Boosting Domain Adaptation by Discovering Latent Domains

Massimiliano Mancini[1,2], Lorenzo Porzi[3], Samuel Rota Bulò[3], Barbara Caputo[1,4], Elisa Ricci[2,5]
[1]Sapienza University of Rome, [2]Fondazione Bruno Kessler, [3]Mapillary Research,
[4]Italian Institute of Technology, [5]University of Trento
{mancini,caputo}@diag.uniroma1.it, {lorenzo,samuel}@mapillary.com, eliricci@fbk.eu

Abstract

Current Domain Adaptation (DA) methods based on deep architectures assume that the source samples arise from a single distribution. However, in practice most datasets can be regarded as mixtures of multiple domains. In these cases exploiting single-source DA methods for learning target classifiers may lead to sub-optimal, if not poor, results. In addition, in many applications it is difficult to manually provide the domain labels for all source data points, i.e. *latent domains should be automatically discovered. This paper introduces a novel Convolutional Neural Network (CNN) architecture which (i) automatically discovers latent domains in visual datasets and (ii) exploits this information to learn robust target classifiers. Our approach is based on the introduction of two main components, which can be embedded into any existing CNN architecture: (i) a side branch that automatically computes the assignment of a source sample to a latent domain and (ii) novel layers that exploit domain membership information to appropriately align the distribution of the CNN internal feature representations to a reference distribution. We test our approach on publicly-available datasets, showing that it outperforms state-of-the-art multi-source DA methods by a large margin.*

1. Introduction

The problem that trained models perform poorly when tested on data from a different distribution is commonly referred to as *domain shift*. This issue is especially relevant in computer vision, as visual data is characterized by large appearance variability, *e.g.* due to differences in resolution, changes in camera pose, occlusions and illumination variations. To address this problem, several transfer learning and domain adaptation approaches have been proposed in the last decade [35].

Domain Adaptation (DA) methods are specifically designed to transfer knowledge from a *source* domain to the domain of interest, *i.e.* the *target* domain, in the form of learned models or invariant feature representations. The problem has been widely studied and both theoretical results [3, 33] and several shallow [10, 15, 17, 22, 30] and deep learning algorithms have been developed [5, 6, 12, 14, 31, 32, 40]. While deep neural networks tend to produce more transferable and domain-invariant features, previous works [8] have shown that the domain shift is only alleviated but not removed.

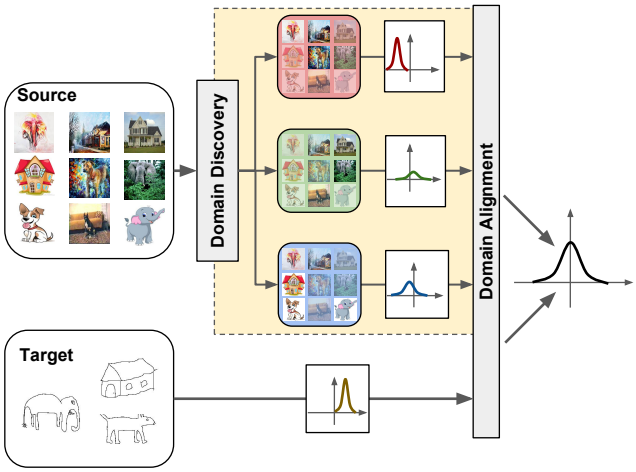

Figure 1: The idea behind our framework. We propose a novel deep architecture which, given a set of images, automatically discover multiple latent source domains and use this information to align the distributions of the internal CNN feature representations of sources and target domains for the purpose of domain adaptation. Image better seen at magnification.

Most works on DA focus on a single-source and single-target scenario. However, in many computer vision applications labeled training data is often generated from multiple distributions, *i.e.* there are multiple source domains. Examples of multi-source DA problems arise when the source set corresponds to images taken with different cameras, collected from the web or associated to multiple points of views. In these cases, a naive application of single-source DA algorithms would not suffice, leading to poor results. Therefore, in the past several research efforts have been devoted to develop DA methods operating on multiple sources

[9, 33, 39]. These approaches assume that the different source domains are known. A more challenging problem arises when labeled training data correspond to latent source domains, *i.e*. we can make a reasonable estimate on the number of source domains available, but we have no information, or only partial, about domain labels. To address this problem, known in the literature as *latent domain discovery*, previous works have proposed methods which simultaneously discover hidden source domains and use them to learn the target classification models [16, 21, 41].

This paper introduces the first deep approach able to automatically discover latent source domains in multi-source domain adaptation settings. Our method is inspired by the recent works [6, 7], which revisit Batch Normalization layers [23] for the purpose of domain adaptation, introducing specific Domain Alignment layers (DA-layers). The main idea behind DA-layers is to cope with domain shift by aligning representations of source and target distributions to a reference Gaussian distribution. Our approach develops from the same intuition. However, to address the additional challenges of discovering and handling multiple latent domains, we propose a novel architecture which is able to (i) learn a set of assignment variables which associate source samples to a latent domain and (ii) exploit this information for aligning the distributions of the internal CNN feature representations and learn a robust target classifier (Fig.2). Our experimental evaluation shows that the proposed approach alleviates the domain discrepancy and outperforms previous multi-source DA techniques on popular benchmarks, such as Office-31 [36] and Office-Caltech [17].

2. Related Work

DA methods with hand-crafted features. Earlier DA approaches operate on hand-crafted features and attempt to reduce the discrepancy between the source and the target domains by adopting different strategies. For instance, instance-based methods [15, 22, 42] develop from the idea of learning classification/regression models by re-weighting source samples according to their similarity with the target data. A different strategy is exploited by feature-based methods, coping with domain shift by learning a common subspace for source and target data such as to obtain domain-invariant representations [10, 17, 30]. Parameter-based methods [43] address the domain shift problem by discovering a set of shared weights between the source and the target models. However, they usually require labeled target data which is not always available.

