{Z}$ are the same across source and target domains and only the marginal distributions of features differ. Since, in the unsupervised DA setting, target labels are not available, imposing or verifying covariate shift is not possible. Moreover, whenever the marginal distributions of labels differ (target shift), a strong enforcement of domain-invariant feature representations necessarily hurts the covariate shift assumption, by imposing the marginals on features to coincide. For this reason, training adversarial-based DA algorithms for many iterations generally yields worse performance [50]. In this work, we show empirically that learning a more robust feature transformation will generally help mitigate this issue.

2.3.2 Choosing the combination of source domains

Looking at the first two terms of the learning bound in Theorem 1 suggests that α could be chosen in an optimistic way, by selecting the source domain in which the sum of the risk of the classifier and the $\mathcal{H}\Delta\mathcal{H}$-divergence w.r.t. the target reaches the lowest value. However, the term $B_\alpha(\delta)$ is proportional to the ℓ^2-norm of α, which is maximum when α is one-hot and minimum when $\alpha_j = 1/M, \forall\, j$. This has an intuitive explanation: choosing a sparse α implies discarding data from the source domains whose component is zero, so the classifier is trained with intrinsically less data and its error tends to increase. This discussion naturally leads to the following formal objective:

$$\min_{\alpha\in\Delta,\, h\in\mathcal{H},\, g\in\mathcal{F}} \sum_{j=1}^{M} \alpha_j \widehat{\varepsilon}_{\mathcal{S}_j}(h\circ g) + \frac{1}{2}\widehat{d}_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S}^g_\alpha, \mathcal{T}^g) + \mu||\alpha||_2, \tag{7}$$

where $\mu = \mu(\delta) > 0$. This objective contrasts with the idea in [51], where the authors essentially choose to minimize the loss for the hardest source domain at each iteration of the learning algorithm, which is a much more pessimistic approach than ours. In either case, though, the minimum combined risk λ_α is uncontrolled and, as discussed in section 2.3.1, this is an issue that must be addressed.

3 Methodology

Given the discussion in the previous section, we now explain our method in detail. It basically consists of two major approaches, explained throughout this section.

3.1 Domain adaptation from a dynamic mixture of sources

We first address the problem of casting the objective (7) into a computationally treatable surrogate. As in many recent works in DA, we resort to neural networks to implement g and h and the empirical $\mathcal{H}\Delta\mathcal{H}$-divergence is implemented with a domain discriminator network, which aims to distinguish between samples of $\mathcal{S}^g_\alpha$ and $\mathcal{T}^g$.[1] Taking the previous considerations and replacing the intractable empirical risks by the usual classification losses, objective (7) is cast as:

$$\min_{\alpha\in\Delta,\theta_g,\theta_h} \max_{\theta_d} \left\{\mathcal{L}(\alpha,\theta_g,\theta_h,\theta_d) = \mathcal{L}_{\text{class}}(\alpha,\theta_g,\theta_h) - \mu_d\mathcal{L}_{\text{disc}}(\alpha,\theta_g,\theta_d) + \mu_s||\alpha||_2^2\right\}, \tag{8}$$

where

$$\mathcal{L}_{\text{class}}(\alpha,\theta_g,\theta_h) = -\sum_{j=1}^{M} \alpha_j \mathbb{E}_{\mathrm{z}\sim\mathcal{S}^g_j}\left[\log p(\mathrm{y} = f_{\mathcal{S}^g_j}(\mathrm{z}) \,|\, \mathrm{z},\theta_h)\right], \tag{9}$$

$$\mathcal{L}_{\text{disc}}(\alpha,\theta_g,\theta_d) = -\mathbb{E}_{\mathrm{z}\sim\mathcal{S}^g_\alpha}\left[\log p(\mathrm{d} = 0 \,|\, \mathrm{z},\theta_d)\right] - \mathbb{E}_{\mathrm{z}\sim\mathcal{T}^g}\left[\log p(\mathrm{d} = 1 \,|\, \mathrm{z},\theta_d)\right]. \tag{10}$$

[1] Theoretically, some care should be taken while designing the architecture of the discriminator to make sure that it parameterizes hypotheses in $\mathcal{H}\Delta\mathcal{H}$, but in practice this constraint is dropped. There are similar bounds using the $\mathcal{H}$-divergence instead [38].

Here, θ_h and θ_d are the parameters of the classifier and domain discriminator networks, respectively, $\mu_d, \mu_s > 0$ are hyperparameters, $\mathrm{y} \in \mathcal{Y}$ is the categorical r.v. associated with the class label, $f_{\mathcal{S}_j^g} : \mathcal{Z} \mapsto \mathcal{Y}$ is the true (multiclass) labeling function for the j-th source domain, d is a Bernoulli r.v. that discriminates source and target domains and the function $g = g(\cdot, \theta_g) : \mathcal{X} \mapsto \mathcal{Z}$ is a neural network, parameterized by θ_g, mapping input samples to features. By linearity of expectations and using the fact that $\mathrm{z} = g(\mathrm{x}, \theta_g)$, for input samples $\mathrm{x} \in \mathcal{X}$, we have:

$$\mathcal{L}_{\text{class}}(\alpha, \theta_g, \theta_h) = -\sum_{j=1}^{M} \alpha_j \mathbb{E}_{\mathrm{x} \sim \mathcal{S}_j} \left[\log p(\mathrm{y} = f_{\mathcal{S}_j}(\mathrm{x}) \mid \mathrm{x}, \theta_g, \theta_h) \right], \tag{11}$$

$$\mathcal{L}_{\text{disc}}(\alpha, \theta_g, \theta_d) = -\sum_{j=1}^{M} \alpha_j \mathbb{E}_{\mathrm{x} \sim \mathcal{S}_j} \left[\log p(\mathrm{d} = 0 \mid \mathrm{x}, \theta_g, \theta_d) \right] - \mathbb{E}_{\mathrm{x} \sim \mathcal{T}} \left[\log p(\mathrm{d} = 1 \mid \mathrm{x}, \theta_g, \theta_d) \right]. \tag{12}$$

In order to satisfy the constraint $\alpha \in \Delta$, we reparameterize the model using $\alpha = \mathrm{softmax}(\beta)$, for an unconstrained parameter $\beta \in \mathbb{R}^M$. Finally, using the data presented in section 2.1, we may compute empirical estimates of losses (11) and (12):

$$\mathcal{L}_{\text{class}}(\beta, \theta_g, \theta_h) \approx -\frac{1}{m} \sum_{j=1}^{M} \frac{\exp(\beta_j)}{\sum_{j'} \exp(\beta_{j'})} \sum_{(x,y) \in S_j^{(m)}} \log p(y \mid x, \theta_g, \theta_h), \tag{13}$$

$$\begin{aligned} \mathcal{L}_{\text{disc}}(\beta, \theta_g, \theta_d) \approx & -\frac{1}{m} \sum_{j=1}^{M} \frac{\exp(\beta_j)}{\sum_{j'} \exp(\beta_{j'})} \sum_{(x,y) \in S_j^{(m)}} \log p(\mathrm{d} = 0 \mid x, \theta_g, \theta_d) \\ & - \frac{1}{m} \sum_{x \in T^{(m)}} \log p(\mathrm{d} = 1 \mid x, \theta_g, \theta_d), \end{aligned} \tag{14}$$

where $S_j^{(m)}$ and $T^{(m)}$ are mini-batches of m examples each from S_j and T, respectively.

