SAPIENZA
UNIVERSITÀ DI ROMA

Learning to see across Domains and Modalities

Department of Computer, Control and Management Engineering "Antonio Ruberti", Sapienza - University of Rome

Dottorato di Ricerca in Ingegneria Informatica – XXXI Ciclo

Candidate

Fabio Maria Carlucci
ID number 1550340

Thesis Advisor

Prof. Barbara Caputo

Co-Advisor

Dr. Tatiana Tommasi

2018/2019

Thesis defended on 22\02\2019
in front of a Board of Examiners composed by:

Prof. Riccardo Torlone

Prof. Alessandro Farinelli

Prof. Paolo Prinetto

Learning to see across Domains and Modalities
Ph.D. thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: fabiom.carlucci@gmail.com

Acknowledgements

This section has always been the hardest one for me to write; not because I don't know who to thank, but because they deserve a real good acknowledgement and I feel the pressure of giving them what's due.

Firstly, I would like to give a huge thank you to my supervisor, Prof. Barbara Caputo. In a very short time she created, almost like a magician, a strong research group out of thin air. Not only that, but her constant support and guidance has been crucial in these years; she's taught me a lot, on all related subjects, but one idea which has stuck is that being able to present well a good idea is as important as the idea itself - researchers need to be good at communicating. I *know* that she has really shaped me into the researcher I am today.

Heartfelt thanks are reserved for Prof. Tinne Tuytelaars and Prof. Bastian Leibe, who have taken the time to accurately read my thesis and provide useful and constructive feedback.

The VANDAL lab has been an integral part of my journey and my thanks go to each of its members. Special thanks are reserved for Tatiana and Arjan, who I have personally pestered with more questions than should be allowed. As senior members, they have guided and pushed us all in the right direction. Tatiana, as my frequent co-author, earns extra thanks for adding her special touch to our papers... I know that without you my publication list would have been shorter! Thanks to Paolo for being my most prolific co-author and good friend. Valentina, Antonio, Nizar, Massimilino and Ilja I thank you all for being awesome friends and colleagues.

How can I not mention Giuseppe? Either when acting best man or simply when having a chat at a pub, you are the friend everybody would want, but not everybody has. Thanks to Sara and Elisa for *adopting* me; getting to know you has been one of the best side-effects of meeting Angela.

Without my family I would not be here today: their constant support and love have been my safe haven in the years. Saying "thank you" to my parents feels almost reductive, but it is all I can do in this section. My brothers, grandparents, aunts, uncles and cousins have all been part of this journey and I feel lucky to have them: thank you for being there.

My family gave me all the tools to get here, but the one person that gave me the (very needed) push to achieve this PhD is Angela, my wife. You have always been by my side and managed to let me be the best possible version of myself; for that I love you and will always be grateful.

Abstract

Deep learning has recently raised hopes and expectations as a general solution for many applications (computer vision, natural language processing, speech recognition, etc.); indeed it has proven effective, but it also showed a strong dependence on large quantities of data. Generally speaking, deep learning models are especially susceptible to overfitting, due to their large number of internal parameters. Luckily, it has also been shown that, even when data is scarce, a successful model can be trained by reusing prior knowledge. Thus, developing techniques for *transfer learning* (as this process is known), in its broadest definition, is a crucial element towards the deployment of effective and accurate intelligent systems into the real world. This thesis will focus on a family of transfer learning methods applied to the task of *visual object recognition*, specifically *image classification*. The visual recognition problem is central to computer vision research: many desired applications, from robotics to information retrieval, demand the ability to correctly identify categories, places, and objects. Transfer learning is a general term, and specific settings have been given specific names: when the learner has access to only unlabeled data from the *target* domain (where the model should perform) and labeled data from a different domain (the *source*), the problem is called *unsupervised domain adaptation* (DA). The first part of this thesis will focus on three methods for this setting. The three presented techniques for domain adaptation are fully distinct: the first one proposes the use of Domain Alignment layers to structurally align the distributions of the source and target domains in feature space. While the general idea of aligning feature distribution is not novel, we distinguish our method by being one of the very few that do so without adding losses. The second method is based on GANs: we propose a bidirectional architecture that jointly learns how to map the source images into the target visual style and vice-versa, thus alleviating the domain shift at the pixel level. The third method features an adversarial learning process that transforms both the images and the features of both domains in order to map them to a common, agnostic, space.

While the first part of the thesis presented general purpose DA methods, the second part will focus on the real life issues of robotic perception, specifically RGB-D recognition. Robotic platforms are usually not limited to color perception; very often they also carry a Depth camera. Unfortunately, the depth modality is rarely used for visual recognition due to the lack of pretrained models from which to transfer and little data to train one on from scratch. We will first explore the use of synthetic data as proxy for real images by training a Convolutional Neural Network (CNN) on virtual depth maps, rendered from $3D$ CAD models, and then testing it on real robotic datasets. The second approach leverages the existence of RGB pretrained models, by learning how to map the depth data into the most discriminative RGB representation and then using existing models for recognition. This second technique is actually a pretty generic Transfer Learning method which can be applied to share knowledge across modalities.

Keywords: deep learning, domain adaptation, transfer learning, rgb-d, depth, generative, recognition

Contents

Chapter 1

Introduction

We are living the *artificial intelligence* revolution. Almost every day an article in the news pops up, telling us how cars will soon be fully autonomous and robots will finally help us to do our house chores. New AI startups are constantly being launched and big companies are expanding their research teams and creating new labs. For this new wave of optimism, we must thank the renaissance of the neural network [126, 84] (today known as *deep learning*) paradigm, which was sparked in 2012 by the spectacular success of Krizhevsky's AlexNet [80] on the ImageNet [32] challenge.

Since 2010, the annual ImageNet Large Scale Visual Recognition Challenge[127] (ILSVRC) is a competition where research teams evaluate their algorithms on the given data set, and compete to achieve higher accuracy on several visual recognition tasks. AlexNet, a convolutional neural network (CNN), managed to outperform all other competing, non deep learning (we call them *shallow* today) methods by a large margin on the object classification (over 1000) categories challenge. This caused a huge paradigm shift in the Computer Vision field (soon extended to many other research fields) which prompted large improvements across many tasks and promoted the widespread optimism in the capabilities of AI we are currently seeing today. Indeed, deep learning has changed the research landscape in visual object recognition over the last few years. Convolutional neural networks have become the new off the shelf state of the art in visual classification and thanks to these improvements we are actually starting to see real life applications, such as effective face and landmark recognition on our smartphones. The robot vision community has also attempted to take advantage of the deep learning trend, as the ability of robots to understand what they see reliably is critical for their deployment in the wild.

Unfortunately, as good as convolutional neural networks may be, they have a strong limitation: training a deep model from scratch requires lots of labeled data (as reference, the ILSVRC recognition dataset contains almost $1.5M$ images). Due to the large number of internal parameters of these deep networks, learning on small sets of data will often lead to overfitting and poor generalization. The good news is that, if we first train on a suitably large and diverse dataset, we obtain a model which, with little supervision, will perform well on a different, but related, task. This process of learning transfer, known as *finetuning* [177, 169, 33, 170], is what makes it possible to exploit the deep learning potential on smaller sets of data. Finetuning a model still requires *some* labeled data belonging to our target domain; this may, in today's age of big data, initially appear as a non-problem but it is not so for a variety of reasons.

The web, with its easy access to large quantities of pictures (Google Image

Figure 1.1. Sample images of certain classes (stapler, water bottle, cellphone, spray can) as seen in a wed dataset, Imagenet [127], on the left, and in a real-life like dataset (JHUIT-50 [87], HelloiCubWorld [38]), on the right. Note that while they should be representing the same things, the images have very little in common

Search[1], Instagram[2], Pinterest[3], etc..) seems the ideal and cheap source of labeled images, but present two significant challenges: data annotation and data bias. Once you download images from the web, you must assign them labels, which is costly, in order to be able to train on them. Most web sources can provide labels, but they usually are noisy [118] and this will affect the final performances. For the scope of this thesis we will assume to be working with hand annotated data and consider our labels noise-free.

Even with perfect annotations, we still have to deal with the issue of data bias[155]. A simple intuition of this problem can be had looking at Fig. 1.1. The images on the left belong to a dataset which was mined from the web, the images on the right were captured by a robot: the framing, the lightning conditions, the resolution, the background clutter are all different. If our model is trained with the images on the left it is easy to understand why it will perform poorly in the real world (another example of data bias can be seen in Fig. 1.2). This is a pretty typical setup: we wanted to perform recognition on a set of classes, we used the web to download some training data (we will call it the *source*) and found out that the model did not work well on real world data. During the deployment of our system we gathered some unlabeled data (our *target*) for free. We know that the labeled source and unlabeled target share the same classes and we would like for our model to perform well on both, ignoring their specific biases. This problem is formally known as that of *unsupervised domain adaptation* (we will define it more rigorously in section 2.1.1).

The first half of this thesis will deal with this issue, by investigating multiple Domain Adaptation (DA) approaches. Domain Adaptation is at its core the quest for principled algorithms enabling the generalization of visual recognition methods. Given at least a source domain for training, the goal is to achieve recognition results as good as those achievable on source test data on any other target domain, in principle belonging to a different probability distribution, without having prior access to labeled images. Solving this problem will represent a major step towards one of the key goals of computer vision, i.e. having machines able to answer the question 'what do you see?' in the wild; hence, its increased popularity in the community

[1] https://images.google.com/
[2] https://www.instagram.com/
[3] https://www.pinterest.com/

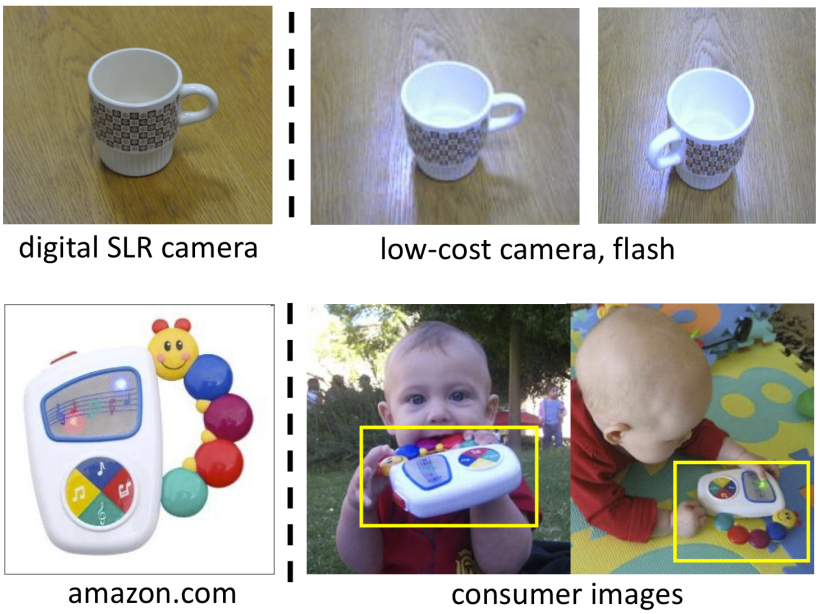


Figure 1.2. A graphical intuition behind the Domain Adaptation problem (image from https://people.eecs.berkeley.edu/ jhoffman/domainadapt.)

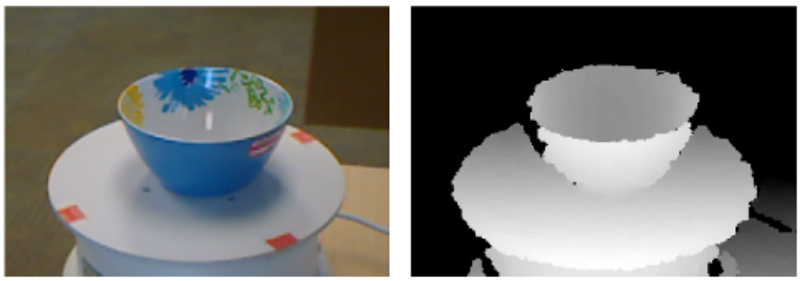


Figure 1.3. A sample RGB-D image from the Washington dataset [81]. Note how the Depth modality provides geometric intuition cues

over the last years (see section 2.1 for a review of recent work)

Of course, DA is not the solution to all CV problems: RGB-D recognition, for example, encounters a different set of challenges. RGB-D sensors (cameras capable of producing color images and depth maps at the same time) are extremely widespread on robotic platforms, but while depth cues could greatly help the recognition process by providing $3D$ intuition (see Fig. 1.3), it is often ignored. Why is this so? RGB-D datasets tend to be too small to train a CNN from scratch and the large difference in appearance between RGB and depth (see Fig. 1.3) severely limits the applicability of current finetuning and DA methods. Furthermore, we do have labels for our target depth dataset, so unsupervised (or semi-supervised) domain adaptation methods are not really suited.

The second part of this thesis investigates specifically how different ways of transferring knowledge can benefit RGB-D recognition.

1.1 Contributions

Working in the context of visual image recognition, the main contributions of this work are two-fold: on one side, we contribute to the field of unsupervised domain adaptation with three novel and distinct techniques, and on the other we explore alternative approaches for those settings in which classical DA methods cannot be applied. We then perform a qualitative and quantitative experimental evaluation of the proposed solutions, to quantify their effectiveness and robustness.

Specifically we present:

a Domain Alignment Layer for domain adaptation [18, 19] which aligns multiple source domains between themselves and to the target at a feature level. Contrary to most previous methods, this solutions does not require a new loss term, as the domain aligning effect is implicit in the architecture. Concretely, what this layer does, is project the input features from each domain into a reference distribution, similarly to what a *Batch Normalization* [73] layer would do, but it does so separately for each domain and learns how much statistic sharing should occur.

a generative approach to apply the style of the target domain to the source and vice versa [128] by exploiting the power of a bidirectional GAN [52]. This technique allows us to bridge the domain gap at the image level by producing source images (of which we have labels) which look exactly as those from the target; clearly a classifier trained on these images will also perform well on the target. In this work we also explore the opposite transformation: making the target look as the source so that, when evaluated on a classifier trained on the source itself, no domain shift would present itself. Since this technique is applied at the image level, it lends itself well to integration with other feature based methods.

a novel approach which learns an agnostic representation for multiple domains [22] by transforming both the images themselves **and** and the feature representation. The intuition here it that sometimes it is easier to reduce the domain shift by working on the images (i.e. different background) and sometimes it is easier in feature space. There is no way of knowing this beforehand, so the best approach is to tweak both representations at the same time. Concretely this happens thanks to a complex network based on image generators and two *Reverse Grad* [43] branches.

a case study on building a synthetic depth dataset for object recognition [20], with the goal of training a model which can then be used on real robotic images. RGB-D sensors are commonly used in robotics, but the Depth modality is often times used only for navigation or segmentation. This is mainly due to the fact that there are no pretrained models for Depth recognition: this work tries to solve the issue by using 3D CAD models farmed from the Web.

a Transfer Learning method useful when source and target exist in different modalities [21] . Most transfer learning and domain adaptation methods require some assumptions to be true; here we present a general approach which allows knowledge sharing across modalities, with no prerequisites on the data[4]. The idea is that if we have a strong pretrained network on the some data (i.e. ImageNet

[4]Clearly, the greater the similarity the greater the potential benefits

[127]) we can map our target data, from a different modality, to the source modality by learning a mapping which maximizes recognition performance.

1.2 Outline

Chapter 2 will provide a formal definition of the considered problems, present an overview of relevant works and introduce the datasets we will use for the experimental evaluation. Reflecting the structure of this thesis, this chapter is divided in two: the first part focuses on visual object recognition, more specifically on unsupervised domain adaptation methods for classification, while the second part will focus on the robotics problem of RGB-D recognition - a modality in which learning to transfer the knowledge is of primary importance.

The domain adaption literature (section 2.1) will present shallow, deep and adversarial methods. We will focus on the different methods used in single-source, multi-source domain adaptation and domain generalization.

Section 2.2, on RGB-D recognition, will review methods for classifying objects using Depth only and then on how Depth and RGB cues are often integrated together. As the transfer learning method we will present in section 4.2 has drawn inspiration these techniques, we will also present some literature pertaining image colorization.