While most earlier DA approaches focus on a single-source and single-target setting, some works have considered the related problem of learning classification models when the training data spans multiple domains [9, 33, 39]. The common idea behind these methods is that when source data arises from multiple distributions, adopting a single source classifier is suboptimal and improved performance can be obtained leveraging information about multiple domains. However, these methods assume that the domain labels for all source samples are known in advance. In practice, in many applications the information about domains is hidden and latent domains must be discovered into the large training set. Few works have considered this problem in the literature. Hoffman *et al*. [21] address this task by modeling domains as Gaussian distributions in the feature space and by estimating the membership of each training sample to a source domain using an iterative approach. Gong *et al*. [16] discover latent domains by devising a non-parametric approach which aims at simultaneously achieving maximum distinctiveness among domains and ensuring that strong discriminative models are learned for each latent domain. In [41] domains are modeled as manifolds and source images representations are learned decoupling information about semantic category and domain. By exploiting these representations the domain assignment labels are inferred using a mutual information based clustering method.

Deep Domain Adaptation. Most recent works on DA consider deep architectures and robust domain-invariant features are learned using either supervised neural networks [5, 6, 12, 14, 31, 40], deep autoencoders [44] or generative adversarial networks [4, 37]. For instance, some methods attempt to align source and target features by minimizing Maximum Mean Discrepancy [31, 32, 38]. Other approaches operate in a domain-adversarial setting, *i.e*. learn domain-agnostic representations by maximizing a domain confusion loss [12, 40]. Domain separation networks are proposed in [5], where feature representations are learned by decoupling the domain-specific component from a shared one. DA-layers are described in [7] which, embedded into an arbitrary CNN architecture, are able to align source and target representation distributions.

While recent deep DA methods significantly outperform approaches based on hand-crafted features, they only consider single-source, single-target settings. To our knowledge, this is the first work proposing a deep architecture for discovering latent source domains and exploiting them for improving classification performance on target data.

3. Method

3.1. Problem Formulation and Notation

In this paper we are interested in predicting labels from an output space $\mathcal{Y}$ (*e.g.* object or scene categories), given elements of an input space $\mathcal{X}$ (*e.g.* images). We further assume that our data belongs to one of several domains: the k *source* domains, characterized by unknown probability distributions $p^{s_1}_{\mathtt{xy}}, \dots, p^{s_\mathsf{k}}_{\mathtt{xy}}$ defined over $\mathcal{X} \times \mathcal{Y}$, and the *target* domain, characterized by $p^{t}_{\mathtt{xy}}$. Note that the number of source domains k is not necessarily known a-

priori, and is left as an hyper-parameter of our method. During training we are given a set of labeled samples from the source domains, and a set of unlabeled samples from the target domain, while we have partial or no access to domain assignment information for the source samples. More formally, we model the source data as a set $\mathcal{S} = \{(x_1^s, y_1^s), \dots, (x_{\mathsf{n}}^s, y_{\mathsf{n}}^s)\}$ of i.i.d. observations from a mixture distribution $p_{\mathsf{xy}}^s = \sum_{i=1}^{\mathsf{k}} \pi_{s_i} p_{\mathsf{xy}}^{s_i}$, where π_{s_i} is the probability of sampling from a source domain s_i. Similarly, the target samples $\mathcal{T} = \{x_1^t, \dots, x_{\mathsf{m}}^t\}$ are i.i.d. observations from the marginal p_{x}^t. Furthermore, we denote by $x_{\mathcal{S}} = \{x_1^s, \dots, x_{\mathsf{n}}^s\}$ and $y_{\mathcal{S}} = \{y_1^s, \dots, y_{\mathsf{n}}^s\}$, the source data and label sets, respectively. We assume to know the domain label for a (possibly empty) subset $\hat{\mathcal{S}} \subset \mathcal{S}$ of source data samples and we denote by $d_{\hat{\mathcal{S}}}$ the domain labels in $\{s_1 \dots, s_{\mathsf{k}}\}$ of the sample points in $x_{\hat{\mathcal{S}}}$. The set of domains labels, including target domain, is given by $\mathcal{D} = \{s_1, \dots, s_{\mathsf{k}}, t\}$.

Our main goal is to learn a predictor that is able to classify samples from the target domain. When tackling this problem we have to deal with three main difficulties: (i) the distributions of source(s) and target can be drastically different, making it hard to apply a classifier learned on one domain to the others, (ii) we lack direct observation of target labels, and (iii) the assignment of each source sample to its domain is unknown, or known for a very limited number of samples only.

Several previous works [5, 6, 12, 14, 31, 40] have tackled the related problem of domain adaptation in the context of deep neural networks, dealing with (i) and (ii) in the case in which all source data comes from a single domain. In particular, some recent works have demonstrated a simple yet effective approach based on the replacement of standard Batch Normalization layers with specific *Domain Alignment layers* [6, 7]. These layers aim to reduce internal domain shift at different levels within the network by re-normalizing features in a domain-dependent way, matching their distributions to a pre-determined one. In the following sections we show how the same idea can be revisited to naturally tackle the case of multiple, unknown source domains. In particular, we propose a novel Multi-domain DA layer (mDA-layer), detailed in Section 3.2, which is able to re-normalize the multi-modal feature distributions encountered in our setting. To do this, our mDA-layers exploit a side-output branch we attach to the main network (see Section 3.3), which predicts domain assignment probabilities for each input sample. Finally, in Section 3.4 we show how the predicted domain probabilities can be exploited, together with the unlabeled target samples, to construct a prior distribution over the network's parameters which is then used to define the training objective for our network.