It is instructive to compare our approach with [51], as that is the most similar to ours. The bound in equation (3) uses one single $\mathcal{H}\Delta\mathcal{H}$-divergence and, therefore, our model comprises one single domain discriminator, which aims to distinguish between the target and the α-weighted mixture of source domains. In [51], they use M discriminator networks, i.e. one per source domain. More importantly, regarding the choice of α, we treat it as one further parameter that can be optimized to minimize the loss. To avoid the objective to collapse into the easiest source domain, as explained in section 2.3.2, we include an extra term that penalizes sparse α's. Unlike us, they do not include the sparsity penalization term and choose α to minimize the worst case scenario, by assigning a greater weight to the maximum loss among the M source domains on each training iteration.

3.2 Consistency regularization on the target domain

Consistency regularization is a key component of many state of the art algorithms in semi-supervised learning. The basic idea is simple: an unlabeled sample and a slightly perturbed version of it should share the same label. Here, this approach is exploited as a sensible heuristic to avoid that domain-invariant features end up hurting the generalization performance of the model on the target domain, as discussed in section 2.3.1.

Specifically, consider a target sample $x \in T$ and a parametric transformation $\omega : \mathcal{X} \times \Xi \mapsto \mathcal{X}$, where Ξ is the space of parameters for the transformation. Further assume that ω is label-preserving, i.e. $f_{\mathcal{T}}(\omega(x, \xi)) = f_{\mathcal{T}}(x), \forall\, x \in \mathcal{X}, \xi \in \Xi$. If ω is rich and strong enough, it should spread the transformed target samples over multiple regions of low density under the target distribution, i.e. it will produce new domains. Then, by enforcing agreement on the predictions of original and transformed target samples, we are encouraging the model to learn to extract features that generalize well across domains. Moreover, some of the augmented samples may fall within regions of higher density under the distribution of the source domains. If this hypothesis holds, by enforcing agreement on the predictions of original and transformed target samples and low classification error on the source domains, we are indirectly promoting low error on the target domain. Figure 1 illustrates this idea.

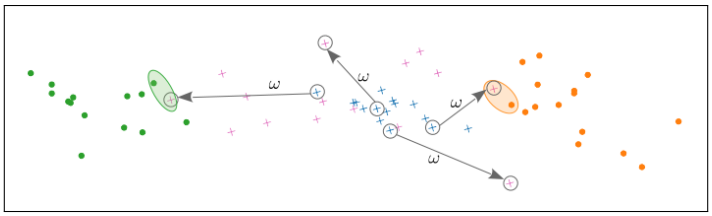


Figure 1: Toy illustration of the desired effect of the consistency regularization, where images are represented as lying on a 2-D space. Green and orange circles represent (labeled) samples from two distinct source domains; blue and purple x-markers represent original and augmented (unlabeled) target samples, respectively. Colored ellipses enclose pairs of augmented and source samples that are close to each other and therefore are likely to share the same label.

The particular type of consistency regularization we adopt here is FixMatch [40], due to its simplicity and good performance. This approach involves using the predicted class of original (or weakly-transformed) samples as pseudo-labels for the (strongly-)transformed samples and, in our setting, is translated into the following loss function:

$$\mathcal{L}_{\text{cons}}(\theta_g, \theta_h) = -\frac{1}{m} \sum_{x \in T^{(m)}} \mathbf{1}\left(\max_{\text{y} \in \mathcal{Y}} p(\text{y} \,|\, x, \theta_g, \theta_h) > \tau \right) \log p(\tilde{y} \,|\, \omega(x, \xi), \theta_g, \theta_h), \tag{15}$$

where $\tilde{y} = \arg\max_{\text{y} \in \mathcal{Y}} p(\text{y} \,|\, x, \theta_g, \theta_h)$ is the pseudo-label, ξ is chosen randomly for each x, and $\tau \geq 0$ is a hyperparameter defining the minimum confidence threshold for the loss to be applied. This threshold prevents the loss to be applied too early in the training process and, in our setting, may discard augmented samples that fall too far from the source distributions. The overall objective is then written as follows:

$$\min_{\alpha \in \Delta, \theta_g, \theta_h} \max_{\theta_d} \left\{ \mathcal{L} = \mathcal{L}_{\text{class}}(\alpha, \theta_g, \theta_h) + \mu_c \mathcal{L}_{\text{cons}}(\theta_g, \theta_h) - \mu_d \mathcal{L}_{\text{disc}}(\alpha, \theta_g, \theta_d) + \mu_s ||\alpha||_2^2 \right\}, \tag{16}$$

where $\mu_c > 0$ controls the relative weight of the consistency loss. A full schematic of our model is provided in the appendix (section B).

It remains to discuss how to build the transformation ω in such a way that it is strong and diverse enough while satisfying the constraint of being label-preserving. This is essentially an application-dependent problem, though. For vision problems, there are multiple simple label-preserving transformations (e.g. translation, rotation, sharpness enhancement, etc.) that can be applied in a pipeline and, as consequence, the transformed image ends up being a strongly distorted version of the original one. This is the idea followed, for instance, in RandAugment [11], which we use here. RandAugment receives as input parameters the magnitude and the number of transformations to be applied and randomly chooses the transformations to apply to each sample in the mini-batch. Here, as in [40], we choose the number and magnitude of the transformations uniformly at random for each mini-batch.

For generic, non-vision problems, adding random noise sampled from some known distribution (e.g. Gaussian) is a trivial realization of ω. Another possibility is to apply a strong dropout [41] transformation at the input and inner layers of the neural network. Although such transformation is not necessarily label-preserving, dropout is known to be a successful regularization technique when applied at fully connected layers. We employ this idea in one of the experiments conducted here.

4 Experiments

4.1 Experimental setting

We now conduct several experiments using standard benchmark datasets for multi-source DA. We follow established evaluation protocols for every dataset and, as far as possible, we use the same network architecture for our model and baselines. Notably, comparing with MDAN [51], our single domain discriminator has the same architecture as each of the M domain discriminators in their model. Consequently, our model has less trainable parameters than theirs. Our parameter β is initialized uniformly at random in $[0, 1]^M$, so the resulting α initially weighs all source domains roughly equally, but it may become sparse as training evolves. We illustrate this behavior in the

Table 4.1: Average accuracy ± standard deviation (%) over 5 independent runs on digits and objects classification. The domain on each column corresponds to the target.