Chapter 3 will delve into the details of the three unsupervised domain adaptation methods we propose in this thesis.

Section 3.1 presents **AutoDIAL** a method for single and multi-source domain adaption based on the use of special domain alignment layers, which reduce the domain shift at the feature level.

In section 3.2 we present **SBADA-GAN**, a GAN based method for single source domain adaptation. In this approach we tackle the domain bias directly at the image level, projecting the source images into the target style and vice versa.

ADAGE, presented in section 3.3, builds on both previous approaches: it is a method for single, multi-source domain adaption and domain generalization. We suggest that for some things it is easier to reduce the domain shift in feature space, while for other it is easier in image space: ADAGE does both at the same time by using an adversarial loss to learn a domain-agnostic representation for the images and features.

Chapter 4 describes two alternative methods which can be used lo learn an effective image classifier for modalities where data is scarce.

Section 4.1 presents a case study on the use of $3D$ CAD models to generate synthetic depth maps as a proxy for real data. We trained a CNN (the **DepthNet**) on the virtual depth images and evaluated its performance on real-life robotic RGB-D datasets.

In section 4.2 we present $(\mathbf{DE})^{\mathbf{2}}\mathbf{CO}$, our Deep Depth Colorization method for converting depth images to RGB. We show this transformation allows us to effectively use models pretrained on ImageNet with great success on RGB-D datasets.

The thesis concludes with a summary discussion and remarks on possible future directions of research in **chapter 5**.

1.3 Publications

The following list gives an overview of the author's publications in chronological order; note that a few of these published papers (marked with a *) are not included in this thesis.

- Kuzborskij, I., Maria Carlucci, F., & Caputo, B. (2016). When naive bayes nearest neighbors meet convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 2100-2109).*

- Carlucci, F. M., Russo, P., & Caputo, B. (2017, May). A deep representation for depth images from synthetic data. In Robotics and Automation (ICRA), 2017 IEEE International Conference on (pp. 1362-1369). IEEE.

- Carlucci, F. M., Porzi, L., Caputo, B., Ricci, E., & Bulò, S. R. (2017, September). Just dial: Domain alignment layers for unsupervised domain adaptation. In International Conference on Image Analysis and Processing (pp. 357-369). Springer, Cham. **(oral - Best Student Paper)***

- D'Innocente, A., Carlucci, F. M., Colosi, M., & Caputo, B. (2017, July). Bridging between computer and robot vision through data augmentation: a case study on object recognition. In International Conference on Computer Vision Systems (pp. 384-393). Springer, Cham. **(Best Paper Award finalist)***

- Carlucci, F. M., Porzi, L., Caputo, B., Ricci, E., & Bulò, S. R. (2017, October). AutoDIAL: Automatic Domain Alignment Layers. In Proceedings of the IEEE International Conference on Computer Vision (pp. 5077-5085).

- Carlucci, F. M., Russo, P., & Caputo, B. (2018). DE^2CO: Deep Depth Colorization. IEEE Robotics and Automation Letters, 3(3), 2386-2393. (also presented at the IEEE International Conference on Robotics and Automation).

- Russo, P., Carlucci, F. M., Tommasi, T., & Caputo, B. (2018). "From source to target and back: symmetric bi-directional adaptive GAN." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Note that an extended version of "AutoDIAL" is currently pending review for publication in the IEEE Transactions on Pattern Analysis and Machine Intelligence journal

1.3.1 Technical Reports pending review:

to be submitted to the IEEE Conference on Computer Vision and Pattern Recognition (2019)

- Carlucci, F. M., Russo, P., Tommasi, T., & Caputo, B. (2018). Agnostic Domain Generalization. arXiv preprint arXiv:1808.01102.

- Carlucci, F. M., D'Innocente, A., Caputo, B., & Tommasi. T. "Jigsaw Domain Generalization." arXiv preprint (2018).

Chapter 2

Background and Related Work

This chapter is divided in two sections: one related to the unsupervised domain adaptation problem (in the context of image classification), and the other pertaining RGB-D recognition, with a specific focus on the depth modality. Each section starts with the problem formulation, continues with a review of related work and finally presents the datasets on which the proposed algorithms will be tested.

2.1 Computer Vision: Domain Adaptation

In this section, first we present a definition of the unsupervised domain adaptation problem and then review previous works. We consider traditional approaches based on shallow models, methods based on deep architectures and more recent techniques based on the generative adversarial paradigm. For each, we describe works on single-source, multi-source domain adaptation and domain generalization. In conclusion we present the datasets commonly used by the community, which we will use to evaluate our solutions

2.1.1 Unsupervised Domain Adaptation

In the introduction of this thesis we gave an intuition of what domain adaptation is and why is it relevant to the deployment of well performing recognition systems in the wild. Here we provide a more formal description:

Domain adaptation [151] aims at solving the learning problem on a target domain T exploiting information from a source domain S, when both the domains and the corresponding tasks are not the same. More specifically, while the tasks have identical label sets $Y^s = Y^t$ they possess (slightly) different conditional distributions $P^s(Y|X) \sim P^t(Y|X)$. The domains are different in terms of marginal data distribution $P^s(X) \neq P^t(X)$, and/or in feature spaces $X^s \neq X^t$. Our goal is to estimate a predictor from S and T that can be used to classify sample points from the target domain. Domain adaptation has been studied in two main settings: one is the semi-supervised case, where the target presents few labeled data, while the other is the unsupervised case that considers only unlabeled examples for the target. In both cases, the source set is generally rich in labeled samples. In this thesis we will focus on the unsupervised case. We can then define the source as $S = \{(x_1, y_1), ..., (x_n, y_n)\} \subset \mathcal{X} \times \mathcal{Y}$ and the target as $T = \{\hat{x}_1, ..., \hat{x}_m\} \subset \mathcal{X}$

Multi-source is a natural extension of the the single source domain adaption setting where multiple source domains are considered instead of a single one. In this case we can redefine the source as: $S = \{(x_1, y_1, d_1), ..., (x_n, y_n, d_n)\} \subset \mathcal{X} \times \mathcal{Y} \times \mathcal{D}$. Generally speaking, the various source domains are also different in terms of marginal data distribution.

Domain Generalization (DG) is an even more challenging variation. In this setting we usually have multiple sources but target data is not available during the training process. In order to best perform on the, unseen, target, DG methods attempt to train the most general and robust classifier possible on the source domains.

2.1.2 Literature survey

Single source domain adaptation

Traditional approaches addressed the problem of reducing the discrepancy between the source and the target distributions by considering two main strategies. The first is based on instance re-weighting [71, 29, 167, 50, 173]: source samples are assigned different importance according to their similarity to the target data. The re-weighted samples are then used to learn a classification model for the target domain. Following this scheme, Huang *et al.* [71] introduced Kernel Mean Matching, a nonparametric method to set source sample weights without explicitly estimating

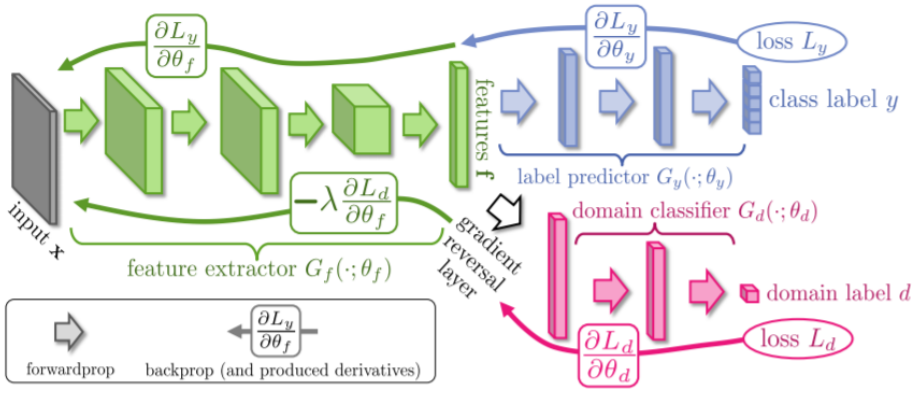


Figure 2.1. The proposed architecture for the use of the reverse gradient, from "Unsupervised Domain Adaptation by Backpropagation" [42]. Our ADAGE framework, in section 3.3, is influenced by this work and depends on the use of gradient reversal layers.

the data distributions. Gong *et al.* [50] proposed to automatically discover a set of landmark datapoints, i.e. to identify the source samples which are most similar to target instances, and used them to create domain-invariant features. Chu *et al.* [29] unified the tasks of source sample selection and classifier learning within a single optimization problem. While these works considered hand-crafted features, recently similar ideas have been applied to deep models. For instance, Zeng *et al.* [173] described an unsupervised domain adaptation approach for pedestrian detection using deep autoencoders to weight the importance of source training samples.

A second strategy for unsupervised domain adaptation is based on feature alignment, i.e. source and target data are projected in a common subspace in order to reduce the distance among the associated distributions. This approach attracted considerable interest in the past years and several different methods have been proposed, both considering shallow models [51, 98, 39] and deep architectures [99, 156, 42, 47, 14].

Focusing on recent deep domain adaptation methods, different schemes have been considered to align feature representations. Earlier works proposed to reduce the domain shift, while learning deep representations, by modeling source and target data distributions in terms of their first [99, 101, 100, 161] and second order [145] statistics. Other works proposed to learn domain-agnostic deep features within a domain-adversarial setting [156, 42]. Haeusser *et al.* [59] described a methodology to learn domain invariant features by reducing the distance among source and target samples of each class, instead of considering the whole sets. Sankaranarayanan *et al.* [132] proposed to learn an embedding that is robust to the shift between source and target distributions by using a combination of a classification loss and an image generation procedure modeled using a variant of Generative Adversarial Networks (GANs)[52]. Other methods attempted to reduce the distribution mismatch between source and target data by embedding into a neural network specific distribution normalization layers [93, 107, 106].

Our approach in section 3.1 belongs to the category of methods employing domain normalization layers for domain adaptation. Similarly to previous work we exploit Batch Normalization in the context of domain adaptation and propose to reduce the discrepancy between source and target distributions by introducing our DA-layers. However, we significantly depart from previous works [93, 107, 106], as our DA layers allow to automatically tune the required degree of adaptation at each level of the

deep network. Furthermore, we also introduce a prior over the network parameters in order to fully benefit from the target samples during training.

Entropy regularization for semi-supervised learning was originally introduced in [54]. This concept has been then exploited in deep neural networks with the name of learning with pseudo-labels [180]. The main idea is to use high confidence predictions as labels for unlabeled samples in order to improve the performance of the classifier [180]. Recently, some domain adaptation methods [128, 131, 101, 100, 161] have exploited this technique for learning deep representations. However, in section 3.1 we demonstrate that this approach is especially effective when applied in combination with our proposed DA-layers.

More recently, several approaches have attempted to reduce the domain shift by directly transforming images using Generative Adversarial Networks [52] (GANs). Vanilla GANs are agnostic to the training samples labels, while conditional GAN variants [111] exploit the class annotation as additional information to both the generator and the discriminator. Some works used multiple GANs: in CoGAN [96] two generators and two discriminators are coupled by weight-sharing to learn the joint distribution of images in two different domains without using pair-wise data. Cycle-GAN [179], Disco-GAN [78] and UNIT [95] encourage the mapping between two domains to be well covered by imposing transitivity: the mapping in one direction followed by the mapping in the opposite direction should arrive where it started. Initially GAN methods were not exploited for domain adaptation, but recently several approaches have attempted to reduce the domain shift by directly transforming images using this approach [13, 137, 149, 70, 128]. [13] proposed a GAN-based approach that adapts source images to appear as if drawn from the target domain; the classifier trained on such data outperformed several domain adaptation methods. [149] introduced a method to generate source images that resemble the target ones, with the extra consistency constraint that the same transformation should keep the target samples identical. All these methods focus on the source-to-target image generation, not considering the inverse procedure, from target to source, which we, in section 3.2, show instead to be beneficial.

It must be noted that, while deep generative models are very effective in specific applications (e.g. adaptation from synthetic to real images[137, 13], digit recognition [13, 137, 149, 70, 128]), their performance in other settings is limited.

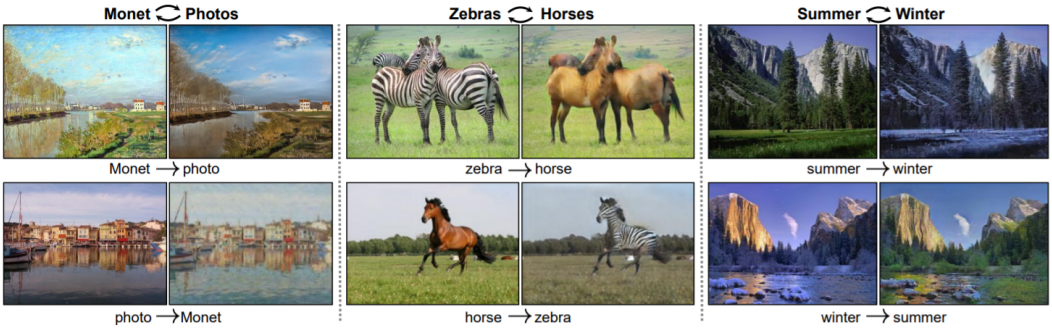


Figure 2.2. Image samples from CycleGAN[179]. Given any two unordered image collections X and Y , their algorithm learns to automatically "translate" an image from one into the other and vice versa. Our SBADA-GAN (section 3.2) approach can be seen as natural extension of this method to the task of domain adaptation.

Multi-source domain adaptation

A reasonable assumption when building visual recognition systems is that we may have access to more than one source domain. In this context, the study of multi-source domain adaptation algorithms is especially important.

This multi-source setting (Multi Domain Adaptation, *MDA*), which originated from A-SVM [168], was initially studied from a theoretical point of view, focusing on theorems indicating how to optimally sub-select the data to be used in learning the source models [30], or proposing principled rules for combining the source-specific classifiers and obtain the ideal target class prediction [108]. Several other works followed this direction in the shallow learning framework. presenting algorithms based on the combination of source hypotheses [147, 34]. A different set of multi-source adaptation approaches are based on learning new data representations, defining a mapping to a latent feature space shared between domains. Some of them are created for single sources but can be easily extended to multiple ones, as the feature replication method of [31] and the unsupervised approach based on Grassman manifolds presented in [53], which supports multiple sources by first computing the Karcher mean of the sources and then exploiting the standard single source version of the algorithm. Other works presented instead dedicated solutions for multiple sources [69, 74, 35, 94].

Recently some deep architectures have been proposed to tackle this challenge. In particular, Xu *et al.* [166] proposed an approach derived from [44] which exploits an adversarial domain discriminator branch for learning domain agnostic features. A similar multi-way adversarial strategy was also introduced in [176]. Mancini *et al.* [107] proposed an approach for multi-source domain adaptation where latent domains are automatically discovered in the training set. Our work in section 3.1 differs significantly from these previous works, as we propose a multi-source domain adaptation method based on DA-layers which automatically learns at each network layer the optimal degree of adaptation.

Domain Generalization Domain Generalization (DG) is a newer line of research first studied by Blanchard et al. [9]; in this setting, no transductive access to the target data is allowed, thus the main objective is to look across multiple sources for shared factors in the hypothesis that they will hold also for any new target domain. A number of shallow methods have been presented in the years. [115] propose a projection-based algorithm, Domain-Invariant Component Analysis (DICA) which extends Kernel PCA by incorporating the distributional variance in order to reduce the dissimilarity across domains and the central subspace. Khosla et al. [77] proposed a multi-task max-margin classifier that explicitly encodes dataset-specific biases in feature space. These biases are used to push the dataset-specific weights to be similar to the global weights. Ghifary et al. [45] proposed Scatter Component Analyis (SCA), a representation learning algorithm for both domain adaptation and domain generalization. SCA, based on a the scatter measure which operates on reproducing kernel Hilbert space, finds a representation that trades between maximizing the separability of classes, minimizing the mismatch between domains, and maximizing the separability of data.