3.2. Multi-domain DA-layers

DA-layers [6, 7, 28] are motivated by the observation that, in general, activations within a neural network follow domain-dependent distributions. As a way to reduce domain shift, the activations are thus normalized in a domain-specific way, shifting them according to a parameterized transformation in order to match their first and second order moments to those of a reference distribution, which is generally chosen to be normal with zero mean and unit standard deviation. While previous works only considered settings with two domains, *i.e.* source and target, the basic idea can in fact be applied to any number of domains, as long as the domain membership of each sample is known. Specifically, denoting as q_{x}^d the distribution of activations for a given feature channel and domain d, an input $x^d \sim q_{\mathsf{x}}^d$ to the DA-layer can be normalized according to

$$\mathrm{DA}(x^d; \mu_d, \sigma_d) = \frac{x^d - \mu_d}{\sqrt{\sigma_d^2 + \epsilon}},$$

where $\mu_d = \mathrm{E}_{x \sim q_{\mathsf{x}}^d}[x]$, $\sigma_d^2 = \mathrm{Var}_{x \sim q_{\mathsf{x}}^d}[x]$ and $\epsilon > 0$ is a small constant to avoid numerical issues. When the statistics μ_d and σ_d^2 are computed over the current batch, this equates in practice to applying standard Batch Normalization separately to the samples of each domain.

As mentioned above, this approach requires full domain knowledge, as, for each d, μ_d and σ_d^2 need to be calculated on the specific samples belonging to d. In our case, however, while the target is clearly distinct from the source, we do not know which specific source domain most or even all of the source samples belong to. To tackle this issue, we propose to model the layer's input distribution as a mixture of Gaussians, with one component for each domain. Specifically, we define a global input distribution $q_{\mathsf{x}} = \sum_d \pi_d q_{\mathsf{x}}^d$, where π_d is the probability of sampling from domain d, and $q_{\mathsf{x}}^d = \mathcal{N}(\mu_d, \sigma_d^2)$ is the domain-specific distribution for d: a normal with mean μ_d and variance σ_d^2. Given a batch of samples $\mathcal{B} = \{x_i\}_{i=1}^{\mathsf{b}}$, a maximum likelihood estimate of the parameters μ_d and σ_d^2 is given by

$$\mu_d = \sum_{i=1}^{\mathsf{b}} \alpha_{i,d} x_i, \quad \sigma_d^2 = \sum_{i=1}^{\mathsf{b}} \alpha_{i,d} (x_i - \mu_d)^2, \tag{1}$$

where

$$\alpha_{i,d} = \frac{q_{\mathsf{d}|\mathsf{x}}(d \mid x_i)}{\sum_{i=1}^{\mathsf{b}} q_{\mathsf{d}|\mathsf{x}}(d \mid x_i)}, \tag{2}$$

and $q_{\mathsf{d}|\mathsf{x}}(d \mid x_i)$ is the conditional probability of x_i belonging to d, given x_i. Clearly, the value of $q_{\mathsf{d}|\mathsf{x}}$ is known for all samples for which we have domain information. In all other cases, the missing domain assignment probabilities are inferred from data, using the *domain prediction* network branch which will be detailed in Section 3.3. Thus, from the

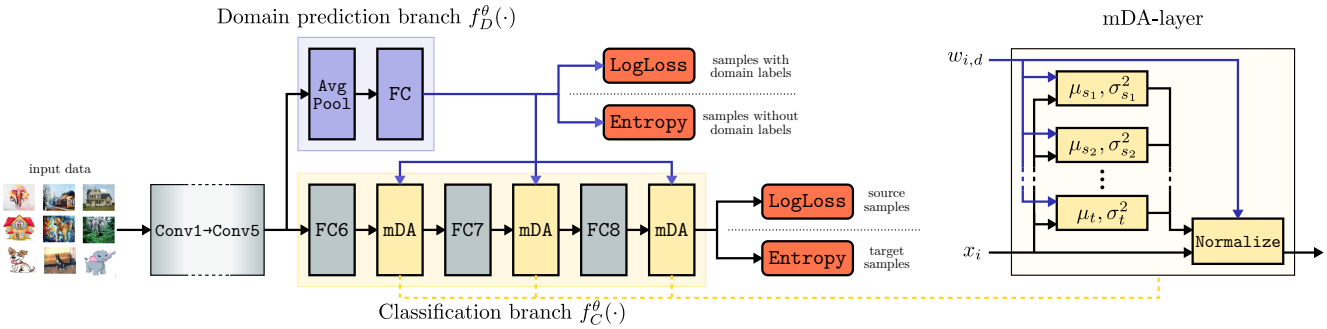

Figure 2: Schematic representation of our method applied to the AlexNet architecture (left) and of an mDA-layer (right).

perspective of the alignment layer, these probabilities become an additional input, which we denote as $w_{i,d}$ for the predicted probability of x_i belonging to d.

By substituting $w_{i,d}$ for $q_{\mathtt{d|x}}(d \mid x_i)$ in (1) and (2), we obtain a new set of empirical estimates for the mixture parameters, which we denote as $\hat{\mu}_d$ and $\hat{\sigma}^2_d$. These parameters are used to normalize the layer's inputs according to

$$\mathrm{mDA}(x_i, \boldsymbol{w}_i; \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}}) = \sum_{d \in \mathcal{D}} w_{i,d} \frac{x_i - \hat{\mu}_d}{\sqrt{\hat{\sigma}^2_d + \epsilon}}, \qquad (3)$$

where $\boldsymbol{w}_i = \{w_{i,d}\}_{d \in \mathcal{D}}$, $\hat{\boldsymbol{\mu}} = \{\hat{\mu}_d\}_{d \in \mathcal{D}}$ and $\hat{\boldsymbol{\sigma}} = \{\hat{\sigma}^2_d\}_{d \in \mathcal{D}}$. As in previous works [6, 7, 23], during back-propagation we calculate the derivatives through the statistics and weights, propagating the gradients to both the main input and the domain assignment probabilities.