	MNIST	MNIST-M	SVHN	Amazon	DSLR	Webcam
DANN-SS [16]	97.9 ±0.4	73.2 ±1.6	72.8 ±3.3	60.5 ±1.4	**100.0** ±0.0	98.0 ±0.3
DANN-MS [16]	97.9 ±1.6	67.5 ±1.4	71.5 ±1.6	61.2 ±1.3	99.9 ±0.2	98.8 ±0.3
MADA [35]	–	–	–	70.3 ±0.3	99.6 ±0.1	97.4 ±0.1
MDAN [51]	98.3 ±0.2	69.1 ±1.2	69.5 ±2.8	65.2 ±0.4	99.3 ±0.2	97.8 ±0.5
MoE [19]	98.6 ±0.2	69.9 ±1.1	81.8 ±0.8	–	–	–
MODA	98.4 ±0.2	77.4 ±1.6	71.7 ±1.5	65.5 ±0.5	99.9 ±0.2	99.0 ±0.3
FM	**99.2** ±0.1	91.1 ±0.4	**90.0** ±0.8	70.3 ±0.6	99.7 ±0.2	**99.2** ±0.4
MODA-FM	98.8 ±0.1	**95.4** ±0.4	89.4 ±1.4	**70.7** ±0.9	**100.0** ±0.0	99.1 ±0.1
Fully supervised	98.9 ±0.1	96.2 ±0.1	90.3 ±0.5	88.1 ±1.6	99.3 ±1.1	99.5 ±0.7

appendix (section C.4). Hyperparameter tuning was performed through cross-validation over source domains and a hyperparameter sensitivity analysis is conducted in the appendix (section C.3). Further details about the experiments, including the label distributions on each domain, network architectures, image transformations, and search ranges for each hyperparameter, are also provided in the appendix (sections C.1, C.5, C.6, and C.7). The PyTorch-based implementation of our model is publicly available.[1] To facilitate the presentation, from now on we refer to our model as MODA-FM (Multi-source mildly Optimistic Domain Adaptation with FixMatch regularization).

Baselines DANN-SS [16]: a single-source DA model, where the best results among all source domains are reported. DANN-MS: the same model as before trained on the combined data from all source domains. MADA [35]: a state of the art model for single-source DA which tries to align conditional distributions by using one domain discriminator per class (results extracted from [35], we report the best accuracy among all source domains). MDAN [51]: a state of the art model for multi-source DA, widely described before. MoE [19]: a state of the art model for multi-source DA that uses one classifier per source domain whose predictions are weighted in an example-dependent way. Fully supervised: fully supervised model trained on the target data, to provide an empirical upper bound on the performance of the DA task. MODA: our model without consistency regularization (i.e., $\mu_c = 0$). FM: our model without domain discriminator, trained on the naively combined data from all source domains and using FixMatch consistency regularization as described in section 3.2.

Digits classification In this experiment, the task is digit classification using 4 datasets: MNIST [27], MNIST-M [16], SVHN [34], and SynthDigits [16]. We take each of the first three datasets as the target in turn, and use the remaining as source domains. The number of training images chosen randomly from each domain, including the target, is 20k. The evaluation is performed in the non-transductive setting, i.e. no target data used during training are used for evaluation. The results are in Table 4.1.

Object classification on Office-31 dataset Office-31 [36] is a standard benchmark dataset for domain adaptation. It comprises 31 object categories extracted from 3 domains: Amazon, which contains 2817 images downloaded from `amazon.com`, DSLR and Webcam, which contain 498 and 795 images captured with DSLR cameras and webcams, respectively, under different environments. In this experiment, we adopt the fully-transductive setting, following [35], where all unlabeled data from the target domain are used for training (except for the fully supervised model, where we use 80% of the data for training and the remaining for testing). All models are implemented using a pre-trained (on ImageNet) ResNet-50 [20] as the base architecture. We take each domain as the target in turn and all remaining are used as sources. The results are presented in Table 4.1.

Sentiment analysis on Amazon Reviews dataset The Amazon Reviews dataset [7] is another multi-domain dataset widely used as a benchmark for domain adaptation. It contains binary (i.e., positive and negative) reviews on 4 types of products: books, DVDs, electronics and kitchen appliances. Here, we follow the experimental setting from [8], where samples were pre-processed to 5k-dimensional TF-IDF feature vectors, thus word order information was not preserved. We choose 2k training samples from each domain randomly and the remaining target samples are used for testing, following the non-transductive setting. Since the data are not images, we use dropout as

[1] `https://github.com/dpernes/modafm`

Table 4.2: Average accuracy $\pm$ standard deviation (%) over 5 independent runs on sentiment analysis (Amazon Reviews). The domain on each column corresponds to the target.

	Books	DVD	Electronics	Kitchen
DANN-SS [16]	77.5 $\pm$1.0	78.9 $\pm$0.9	84.2 $\pm$0.2	85.8 $\pm$0.4
DANN-MS [16]	78.9 $\pm$0.2	80.8 $\pm$1.0	84.7 $\pm$0.6	87.0 $\pm$0.4
MDAN [51]	79.1 $\pm$0.3	81.3 $\pm$0.8	84.6 $\pm$0.3	85.6 $\pm$1.6
MoE [19]	80.3 $\pm$0.3	81.9 $\pm$0.6	85.2 $\pm$0.6	87.4 $\pm$0.5
MODA	79.0 $\pm$0.2	80.7 $\pm$1.2	84.7 $\pm$0.3	86.8 $\pm$0.6
FM	78.8 $\pm$2.0	81.0 $\pm$1.5	**85.6** $\pm$0.9	87.6 $\pm$0.7
MODA-FM	**80.9** $\pm$1.1	**82.0** $\pm$1.0	85.3 $\pm$0.4	**88.4** $\pm$0.3
Fully supervised	83.6 $\pm$0.4	83.6 $\pm$0.6	85.4 $\pm$0.4	87.8 $\pm$0.2

our (pseudo-)label-preserving transformation. Specifically, on each training iteration, we randomly choose a dropout rate (in a pre-specified range) to be applied at the input and hidden layers of the MLP and the corresponding output prediction is used as the pseudo-label $\tilde{y}$. No dropout is applied for source and non-augmented target samples. All domains are taken as the target in turn and all remaining are used as sources. The results are in Table 4.2.

4.2 Discussion

Tables 4.1 and 4.2 show that our model outperforms the baselines in most settings and performs comparably to the fully-supervised model in many. When no consistency regularization is used (MODA), our approach exhibits a higher accuracy than DANN-SS, DANN-MS and MDAN in most cases. This observation shows that our mildly optimistic combination of source domains works better than using only the data from the best source domain (DANN-SS) or naively combining the data from all source domains (DANN-MS). It also suggests that MDAN wastes too much computational effort on optimizing itself for the hardest source domain. MADA and MoE perform better than our non-regularized model in some cases, which is not surprising since both methods try to mitigate somehow the curse of domain-invariant representations. Their advantage is, therefore, mostly noticeable when the target shift is large (e.g., SVHN and all domains in Office-31 – see the appendix, section C.1), but the performance is still below MODA-FM.

Very significant performance gains are observed when we apply the consistency regularization, particularly in the visual datasets (digits and Office-31). These gains are more expressive in the most challenging settings, i.e. when the target data are perceptually very different from the source data (e.g., MNIST-M – see the appendix, section C.8) or when the target shift is large. The latter observation suggests that this regularization succeeds on mitigating the curse of domain-invariant representations, as hypothesized before. This is strongly corroborated by an experiment we present in the appendix (section C.2), where we show that the model accuracy on the target domain keeps stably high when the model is trained for a large number of epochs. Interestingly, though, we observe that, in some settings, FM outperforms MODA-FM, although the differences are small. In these cases domain-invariant representations are slightly hurting the performance and/or the source weights α are sub-optimal. Finally, it is worth highlighting the positive effect provided by using dropout as the label-preserving transformation ω in the experiment with Amazon Reviews. This observation suggests that this methodology can be applied successfully to non-visual data too.