Deep learning methods usually search for shared factors either at *model-level* to regularize the learning process on the sources, or at *feature-level* to learn some domain-shared representation. When focusing on deep DG methods, model-level strategies are presented in [105] and [90]. The first work proposes a weighting procedure on the source models, while the second aims at separating the source knowledge into domain-specific and domain-agnostic sub-models. A meta-learning approach was recently presented in [89]: it starts by creating virtual testing domains within

Figure 2.3. Samples from the digits datasets. From left to right: MNIST, SYNTH, SVHN, MNIST-M and USPS

each source mini-batch and then it trains a network to minimize the classification loss, while also ensuring that the taken direction leads to an improvement on the virtual testing loss. Regarding feature-based methods, [114] proposes to exploit a Siamese architecture to learn an embedding space where samples from different source domains but same labels are projected nearby, while samples from different domains and different labels are mapped far apart. Both the works [46, 91] exploit deep learning autoencoders for domain generalization still focusing on representation learning. A new way to tackle DG was recently presented in [135]. Instead of aiming at the reduction of domain-specific signals, this work introduces a form of data augmentation based on domain-guided perturbations of the input instances. A label classifier is then trained on both the original instance and the data produced by the described augmentation process.

As a final remark, we note that, despite for both DA and DG there exist multiple methods based on features and representation learning, image-adaptive solutions are exclusive for the single source domain adaptation setting. The only methods in DG that somehow involve images are [46, 135] but the focus is either on source-to-source reconstruction or on data augmentation. With ADAGE, a method we introduce in section 3.3, we introduce a joint image and feature adaptation method: it learns how to project images into a network-understandable agnostic visual space that paves the way for completing the domain gap closure in the feature space.

2.1.3 Datasets

The following section presents the datasets and the protocols which will be used to evaluate the DA methods proposed in chapter 3. We first present some simple datasets, commonly used by generative adversarial methods, and then describe the more complex scenarios on which DA methods are routinely tested

Digits like datasets

The following datasets are fairly simple, with moderate intra class variability. They are commonly used to evalute GAN based methods, which, to date, still struggle with more complex data

MNIST [86] is the dataset on which the first convolutional neural network was trained. It has a training set of $60k$ examples, and a test set of $10k$ examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image. The images contain a single digit numbers on a black background.

MNIST-M [43] is a variant where the background is substituted by a randomly extracted patch obtained from color photos of BSDS500 [4].

Figure 2.4. Samples from the signs datasets. From left to right: Synth Signs and GTSRB

USPS [41] is a digit dataset automatically scanned from envelopes by the U.S. Postal Service containing a total of $9,298$ 16×16 pixel grayscale samples; the images are centered, normalized and show a broad range of font styles.

SVHN [117] is the challenging real-world Street View House Number dataset. It contains over 600k 32×32 pixel color samples, while we focused on the smaller version of almost 100k cropped digits. Besides presenting a great variety of shapes and textures, images from this dataset often contain extraneous numbers in addition to the labeled, centered one.

SYNTH Digits this collection [43] consists of 500k images generated from Windows^{TM} fonts by varying the text (that includes different one-, two-, and three-digit numbers), positioning, orientation, background and stroke colors, and the amount of blur

Synth Signs the Synthetic Signs collection [112] contains $100k$ samples of common street signs obtained from Wikipedia and artificially transformed to simulate various imaging conditions.

GTSRBThe German Traffic Signs Recognition Benchmark (GTSRB, [142]) consists of $51,839$ cropped images of German traffic signs.

Real life datasets

These datasets are more complex and unconstrained; they better mimic real world applications. Most of them are commonly used for evaluation of recent deep learning based methods.

The **Office 31**[129] dataset has been the standard benchmark for testing domain-adaptation methods for many years. It contains 4652 images organized in 31 classes from three different domains: Amazon (A), DSRL (D) and Webcam (W). Amazon images are collected from `amazon.com`, Webcam and DSLR images were manually gathered in an office environment. As can be seen in Fig. 2.5, the domain gap is pretty large between Amazon and the remaining two, and small between DSLR and Webcam. In our experiments we consider all possible source/target combinations of these domains and adopt the *full protocol* setting [50], i.e. we train on the entire labeled source and unlabeled target data and test on annotated target samples. For the multi-source setting, we instead follow the protocol described in [166].

Office-Home [161] is a recently released dataset, built to overcome the size limitation of previous domain adaptation settings. It consists of 4 domains, each domain containing 65 matching categories of everyday objects from the home and office environment. Looking at Fig. 2.6 we can see how there exists a large domain gap between domains containing drawings and domains containing photos. In total this dataset is composed by 15.500 images gathered from the web. The domains are:

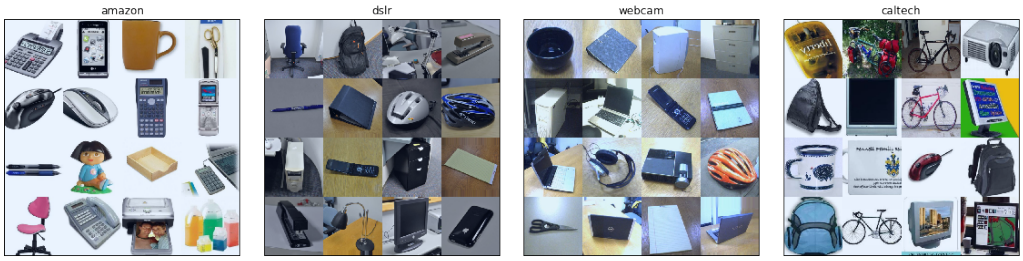


Figure 2.5. Samples from the Office and Caltech settings. From left to right: Amazon, DSLR, Webcam and Caltech

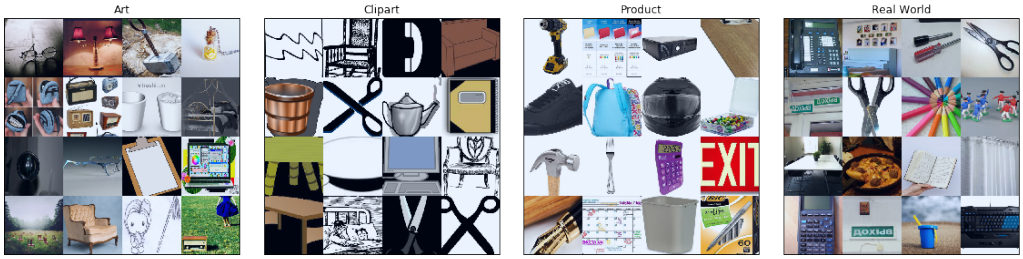


Figure 2.6. Samples from the Office-Home setting. From left to right: Art, Clipart, Product and Real World

Art, *Clipart*, *Product* and *Real-World*.

The **Office-Caltech** [51] dataset is obtained by selecting the subset of 10 common categories in the Office31 and the Caltech256[55] datasets. It contains 2533 images of which about half belong to Caltech256. Each of Amazon (A), DSLR (D), Webcam (W) and Caltech256 (C) are regarded as separate domains. In our experiments we only consider the source/target combinations containing C as either the source or target domain.

To further perform an analysis on a large-scale dataset, we also consider the **Cross-Dataset Testbed** introduced in [154] and specifically the **Caltech-ImageNet** setting. This dataset was obtained by collecting the images corresponding to the 40 classes shared between the Caltech256 (C) and the Imagenet (I) [32] datasets. To facilitate comparison with previous works [156, 152, 143] we perform experiments in two different settings. The first setting, adopted in [152, 156], considers 5 splits obtained by selecting 5534 images from ImageNet and 4366 images from Caltech256 across all 40 categories. The second setting, adopted in [143], uses 3847 images for Caltech256 and 4000 images for ImageNet.

Figure 2.7. 1000 samples from the ImageNet dataset

Figure 2.8. To the left: an RGB image of a white cup, on a white plate, sitting on a white table with a white background. Similar scenes lack in texture and are extremely hard to correctly classify. To the right: the depth image of a similar scene. Note how the cup is much more visible against the background and the gradient on the table gives us geometrical intuition.

2.2 Robotic Vision: RGB-D recognition

We start this section by explaining why RGB-D recognition is important and how it is related to this thesis. We then give an overview of how this problem has been tackled by others and, in the end, we present the RGB-D datasets commonly used by the community which we will use to evaluate our proposed solutions.

2.2.1 Problem statement

Perception of the outside world is probably one of the key ingredients for the success of autonomous robots interacting with an unstructured and unconstrained environment. Planning, reasoning and accurate motion control all depend on the agent having a correct internal representation of the space it's moving in. Adding multiple sensing modalities is one way to make the perception task easier. For example, RGB recognition systems have been know to struggle with low-contrast, textureless, images, such as the left picture in Fig. 2.8, but depth cameras can produce images of the same scene which are much easier to interpret. Indeed, due to their low cost and ease of use, depth cameras are one of the most commonly available sensors in robotics. Depth is used for many things[60], from SLAM [37] to segmentation[64, 65], to pose estimation[125, 76]. As discussed above, the depth modality can also clearly contribute to the recognition performance, as it can perceive different features (ex. geometrical cues) from the RGB ones.

In section 2.2.2 we will survey most recent works dealing with RGB-D recognition. Almost all of them share one commonality: they use an ImageNet pretrained network for RGB perception and then exploit multiple workarounds to deal with the Depth modality. Why is this necessary? Simply put, most RGB-D datasets are small (in semantic variability if not in sheer size) and a pretrained network of some type is needed to avoid overfitting issues. Here lies the problem: it is extremely unlikely we will ever have an RGB-D dataset comparable in size to ImageNet.

Most large RGB datasets (ImageNet included) were farmed from the web, which is a true treasure for this kind of data - but datasets for other modalities (Depth, Infrared, Multispectral cameras) will never be found online with the same ease. For

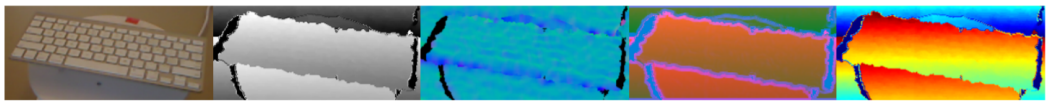


Figure 2.9. Different approaches for color encoding of depth images. From left to right: RGB, depth-gray, surface normals, HHA, colorjet. (image from "Multimodal Deep Learning for Robust RGB-D Object Recognition" [36]

these cases we must accept that we either train a model on synthetic data (which we explore in section 4.1) or we find a smart, possibly data-driven, way to exploit what we learned on RGB for these other modalities (which we will describe in section 4.2).

Classical DA approaches are not easily applicable here: firstly, our *target*, the Depth modality, is fully labeled (so even semi-supervised approaches do not really fit) and secondly our images are not immediately usable by RGB pretrained models, at least without some sort of preprocessing.

2.2.2 Literature survey

Object recognition from RGB-D data traditionally relied on hand-crafted features such as SIFT [102] and spin images [82], combined together through vector quantization in a Bag-of-Words encoding [82]. This heuristic approach has been surpassed by end-to-end feature learning architectures, able to define suitable features in a data-driven fashion [10, 141, 5]. All these methods have been designed to cope with a limited amount of training data (of the order of $10^3 - 10^4$ depth images), thus they are able to only partially exploit the generalization abilities of deep learning as feature extractors experienced in the computer vision community [80, 136], where databases of 10^6 RGB images like ImageNet [127] or Places [178] are available.

An alternative route is that of re-using deep learning architectures trained on ImageNet through pre-defined encoding [56] or colorization. Colorization of depth images can be seen as a transfer learning process across modalities, and several works explored this avenue within the deep learning framework: HHA, a method proposed by Gupta, Saurabh, et al. [56], is a mapping where one channel encodes the horizontal disparity, one the height above ground and the third the pixelwise angle between the surface normal and the gravity vector. Schwarz et al [133] proposed a colorization pipeline where colors are assigned to the image pixels according to the distance of the vertexes of a rendered mesh to the center of the object. Bo et al.[11] convert the depth map to surface normals and then re-interprets it as RGB values, while Aekerberg et al. [1] builds on this approach and suggests an effective preprocessing pipeline to increase performance. Besides the naive grayscale method, the rest of the mentioned colorization schemes are computationally expensive. Eitel et al [36] used a color mapping technique known as *ColorJet*, showing this simple method to be competitive with more sophisticated approaches.

In the context of RGB-D object detection, a recent stream of works explicitly addressed cross modal transfer learning through sharing of weights across architectures [68], [67] and [57]. This last work is conceptually close to one approach we will present in section 4.2, as it proposes to learn how to transfer RGB extracted information to the Depth domain through distillation [66]. While [57] has proved very successful in the object detection realm, it presents some constraints that might potentially be problematic in object recognition, from the requirement of paired RGB-D images, to specific data preprocessing and preparation for training. As opposed to this, our algorithm does not require explicit pairing of images in the two

modalities, can be applied successfully on raw pixel data and does not require other data preparation for training.

Some approaches coupled non linear learning methods with various forms of spatial encodings [26, 25, 24, 88]. Hasan et al [172] pushed further this multi-modal approach, proposing an architecture merging together RGB, depth and 3D point cloud information. Another notable feature is the encoding of an implicit multi scale representation through a rich coarse-to-fine feature extraction approach.

All these works, and others [172, 20], make use of an ad-hoc mapping for converting depth images into three channels. This conversion is vital as the dataset has to be compatible with the pre-trained CNN. Depth data is encoded as a 2D array where each element represents an approximate distance between the sensor and the object. Depth information is often depicted and stored as a single monochrome image. Compared to regular RGB cameras, the depth resolution is relatively low, especially when the frame is cropped to focus on a particular object. In section 4.2 we will address this issue, by avoiding heuristic choices in our approach and instead relying instead on an end-to-end, residual based deep architecture to learn the optimal mapping for the cross modal knowledge transfer.

Our work is also related to the colorization of grayscale images using deep nets. Cheng et al [27] proposed a colorization pipeline based on three different hand-designed feature extractors to determine the features from different levels of an input image. Larsson et al [83] used an architecture consisting of two parts. The first part is a fully convolutional version of VGG-16 used as feature extractor, and the second part is a fully-connected layer with 1024 channels predicting the distributions of hue and the chroma for each pixel given its feature descriptors from the previous level. Iizuka et al [72] proposed an end-to-end network able to learn global and local features, exploiting the classification labels for better image colorization. Their architecture consists of several networks followed by fusion layer for the colorization task. Sun et al. [146] propose to use large scale CAD rendered data to leverage depth information without using low level features or colorization. In Asif et al. [6], hierarchical cascaded forests were used for computing grasp poses and perform object classification, exploiting several different features like orientation angle maps, surface normals and depth information colored with *Jet* method. Our work differs from this last research thread in the specific architecture proposed, and in its main goal, as here we are interested in learning optimal mapping for categorization rather than for colorization of grayscale images.

All these works build on top of CNNs pre-trained over ImageNet, for all modal channels. Thus, the very same filters are used to extract features from all of them. As empirically successful as this might be, it is a questionable strategy, as RGB and depth images are perceptually very different, and as such they would benefit from approaches able to learn data-specific features (fig. 4.1).

Another possibility, which we explore in section 4.1 is to train the model on synthetic data; it is usually much cheaper to build a computer generated dataset than a real one and, often times, it allows for a greater variability. Our method matches this challenge, learning RGB features from RGB data and depth features from synthetically generated data, within a deep learning framework. The use of realistic synthetic data in conjunction with deep learning architectures is a promising emerging trend [123, 110, 164, 61]. We are not aware of previous work attempting to use synthetic data to learn depth representations, with or without deep learning techniques.

Figure 2.10. Samples from the Jhuit[87], Washington-50[81] and BigBIRD[140] datasets, RGB modality. Note how items in the BigBIRD datasets are very small: for all our Depth recognition experiments we used the provided masks to crop the images (see Fig. 2.11).