3.3. Domain prediction

As explained in the previous Section 3.2, our mDA-layers take as input a set of domain assignment probabilities for each input sample, which need to be predicted. While different mDA-layers in a network have in general different input distributions, the assignment of sample points to domains should be coherent across them. Specifically, sample points at different mDA-layers corresponding to a single input element to the network should share the same probabilities. As a practical example, in the typical case in which mDA-layers are used in a CNN to normalize convolutional activations, the network would predict a single set of probabilities for each input image, which would then be given as input to all mDA-layers and broadcasted across all spatial locations and feature channels corresponding to that image. Following these consideration, we compute domain assignment probabilities using a distinct section of the network, which we call the *domain prediction* branch, while we refer to the main section of the network as the *classification* branch. The two branches share the bottom-most layers and parameters as depicted in Figure 2. The domain prediction branch is implemented as a minimal set of layers followed by a soft max operation with k outputs for the k latent source domains (more details follow in Section 4).

As the domain membership of target samples is always assumed to be known, we do not predict domain assignment probabilities for the target. Furthermore, for each sample x_i with known domain membership $\hat{d}$, we fix in each mDA-layer $w_{i,d} = 1$ if $d = \hat{d}$, otherwise $w_{i,d} = 0$.

We split the network into a domain prediction branch and classification branch at some low level layer. This choice is motivated by the observation [1] that features tend to become increasingly more domain invariant going deeper into the network, meaning that it becomes increasingly harder to compute a sample's domain as a function of deeper features. In fact, as pointed out in [6], this phenomenon is even more evident in networks that include Domain Alignment layers.

3.4. Training the network

We want to estimate $\theta \in \Theta$, which comprises all trainable parameters of the classification and domain prediction branches, while taking advantage of both labeled and unlabeled data. A main difficulty lies in the fact that, when employing a discriminative model, the unlabeled samples cannot be used to express the data likelihood. However, following the approach sketched in [6], we can exploit the unlabeled data to define a prior distribution over the network's parameters. By doing this, we define a posterior distribution over θ given all data and labels as follows:

$$\pi(\theta|\mathcal{S}, \mathcal{T}, \hat{\mathcal{S}}) \propto \pi(y_{\mathcal{S}}|x_{\mathcal{S}}, \theta) \cdot \pi(d_{\hat{\mathcal{S}}}|x_{\hat{\mathcal{S}}}, \theta)\pi(\theta|\mathcal{T})\pi(\theta|x_{\mathcal{S}\setminus\hat{\mathcal{S}}}), \qquad (4)$$

where for notational convenience we have omitted some dependences. By maximizing (4) over Θ we obtain a maximum-a-posterior estimate $\hat{\theta}$ for the parameters:

$$\hat{\theta} \in \arg\max_{\theta \in \Theta} \pi(\theta|\mathcal{S}, \mathcal{T}, \hat{\mathcal{S}}). \qquad (5)$$

The first term on the right hand side of (4) is the likelihood of θ w.r.t. the source dataset, and can be written as

$$\pi(y_{\mathcal{S}}|x_{\mathcal{S}}, \theta) = \prod_{i=1}^{\mathtt{n}} f_C^\theta(y_i^s; x_i^s) \qquad (6)$$

due to the i.i.d. assumption on the training samples. Here we denote by $f_C^\theta(y_i^s; x_i^s)$ the output of the *classification branch* of the network for a source sample, *i.e.* the predicted probability of x_i^s having class y_i^s, and, for convenience of notation, we omit the dependence of f_C^θ on the target samples induced by the mDA-layers. Similarly, the second term in (4) is the likelihood of θ w.r.t. the known domains:

$$\pi(d_{\hat{\mathcal{S}}}|x_{\hat{\mathcal{S}}}, \theta) = \prod_{x_i \in x_{\hat{\mathcal{S}}}} f_D^\theta(d_i; x_i),$$

where d_i is the domain corresponding to $x_i \in x_{\hat{\mathcal{S}}}$. In the previous equation, $f_D^\theta(d; x)$ denotes the output of the *domain prediction* branch for a sample x and domain d, *i.e.* the predicted probability of x belonging to d.

To define our prior $\pi(\theta|\mathcal{T})$ over the parameters, we exploit all available unlabeled data, biasing our classifier towards exhibiting low uncertainty on the unlabeled samples, similarly to [6]. However, in addition, we introduce a prior term $\pi(\theta|x_{\mathcal{S}\setminus\hat{\mathcal{S}}})$, which exploits source sample points with missing domain labels. Uncertainty when predicting class labels can be measured in terms of the empirical entropy

$$h_C(\theta|x_{\mathcal{S}}) = -\frac{1}{\mathsf{m}} \sum_{i=1}^{\mathsf{m}} \sum_{y \in \mathcal{Y}} f_C^\theta(y; x_i^t) \log f_C^\theta(y; x_i^t),$$

and similarly for the uncertainty when predicting domains.

$$h_D(\theta|x_{\mathcal{S}\setminus\hat{\mathcal{S}}}) = -\frac{1}{|x_{\mathcal{S}\setminus\hat{\mathcal{S}}}|} \sum_{x \in x_{\mathcal{S}\setminus\hat{\mathcal{S}}}} \sum_{i=1}^{\mathsf{k}} f_D^\theta(s_i; x) \log f_D^\theta(s_i; x).$$

Now, $\pi(\theta|\mathcal{T})$ can be obtained as the distribution with maximum entropy under the constraints $\int \pi(\theta|x_{\mathcal{S}}) h_C(\theta|x_{\mathcal{S}}) d\theta = \varepsilon_C$ and, similarly, $\pi(\theta|x_{\mathcal{S}\setminus\hat{\mathcal{S}}})$ can be regarded as a maximum entropy distribution under the constraint $\int \pi(\theta|x_{\mathcal{S}\setminus\hat{\mathcal{S}}}) h_D(\theta|x_{\mathcal{S}\setminus\hat{\mathcal{S}}}) d\theta = \varepsilon_D$, where $\varepsilon_C > 0$ and $\varepsilon_D > 0$ define the desired average uncertainties for class and domain predictions, respectively. These optimization problems can be shown to have solutions:

$$\pi(\theta|\mathcal{T}) \propto \exp(-\lambda_C h_C(\theta|\mathcal{T}))$$
$$\pi(\theta|x_{\mathcal{S}\setminus\hat{\mathcal{S}}}) \propto \exp(-\lambda_D h_D(\theta|x_{\mathcal{S}\setminus\hat{\mathcal{S}}})),$$

where λ_C and λ_D are the Lagrange multipliers corresponding to ε_C and ε_D, respectively.