5 Conclusion

We have presented a novel algorithm that achieves state of the art results in unsupervised multi-source domain adaptation. In our approach, the problem is formulated as DA from a single source domain whose distribution corresponds to a mixture of the original source domains. The mixture weights are adjusted dynamically throughout the training process, according to a mildly optimistic objective. Additionally, we employ FixMatch on the target samples, a form of consistency regularization that proves to have a strong impact on the model performance and to be capable of mitigating the curse of domain invariant representations. This regularization relies on a label-preserving transformation, which is hard to construct for non-visual data. Moreover, better results could be achieved if both the label-preserving transformation and the source weights were learned to approximate the augmented target samples close to the source samples. Both problems are interesting lines for future research.

Acknowledgments and Disclosure of Funding

This work was funded by FCT - Fundação para a Ciência e a Tecnologia within Ph.D. grant number SFRH/BD/129600/2017.

References

[1] H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, and M. Marchand. Domain-adversarial neural networks. *arXiv preprint arXiv:1412.4446*, 2014.

[2] M. Baktashmotlagh, M. T. Harandi, B. C. Lovell, and M. Salzmann. Unsupervised domain adaptation by domain invariant projection. In *Proceedings of the IEEE International Conference on Computer Vision*, 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] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

[5] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira. Analysis of representations for domain adaptation. In *Advances in Neural Information Processing Systems*, pages 137–144, 2007.

[6] J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. Wortman. Learning bounds for domain adaptation. In *Advances in Neural Information Processing Systems*, pages 129–136, 2008.

[7] J. Blitzer, M. Dredze, and F. Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 440–447, 2007.

[8] M. Chen, Z. Xu, K. Weinberger, and F. Sha. Marginalized denoising autoencoders for domain adaptation. *arXiv preprint arXiv:1206.4683*, 2012.

[9] C. Cortes, Y. Mansour, and M. Mohri. Learning bounds for importance weighting. In *Advances in Neural Information Processing Systems*, pages 442–450, 2010.

[10] C. Cortes and M. Mohri. Domain adaptation and sample bias correction theory and algorithm for regression. *Theoretical Computer Science*, 519:103–126, 2014.

[11] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le. Randaugment: Practical data augmentation with no separate search. *arXiv preprint arXiv:1909.13719*, 2019.

[12] H. Daumé III, A. Kumar, and A. Saha. Frustratingly easy semi-supervised domain adaptation. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 53–59. Association for Computational Linguistics, 2010.

[13] J. Donahue, J. Hoffman, E. Rodner, K. Saenko, and T. Darrell. Semi-supervised domain adaptation with instance constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 668–675, 2013.

[14] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2960–2967, 2013.

[15] P. M. Ferreira, D. Pernes, A. Rebelo, and J. S. Cardoso. Desire: Deep signer-invariant representations for sign language recognition. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2019.

[16] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, pages 1180–1189, 2015.

[17] M. Gong, K. Zhang, T. Liu, D. Tao, C. Glymour, and B. Schölkopf. Domain adaptation with conditional transferable components. In *International Conference on Machine Learning*, pages 2839–2848, 2016.

[18] R. Gopalan, R. Li, and R. Chellappa. Unsupervised adaptation across domain shifts by generating intermediate data representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2288–2302, 2013.

[19] J. Guo, D. Shah, and R. Barzilay. Multi-source domain adaptation with mixture of experts. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4694–4703, 2018.

[20] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[21] J. Hoffman, M. Mohri, and N. Zhang. Algorithms and theory for multiple-source adaptation. In *Advances in Neural Information Processing Systems*, pages 8246–8256, 2018.

[22] A. Iyer, S. Nath, and S. Sarawagi. Maximum mean discrepancy for class ratio estimation: Convergence bounds and kernel selection. In *International Conference on Machine Learning*, pages 530–538, 2014.

[23] I.-H. Jhuo, D. Liu, D. Lee, and S.-F. Chang. Robust visual domain adaptation with low-rank reconstruction. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2168–2175. IEEE, 2012.

[24] G. Kang, L. Jiang, Y. Yang, and A. G. Hauptmann. Contrastive adaptation network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4893–4902, 2019.

[25] Y.-B. Kim, K. Stratos, and D. Kim. Domain attention with an ensemble of experts. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 643–653, 2017.

[26] A. Kumar, A. Saha, and H. Daume. Co-regularization based semi-supervised domain adaptation. In *Advances in Neural Information Processing Systems*, pages 478–486, 2010.

[27] 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.

[28] Z. C. Lipton, Y.-X. Wang, and A. Smola. Detecting and correcting for label shift with black box predictors. *arXiv preprint arXiv:1802.03916*, 2018.

[29] M. Long, Y. Cao, J. Wang, and M. I. Jordan. Learning transferable features with deep adaptation networks. *arXiv preprint arXiv:1502.02791*, 2015.

[30] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu. Transfer feature learning with joint distribution adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2200–2207, 2013.

[31] M. Long, H. Zhu, J. Wang, and M. I. Jordan. Unsupervised domain adaptation with residual transfer networks. In *Advances in Neural Information Processing Systems*, pages 136–144, 2016.

[32] C. Louizos, K. Swersky, Y. Li, M. Welling, and R. Zemel. The variational fair autoencoder. In *International Conference on Learning Representations*, 2016.

[33] Y. Mansour, M. Mohri, and A. Rostamizadeh. Domain adaptation: Learning bounds and algorithms. In *22nd Conference on Learning Theory, COLT 2009*, 2009.

[34] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[35] Z. Pei, Z. Cao, M. Long, and J. Wang. Multi-adversarial domain adaptation. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[36] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *European Conference on Computer Vision*, pages 213–226. Springer, 2010.

[37] K. Saito, D. Kim, S. Sclaroff, T. Darrell, and K. Saenko. Semi-supervised domain adaptation via minimax entropy. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8050–8058, 2019.

[38] A. Schoenauer-Sebag, L. Heinrich, M. Schoenauer, M. Sebag, L. Wu, and S. Altschuler. Multi-domain adversarial learning. In *International Conference on Learning Representations*, 2019.

[39] 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.

[40] K. Sohn, D. Berthelot, C.-L. Li, Z. Zhang, N. Carlini, E. D. Cubuk, A. Kurakin, H. Zhang, and C. Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *arXiv preprint arXiv:2001.07685*, 2020.

[41] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[42] A. Storkey. When training and test sets are different: characterizing learning transfer. *Dataset shift in machine learning*, pages 3–28, 2009.

[43] M. Sugiyama, T. Suzuki, S. Nakajima, H. Kashima, P. von Bünau, and M. Kawanabe. Direct importance estimation for covariate shift adaptation. *Annals of the Institute of Statistical Mathematics*, 60(4):699–746, 2008.

[44] B. Sun, J. Feng, and K. Saenko. Return of frustratingly easy domain adaptation. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[45] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7167–7176, 2017.

[46] T. Yao, Y. Pan, C.-W. Ngo, H. Li, and T. Mei. Semi-supervised domain adaptation with subspace learning for visual recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2142–2150, 2015.

[47] K. Zhang, M. Gong, and B. Schölkopf. Multi-source domain adaptation: A causal view. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[48] K. Zhang, M. Gong, P. Stojanov, B. Huang, and C. Glymour. Domain adaptation as a problem of inference on graphical models. *arXiv preprint arXiv:2002.03278*, 2020.