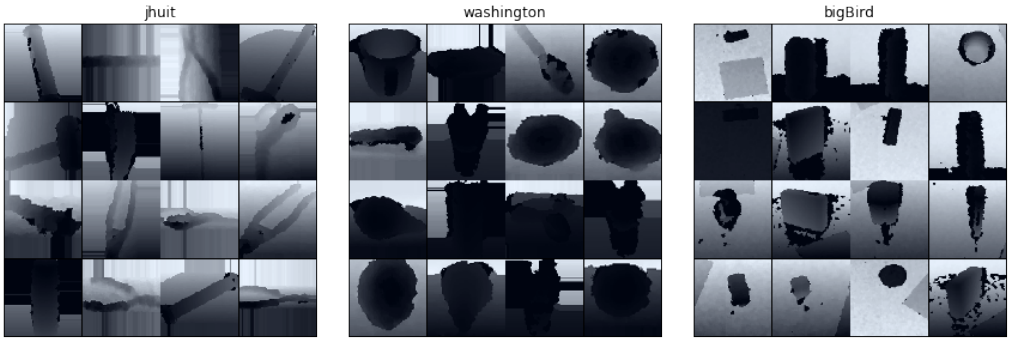


Figure 2.11. Samples from the Jhuit[87], Washington[81] and BigBIRD[140] datasets, Depth modality. For visualization purposes, the 16bit raw data has been normalized to 8bits.

2.2.3 Datasets

The datasets and the protocols used to evaluate the methods proposed in chapter 4 are listed here

We considered three datasets (samples can be seen in Fig 2.10 and 2.11): the **Washington RGB-D** [81], the **JHUIT-50** [87] and the **BigBIRD** [140] object datasets, which are the main public datasets for RGB-D object recognition. Each of them contains paired RGB and Depth images.

The first consists of $41,877$ RGB-D images organized into 300 instances divided in 51 classes. We performed experiments on the object categorization setting, where we followed the evaluation protocol defined in [81]. Following the guidelines defined by the authors[81] we crop the images with the provided bounding boxes.

The JHUIT-50 is a challenging recent dataset that focuses on the problem of fine-grained classification. It contains 50 object instances, often very similar with each other (e.g. 9 different kinds of screwdrivers). As such, it presents different recognition challenges compared to the Washington database. Here we followed the evaluation procedure defined in [87].

BigBIRD is the biggest of the datasets we considered: it contains 121 object

instances and 75.000 images. Unfortunately, it is an extremely unforgiving dataset for evaluating depth features: many objects are extremely similar, and many are boxes, which are indistinguishable without texture information. To partially mitigate this, we grouped together all classes annotated with the same first word: for example *nutrigrain apple cinnamon* and *nutrigrain blueberry* were grouped into *nutrigrain*. With this procedure, we reduced the number of classes to 61 (while keeping all of the samples). As items are quite small (check the rightmost image in Fig.2.10) , we used the object masks provided by [140] to crop around the object (results are in Fig. 2.11). Evaluation-wise, we followed the protocol defined in [87].

Chapter 3

Learning to see across domains

This chapter presents different strategies that allow us to deal with the Unsupervised Domain Adaptation problem. Three methods are presented: AutoDIAL, SBADA-GAN and ADAGE. The first uses a novel layer, the Domain Alignment layer, which structurally enforces statistic similarity between domains by applying a transformations that maps all data sources into a predefined distribution. The second is GAN[52] based method which tackles the domain shift issue in image space. The last method tweaks both the images and the feature representation to have all domains lie in one common, agnostic, subspace.

3.1 Feature Alignment: AutoDIAL

One of the main challenges for developing visual recognition systems working in the wild is to devise computational models immune from the domain shift problem, i.e. accurate when test data are drawn from a (slightly) different data distribution than training samples. In the last decade, several research efforts have been devoted to devise algorithmic solutions for this issue. Recent attempts to mitigate domain shift have resulted into deep learning models for domain adaptation which learn domain-invariant representations by introducing appropriate loss terms, by casting the problem within an adversarial learning framework or by embedding into deep network specific domain normalization layers. This section describes a novel approach for unsupervised domain adaptation. Similarly to previous works we propose to align the learned representations by embedding them into appropriate network feature normalization layers. Opposite to previous works, our Domain Alignment Layers *are designed not only to match the source and target feature distributions but also to* automatically *learn the degree of feature alignment required at different levels of the deep network. Differently from most previous deep domain adaptation methods, our approach is able to operate in a multi-source setting. Thorough experiments on four publicly available benchmarks confirm the effectiveness of our approach.*

The quest to create intelligent systems able to adapt to varying environmental conditions is a deep-seated human pursuit. Focusing on computer vision, adaptation is a crucial aspect, as visual recognition systems are required to operate under different conditions, corresponding to varying illuminations, changing point of view, diverse image resolution, etc. Therefore, it is not surprising that over the years researchers have devoted significant efforts into developing algorithms and techniques for domain adaptation.

Domain adaptation methods attempt to address the so-called *domain shift* problem, i.e. the fact that in many real world applications there is a discrepancy between the distributions of training and test samples. This distribution mismatch adversely affects the performance of visual recognition models. Despite the significant advances brought by deep learning, several works have shown that the domain shift problem is alleviated when using deep feature representations, but not solved [33].

In the last few years the research community has devoted significant efforts in addressing this issue. In particular, several previous studies have considered the problem of *unsupervised domain adaptation*, where only the training samples from a source domain have labels and the goal is to predict the labels of test samples in a target domain. This problem is highly relevant in many applications as annotating data is not only a costly and time-consuming operation but, in some cases, it is not possible at all. Several approaches have been proposed for unsupervised domain adaptation, both considering hand-crafted features [71, 50, 51, 98, 39] and learned deep representations [99, 156, 42, 101, 47, 93, 19]. Focusing on deep architectures, several strategies have been proposed in the last few years. Some methods attempt to reduce the discrepancy among source and target distributions by learning features which are invariant to the domain shift. Earlier works propose to minimize the Maximum Mean Discrepancy (MMD) [99, 101]: the distributions of the learned source and target representations are optimized to be as similar as possible by minimizing the distance between their mean embeddings. Other studies [156, 42] consider a domain-confusion loss, i.e. learn domain-agnostic representations by introducing an auxiliary classifier predicting if a sample comes from the source or from the target domain. More recently, researchers have also investigated alternative directions. For instance, Ghifary *et al.* [47] designed an encoder-decoder network

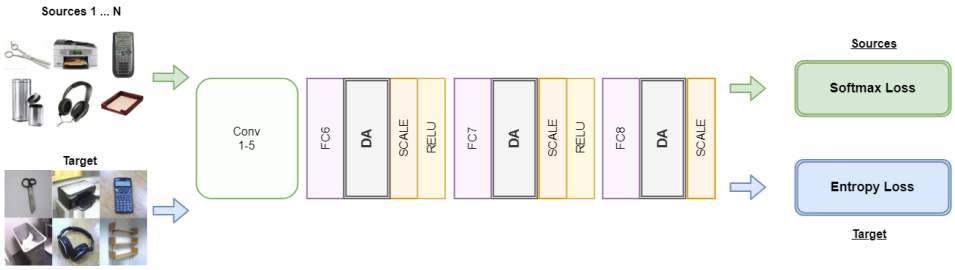


Figure 3.1. AutoDIAL as applied on AlexNet [80]. Source and target images are provided as input to the network. After passing through the first layers, they enter our DA-layer where source and target distributions are aligned. The DA-Layer is shown in detail in Fig. 3.2. After flowing through the whole network, source samples contribute to a Softmax loss, while target samples contribute to an Entropy loss, which promotes classification models which maximally separate unlabeled data. Note that we use multiple DA-layers to align learned feature representations at different levels.

to jointly learn source labels and reconstruct unsupervised target images. Some methods tackle the domain shift problem by transforming images using Generative Adversarial Networks (GANs) [13, 137, 149, 70, 128]. [14, 93, 19], Other works investigate the possibility of reducing the distribution discrepancy among source and target domain by embedding into a neural network specific distribution normalization layers [93, 19, 107, 106]. These methods are especially interesting since, differently from most previous works learning deep domain-invariant features, they do not consider in the optimization additional loss terms for minimizing the distance among distributions (e.g. MMD, domain-confusion), thus they do not require to tune the associated hyper-parameters. Furthermore, opposite to previous approaches based on GANs, the optimization process has more favorable convergence properties.

The approach described in this section belongs to the latter category of methods. Specifically, we introduce novel *Domain Alignment* layers (DA-layers) (Fig. 3.1 and 3.2) which are embedded at different levels of a deep architecture to align the learned source and target feature distributions to a canonical one. Different from previous works based on distribution normalization layers [93, 107, 106] which decide *a priori* which layers should be adapted, we endow our DA-layers with the ability to *automatically* learn the degree of alignment that should be pursued at different levels of the network. This is to our knowledge the first work that tries to pursue this objective. Furthermore, inspired by [54], we propose to leverage information from the target domain to construct a prior distribution on the network parameters, biasing the learned solution towards models that are able to separate well the classes in the target domain (see subsection 3.1.1). Our DA-layers and the considered prior distribution work in synergy during learning: the first aligning the source and target feature distributions, the second encouraging the network to learn features that lead to maximally separated target classes. We call our algorithm AutoDIAL– DomaIn Alignment Layers with Automatic alignment parameters. Our extensive experimental evaluation demonstrates that AutoDIAL greatly alleviates the domain discrepancy and outperforms state of the art techniques on popular domain adaptation benchmarks: Office-31 [129], Office-Caltech [51], the Caltech-ImageNet setting of the Cross-Dataset Testbed[154] and the recent Office-Home [161] dataset.

This section extends our earlier works [18, 19] through considering a multi-source domain adaptation setting. In particular we modify the proposed DA-layers in order to enable feature normalization of data coming from multiple domains. Moreover, we enrich the related works, provide additional details on the proposed

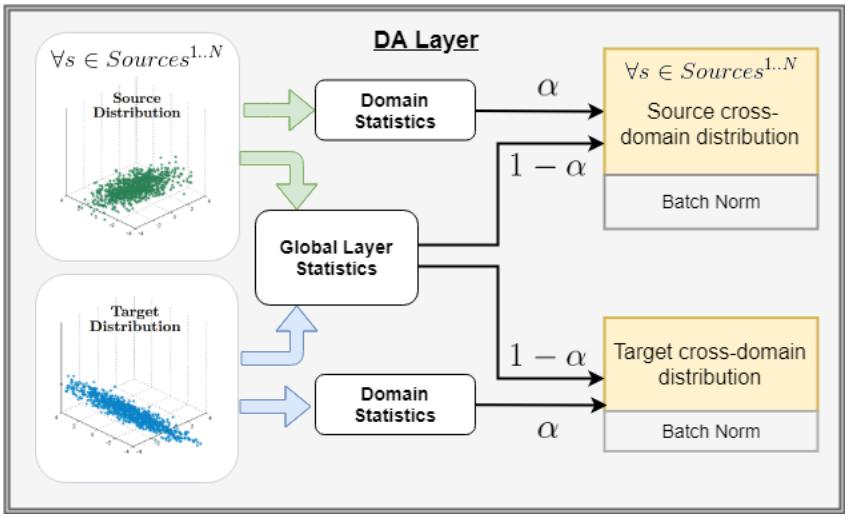


Figure 3.2. The DA-layer learns the global statistics of all domains and normalizes the source and target mini-batches according to the computed mean and variance, different for each domain (see subsection 3.1.1). The amount by which each distribution is influenced by the global one and therefore the degree of domain alignment, depends on a parameter, $\alpha \in [0.0, 1.0]$, which is also automatically learned.

method and significantly expand our experimental results and analysis considering an additional large scale dataset (i.e. the Office-Home [161] dataset) and more recent deep architectures.

To summarize, our contribution in this work are threefold. First, we present an approach for unsupervised domain adaptation, based on the introduction of DA-layers to explicitly address the domain shift problem, which act in synergy with an entropy loss which exploits unsupervised target data during learning. Our solution simultaneously aligns feature representations and learns where and to which extent adaptation should take place. We also show that the proposed approach can be naturally extended to a multi-source setting. Second, in contrast to previous works optimizing domain discrepancy regularization terms [101, 156, 42, 99], our DA-layers do not require any additional hyper-parameters. Third, we perform an extensive experimental analysis on four different benchmarks. We find that our unsupervised domain adaptation approach outperforms state-of-the-art methods and can be applied to different CNN architectures, consistently improving their performance in domain adaptation problems.

The remainder of this section is organized as follows. The proposed DA-layers and our novel unsupervised domain adaptation approach are described in subsection 3.1.1. The experimental results and analysis are provided in subsection 3.1.2, and we conclude in subsection 3.1.3.

3.1.1 Automatic DomaIn Alignment Layers

Let $\mathcal{X}$ be the input space (e.g. images) and $\mathcal{Y}$ the output space (e.g. image categories) of our learning task. In the typical Unsupervised Domain Adaptation (UDA) setting, we are given labeled data points from a *source* domain and the goal is to train a classifier for data points coming from a *target* domain. A particular variant of UDA, which will be addressed in this section, considers *multiple* source domains instead of a single one. Source and target distributions are in general different and unknown, but

we are provided with an *i.i.d* sample $\mathcal{S} = \{(x_1, y_1, d_1), \ldots, (x_n, y_n, d_n)\} \subset \mathcal{X} \times \mathcal{Y} \times \mathcal{D}$ from the source domains and an unlabeled *i.i.d* sample $\mathcal{T} = \{\hat{x}_1, \ldots, \hat{x}_m\} \subset \mathcal{X}$ from the target domain, where $\mathcal{D}$ is the set of source domain labels.

Our goal is to estimate a predictor from $\mathcal{S}$ and $\mathcal{T}$ that can be used to classify sample points from the target domain. This task is particularly challenging because on one hand we lack direct observations of labels from the target domain and on the other hand the discrepancy between the source and target domain distributions prevents a predictor trained on $\mathcal{S}$ to be readily applied to the target domain.

A number of state of the art methods try to reduce the domain discrepancy by performing some form of alignment at the feature or classifier level. In particular, the recent, most successful methods try to *couple* the training process and the domain adaptation step within *deep* neural architectures [42, 101, 99], as this solution enables alignments at different levels of abstraction. The approach we propose embraces the same philosophy, while departing from the assumption that domain alignment can by pursued by applying the *same* predictor to the source and target domains. This is motivated by an impossibility theorem [8], which intuitively states that no learner relying on the *covariate shift* hypothesis, and achieving a low discrepancy between the source and target unlabeled distributions, is guaranteed to succeed in domain adaptation without further relatedness assumptions between training and target distributions. For this reason, we assume that the source and target predictors are in general *different* functions. Nonetheless, all predictors depend on a common parameter θ belonging to a set Θ, which couples them explicitly, while not being directly involved in the alignment of the source and target domains. This contrasts with the majority of state of the art methods that augment the loss function used to train their predictors with a regularization term penalizing discrepancies between source and target representations (see, e.g. [42, 101, 99]). The perspective we take is different and is close in spirit to AdaBN [93]. It consists in hard-coding the desired domain-invariance properties into the source and target predictors through the introduction of so-called *Domain-Alignment layers* (DA-layers). Moreover, we sidestep the problem of deciding which layers should be aligned, and to what extent, by endowing the architecture with the ability to *automatically* tune the degree of alignment that should be considered in each domain-alignment layer. The rest of this section is devoted to providing the details of our method, which extends our previous work [18] to the case of multiple source domains.

Domain-Specific Predictors

Each domain is associated with a predictor that is implemented as a deep neural network and each domain-specific predictor is applied only to sample points from the corresponding domain. All predictors are actually almost identical, as they share the same structure and the same weights (given by the parameter θ). However, the networks contain also a number of special layers, the DA-layers , which implement a domain-specific operation. Indeed, the role of such layers is to apply a data transformation that aligns the observed input distribution with a reference distribution. Since in general the input distributions of the domain-specific predictors differ, while the reference distribution stays the same, we have that the predictors undergo different transformations in the corresponding DA-layers . Consequently, the source and target predictors de facto implement different functions, which is important for the reasons given in subsec. 3.1.1.