In practice, the optimization in (5) can be replaced by the equivalent minimization of the negative logarithm of the likelihood, obtaining our loss function:

$$\begin{aligned} L(\theta) = &- \frac{1}{\mathsf{n}} \sum_{i=1}^{\mathsf{n}} \log f_C^\theta(y_i^s; x_i^s) \\ &- \lambda_t \frac{1}{|x_{\hat{\mathcal{S}}}|} \sum_{x_i \in x_{\hat{\mathcal{S}}}} \log f_D^\theta(d_i; x_i) \\ &- \lambda_C \frac{1}{\mathsf{m}} \sum_{i=1}^{\mathsf{m}} \sum_{y \in \mathcal{Y}} f_C^\theta(y; x_i^t) \log f_C^\theta(y; x_i^t) \\ &- \lambda_D \frac{1}{|x_{\mathcal{S}\setminus\hat{\mathcal{S}}}|} \sum_{x \in x_{\mathcal{S}\setminus\hat{\mathcal{S}}}} \sum_{i=1}^{\mathsf{k}} f_D^\theta(s_i; x) \log f_D^\theta(s_i; x). \end{aligned} \tag{7}$$

The four terms, balanced by the hyper-parameters λ_t, λ_C and λ_D, can be interpreted as two log-losses and two entropy losses applied to the classification and domain prediction branches of the network, respectively to samples with known and unknown labels. Interestingly, since the classification branch has a dependence on the domain prediction branch via the mDA-layers, by optimizing (7), the network learns to predict domain assignment probabilities that result in a low classification loss. In other words, the network is free to predict domain memberships that do not necessarily reflect the real ones, as long as this helps improving its classification performance.

4. Experiments

4.1. Experimental Setup

Datasets. In our evaluation we consider several common DA benchmarks: the combination of the USPS [11], MNIST [26] and MNIST-m [12] datasets, the Office-31 [36] dataset, Office-Caltech [17] and the PACS [27] dataset.

MNIST, MNIST-m and USPS are three standard datasets for digits recognition. USPS [11] is a dataset built using digits scanned from U.S. envelopes, MNIST [26] is the popular benchmark for digits recognition and MNIST-m [12] its counterpart obtained by blending the original images with colored patches extracted from BSD500 photos [2]. Due to their different representations (*e.g.* colored vs gray-scale), these datasets have been adopted as a DA benchmark by many previous works [4, 5, 12]. Here, we consider a multi source DA setting, using MNIST and MNIST-m as sources and USPS as target, training on the union of the training sets and testing on the test set of USPS.

Office-31 is a standard DA benchmark which contains images of 31 object categories collected from 3 different sources: Webcam (W), DSLR camera (D) and the Amazon website (A). Following [41], we perform our tests in the multi-source setting, where each domain is in turn considered as target, while the others are used as source.

Office-Caltech [17] is obtained by selecting the subset of 10 common categories in the Office31 and the Cal-

tech256 [19] datasets. It contains 2533 images, about half of which belong to Caltech256. The different domains are Amazon (A), DSLR (D), Webcam (W) and Caltech256 (C). In our experiments we consider the set of source/target combinations used in [16].

PACS [27] is a recently proposed benchmark which is especially interesting due to the significant domain shift between different domains. It contains images of 7 categories (*dog, elephant, giraffe, guitar, horse*) extracted from 4 different representations, *i.e*. Photo (P), Art paintings (A), Cartoon (C) and Sketch (S). Following the experimental protocol in [27], we train our model considering 3 domains as sources and the remaining as target, using all the images of each domain.Differently from [27] we consider a DA setting (*i.e*. target data is available at training time) and we do not address the problem of domain generalization.

Networks and training protocols. We apply our approach to three different CNN architectures: the MNIST network described in [12], AlexNet [25] and ResNet [20]. We choose AlexNet due to its widespread use in state of the art DA approaches [6, 12, 31, 32], while ResNet is taken as an exemplar for recent architectures employing batch-normalization layers. Both AlexNet and ResNet are first pre-trained on ImageNet and then fine-tuned on the datasets of interest. The MNIST architecture in [12, 13] is chosen following previous works considering digits datasets.

For the evaluation on digits datasets we employ the MNIST architecture described in [12]. Since the original architecture does not contain BN layers, we add mDA-layers after each layer with parameters. We train the architecture following the schedule defined in [12], with a batch-size containing 128 images per domain. The side-branch starts from the `conv1` layer, applies a second convolution with the same parameters of `conv2` and a fully-connected layer with 100 output channel, before the final domain-classifier.

For the experiments on the Office-31 and Office-Caltech datasets we employ the AlexNet architecture. We follow a setup similar to the one proposed in [6, 7], fixing the parameters of all convolutional layers with mDA-layers inserted following each fully-connected layer and before their corresponding activation functions. The domain prediction branch is attached to the last pooling layer following `conv5`. It is composed of a global average pooling, followed by a fully connected layer and a softmax operation to produce the final domain probabilities. The training schedule and hyperparameters are set following [6].