[49] K. Zhang, B. Schölkopf, K. Muandet, and Z. Wang. Domain adaptation under target and conditional shift. In *International Conference on Machine Learning*, pages 819–827, 2013.

[50] H. Zhao, R. T. Des Combes, K. Zhang, and G. Gordon. On learning invariant representations for domain adaptation. In *International Conference on Machine Learning*, pages 7523–7532, 2019.

[51] H. Zhao, S. Zhang, G. Wu, J. M. Moura, J. P. Costeira, and G. J. Gordon. Adversarial multiple source domain adaptation. In *Advances in Neural Information Processing Systems*, pages 8559–8570, 2018.

A Proof of Theorem 1

The presented bound is an immediate consequence of two theorems of Ben-David [4] and Blitzer [6], which we present here as lemmas:

Lemma 1. *(Lemma 4 in [6]) Let $\mathcal{H}$ be a hypothesis class with VC-dimension d. For each $j \in \{1, 2, ..., M\}$, consider a labeled set of n/M samples drawn from the source domain $\mathcal{S}_j$. For any $h \in \mathcal{H}$ and any $\alpha \in \Delta$, with probability at least $1 - \delta$ over the choice of samples,*

$$|\widehat{\varepsilon}_\alpha(h) - \varepsilon_\alpha(h)| \leq 2\sqrt{\frac{M(2d\log(2(n+1)) + \log(\frac{4}{\delta}))}{n} \sum_{j=1}^{M} \alpha_j^2}. \tag{17}$$

Lemma 2. *(Theorem 2 in [4]) Let $\mathcal{H}$ be a hypothesis class with VC-dimension d. Consider n unlabeled samples drawn from each of the two domains $\mathcal{S}$ (source) and $\mathcal{T}$ (target). Then, for every $h \in \mathcal{H}$, with probability at least $1 - \delta$ over the choice of samples,*

$$\varepsilon_\mathcal{T}(h) \leq \varepsilon_\mathcal{S}(h) + \frac{1}{2}\widehat{d}_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S}, \mathcal{T}) + 2\sqrt{\frac{2d\log(2n) + \log(\frac{2}{\delta})}{n}} + \lambda, \tag{18}$$

where $\lambda = \min_{h\in\mathcal{H}} \varepsilon_\mathcal{S}(h) + \varepsilon_\mathcal{T}(h)$.

Proving our result is now straightforward. We can apply Lemma 2 taking $\mathcal{S}_\alpha = \sum_{j=1}^{M} \alpha_j \mathcal{S}_j$ as the source domain. Note that $\varepsilon_{S_\alpha}(h) = \sum_{j=1}^{M} \alpha_j \varepsilon_{S_j}(h)$ for any $h \in \mathcal{H}$, and therefore $\lambda_\alpha = \min_{h\in\mathcal{H}} \sum_{j=1}^{M} \alpha_j \varepsilon_{\mathcal{S}_j}(h) + \varepsilon_\mathcal{T}(h)$. Combining the obtained bound with Lemma 1 through the union bound yields the desired result. □

B Model overview

To enhance clarity, a detailed schematic of our model and loss terms is shown in Figure 2. The three main blocks are the feature extractor $g(\cdot, \theta_g)$, the classifier $h(\cdot, \theta_h)$, and the domain discriminator $d(\cdot, \theta_d)$. It is worth noting the usage of gradient reversal layers [16] at the input of the domain discriminator and before the α that is used in the discriminator loss $\mathcal{L}_{\text{disc}}$. This layer is the identity function in the forward pass but reverses the gradient in the backward pass, by multiplying it by a negative constant (here, -1 is used). Hence, the usage of this layer allows the model to be trained using standard backpropagation and gradient descent (or any of its variants) over all model parameters.

C Experiments – further results and details

C.1 Label distributions

As remarked throughout this work and formally discussed and proven in [50], having different marginal label distributions across source and target domains poses difficulties to the DA task and,

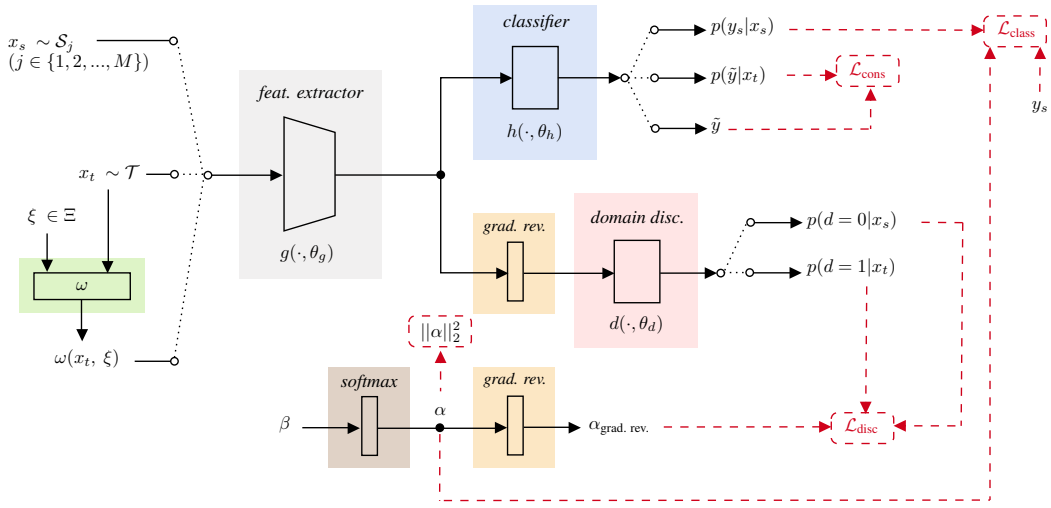
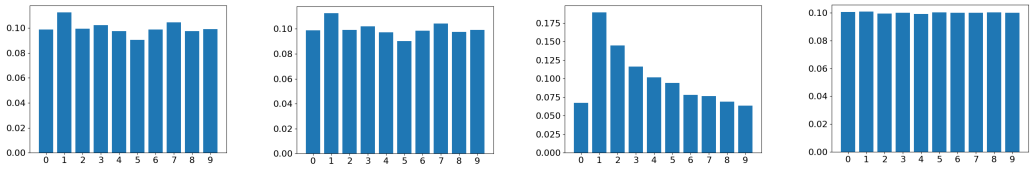


Figure 2: Block diagram representing the proposed model (MODA-FM).



(a) MNIST (b) MNIST-M (c) SVHN (d) SynthDigits

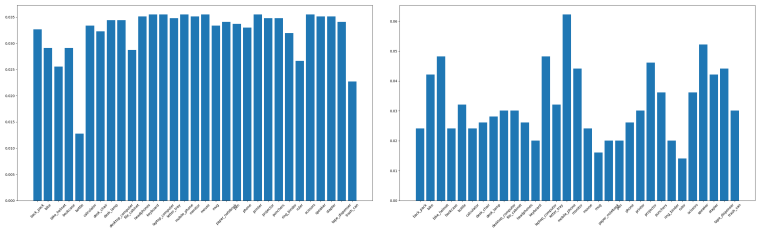
Figure 3: Label distributions in the digits datasets.

under this scenario, domain-invariant representations can increase the error on the target domain. Since we claim that the adopted consistency regularization can help mitigate this issue, it is relevant to observe the distributions of labels across domains in the datasets used in this work, which are shown in Figures 3, 4 and 5. We also provide the pairwise Jensen-Shannon distances for the label distributions of all domains in Tables C.1, C.2 and C.3. The Jensen-Shannon distance is convenient since it is a metric (so it is symmetric, non-negative and zero if and only if the two distributions coincide) and upper bounded by $\sqrt{\log(2)}$ (≈ 0.8326). Therefore, it provides a comprehensive measure to evaluate the dissimilarity between two distributions.