The actual implementation of our DA-layers is inspired by AdaBN [93], where Batch Normalization layers are used to independently align source and target distributions to a standard normal distribution, by matching the first- and second-

order moments. The approach they propose consists in training on the source domain a network having BN-layers, thus obtaining the source predictor, and deriving the target predictor as a post-processing step, which re-estimates the BN statistics using target samples only. Accordingly, the source and target predictors share the same network parameters but have different BN statistics, thus rendering the predictors different functions.

The approach we propose sticks to the same idea of using BN-layers to align domains, but we introduce fundamental changes. One limitation of AdaBN is that the target observations have no influence on the network parameters, as they are not observed during training. Our approach overcomes this limitation by coupling the network parameters to target and source observations at training time. This is achieved in two ways: first we introduce an entropy-based prior distribution for the network parameters based on the target observations; second, we endow the architecture with the ability of learning the degree of adaptation by introducing a parametrized, cross-domain bias to the input distribution of each domain-specific DA-layer . The rest of this subsection is devoted to describe the new layer, while we defer to the next subsection the description of the prior distribution.

DA-layer . The goal of our DA-layers is to align the input distribution to a reference distribution. Here, we consider the case of having a standard normal as reference distribution, but other distributions can be considered [19]. Under this assumption, our layer takes the same form of Batch Normalization, but as opposed to BN, which computes first and second-order moments from the input distribution – in our case samples from the same domain in the mini-batch –, we let the latter statistics to be contaminated by samples from the other domains, thus introducing a cross-domain bias. Since the domain-specific predictors share the same network topology, each DA-layer in one predictor has a matching DA-layer in all other domain-specific predictors. Let x denote an input to the DA-layer of a given domain for a given feature channel and spatial location. Assume q to be the distribution of x and assume $\bar{q}$ to be the distribution of inputs to all matching DA-layers . Let $q_\alpha = \alpha q + (1-\alpha)\bar{q}$ be the cross-domain distribution mixed by a factor $\alpha \in [0,1]$, which ranges from considering only a domain-specific distribution q if $\alpha = 1$, to considering the mixture distribution $\bar{q}$ of all domains if $\alpha = 0$. Then, the output of the DA-layer is given by

$$\texttt{DA}(x;\alpha) = \frac{x - \mu_\alpha}{\sqrt{\epsilon + \sigma_\alpha^2}}, \tag{3.1}$$

where $\epsilon > 0$ is a small number to avoid numerical instabilities in case of zero variance, $\mu_\alpha = \texttt{E}_{x\sim q_\alpha}[x]$, $\sigma_\alpha^2 = \texttt{Var}_{x\sim q_\alpha}[x]$. Akin to BN, we estimate the statistics based on the mini-batch and derive similarly the gradients through the statistics (see Supplementary Material). Note that in this work we assume the α parameter to be learned and shared between matching DA-layers of each domain-specific predictor. However, α's can differ across DA-layers within the same predictor.

The rationale behind the introduction of the mixing factor α is that we can move from having an independent alignment of the domains akin to AdaBN, when $\alpha = 1$, to having a coupled normalization when $\alpha = 0$. In the former case the DA-layer computes different functions in each domain-specific predictor and is equivalent to considering a full degree of domain alignment. The latter case, instead, yields the same function in each predictor thus transforming the domains equally, which yields no domain alignment. Since the mixing parameter α is not fixed a priori but learned during the training phase, we obtain as a result that the network can

decide how strong the domain alignment should be at each level of the architecture where DA-layer is applied. More details about the actual CNN architectures used to implement the domain-specific predictors are given in subsection 3.1.2.

Training

During the training phase we estimate the parameter θ, which holds the neural network weights shared by the domain predictors including the mixing factors αs pertaining to the DA-layers , using the observations $\mathcal{S}$ from the source domains and $\mathcal{T}$ from the target domain. As we stick to a discriminative model, the unlabeled target observations cannot be employed to express the data likelihood. However, we can exploit $\mathcal{T}$ to construct a prior distribution of the parameter θ. Accordingly, we shape a posterior distribution of θ given the observations $\mathcal{S}$ and $\mathcal{T}$ as

$$P(\theta|\mathcal{S},\mathcal{T}) \propto P(y_{\mathcal{S}}|x_{\mathcal{S}},d_{\mathcal{S}},\theta)P(\theta|\mathcal{T})\,, \tag{3.2}$$

where $y_{\mathcal{S}} = \{y_1,\ldots,y_n\}$ and $x_{\mathcal{S}} = \{x_1,\ldots,x_n\}$ collect the labels and data points of the observations in $\mathcal{S}$, respectively, and $d_{\mathcal{S}} = \{d_1,\ldots,d_n\}$ collects the source domain labels. For notational convenience, we dropped dependences that are induced by DA-layers only and we will keep this simplification in the rest of the section (e.g. the prior term in (3.2) depends also on $x_{\mathcal{S}}$ and $d_{\mathcal{S}}$ and the likelihood term depends also on $\mathcal{T}$). The posterior distribution is maximized over Θ to obtain a maximum a posteriori estimate $\hat{\theta}$ of the parameter used in the source and target predictors:

$$\theta^* \in \arg\max_{\theta\in\Theta} P(\theta|\mathcal{S},\mathcal{T})\,. \tag{3.3}$$

The term $P(y_{\mathcal{S}}|x_{\mathcal{S}},d_{\mathcal{S}},\theta)$ in (3.2) represents the likelihood of θ with respect to observations from the source domains, while $P(\theta|\mathcal{T})$ is the prior term depending on the target domain sample, which acts as a regularizer in the classical learning theory sense. Both terms actually, depend on all domains due to the cross-domain statistics that we have in our DA-layers for $0 \leq \alpha < 1$ and are estimated from observations from the source *and* target domains.

The likelihood decomposes into the following product over observation, due to the data sample *i.i.d.* assumption:

$$P(y_{\mathcal{S}}|x_{\mathcal{S}},d_{\mathcal{S}},\theta) = \prod_{i=1}^{n} f^{\theta}_{d_i}(y_i;x_i)\,, \tag{3.4}$$

where $f^{\theta}_{d_i}(y_i;x_i)$ is the probability that sample point x_i takes label y_i according to the predictor for domain d_i.

Before delving into the details of the prior term, we would like to remark on the absence of an explicit component in the probabilistic model that tries to align the source and target domains. This is because in our model the domain-alignment step is taken over by each predictor, independently, via the domain-alignment layers as shown in the previous subsection.

Prior distribution. The prior distribution of the parameter θ shared by the source and target predictors is constructed from the observed, target data distribution. This choice is motivated by the theoretical possibility of squeezing more bits of information from unlabeled data points insofar as they exhibit low levels of class overlap [120]. Accordingly, it is reasonable to bias a priori a predictor based on the

degree of label uncertainty that is observed when the same predictor is applied to the target samples. Uncertainty in this sense can be measured for an hypothesis θ in terms of the following empirical entropy of the target predictor

$$h(\theta|\mathcal{T}) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{y\in\mathcal{Y}} f_t^\theta(y;\hat{x}_i)\log f_t^\theta(y;\hat{x}_i)\,, \tag{3.5}$$

where $f_t(y;\hat{x}_i)$ represents the probability that sample point $\hat{x}_i$ takes label y according to the target predictor.

It is now possible to derive a prior distribution $P(\theta|\mathcal{T})$ in terms of the label uncertainty measure $h(\theta|\mathcal{T})$ by requiring the prior distribution to maximize the entropy under the constraint $\int h(\theta|\mathcal{T})P(\theta|\mathcal{T})d\theta = \varepsilon$, where the constant $\varepsilon > 0$ specifies how small the label uncertainty should be on average. This yields a concave, variational optimization problem with solution:

$$P(\theta|\mathcal{T}) \propto \exp\left(-\lambda\, h(\theta|\mathcal{T})\right)\,, \tag{3.6}$$

where λ is the Lagrange multiplier corresponding to ε. The resulting prior distribution satisfies the desired property of preferring models that exhibit well separated classes (i.e. having lower values of $h(\theta|\mathcal{T})$), thus enabling the exploitation of the information content of unlabeled target observations within a discriminative setting [54].

Prior distributions of this kind have been adopted also in other works [101] in order to exploit more information from the target distribution, but have never been used before in conjunction to explicit domain alignment methods (i.e. not based on additional regularization terms such as MMD and domain-confusion) like the one we are proposing.

Inference. Once we have estimated the optimal network parameters θ^* by solving (3.3), we can remove the dependence of the target predictor on $\mathcal{T}$ on source domain observations. In fact, after fixing θ^*, the input distribution to each DA-layer also becomes fixed, and we can thus compute and store the required statistics once at all, akin to standard BN.

Implementation Notes

DA-layer can be implemented as a mostly straightforward modification of standard Batch Normalization. We refer the reader to the supplementary material for a complete derivation. In our implementation in particular, we treat each set of matching DA-layers in the domain specific predictors as a single network layer which simultaneously computes the normalization functions in Equation (3.1) for all domains and learns the α parameter. During training, each DA-layer receives the domain labels corresponding to its input samples, and is thus able to easily differentiate between them. Similarly to standard BN, we keep separate moving averages of each domain's statistics and for the global statistics. Since α is constrained to $[0, 1]$, we clip its value in the allowed range in each forward pass of the network.

By replacing the optimization problem in (3.3) with the equivalent minimization of the negative logarithm of $P(\theta|\mathcal{S},\mathcal{T})$ and combining (3.2), (3.4), (3.5) and (3.6)

we obtain a loss function $L(\theta) = L^s(\theta) + \lambda L^t(\theta)$, where:

$$L^s(\theta) = -\frac{1}{n}\sum_{i=1}^{n} \log f^{\theta}_{d_i}(y_i; x_i)\,,$$
$$L^t(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{y\in\mathcal{Y}} f^{\theta}_t(y; \hat{x}_i) \log f^{\theta}_t(y; \hat{x}_i)\,.$$

The term $L^s(\theta)$ is the standard log-loss applied to the source samples, while $L^t(\theta)$ is an entropy loss applied to the target samples. The second term can be implemented by feeding $f^{\theta}_t(y; x^t_i)$ to both inputs of a cross-entropy loss layer, where supported by the deep learning toolkit of choice.

3.1.2 Experiments

In this section we extensively evaluate our approach and compare it with state of the art unsupervised domain adaptation methods. We also provide a detailed analysis of the proposed framework, performing a sensitivity study and demonstrating empirically the effect of our contributions. Note that all the results in the following are reported as averages over five training/testing runs. Full details on the datasets and the used protocol can be found in subsection 2.1.3.

Networks and Training

We apply the proposed method to three widely used CNNs architectures, i.e. AlexNet [80], VGGf[23], and Inception-BN [73]. We train our networks using mini-batch stochastic gradient descent with momentum, as implemented in the Caffe library, using the following meta-parameters: weight decay 5×10^{-4}, momentum 0.9, initial learning rate 10^{-3}. We augment the input data by scaling all images to 256×256 pixels, randomly cropping 227×227 pixels (for AlexNet and VGGf) or 224×224 pixels (Inception-BN) patches and performing random flips. In all experiments we choose the parameter λ by cross-validation on the source set according to the protocol in [101].

AlexNet [80] and VGGf [23] are two well-know architectures with five convolutional and three fully-connected layers, denoted as `fc6`, `fc7` and `fc8`. The outputs of `fc6` and `fc7` are commonly used in the domain-adaptation literature as pre-trained feature representations [33, 143] for traditional machine learning approaches. In our experiments we modify both architectures by appending a DA-layer to each fully-connected layer. Differently from the original networks, we *do not* perform dropout on the outputs of `fc6` and `fc7`.

We initialize the network parameters from a publicly-available model trained on the ILSVRC-2012 data, we freeze all the convolutional layers, and increase the learning rate of `fc8` by a factor of 10. During training, each mini-batch contains a number of source and target samples proportional to the size of the corresponding dataset, while the batch size remains fixed at 256. We train for a total of 60 epochs (where "epoch" refers to a complete pass over the source set), reducing the learning rate by a factor 10 after 54 epochs.

Inception-BN [73] is a very deep architecture obtained by concatenating "inception" blocks. Each block is composed of several parallel convolutions with batch normalization and pooling layers. To apply the proposed method to Inception-BN, we replace each batch-normalization layer with a DA-layer . Similarly to AlexNet, we initialize the network's parameters from a publicly-available model trained on

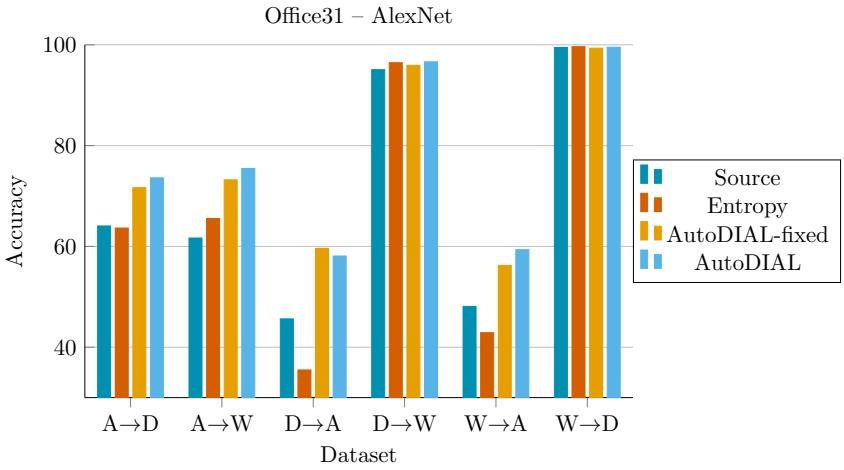


Figure 3.3. Accuracy on the Office31 dataset when considering different architectures based on AlexNet.

the ILSVRC-2012 data and freeze the first three inception blocks. The α parameter is also fixed to a value of 0 in the DA-layers of the first three blocks, which is equivalent to preserving the original batch normalization layers. Due to GPU memory constraints, we use a much smaller batch size than for AlexNet and fix the number of source and target samples in each batch to, respectively, 32 and 16. In the Office-31 experiments we train for 1200 iterations, reducing the learning rate by a factor 10 after 1000 iterations. In the Cross-Dataset Testbed experiments we train for 2000 iterations, reducing the learning rate after 1500.

Analysis of the proposed method

Here we conduct an in-depth analysis of the full proposed approach, evaluating the impact of our three main contributions: i) aligning features by matching source and target distributions to a reference one; ii) learning the adaptation coefficients α; iii) applying an entropy-based regularization term.

As a first set of experiments, we perform an ablation study on the Office31 dataset and report the results in Fig. 3.3.

Specifically, we compare the performance of four variations of the AlexNet network: trained on source data only (Source); with the addition of the entropy loss (Entropy); with DA-layers and α fixed to 1 (AutoDIAL-fixed); with DA-layers and learned α (AutoDIAL). Here the advantage of learning α is evident, as AutoDIAL outperforms AutoDIAL-fixed in all but one of the experimental settings. Interestingly, the addition of the entropy term by itself seems to have mixed effects on the final accuracy: in D→A and W→A in particular, the performance drastically decreases in Entropy compared to Source. This is not surprising as these two settings correspond to cases where the number of labeled source samples is very limited and the domain shift is more severe. However, using DA-layers in conjunction with the entropy loss always leads to a sizable performance increase. These results confirms the validity of our contribution: the entropy regularization term is especially beneficial when source and target data representations are aligned.