For the experiments on the PACS dataset we consider the ResNet architecture and we choose the 18-layers setup described in [20] and denoted as ResNet18. This architecture comprises an initial 7×7 convolution, denoted as `conv1`, followed by 4 main modules, denoted as `conv2` – `conv5`, each containing two residual blocks. To apply our approach, we replace each Batch Normalization layer in the network with an mDA-layer. The domain prediction branch is attached to `conv1`, and is formed by adding a residual block (with the same number of filters as the ones in `conv2`) and a global average pooling layer followed by a fully connected layer and a softmax. For training we use a weight-decay of 10^{-6}, with the same initial learning rate and momentum adopted for AlexNet. The network is trained for 1200 iterations with a batch-size of 48, equally divided between the domains. The learning rate is scaled by a factor 0.1 after 75% of the iterations. More details about the training procedures can be found in the supplementary material.

Regarding the hyper-parameters of our method, we set the number of source domains k equal to $Q-1$, where Q is the number of different datasets used in each single experiment. Following [6], in the experiments with AlexNet architecture we fix $\lambda_C = \lambda_D = 0.2$. Similarly, for the experiments on digits classification, we keep the weights λ_C, λ_D of the two entropy losses fixed to the same value (0.1). For ResNet we select the values $\lambda_C = 0.1$ and $\lambda_D = 0.0001$ through cross-validation, following the procedure adopted in [6, 30]. When domain labels are available for a subset of source samples, we fix $\lambda_t = 0.5$.

We implement[1] all the models with the Caffe [24] framework and our evaluation is performed using a NVIDIA GeForce 1070 GTX GPU. We initialize both AlexNet and ResNet networks through their models pre-trained on ImageNet. For AlexNet we take the pre-trained model available in Caffe, while for ResNet we use the converted version of the original model developed in Torch [2].

4.2. Results

In this section we report the results of our evaluation. We first analyze the proposed approach, demonstrating the advantages of considering multiple sources and discovering latent domains. We then compare the proposed method with state-of-the-art approaches. For all the experiments we report the results in terms of accuracy, repeating the experiments 5 times and averaging the results.

Analysis of the Proposed Approach. In a first series of experiments, we test the performance of our approach on the MNIST-MNIST-m to USPS benchmark. We compare our method with different baselines: (i) the network trained on the union of all source domains (*Single source (unified)*), (ii) the model which leads to the best performance among those trained on each single source domain (*Best single source*) (iii) the domain adaptation method DIAL in [7] which uses as source set the union of all source domains (*DIAL [7] - Single source (unified)*) and (iv) the DIAL model which leads to the best performance among those

[1] Code available at https://github.com/mancinimassimiliano/latent_domains_DA.git

[2] https://github.com/HolmesShuan/ResNet-18-Caffemodel-on-ImageNet

Table 1: Digits datasets: comparison of different models in the multi-source scenario. MNIST (M) and MNIST-m (Mm) are taken as source domains, USPS (U) as target.

Method	M-Mm to U
Single source (unified)	57.1
Best single source	59.8
DIAL [7] - Single source (unified)	81.7
DIAL [7] - Best single source	81.9
Ours k = 2	82.5
Ours k = 3	82.2
Ours k = 4	82.7
Ours k = 5	82.4
Multi-source DA	84.2

Table 2: PACS dataset: comparison of different methods using the ResNet architecture. The first row indicates the target domain, while all the others are considered as sources.

Method	Sketch	Photo	Art	Cartoon	Mean
ResNet [20]	60.1	92.9	74.7	72.4	75.0
DIAL [7]	66.8	**97.0**	87.3	85.5	84.2
Ours	**69.6**	**97.0**	**87.7**	**86.9**	**85.3**
Multi-source DA	71.6	96.6	87.5	87.0	85.7

trained on each single source domain (*DIAL [7] - Best single source*). Moreover, we report the results of our approach in the ideal case where the multiple source domains are known and we do not need to discover them (*Multi-source DA*). For our approach, we consider several different values for the hyper-parameter k, *i.e*. the number of discovered source domains. All these methods are based on the MNIST network in [12] with the addition of BN layers.

Table 1 shows the results of our comparison. By looking at the table several observations can be made. First, there is a large performance gap between models trained only on source data and DA methods, confirming the fact that deep architectures do not solve the domain shift problem [8]. Second, in analogy with previous works on DA [9, 33, 39], we found that considering multiple sources is beneficial for reducing the domain shift with respect to learning a model on the unified source set. Finally, and more importantly, when the domain labels are not available, our approach is successful in discovering latent domains and in exploiting this information for improving accuracy classification on target data, partially filling the performance gap between the single source models and Multi-source DA. Interestingly, the performance of the algorithm are comparable when the number of latent domains k changes, highlighting the robustness of our model to different values of k. This motivates our choice to always fix k to the known number of domains in the next experiments.

In a second series of experiments we consider the PACS dataset. We compare the proposed approach with the original ResNet architecture trained only on source data and with DA method DIAL [7] trained on the unified source set. As in the previous experiments, we report the results of the ideal multi-source DA setting, *i.e*. our approach is applied to multiple known source domains. Table 2 shows our results. As expected, DA models are especially beneficial when considering the PACS dataset. Moreover, the multi-source DA network outperforms the single source one. Remarkably, our model is able to infer domain information automatically without supervision. In fact, its accuracy is either comparable with the multi-source model (*i.e*. for Photo, Art and Cartoon) or in between the single-source, *i.e*. DIAL, and the multi-source models (*i.e*. Sketch).

Looking at the partial results, it is especially interesting to see that the improvements of our approach and the multi-source model over DIAL trained on the unified source set are especially significant when either the Sketch or the Cartoon domains are employed as target set. Since these domains are less represented in the ImageNet database, we believe that the corresponding features derived from the pre-trained model are less discriminative. In this case DA methods play a significant role.