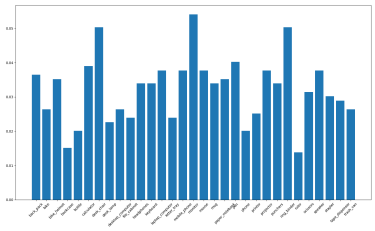
In the digits datasets (Figure 3, Table C.1), MNIST and MNIST-M have very similar label distributions and SynthDigits has an almost uniform label distribution. SVHN has the most skewed distribution, which is radically different from the remaining. Thus, this is the most challenging domain from this perspective. In Office-31 (Figure 4, Table C.2), label distributions across domains differ significantly. In the Amazon Reviews dataset (Figure 5, Table C.3), the label distribution is almost uniform for all domains.

Table C.1: Jensen-Shannon distances between the label distributions of each pair of digits domains. On each column, the largest distance is in bold and the smallest is underlined.

	MNIST	MNIST-M	SVHN	SynthDigits
MNIST	0	$\underline{2.73 \cdot 10^{-4}}$	$\underline{1.17 \cdot 10^{-1}}$	$\underline{1.83 \cdot 10^{-2}}$
MNIST-M	$\underline{2.73 \cdot 10^{-4}}$	0	$\underline{1.17 \cdot 10^{-1}}$	$1.84 \cdot 10^{-2}$
SVHN	$\mathbf{1.17 \cdot 10^{-1}}$	$\mathbf{1.17 \cdot 10^{-1}}$	0	$\mathbf{1.26 \cdot 10^{-1}}$
SynthDigits	$1.83 \cdot 10^{-2}$	$1.84 \cdot 10^{-2}$	$\mathbf{1.26 \cdot 10^{-1}}$	0
Average	$4.51 \cdot 10^{-2}$	$4.52 \cdot 10^{-2}$	$1.20 \cdot 10^{-1}$	$5.42 \cdot 10^{-2}$



(a) Amazon (b) DSLR

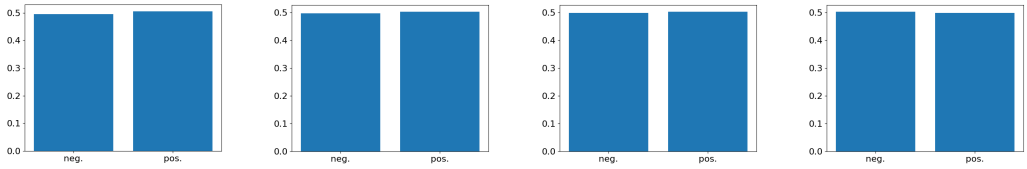


(c) Webcam

Figure 4: Label distributions in Office-31.

Table C.2: Jensen-Shannon distances between the label distributions of each pair of domains in Office-31. On each column, the largest distance is in bold.

	Amazon	DSLR	Webcam
Amazon	0	$1.33 \cdot 10^{-1}$	$9.76 \cdot 10^{-2}$
DSLR	$\mathbf{1.33 \cdot 10^{-1}}$	0	$\mathbf{1.45 \cdot 10^{-1}}$
Webcam	$9.76 \cdot 10^{-2}$	$\mathbf{1.45 \cdot 10^{-1}}$	0
Average	$1.15 \cdot 10^{-1}$	$1.39 \cdot 10^{-1}$	$1.21 \cdot 10^{-1}$

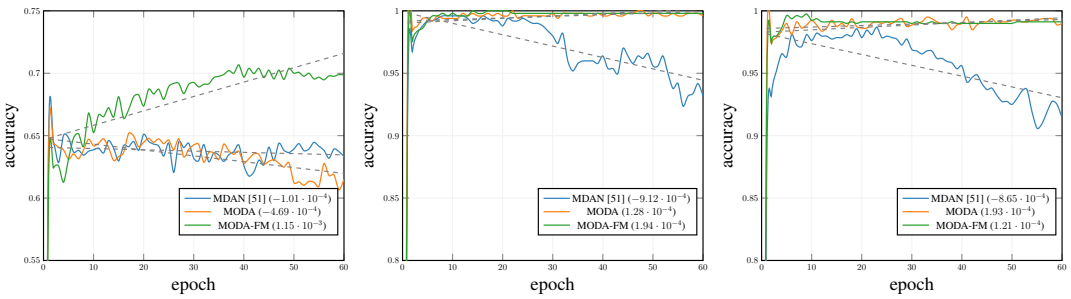


(a) Books (b) DVD (c) Electronics (d) Kitchen

Figure 5: Label distributions in Amazon Reviews.

Table C.3: Jensen-Shannon distances between the label distributions of each pair of domains in Amazon Reviews. On each column, the largest distance is in bold and the smallest is underlined.

	Books	DVD	Electronics	Kitchen
Books	0	$1.67 \cdot 10^{-3}$	$1.93 \cdot 10^{-3}$	$\mathbf{5.09 \cdot 10^{-3}}$
DVD	$\underline{1.67 \cdot 10^{-3}}$	0	$\underline{2.53 \cdot 10^{-4}}$	$3.42 \cdot 10^{-3}$
Electronics	$1.93 \cdot 10^{-3}$	$\underline{2.53 \cdot 10^{-4}}$	0	$\underline{3.17 \cdot 10^{-3}}$
Kitchen	$\mathbf{5.09 \cdot 10^{-3}}$	$\mathbf{3.42 \cdot 10^{-3}}$	$\mathbf{3.17 \cdot 10^{-3}}$	0
Average	$2.90 \cdot 10^{-3}$	$1.78 \cdot 10^{-3}$	$1.78 \cdot 10^{-3}$	$3.89 \cdot 10^{-3}$



(a) Amazon (b) DSLR (c) Webcam

Figure 6: Test accuracy along 60 training epochs for our model and two baselines in Office-31. The tendency line for each curve is also shown (dashed lines) and the respective slope is indicated in brackets in each plot legend. The domain indicated below each plot is the target.

C.2 Effect of over-training

When the label distributions differ across domains, training the feature extractor and domain discriminator for a large number of iterations tends to lead to an increased target error. This is a direct effect of the curse of domain-invariant representations and it has been verified experimentally in [50].