In Fig. 3.4 we plot the values of α learned by the DA-layers in AutoDIAL – AlexNet and AutoDIAL – Inception-BN on the Office31 dataset. In both networks we observe that lower layers tend to learn values closer to 1, i.e. require an higher degree

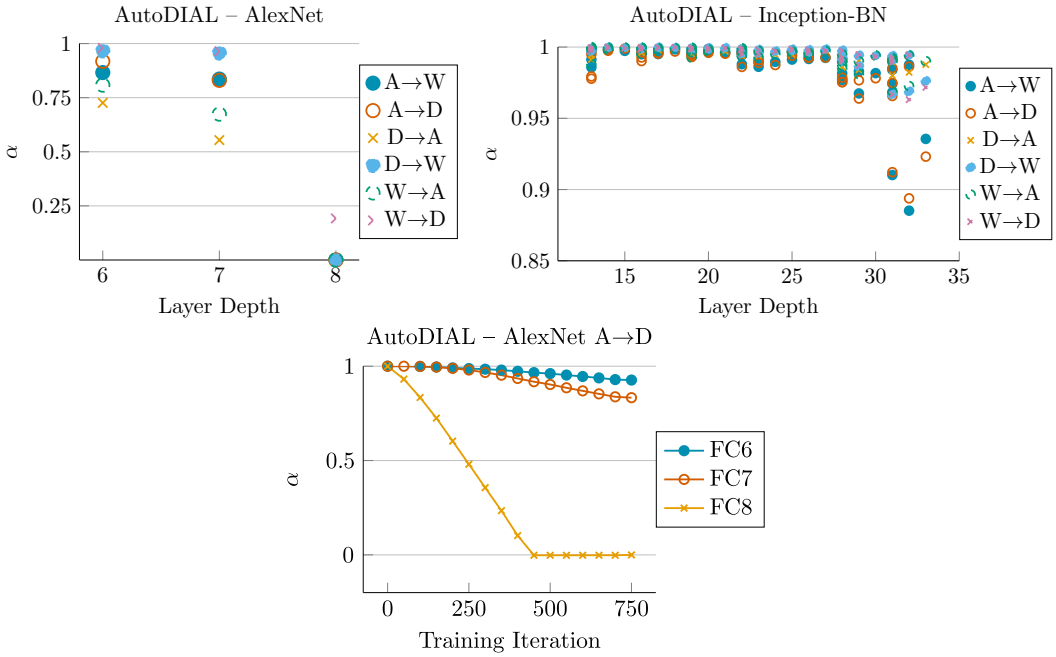


Figure 3.4. α parameters learned on the Office31 dataset, plotted as a function of layer depth (left and center) and training iteration (right).

of adaptation compared to the layers closer to the classifier. This behavior, however, seems to be more pronounced in AutoDIAL – AlexNet compared to AutoDIAL – Inception-BN. Our results agree with recent findings in the literature [2], according to which lower layers in a network are subject to domain shift even more than the very last layers. During training, the α are able to converge to their final values in a few iterations (Fig. 3.4, right).

Comparison with State of the Art methods

In this section we compare our approach with state-of-the art deep domain adaptation methods. We first consider the Office-31 dataset. The results of our evaluation, obtained embedding the proposed DA-layers in the AlexNet, VGGf and the Inception-BN networks as explained in subsection 3.1.2, are summarized in Tables 3.1,

Method	Source / Target	Amazon DSLR	Amazon Webcam	DSLR Amazon	DSLR Webcam	Webcam Amazon	Webcam DSLR	Average
AlexNet – source [80]		63.8	61.6	51.1	95.4	49.8	99.0	70.1
DDC [159]		64.4	61.8	52.1	95.0	52.2	98.5	70.6
DAN [99]		67.0	68.5	54.0	96.0	53.1	99.0	72.9
ReverseGrad [42]		67.1	72.6	54.5	96.4	52.7	99.2	72.7
DRCN [47]		66.8	68.7	56.0	96.4	54.9	99.0	73.6
RTN [101]		71.0	73.3	50.5	**96.8**	51.0	**99.6**	73.7
JAN [100]		71.8	74.9	**58.3**	96.6	55.0	99.5	76.0
AutoDIAL – AlexNet		**73.6**	**75.5**	58.1	96.6	**59.4**	99.5	**77.1**

Table 3.1. AlexNet-based approaches on Office31 / full sampling protocol.

Method	Source / Target	Amazon DSLR	Amazon Webcam	DSLR Amazon	DSLR Webcam	Webcam Amazon	Webcam DSLR	Average
VGGf – source [23]		61.3	58.3	46.5	93.8	47.7	98.7	67.7
GFK [161]		48.6	52.1	41.8	89.2	49.0	93.2	62.3
TCA [161]		51.0	49.4	48.1	93.1	48.8	96.8	64.5
CORAL [161]		54.4	51.7	48.3	96.0	47.3	98.6	66.0
JDA [161]		59.2	58.6	51.4	96.9	52.3	97.8	69.4
DAN [161]		67.0	67.8	50.4	95.9	52.3	99.4	72.1
DANN [161]		**72.9**	72.7	56.3	96.5	53.2	**99.4**	75.2
DAH-e [161]		66.3	66.2	56.0	94.6	53.9	97.0	72.3
DAH [161]		66.5	68.3	55.5	96.1	53.0	98.8	73.0
AutoDIAL – VGGf		72.2	**73.1**	**61.0**	**97.6**	**57.9**	98.3	**76.7**

Table 3.2. VGGf-based approaches on Office31 / full sampling protocol.

Method	Source / Target	Ar Cl	Ar Pr	Ar Rw	Cl Ar	Cl Pr	Cl Rw	Pr Ar	Pr Cl	Pr Rw	Rw Ar	Rw Cl	Rw Pr	Average
VGGf - source[23]		30.1	42.9	54.4	31.6	45.8	47.9	29.8	30.1	55.0	43.9	35.6	62.6	42.5
GFK [161]		21.6	31.7	38.8	21.6	34.9	34.2	24.5	25.7	42.9	32.9	29.0	50.9	32.4
TCA [161]		19.9	32.1	35.7	19.0	31.4	31.7	21.9	23.6	42.1	30.7	27.2	48.7	30.3
CORAL [161]		27.1	36.2	44.3	26.1	40.0	40.3	27.8	30.5	50.6	38.5	36.4	57.1	37.9
JDA [161]		25.3	36.0	42.9	24.5	40.2	40.9	26.0	32.7	49.3	35.1	35.4	55.4	37.0
DAN [161]		30.7	42.2	54.1	32.8	47.6	49.8	29.1	34.1	56.7	43.6	38.3	62.7	43.5
DANN [161]		33.3	43.0	54.4	32.3	49.1	49.8	30.5	38.1	56.8	44.7	42.7	64.7	44.9
DAH-e [161]		29.2	35.7	48.3	33.8	48.2	47.5	29.9	38.8	55.6	41.2	45.0	59.1	42.7
DAH [161]		31.6	40.8	51.7	34.7	51.9	52.8	29.9	**39.6**	60.7	45.0	45.1	62.5	45.5
AutoDIAL – VGGf		**35.7**	**53.7**	**61.6**	**38.9**	**58.7**	**61.3**	**37.8**	39.1	**65.8**	**48.5**	**46.2**	**70.0**	**51.4**

Table 3.3. VGGf-based approaches on Office-Home[161]. {Art (Ar), Clipart (Cl), Product (Pr), Real-World (Rw)}

Method	Source / Target	Amazon DSLR	Amazon Webcam	DSLR Amazon	DSLR Webcam	Webcam Amazon	Webcam DSLR	Average
Inception-BN – source [73]		70.5	70.3	60.1	94.3	57.9	**100.0**	75.5
AdaBN [93]		73.1	74.2	59.8	95.7	57.4	99.8	76.7
AdaBN + CORAL [93]		72.7	75.4	59.0	96.2	60.5	99.6	77.2
DDC [159]		73.2	72.5	61.6	95.5	61.6	98.1	77.1
DAN [99]		74.4	76.0	61.5	95.9	60.3	98.6	77.8
JAN [100]		77.5	78.1	**68.4**	96.4	**65.0**	99.3	80.8
AutoDIAL – Inception-BN		**82.3**	**84.2**	64.6	**97.9**	64.2	99.9	**82.2**

Table 3.4. Inception-based approaches on Office31 / full sampling protocol.

Method	Source Target	Amazon Caltech	Webcam Caltech	DSLR Caltech	Caltech Amazon	Caltech Webcam	Caltech DSLR	Average
AlexNet – source [80]		83.8	76.1	80.8	91.1	83.1	89.0	84.0
DDC [159]		85.0	78.0	81.1	91.9	85.4	88.8	85.0
DAN [99]		85.1	84.3	82.4	92.0	90.6	90.5	87.5
RTN [101]		**88.1**	86.6	84.6	93.7	**96.9**	**94.2**	**90.6**
AutoDIAL – AlexNet		87.4	**86.8**	**86.9**	**94.3**	96.3	90.1	90.3

Table 3.5. Office-Caltech results using the full protocol.

Method	Source Target	Caltech Imagenet	Imagenet Caltech
SDT [156]		–	73.6
Tommasi *et al.* [152]		–	75.4
Inception-BN – source [73]		82.1	88.4
AdaBN [93]		82.2	87.3
AutoDIAL – Inception-BN		**85.2**	**90.5**

Table 3.6. Cross-Dataset Testbed results using the protocol in [154].

3.2 and 3.4, respectively. As baselines, we consider: Deep Adaptation Networks (DAN) [99], Deep Domain Confusion (DDC) [159], the ReverseGrad network [42], Residual Transfer Network (RTN) [101], Joint Adaptation Network (JAN) [100], Deep Reconstruction-Classification Network (DRCN) [47] and AdaBN [93] with and without CORAL feature alignment [143]. The results associated to the baseline methods are derived from the original papers. When using the VGGf network we instead compare with Deep Hashing Networks[161] and the baselines reported in their work. As a reference, we further report the results obtained considering standard AlexNet, VGGf and Inception-BN networks trained only on source data.

Among the deep methods based on the AlexNet architecture, AutoDIAL – AlexNet shows the best average performance, clearly demonstrating the benefit of the proposed adaptation strategy. Similar results are found in the experiments with VGGf and Inception-BN networks, where our approach also outperforms all baselines.

Method	Source Target	Caltech Imagenet	Imagenet Caltech
SA [39]		43.7	52.0
GFK [51]		52.0	58.5
TCA [122]		48.6	54.0
CORAL [143]		66.2	74.7
Inception-BN – source [73]		82.1	88.4
AdaBN [93]		81.9	86.5
AutoDIAL – Inception-BN		**84.2**	**89.8**

Table 3.7. Cross-Dataset Testbed results using the protocol in [143].

Source: Target:	A-W Dslr	A-D Webcam	D-W Amazon	Mean
Source only	98.2	92.7	51.6	80.8
sFRAME [165]	54.5	52.2	32.1	46.3
SGF [53]	39.0	52.0	28.0	39.7
DCTN [166]	**99.6**	96.9	54.9	83.8
AdaBN (Inception-BN) [93]	94.2	**97.2**	59.3	83.6
AutoDIAL – AlexNet	97.2	95.3	62.7	**85.1**

Table 3.8. AutoDIAL results on the multi-source Office31 setting

It is interesting to compare AutoDIAL with the AdaBN method [93], as this approach also develops from a similar intuition than ours. Our results clearly demonstrate the added value of our contributions: the introduction of the alignment parameters α, together with the adoption of the entropy regularization term, produce a significant boost in performance. It is interesting to note that the relative increase in accuracy from the source-only Inception-BN to AutoDIAL – Inception-BN is higher than that from the source only AlexNet to AutoDIAL – AlexNet. The considerable success of our method in conjunction with Inception-BN can be attributed to the fact that, differently from AlexNet, this network is pre-trained with batch normalization, and thus initialized with weights that are already calibrated for normalized features.

In our second set of experiments we analyze the performance of several approaches on the Office-Caltech dataset. The results are reported in Table 3.5. We restrict our attention to methods based on deep architectures and, for a fair comparison, we consider all AlexNet-based approaches. Here we report results obtained with DDC [159], DAN [99], and the recent Residual Transfer Network (RTN) in [101]. As it is clear from the table, our method and RTN have roughly the same performance (90.6% vs 90.4% on average), while they significantly outperform the other baselines.

As a third set of experiments we evaluate our method on the 12 settings of Office-Home. As shown in table 3.3, AutoDIAL comfortably outperforms the previous state of the art on this setting.

For our multi-source experiments, we benchmark our approach on Office, testing on each of the three domains, while using the remaining two as sources (results in table 3.8). Both our variants, α free or fixed, outperform the existing state of the art in this setting.

Finally, we perform some large scale experiments on the Caltech-ImageNet subset of the Cross-Dataset Testbed of [154]. As explained above, to facilitate comparison with previous works which have also considered this dataset we perform experiments in two different settings. As baselines we consider Geodesic Flow Kernel (GFK) [51], Subspace Alignment (SA) [39]), CORAL [143], Transfer Component Analysis (TCA) [122], Simultaneous Deep Transfer (SDT) [156], and the recent method in [152]. Table 3.6 and Table 3.7 show our results.

The proposed approach significantly outperforms previous methods and sets the new state of the art on this dataset. The higher performance of our method is not only due to the use of Inception-BN but also due to the effectiveness of our contributions. Indeed, the proposed alignment strategy, combined with the adoption of the entropy regularization term, makes our approach more effective than previous adaptation techniques based on Inception-BN, i.e. AdaBN [93].

3.1.3 Conclusion

We presented AutoDIAL, a novel framework for unsupervised, deep domain adaptation. The core of our contribution is the introduction of novel Domain Alignment layers, which reduce the domain shift by matching source and target distributions to a reference one. Our DA-layers are endowed with a set of alignment parameters, also learned by the network, which allow the CNN not only to align the source and target feature representations but also to automatically decide at each layer the required degree of adaptation. Our framework exploits target data both by computing statistics in the DA-layers and by introducing an entropy loss which promotes classification models with high confidence on unlabeled samples. The results of our experiments demonstrate that our approach outperforms state of the art domain adaptation methods.

While this section focuses on the challenging problem of unsupervised domain-adaptation, our approach can be also exploited in a semi-supervised setting. Future works will be devoted to analyze the effectiveness of AutoDIAL in this scenario.

3.2 Image Alignment: SBADA-GAN

The effectiveness of GANs in producing images according to a specific visual domain has shown potential in unsupervised domain adaptation. Source labeled images have been modified to mimic target samples for training classifiers in the target domain, and inverse mappings from the target to the source domain have also been evaluated.

In this section we aim at getting the best of both worlds by introducing a symmetric mapping among domains. We jointly optimize bi-directional image transformations combining them with target self-labeling. We define a new class consistency loss that aligns the generators in the two directions, imposing to preserve the class identity of an image passing through both domain mappings. A detailed analysis of the reconstructed images, a thorough ablation study and extensive experiments on six different settings confirm the power of our approach.

The ability to generalize across domains is challenging when there is ample labeled data on which to train a deep network (source domain), but no annotated data for the target domain. To attack this issue, a wide array of methods have been proposed, most of them aiming at reducing the shift between the source and target distributions (see Sec. 2.1 for a review of previous work). An alternative is mapping the source data into the target domain, either by modifying the image representation [43] or by directly generating a new version of the source images [13]. Several authors proposed approaches that follow both these strategies by building over Generative Adversarial Networks (GANs) [52]. A similar but inverse method maps the target data into the source domain, where there is already an abundance of labeled images [158].

We argue that these two mapping directions should not be alternative, but complementary. Indeed, the main ingredient for adaptation is the ability of transferring successfully the style of one domain to the images of the other. This, given a fixed generative architecture, will depend on the application: there may be cases where mapping from the source to the target is easier, and cases where it is true otherwise. By pursuing both directions in a unified architecture, we can obtain a system more robust and more general than previous adaptation algorithms.