We also conduct experiments on the Office31 dataset. As baselines we consider the standard AlexNet architecture trained on source data, AlexNet with Batch Normalization added after each fully-connected layer and the DA model of [7] with all source domains unified in a single set. Again, the multi-source DA model obtained assuming the domain labels known for each source sample is taken as upper bound. The results reported in Table 3 trigger two main observations. First, in this dataset there is a small margin for improvement when using a multi-source model with respect to adopting a single source one. This is in accordance with findings in [27], where it is shown that, with respect to PACS dataset, in Office31 the domain shift with deep features is limited and it is linked mainly to changes in background (*i.e*. Webcam-Amazon, DSLR-Amazon) or acquisition camera (DSLR-Webcam). Second, in this case our approach only slightly improves performance over the single-source DA model, suggesting that accuracy in automatically inferring latent domains may not be sufficient for learning better target classifiers.

To further analyze this fact and to demonstrate the flexibility of our framework, we also perform an experiment in a semi-supervised setting. In particular, we consider different levels of supervision in terms of domain information and analyze how the performance of our method change at varying number of labeled source samples. The results of this experiment are reported in Fig. 3. Looking at the figure we can see that by using just few domain labels (5% of the source samples), our model is able to completely fill the performance gap between the unsupervised and the multi-source model. Furthermore, by increasing the level of supervision the accuracy saturates towards the value corresponding to the multi-source model.

Table 3: Office-31 dataset: comparison of different methods using AlexNet. In the first row we indicate the source (top) and the target domains (bottom).

Method / Source / Target	A-D W	A-W D	W-D A	Mean
AlexNet [25]	89.1	94.6	49.1	77.6
AlexNet+BN	92.9	**95.2**	60.1	82.7
DIAL [7]	94.3	93.8	62.5	83.5
Ours	**94.6**	93.7	**62.6**	**83.6**
Multi-source DA	95.8	94.8	62.9	84.5

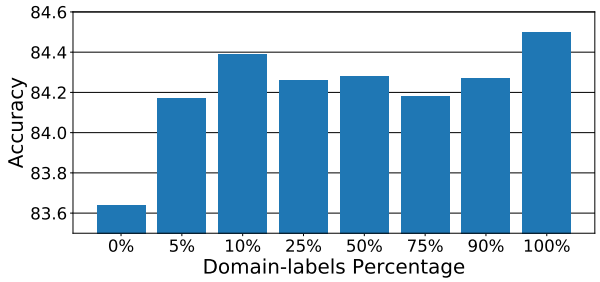

Figure 3: Office31 dataset. Performance at varying number of domain labels (%) for source samples.

Comparison with state of the art. In this section we compare the performance of our model with previous works on DA which also consider the problem of inferring latent domains [16, 21, 41]. As stated in Section 2, there are no previous works adopting deep learning models (i) in a multi-source setting and (ii) discovering hidden domains. Therefore, the considered baseline methods [16, 21, 41] only employ handcrafted features. For these approaches we report results taken from the original papers. To further analyze the impact of different feature representations, we also report results obtained running the method of Gong *et al*. [16] using features from the last layer of the AlexNet architecture. For a fair comparison, in this series of experiments we extract features from the `fc7` layer, without fine-tuning, applying mDA layers to these features and after the classifier.

We first consider the Office31 dataset, as this benchmark has been used in [21, 41]. Table 4 shows the results of our comparison. Our model outperforms all the baselines, with a clear margin in terms of accuracy. Importantly, even when the method in [16] is applied to features derived from AlexNet, still our approach leads to higher accuracy. For the sake of completeness, in the same table we also report results from previous multi-source DA methods [18, 29, 34]. Notice that also these methods are based on shallow models. While these approaches significantly outperform [21] and [41], still their accuracy is much lower than ours.

To compare with [16, 21], we also consider the Office-Caltech dataset. Following [16], we test both single target (Amazon) and multi-target (Amazon-Caltech and Webcam-DSLR) scenarios, for our model can be easily extended to the latter case. We assume to know which samples belong to the source domains and which samples to the target domains. Then, we apply two different mDA modules: one for discovering latent source domains and one for discovering latent target domains. To this extent we need two domain prediction branches: in our implementation they share only the input features, while their parameters are independently learned. Notice that, since we do not assume to know the target domain to which a sample belongs, the task is even harder since we require a domain prediction step also at test time. Again, our approach outperforms all baselines, even the method in [16] adopting features derived from AlexNet.

Table 4: Office-31: comparison with state-of-the-art algorithms. In the first row we indicate the source (top) and the target domains (bottom).

Method / Sources / Target	A-D W	A-W D	W-D A	Mean
Hoffman *et al*. [21]	24.8	42.7	12.8	26.8
Xiong *et al*. [41]	29.3	43.6	13.3	28.7
Gong *et al*. (AlexNet) [16]	91.8	**94.6**	48.9	78.4
Ours	**93.1**	94.3	**64.2**	**83.9**
Gopalan *et al*. [18]	51.3	36.1	35.8	41.1
Nguyen *et al*. [34]	64.5	68.6	41.8	58.3
Lin *et al*. [29]	73.2	81.3	41.1	65.2

Table 5: Office-Caltech dataset: comparison with state-of-the-art algorithms. In the first row we indicate the source (top) and the target domains (bottom).

Method / Source / Target	A-C W-D	W-D A-C	C-W-D A	Mean
Gong *et al*. [16] - original	41.7	35.8	41.0	39.5
Hoffman *et al*. [21] - ensemble	31.7	34.4	38.9	35.0
Hoffman *et al*. [21] - matching	39.6	34.0	34.6	36.1
Gong *et al*. [16] - ensemble	38.7	35.8	42.8	39.1
Gong *et al*. [16] - matching	42.6	35.5	44.6	40.9
Gong *et al*. (AlexNet) [16]	87.8	87.9	93.6	89.8
Ours	**93.5**	**88.2**	**93.7**	**91.8**

5. Conclusions

In this work we presented a novel deep DA model for automatically discovering latent domains within visual datasets. The proposed deep architecture is based on a side-branch which computes the assignment of a source sample to a latent domain. These assignments are then exploited within the main network by novel domain alignment layers which reduce the domain shift by aligning the feature distributions of the discovered sources and the target domains. Our experimental results demonstrate the ability of our model to efficiently exploit the discovered latent domains for addressing challenging domain adaptation tasks.