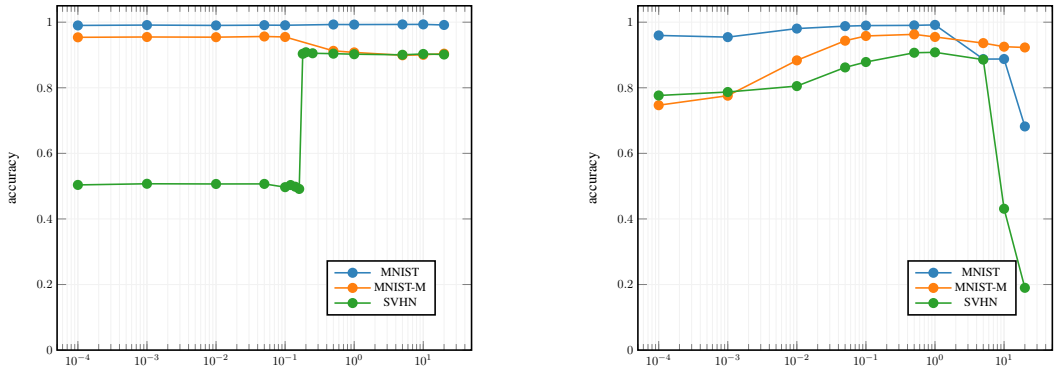
In this experiment, we want to evaluate if the increased robustness of the feature extractor provided by the adopted consistency regularization helps to mitigate this issue. For this purpose, we use Office-31, as this is the dataset where the label distributions are most different across domains (see section C.1). We train our model and two baselines (MDAN [51] and MODA) for 60 epochs and we observe the evolution of the model accuracy on the target data. The plots are shown in Figure 6. As we see there, in MODA-FM the accuracy keeps stable after reaching the maximum in all domains. Contrarily, in MODA and especially in MDAN, the accuracy tends to decay after reaching the maximum. These observations strongly suggest that MODA-FM succeeds on mitigating the curse of domain-invariant representations. In MODA the problem is less pronounced than in MDAN probably because the latter will continue optimizing itself until it can produce domain-invariant representations for all source domains, whereas the former will simply ignore the hardest source domains by assigning them a low weight (possibly even zero).

C.3 Hyperparameter sensitivity analysis

Our model comprises three main hyperparameters: μ_d, μ_s and μ_c. The hyperparameter μ_d controls the relative weight of the domain discriminator loss and is present in every work on adversarial DA, thus its effect has already been studied extensively. For this reason, we focus on the effect of μ_s and μ_c. We use the digits datasets and evaluate the accuracy on each target domain while varying either μ_s or μ_c and keeping the other one constant at the optimal value.

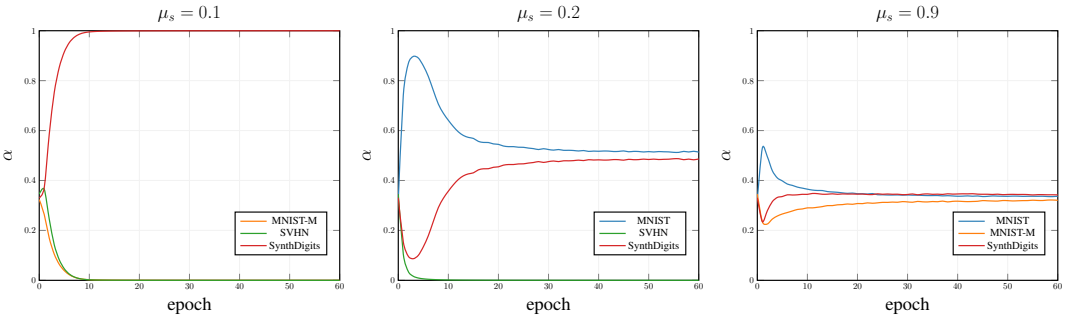
A sufficiently large value of μ_s forces α to converge to a vector with all components equal to $1/M$, weighting all source domains equally. Setting μ_s close to zero corresponds to the most optimistic scenario: the sparsity penalization is dropped and so, after sufficiently many training iterations, α would converge to a one-hot vector, choosing the source domain with the minimum difference of classification and discrimination losses ($\mathcal{L}_{class} - \mu_d \mathcal{L}_{disc}$). Figure 7a shows that, for MNIST, similar results are obtained with any of those strategies. The fact that digits classification in MNIST is significantly easier than in any of the remaining datasets likely explains this behavior. Domain adaptation for MNIST-M is slightly favored by smaller values of μ_s, meaning that a more optimistic approach works better here. For SVHN, the opposite holds and the results are very poor when low sparsity regularization is employed. There is even a strong discontinuity in the accuracy plot for this domain around $\mu_s \approx 0.2$. This discontinuity divides situations where α collapses into the easiest source domain ($\mu_s < 0.2$) and those where it does not collapse ($\mu_s > 0.2$). We elaborate more on this in section C.4. In all cases, we observe that a $\mu_s \in [0.2, 0.6]$ provides near-optimal results.

The effect of varying μ_c is plotted in Figure 7b. Values of μ_c close to zero drop the consistency regularization and, therefore, we obtain results close to those of MODA. Significant performance gains are obtained for $\mu_c > 0.01$ and the optimal is reached for $\mu_c \in [0.4, 1.0]$ for all domains.



(a) μ_s (b) μ_c

Figure 7: Test accuracy as a function of hyperparameters μ_s (a) and μ_c (b) using the digits datasets. The domain corresponding to each line is the target.



(a) MNIST (b) MNIST-M (c) SVHN

Figure 8: Source weights α for each domain along 60 training epochs in the digits datasets. The target domain and the value of μ_s that was used are indicated below and above each plot, respectively.

C.4 Evolution of the source weights

Here, we study the evolution of the source weights α along training. The behavior is determined by the evolution of the classification and discrimination losses of each source domain and also by the choice of the hyperparameter μ_s. The effect of varying μ_s on the model performance was analyzed in section C.3. Now, we are interested in observing the dynamics of α along the training epochs, for different source domains, using the corresponding optimal μ_s. We use the digits datasets for this purpose and we present the results in Figure 8.

For MNIST, where the lowest μ_s is used, we observe that the mixture coefficients rapidly collapse into the easiest source domain (SynthDigits). Nonetheless, in section C.3 we have observed that the target accuracy for MNIST was almost insensitive to the choice of μ_s, meaning that using only the data from the easiest source domain or weighting all source domains equally would produce identical results. When we take MNIST-M and SVHN as target domains, with a slightly increased μ_s, a different behavior occurs. Early in the training process, the weight for the easiest source domain (MNIST) increases rapidly, following the fast decrease in the corresponding loss. Later, as the loss for the easiest source domain plateaus and the remaining keep decreasing, the weights for the remaining active source domains start to increase. This behavior explains the discontinuity we have observed in the plot of target accuracy vs. μ_s for SVHN (Figure 7a): if μ_s is sufficiently small to allow α to rapidly collapse into MNIST, the data of the remaining source domains is simply discarded for the remaining of the training process and the corresponding weights will never increase.

Table C.4: Values and search ranges for all hyperparameters in all experiments.