With this idea in mind, we designed SBADA-GAN: Symmetric Bi-directional ADAptive Generative Adversarial Network. Its features are (see Figure 3.5):

- it exploits two generative adversarial losses that encourage the network to produce target-like images from the source samples and source-like images from the target samples. Moreover, it jointly minimizes two classification losses, one on the original source images and the other on the transformed target-like source images;
- it uses the source classifier to annotate the source-like transformed target images. Such pseudo-labels help regularizing the same classifier while improving the target-to-source generator model by backpropagation;
- it introduces a new semantic constraint on the source images: the *class consistency loss*. It imposes that by mapping source images towards the target domain and then again towards the source domain they should get back to their ground truth class. This last condition is less restrictive than a standard reconstruction loss [179, 78], as it deals only with the image annotation and not with the image appearance. Still, our experiments show that it is highly effective in aligning the domain mappings in the two directions;

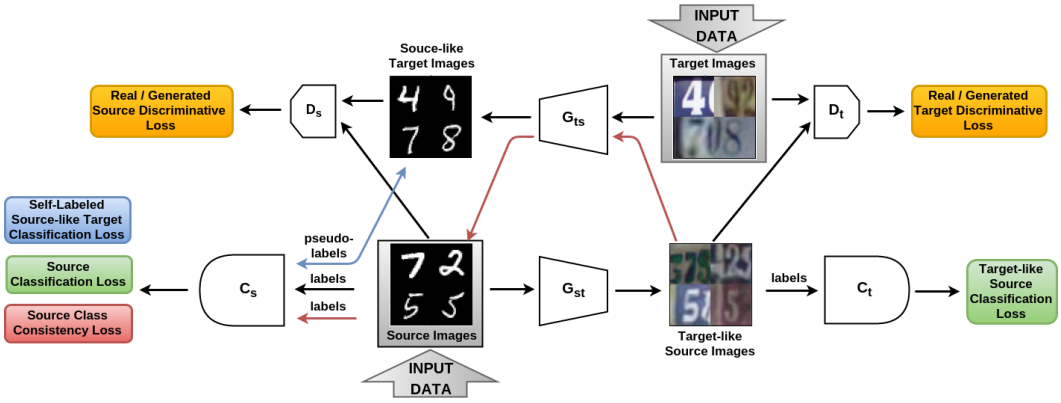


Figure 3.5. SBADA-GAN, training: the data flow starts from the Input Data arrows. The bottom and top row show respectively the source-to-target and target-to-source symmetric directions. The generative models G_{st} and G_{ts} transform the source images to the target domain and vice versa. D_s and D_t discriminate real from generated images of source and target. Finally the classifiers C_s and C_t are trained to recognize respectively the original source images and their target-like transformed versions. The bi-directional blue arrow indicates that the source-like target images are automatically annotated and the assigned pseudo-labels are re-used by the classifier C_s. The red arrows describe the class consistency condition by which source images transformed to the target domain through G_{st} and back to the source domain through G_{ts} should maintain their ground truth label.

- at test time the two trained classifiers are used respectively on the original target images and on their source-like transformed version. The two predictions are integrated to produce the final annotation.

Our architecture yields realistic image reconstructions while competing against previous state-of-the-art classifiers and exceeding them on four out of six different unsupervised adaptation settings. An ablation study showcasing the importance of each component in the architecture, and investigating the robustness with respect to its hyperparameters, sheds light on the inner workings of the approach, while providing further evidence of its value.

Model We focus on unsupervised cross domain classification. Let us start from a dataset $\boldsymbol{X}_s = \{\boldsymbol{x}_s^i, y_s^i\}_{i=0}^{N_s}$ drawn from a labeled source domain $\mathcal{S}$, and a dataset $\boldsymbol{X}_t = \{\boldsymbol{x}_t^j\}_{j=0}^{N_t}$ from a different unlabeled target domain $\mathcal{T}$, sharing the same set of categories. The task is to maximize the classification accuracy on $\boldsymbol{X}_t$ while training on $\boldsymbol{X}_s$. To reduce the domain gap, we propose to adapt the source images such that they appear as sampled from the target domain by training a generator model G_{st} that maps any source samples $\boldsymbol{x}_s^i$ to its target-like version $\boldsymbol{x}_{st}^i = G_{st}(\boldsymbol{x}_s^i)$ defining the set $\boldsymbol{X}_{st} = \{\boldsymbol{x}_{st}^i, y_s^i\}_{i=0}^{N_s}$ (see Figure 3.5, bottom row). The model is also augmented with a discriminator D_t and a classifier C_t. The former takes as input the target images $\boldsymbol{X}_t$ and target-like source transformed images $\boldsymbol{X}_{st}$, learning to recognize them as two different sets. The latter takes as input each of the transformed images $\boldsymbol{x}_{st}^i$ and learns to assign its task-specific label y_s^i. During the training procedure for this model, information about the domain recognition likelihood produced by D_t is used adversarially to guide and optimize the performance of the generator G_{st}. Similarly, the generator also benefits from backpropagation in the classifier training procedure.

Besides the source-to-target transformation, we also consider the inverse target-to-source direction by using a symmetric architecture (see Figure 3.5, top row). Here any target image $\boldsymbol{x}_t^j$ is given as input to a generator model G_{ts} transforming it to its source-like version $\boldsymbol{x}_{ts}^j = G_{ts}(\boldsymbol{x}_t^j)$, defining the set $\boldsymbol{X}_{ts} = \{\boldsymbol{x}_{ts}^j\}_{j=0}^{N_t}$. As before, the model is augmented with a discriminator D_s which takes as input both $\boldsymbol{X}_{ts}$ and $\boldsymbol{X}_s$ and learns to recognize them as two different sets, adversarially helping the generator.

Since the target images are unlabeled, no classifier can be trained in the target-to-source direction as a further support for the generator model. We overcome this issue by *self-labeling* (see Figure 3.5, blue arrow). The original source data $\boldsymbol{X}_s$ is used to train a classifier C_s. Once it has reached convergence, we apply the learned model to annotate each of the source-like transformed target images $\boldsymbol{x}_{ts}^j$. These samples, with the assigned pseudo-labels $y_{t_{self}}^j = \arg\max_y(C_s(G_{ts}(\boldsymbol{x}_t^j))$, are then used transductively as input to C_s while information about the performance of the model on them is backpropagated to guide and improve the generator G_{ts}. Self-labeling has a long track record of success for domain adaptation: it proved to be effective both with shallow models [16, 58, 113], as well as with the most recent deep architectures [134, 157, 131]. In our case the classification loss on pseudo-labeled samples is combined with our other losses, which helps making sure we move towards the optimal solution: in case of a moderate domain shift, the correct pseudo-labels help to regularize the learning process, while in case of large domain shift, the possible mislabeled samples do not hinder the performance (see Sec. 3.2.5 for a detailed discussion on the experimental results).

Finally, the symmetry in the source-to-target and target-to-source transformations is enhanced by aligning the two generator models such that, when used in sequence, they bring a sample back to its starting point. Since our main focus is classification, we are interested in preserving the class identity of each sample rather than its overall appearance. Thus, instead of a standard image-based reconstruction condition we introduce a *class consistency* condition (see Figure 3.5, red arrows). Specifically, we impose that any source image $\boldsymbol{x}_s^i$ adapted to the target domain through $G_{st}(\boldsymbol{x}_s^i)$ and transformed back towards the source domain through $G_{ts}(G_{st}(\boldsymbol{x}_s^i))$ is correctly classified by C_s. This condition helps by imposing a further joint optimization of the two generators.

Learning Here we formalize the description above. To begin with, we specify that the generators take as input a noise vector $\boldsymbol{z} \in \mathcal{N}(0,1)$ besides the images, this allows some extra degree of freedom to model external variations. We also better define the discriminators as $D_s(\boldsymbol{x})$, $D_t(\boldsymbol{x})$ and the classifiers as $C_s(\boldsymbol{x})$, $C_t(\boldsymbol{x})$. Of course each of these models depends from its parameters but we do not explicitly indicate them to simplify the notation. For the same reason we also drop the superscripts i, j.

The source-to-target part of the network optimizes the following objective function:

$$\min_{G_{st},C_t} \max_{D_t} \quad \alpha\mathcal{L}_{D_t}(D_t, G_{st}) + \beta\mathcal{L}_{C_t}(G_{st}, C_t) \ , \tag{3.7}$$

where the classification loss $\mathcal{L}_{C_t}$ is a standard *softmax cross-entropy*

$$\mathcal{L}_{C_t}(G_{st}, C_t) = \mathbb{E}_{\substack{\{\boldsymbol{x}_s,\boldsymbol{y}_s\}\sim\mathcal{S} \\ \boldsymbol{z}_s\sim noise}}[-\boldsymbol{y}_s \cdot \log(\hat{\boldsymbol{y}}_s)] \ , \tag{3.8}$$

evaluated on the source samples transformed by the generator G_{st}, so that $\hat{\boldsymbol{y}}_s = C_t(G_{st}(\boldsymbol{x}_s, \boldsymbol{z}_s))$ and $\boldsymbol{y}_s$ is the one-hot encoding of the class label y_s. For the discriminator, instead of the less robust binary cross-entropy, we followed [109] and chose

a *least square* loss:

$$\mathcal{L}_{D_t}(D_t, G_{st}) = \mathbb{E}_{\boldsymbol{x}_t \sim T}[(D_t(\boldsymbol{x}_t) - 1)^2] + \mathbb{E}_{\substack{\boldsymbol{x}_s \sim S \\ \boldsymbol{z}_s \sim noise}} [(D_t(G_{st}(\boldsymbol{x}_s, \boldsymbol{z}_s)))^2] \ . \tag{3.9}$$

The objective function for the target-to-source part of the network is:

$$\min_{G_{ts}, C_s} \max_{D_s} \ \gamma \mathcal{L}_{D_s}(D_s, G_{ts}) + \mu \mathcal{L}_{C_s}(C_s) + \eta \mathcal{L}_{self}(G_{ts}, C_s) \ , \tag{3.10}$$

where the discriminative loss is analogous to eq. (3.9), while the classification loss is analogous to eq. (3.8) but evaluated on the original source samples with $\hat{\boldsymbol{y}}_s = C_s(\boldsymbol{x}_s)$, thus it neither has any dependence on the generator that transforms the target samples G_{ts}, nor it provides feedback to it. The *self* loss is again a classification softmax cross-entropy:

$$\mathcal{L}_{self}(G_{ts}, C_s) = \mathbb{E}_{\substack{\{\boldsymbol{x}_t, \boldsymbol{y}_{t_{self}}\} \sim \mathcal{T} \\ \boldsymbol{z}_t \sim noise}} [-\boldsymbol{y}_{t_{self}} \cdot \log(\hat{\boldsymbol{y}}_{t_{self}})] \ . \tag{3.11}$$

where $\hat{\boldsymbol{y}}_{t_{self}} = C_s(G_{ts}(\boldsymbol{x}_t, \boldsymbol{z}_t))$ and $\boldsymbol{y}_{\boldsymbol{t_{self}}}$ is the one-hot vector encoding of the assigned label $y_{t_{self}}$. This loss back-propagates to the generator G_{ts} which is encouraged to preserve the annotated category within the transformation.

Finally, we developed a novel *class consistency* loss by minimizing the error of the classifier C_s when applied on the concatenated transformation of G_{ts} and G_{st} to produce $\hat{\boldsymbol{y}}_{cons} = (C_s(G_{ts}(G_{st}(\boldsymbol{x}_s, \boldsymbol{z}_s), \boldsymbol{z}_t)))$:

$$\mathcal{L}_{cons}(G_{ts}, G_{st}, C_s) = \mathbb{E}_{\substack{\{\boldsymbol{x}_s, \boldsymbol{y}_s\} \sim S \\ \boldsymbol{z}_s, \boldsymbol{z}_t \sim noise}} [-\boldsymbol{y}_s \cdot \log(\hat{\boldsymbol{y}}_{cons})] \ . \tag{3.12}$$

This loss has the important role of aligning the generators in the two directions and strongly connecting the two main parts of our architecture.

By collecting all the presented parts, we conclude with the complete SBADA-GAN loss:

$$\mathcal{L}_{SBADA-GAN}(G_{st}, G_{ts}, C_s, C_t, D_s, D_t) = \alpha \mathcal{L}_{D_t} + \beta \mathcal{L}_{C_t} + \gamma \mathcal{L}_{D_s} + \mu \mathcal{L}_{C_s} + \eta \mathcal{L}_{self} + \nu \mathcal{L}_{cons} \ . \tag{3.13}$$

Here $(\alpha, \beta, \gamma, \mu, \eta, \nu) \geq 0$ are weights that control the interaction of the loss terms. While the combination of six different losses might appear daunting, it is not unusual [15]. Here, it stems from the symmetric bi-directional nature of the overall architecture. Indeed each directional branch has three losses as it is custom practice in the GAN-based domain adaptation literature [158, 13]. Moreover, the ablation study reported in Sec. 3.2.5 indicates that the system is remarkably robust to changes in the hyperparameter values.

Testing The classifier C_t is trained on $\boldsymbol{X}_{st}$ generated images that mimic the target domain style, and is then tested on the original target samples $\boldsymbol{X}_t$. The classifier C_s is trained on $\boldsymbol{X}_s$ source data, and then tested on $\boldsymbol{X}_{ts}$ samples, that are the target images modified to mimic the source domain style. These classifiers make mistakes of different type assigning also a different confidence rank to each of the possible labels. Overall the two classification models complement each other. We take advantage of this with a simple ensemble method $\sigma C_s(G_{ts}(\boldsymbol{x}_t, \boldsymbol{z}_t)) + \tau C_t(\boldsymbol{x}_t)$ which linearly combines their probability output, providing a further gain in performance. A schematic illustration of the testing procedure is shown in Figure 3.6. We set the combination weights σ, τ through cross validation (see Sec. 3.2.2 for further details).

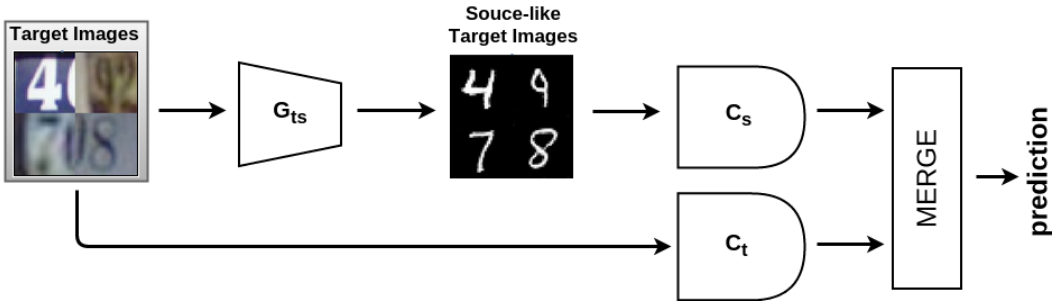


Figure 3.6. SBADA-GAN, test: the two pre-trained classifiers are applied respectively on the target images and on the transformed source-like target images. Their outputs are linearly combined for the final prediction.

3.2.1 Datasets and Adaptation Scenarios

We evaluate SBADA-GAN on several unsupervised adaptation scenarios , considering the following widely used datasets and settings (dataset details can be found in section 2.1.3):

MNIST → MNIST-M: We follow the evaluation protocol of [15, 13, 43].

MNIST ↔ USPS: We follow the evaluation protocol of [13].

SVHN ↔ MNIST: Most previous works simplified the data by considering a grayscale version, instead we apply our method to the original RGB images. We resize the MNIST images to 32×32 pixels and use the protocol by [15, 48]. We also test SBADA-GAN on a traffic sign scenario.

Synth Signs → GTSRB: Both databases contain samples from 43 classes, thus defining a larger classification task than that on the 10 digits. We adopt the protocol proposed in [59].

3.2.2 Implementation details

We composed SBADA-GAN starting from two symmetric GANs, each with an architecture[1] analogous to that used in [13].

The model is coded[2] in python and we ran all our experiments in the Keras framework [28]. We use the ADAM [79] optimizer with learning rates for the discriminator and the generator both set to 10^{-4}. The batch size is set to 32 and we train the model for 500 epochs not noticing any overfitting, which suggests that further epochs might be beneficial. The α and γ loss weights (discriminator losses) are set to 1, β and μ (classifier losses) are set to 10, to prevent that generator from indirectly switching labels (for instance, transform 7's into 1's). The class consistency loss weight ν is set to 1.

All training procedures start with the self-labeling loss weight, η, set to zero, as this loss hinders convergence until the classifier is fully trained. After the model converges (losses stop oscillating, usually after 250 epochs) η is set to 1 to further increase performance. Finally the parameters to combine the classifiers at test time are $\sigma \in [0, 0.1, 0.2, \ldots, 1]$ and $\tau = (1 - \sigma)$ chosen on a validation set of 1000 random samples from the target in each different setting.