Acknowledgements. We acknowledge financial support from ERC grant 637076 - RoboExNovo and project *DIGIMAP*, grant 860375, funded by the Austrian Research Promotion Agency (FFG).

References

[1] R. Aljundi and T. Tuytelaars. Lightweight unsupervised domain adaptation by convolutional filter reconstruction. In *ECCV TASK-CV Workshops*, 2016. 4

[2] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, 2011. 5

[3] 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):151–175, 2010. 1

[4] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *CVPR*, 2017. 2, 5

[5] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan. Domain separation networks. In *NIPS*, 2016. 1, 2, 3, 5

[6] F. M. Carlucci, L. Porzi, B. Caputo, E. Ricci, and S. Rota Bulò. Autodial: Automatic domain alignment layers. *ICCV*, 2017. 1, 2, 3, 4, 5, 6

[7] F. M. Carlucci, L. Porzi, B. Caputo, E. Ricci, and S. Rota Bulò. Just dial: Domain alignment layers for unsupervised domain adaptation. *arXiv preprint arXiv:1702.06332*, 2017. 2, 3, 4, 6, 7, 8

[8] 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 *ICML*, 2014. 1, 7

[9] L. Duan, I. W. Tsang, D. Xu, and T.-S. Chua. Domain adaptation from multiple sources via auxiliary classifiers. In *ICML*, 2009. 2, 7

[10] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *ICCV*, 2013. 1, 2

[11] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001. 5

[12] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. *ICML*, 2015. 1, 2, 3, 5, 6, 7

[13] 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. 6

[14] M. Ghifary, W. B. Kleijn, M. Zhang, D. Balduzzi, and W. Li. Deep reconstruction-classification networks for unsupervised domain adaptation. In *ECCV*, 2016. 1, 2, 3

[15] B. Gong, K. Grauman, and F. Sha. Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation. In *ICML*, 2013. 1, 2

[16] B. Gong, K. Grauman, and F. Sha. Reshaping visual datasets for domain adaptation. In *NIPS*, 2013. 2, 6, 8

[17] B. Gong, Y. Shi, F. Sha, and K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *CVPR*, pages 2066–2073. IEEE, 2012. 1, 2, 5

[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, 2014. 8

[19] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. 2007. 6

[20] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 6, 7

[21] J. Hoffman, B. Kulis, T. Darrell, and K. Saenko. Discovering latent domains for multisource domain adaptation. In *ECCV*, 2012. 2, 8

[22] J. Huang, A. Gretton, K. M. Borgwardt, B. Schölkopf, and A. J. Smola. Correcting sample selection bias by unlabeled data. In *NIPS*, 2006. 1, 2

[23] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 2, 4

[24] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM-Multimedia*, pages 675–678. ACM, 2014. 6

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012. 6, 8

[26] 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. 5

[27] D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales. Deeper, broader and artier domain generalization. *arXiv preprint arXiv:1710.03077*, 2017. 5, 6, 7

[28] Y. Li, N. Wang, J. Shi, J. Liu, and X. Hou. Revisiting batch normalization for practical domain adaptation. *arXiv preprint arXiv:1603.04779*, 2016. 3

[29] Y. Lin, J. Chen, Y. Cao, Y. Zhou, L. Zhang, Y. Y. Tang, and S. Wang. Cross-domain recognition by identifying joint subspaces of source domain and target domain. *IEEE Transactions on Cybernetics*, 47(4):1090–1101, 2017. 8

[30] M. Long, G. Ding, J. Wang, J. Sun, Y. Guo, and P. S. Yu. Transfer sparse coding for robust image representation. In *CVPR*, 2013. 1, 2, 6

[31] M. Long and J. Wang. Learning transferable features with deep adaptation networks. In *ICML*, 2015. 1, 2, 3, 6

[32] M. Long, J. Wang, and M. I. Jordan. Unsupervised domain adaptation with residual transfer networks. *NIPS*, 2016. 1, 2, 6

[33] Y. Mansour, M. Mohri, and A. Rostamizadeh. Domain adaptation: Learning bounds and algorithms. *arXiv preprint arXiv:0902.3430*, 2009. 1, 2, 7

[34] H. V. Nguyen, H. T. Ho, V. M. Patel, and R. Chellappa. Dash-n: Joint hierarchical domain adaptation and feature learning. *IEEE Transactions on Image Processing*, 24(12):5479–5491, 2015. 8

[35] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010. 1

[36] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. *ECCV*, pages 213–226, 2010. 2, 5

[37] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. In *CVPR*, pages 2107–2116, 2017. 2

[38] B. Sun and K. Saenko. Deep coral: Correlation alignment for deep domain adaptation. *arXiv preprint arXiv:1607.01719*,

2016. 2

[39] Q. Sun, R. Chattopadhyay, S. Panchanathan, and J. Ye. A two-stage weighting framework for multi-source domain adaptation. In *NIPS*, 2011. 2, 7

[40] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko. Simultaneous deep transfer across domains and tasks. In *ICCV*, 2015. 1, 2, 3

[41] C. Xiong, S. McCloskey, S.-H. Hsieh, and J. J. Corso. Latent domains modeling for visual domain adaptation. In *AAAI*, 2014. 2, 5, 8

[42] M. Yamada, L. Sigal, and M. Raptis. No bias left behind: Covariate shift adaptation for discriminative 3d pose estimation. In *ECCV*, 2012. 2

[43] J. Yang, R. Yan, and A. G. Hauptmann. Adapting svm classifiers to data with shifted distributions. In *ICDM Workshops 2007*, 2007. 2

[44] X. Zeng, W. Ouyang, M. Wang, and X. Wang. Deep learning of scene-specific classifier for pedestrian detection. In *ECCV*, 2014. 2