Hyperparameter	Digits	Office-31	Amazon Reviews
μ_d (8)	$[10^{-4}, 10^{0}]$	$[10^{-4}, 10^{0}]$	$[10^{-4}, 10^{0}]$
μ_s (8)	$[10^{-5}, 10^{0}]$	10^{-2}	$[10^{-6}, 10^{-1}]$
μ_c (16)	$[10^{-2}, 10^{0}]$	$[10^{-2}, 10^{0}]$	$[10^{-2}, 10^{0}]$
τ (15)	0.9	0.9	0.9
optimizer	AdaDelta	AdaDelta	AdaDelta
no. epochs	$\{10, 20, \ldots, 60\}$	$\{10, 15, \ldots, 60\}$	$\{5, 10, \ldots, 40\}$
learning rate	0.1	0.1	1.0
batch size	8	8	20
RandAugment: n	2	2	n.a.
RandAugment: m_{min}	3	3	n.a.
RandAugment: m_{max}	10	10	n.a.
dropout: p_{min}	n.a.	n.a.	0.2
dropout: p_{max}	n.a.	n.a.	0.8

C.5 Network architectures

The following notation is used to designate network layers: `Conv(n, k)` – 2-D convolutional layer with `n` output channels, square kernels with size `k`, and unit stride, followed by a rectified linear activation; `MaxPool(k)` – 2-D max-pooling over non-overlapping regions of `k` $\times$ `k` pixels; `AdaptAvgPool(k)` – 2-D adaptive average pooling where the output size is `k` $\times$ `k` pixels. `FC(n)` – fully-connected layer with `n` output neurons, followed by a rectified linear activation except for output layers; `Dropout(p)` – dropout layer where `p` is the probability of zeroing an element. A gradient reversal layer [16] is present at the input of the domain discriminator in all models. All classifiers and domain discriminators are followed by a softmax layer that maps the output to the corresponding class probabilities.

Digits classification Input images have shape $3 \times 32 \times 32$. Feature extractor: `Conv(64, 3)` $\rightarrow$ `MaxPool(2)` $\rightarrow$ `Conv(128, 3)` $\rightarrow$ `MaxPool(2)` $\rightarrow$ `Conv(256, 3)`. Task classifier: `MaxPool(2)` $\rightarrow$ `Conv(256, 3)` $\rightarrow$ `FC(2048)` $\rightarrow$ `FC(1024)` $\rightarrow$ `FC(10)`. Domain discriminator: `MaxPool(2)` $\rightarrow$ `FC(2048)` $\rightarrow$ `FC(2048)` $\rightarrow$ `FC(1024)` $\rightarrow$ `FC(2)`.

Object classification (Office-31 dataset) Input images have shape $3 \times 256 \times 256$. Feature extractor: ResNet-50 up to stage 3. Task classifier / domain discriminator: ResNet-50 stage 4 $\rightarrow$ `AdaptAvgPool(1)` $\rightarrow$ `FC(#classes)`, where `#classes` $= 31$ for the task classifier and `#classes` $= 2$ for the domain discriminator.

Sentiment analysis (Amazon Reviews dataset) Input features have dimension 5000. Feature extractor: `Dropout(p)` $\rightarrow$ `FC(1000)` $\rightarrow$ `Dropout(p)` $\rightarrow$ `FC(500)` $\rightarrow$ `Dropout(p)` $\rightarrow$ `FC(100)` $\rightarrow$ `Dropout(p)`, where `p` $= 0$ except for producing the augmented samples used in the consistency regularization. Task classifier / domain discriminator: `FC(2)`.

C.6 Choice of hyperparameters

Table C.4 presents the values assigned to each hyperparameter. For some hyperparameters, a set or range of values is presented instead of a single value, meaning that the value for that hyperparameter was selected via cross-validation on the given set/range. This cross-validation consisted on 20 iterations of random search for each experiment, where no data from the target domain was used and each source domain was taken as target in turn. The hyperparameter combination that yielded the best average result was chosen.

C.7 Image transformations

The list of image transformations used in RandAugment is provided in Table C.5. The cutout transformation is the first transformation to be applied to every image. This transformation occludes a randomly located square region with a length equal to 30% of the image length. The remaining n transformations are chosen at random from the provided list. All transformations were normalized

such that a transformation of magnitude $m = 0$ corresponds to the identity (i.e., unchanged image) and a magnitude $m = 30$ (maximum admissible value) corresponds to the maximum distortion. We implemented the image transformations using the Python Imaging Library (PIL).

Table C.5: RandAugment image transformations and respective ranges.

Transformation	Range	Transformation	Range
AutoContrast	$\{0, 1\}$	Rotate	$[-30°, 30°]$
Brightness	$[0.1, 1.9]$	Sharpness	$[0.1, 1.9]$
Color	$[0.1, 1.9]$	ShearX	$[0, 0.3]$
Contrast	$[0.1, 1.9]$	ShearY	$[0, 0.3]$
Equalize	$\{0, 1\}$	Solarize	$[0, 255]$
FlipX [2]	$\{0, 1\}$	TranslateX	$[-0.45, 0.45]$
Invert	$\{0, 1\}$	TranslateY	$[-0.45, 0.45]$
Posterize	$[4, 8]$		

C.8 Sample images

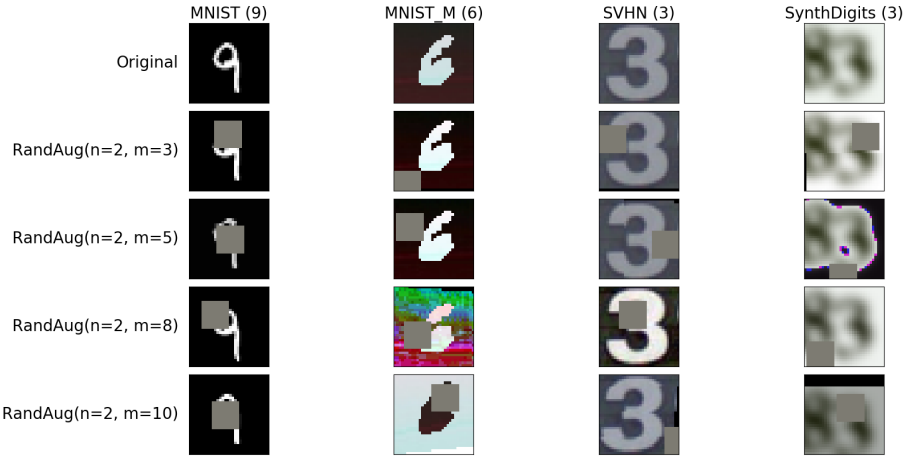


Figure 9: Sample original and transformed images from the digits datasets. The ground-truth label is in brackets.

[2]Excluded in the digits experiment since it is not label-preserving.

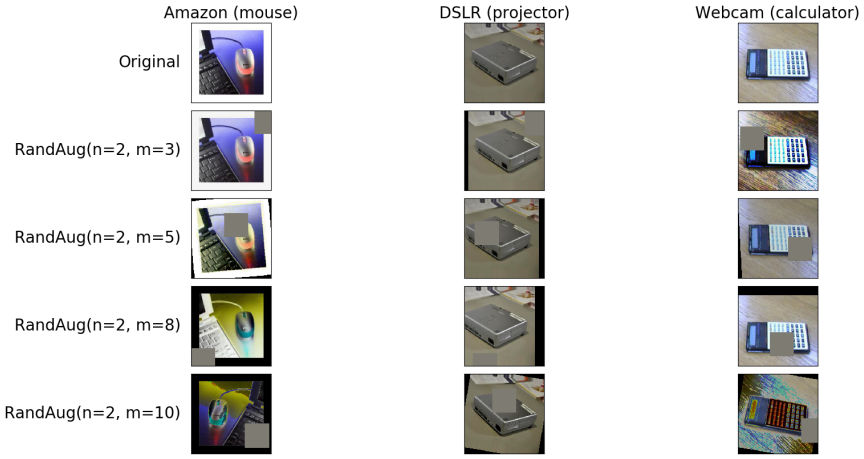


Figure 10: Sample original and transformed images from each domain in the Office-31 dataset. The ground-truth label is in brackets.