[1] See all the model details in the supplementary material.
[2] Code available at https://github.com/engharat/SBADAGAN

	MNIST→ USPS	USPS→MNIST	MNIST→MNIST-M	SVHN→MNIST	MNIST→SVHN	Synth Signs→GTSRB
Source Only	78.9	57.1 ± 1.7	63.6	60.1 ± 1.1	26.0 ± 1.2	79.0
CORAL [144]	81.7	-	57.7	63.1	-	86.9
MMD [157]	81.1	-	76.9	71.1	-	91.1
DANN [43]	85.1	73.0 ± 2.0	77.4	73.9	35.7	88.7
DSN [15]	91.3	-	83.2	82.7	-	93.1
CoGAN [96]	91.2	89.1 ± 0.8	62.0	not conv.	-	-
ADDA [158]	89.4 ± 0.2	90.1 ± 0.8	-	76.0 ± 1.8	-	-
DRCN [48]	91.8 ± 0.1	73.7 ± 0.1	-	82.0 ± 0.2	40.1 ± 0.1	-
PixelDA [13]	95.9	-	98.2	-	-	-
DTN [149]	-	-	-	84.4	-	-
TRUDA [134]	-	-	86.7	78.8	40.3	-
ATT [131]	-	-	94.2	86.2	52.8	96.2
UNIT [95]	95.9	93.5	-	90.5	-	-
DA_{ass} fix. par. [59]	-	-	89.5	95.7	-	82.8
DA_{ass} [59]	-	-	89.5	**97.6**	-	**97.7**
Target Only	96.5	99.2 ± 0.1	96.4	99.5	96.7	98.2
SBADA-GAN C_t	96.7	94.4	99.1	72.2	59.2	95.9
SBADA-GAN C_s	97.1	87.5	98.4	74.2	50.9	95.7
SBADA-GAN	**97.6**	**95.0**	**99.4**	76.1	**61.1**	96.7
GenToAdapt [132]	92.5 ± 0.7	90.8 ± 1.3	-	84.7 ± 0.9	36.4 ± 1.2	-
CyCADA [70]	94.8 ± 0.2	95.7 ± 0.2	-	88.3 ± 0.2	-	-
Self-Ensembling [40]	98.3 ± 0.1	99.5 ± 0.4	-	99.2 ± 0.3	42.0 ± 5.7	98.3 ± 0.3

Table 3.9. Comparison against previous work. SBADA-GAN C_t, C_s reports respectively the accuracies produced by the classifier trained in the target domain space, and the results produced by training in the source domain space and testing on the target images mapped to this space. SBADA-GAN reports the results obtained by a weighted combination of the softmax outputs of these two classifiers. Note that all competitors convert SVHN to grayscale, while we deal with the more complex original RGB version. The last three rows report results from online available pre-print papers.

3.2.3 Quantitative Results

Table 3.9 shows results on our evaluation settings. The top of the table reports results by thirteen competing baselines published over the last two years. The Source-Only and Target-Only rows contain reference results corresponding to the no-adaptation case and to the target fully supervised case. For SBADA-GAN, besides the full method, we also report the accuracy obtained by the separate classifiers (C_s,C_t) before the linear combination.

SBADA-GAN improves over the state of the art in four out of six settings. In these cases the advantage with respect to its competitors is already visible in the separate C_s and C_t results and it increases when considering the full combination procedure. Moreover, the gain in performance of SBADA-GAN reaches up to +8 percentage points in the MNIST→SVHN experiment. This setting was disregarded in many previous works: differently from its inverse SVHN→MNIST, it requires a difficult adaptation from the grayscale handwritten digits domain to the widely variable and colorful street view house number domain. Thanks to its bi-directionality, SBADA-GAN leverages on the inverse target to source mapping to produce highly accuracy results.

Conversely, in the SVHN→MNIST case SBADA-GAN ranks eighth out of the thirteen baselines in terms of performance. Our accuracy is on par with ADDA's [158]: the two approaches share the same classifier architecture, although the number of fully-connected neurons of SBADA-GAN is five time lower. Moreover, compared to DRCN [48], the classifiers of SBADA-GAN are shallower with a reduced number of convolutional layers. Overall here SBADA-GAN suffers of some typical drawbacks of GAN-based domain adaptation methods: although the style of a domain can be easily transferred in the raw pixel space, the generative process does not have any explicit constraint on reducing the overall data distribution shift as instead done by the alternative feature-based domain adaptation approaches. Thus, methods like

Setting	S	T map to S	S map to T	T
MNIST → USPS	0.206	0.219	0.106	0.102
MNIST → MNIST-M	0.206	0.207	0.035	0.032
MNIST → SVHN	0.206	0.292	0.027	0.012
Synth S. → GTSRB	0.105	0.136	0.128	0.154

Table 3.10. Dataset mean SSIM: this measure of data variability suggests that our method successfully generates images with not only the same style of a chosen domain, but also similar perceptual variability.

DA_{ass} [59], DTN [149] and DSN [15] deal better with the large domain gap of the SVHN→MNIST setting.

Finally, in the Synth Signs → GTSRB experiment, SBADA-GAN is just slightly worse than DA_{ass}, but outperforms all the other competing methods. The comparison remains in favor of SBADA-GAN when considering that its performance is robust to hyperparameter variations (see Sec. 3.2.5 for more details).

3.2.4 Qualitative Results

To complement the quantitative evaluation, we look at the quality of the images generated by SBADA-GAN. First, we see from Figure 3.7 how the generated images mimic the style of the chosen domain, even when going from the simple MNIST digits to the SVHN colorful house numbers.

Visually inspecting the data distribution before and after domain mapping defines a second qualitative evaluation metric. We use t-SNE [104] to project the data from their raw pixel space to a simplified 2D embedding. Figure 3.8 shows that the transformed dataset tends to replicate faithfully the distribution of the chosen final domain.

A further measure of the quality of the SBADA-GAN generators comes from the diversity of the produced images. Indeed, GAN's generators may collapse and output a single prototype that maximally fools the discriminators. To evaluate the diversity of samples generated by SBADA-GAN we choose the Structural Similarity (SSIM, [162]) that correlates well with the human perceptual similarity judgments. Its values range between 0 and 1 with higher values corresponding to more similar images. We follow the same procedure used in [119] by randomly choosing 1000 pairs of generated images within a given class. We also repeat the evaluation over all the classes and calculate the average results. Table 3.10 shows the results of the mean SSIM metric and indicates that the SBADA-GAN generated images not only mimic the same style, but also successfully reproduce the variability of a chosen domain.

3.2.5 Method Analysis

Ablation Study Starting from the core source-to-target GAN module we analyze the effect of adding all the other model parts. At first we add the symmetric target-to-source GAN model. These two parts are then combined and the domain transformation loop is closed by adding the class consistency condition. Finally the model is completed by introducing the target self-labeling procedure. We empirically test each of these model steps on the MNIST→USPS setting and report the results in Table 3.11. We see the gain achieved by progressively adding the different components, with the largest advantage obtained by the introduction of self-labeling.

(a) MNIST to USPS (b) USPS to MNIST

(c) MNIST to MNIST-M (d) MNIST-M to MNIST

(e) MNIST to SVHN (f) SVHN to MNIST

(g) Synth S. to GTSRB (h) GTSRB to Synth S.

Figure 3.7. Examples of generated digits: we show the image transformation from the original domain to the paired one as indicated under every sub-figure. For each of the (a)-(h) cases, the original/generated images are in the top/bottom row.

An analogous boost due to self-labeling is also visible in all the other experimental settings with the exception of MNIST↔SVHN, where the accuracy remains unchanged if η is equal or larger than zero. A further analysis reveals that here the recognition accuracy of the source classifier applied to the source-like transformed target images is quite low (about 65%, while in all the other settings reaches $80-90\%$), thus the pseudo-labels cannot be considered reliable. Still, using them does not hinder the overall performance.

The crucial effect of the class consistency loss can be better observed by looking at the generated images and comparing them with those obtained in two alternative cases: setting $\nu = 0$, i.e. not using any consistency condition between the two generators G_{st} and G_{ts}, or substituting our class consistency loss with the standard cycle consistency loss [179, 78] based on image reconstruction. For this evaluation we choose the MNIST→SVHN case which has the strongest domain shift and we show the generated images in Figure 3.9. When the consistency loss is not activated,

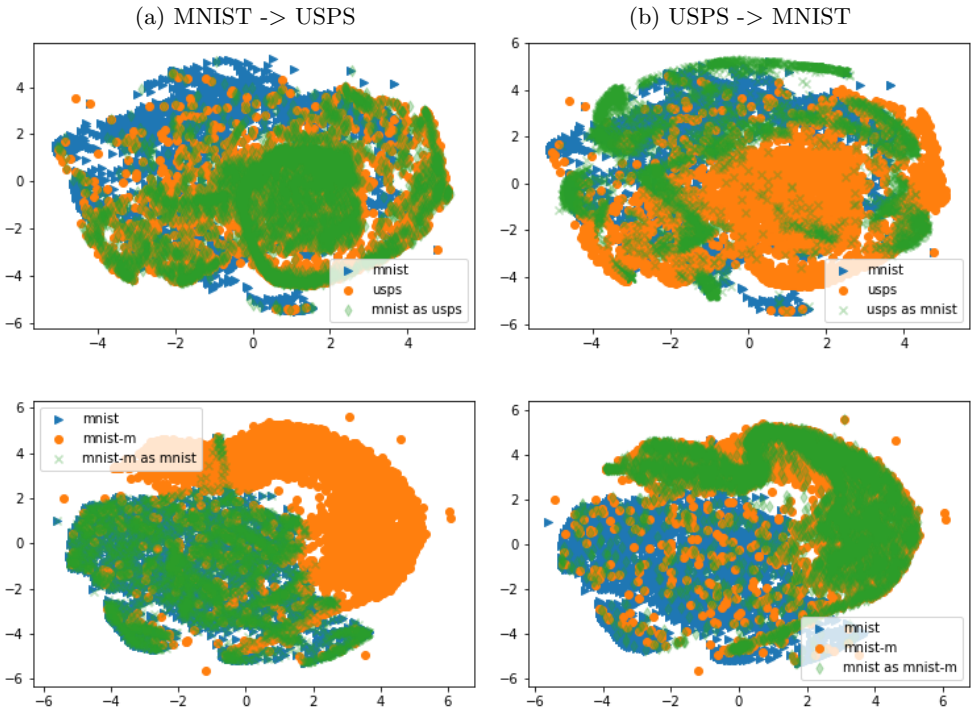


Figure 3.8. t-SNE visualization of source, target and source mapped to target images. Note how the mapped source covers faithfully the target space both in the (a),(b) case with moderated domain shift and in the more challenging (c),(d) setting.

the G_{ts} output images are realistic, but fail at reproducing the correct input digit and provide misleading information to the classifier. On the other hand, using the cycle-consistency loss preserves the input digit but fails in rendering a realistic sample in the correct domain style. Finally, our class consistency loss allows to preserve the distinct features belonging to a category while still leaving enough freedom to the generation process, thus it succeeds in both preserving the digits and rendering realistic samples.

CycleGAN vs SBADA-GAN To further clarify the difference between the two methods, we remind that CycleGAN is unsupervised and works only when transferring style across similarly shaped categories (e.g. horses→zebras), not across domains. SBADA-GAN instead deals with domains containing multiple categories. The images samples in Figure 3.9(b) are indeed obtained with CycleGAN: training on them produces an accuracy of 25.5%, much lower than the corresponding 61.1% of SBAD-GAN. Moreover, CycleGAN has a single transformed image as output, while SBADA-GAN exploits a noise vector as input producing multiple outputs for each input image: this is critical for classification as it provides variability through data augmentation, it avoids overfitting and eases generalization. For completeness we also ran an experiment on the challenging Office Dataset [130]: here both the images produced by CycleGAN and SBADA-GAN (see Figure 3.10) are given as input to a pre-trained AlexNet and the classification accuracy is respectively 52.0% and 50.7%, both lower than the reference 61.6% result produced by the baseline without

S→T GAN		T→S GAN		Class Consist.	Self Label.	Accuracy
$\mathcal{L}_{D_t}$	$\mathcal{L}_{C_t}$	$\mathcal{L}_{D_s}$	$\mathcal{L}_{C_s}$	$\mathcal{L}_{cons}$	$\mathcal{L}_{self}$	MNIST→USPS
✓	✓					94.23
		✓	✓			91.55
✓	✓	✓	✓			94.90
✓	✓	✓	✓	✓		95.45
✓	✓	✓	✓	✓	✓	97.60

Table 3.11. Analysis of the role of each SBADA-GAN component. We ran experiments by turning on the different losses of the model as indicated by the checkmarks.

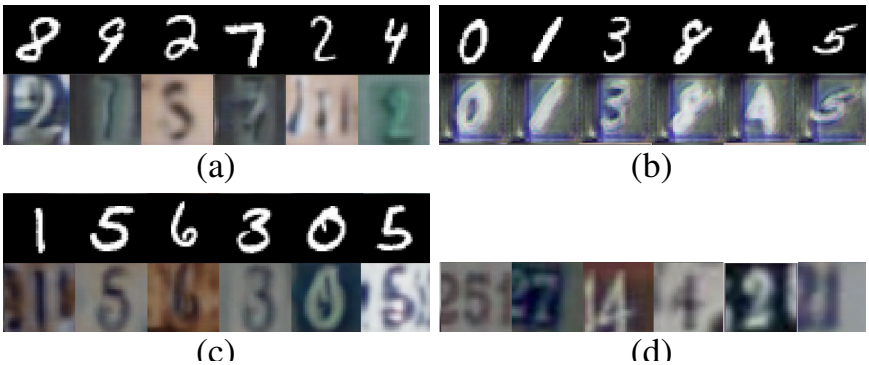


Figure 3.9. G_{ts} outputs (lower line) and their respective inputs (upper line) obtained with: (a) no consistency loss, (b) image-based cycle consistency loss [179, 78], (c) our class consistency loss. In (d) we show some real SVHN samples as a reference.

adaptation. These results confirm the known difficulty of GAN-based method to deal with domain shifts due to poses and shapes.

Robustness Study SBADA-GAN is robust to the specific choice of the consistency loss weight ν, given that it is different from zero. Changing it in $[0.1, 1, 10]$ induces a maximum variation of 0.6 percentage points in accuracy over the different settings. An analogous evaluation performed on the classification loss weights (β,μ) reveals that changing them in the same range used for ν causes a maximum overall performance variation of 0.2 percentage points. Furthermore SBADA-GAN is minimally sensitive to the batch size used: halving it from 32 to 16 samples while keeping the same number of learning epochs reduces the performance only of about 0.2 percentage points. Such robustness is particularly relevant when compared to competing methods. For instance the most recent DA_{ass} [59] needs a perfectly balanced source and target distribution of classes in each batch, a condition difficult to satisfy in real world scenarios, and halving the originally large batch size reduces by 3.5 percentage points the final accuracy. Moreover, changing the weights of the losses that enforce associations across domains with a range analogous to that used for the SBADA-GAN parameters induces a drop in performance up to 16 accuracy percentage points[3].

[3] More details are provided in the supplementary material.

(a) Amazon (b) Webcam

(c) CycleGAN (d) SBADA-GAN

Figure 3.10. CycleGAN [179] vs SBADA-GAN on the Amazon-Webcam experiment of the Office Dataset [130].

3.2.6 Conclusion

This section presented SBADA-GAN, an adaptive adversarial domain adaptation architecture that simultaneously maps source samples into the target domain and vice versa with the aim to learn and use both classifiers at test time. To achieve this, self-labeling is exploited to regularize the classifier trained on the source, and we impose a class consistency loss that improves greatly the stability of the architecture, as well as the quality of the reconstructed images in both domains.

We explain the success of SBADA-GAN in several ways. To begin with, thanks to the the bi-directional mapping we avoid deciding a priori which is the best strategy for a specific task. Also, the combination of the two network directions improves performance providing empirical evidence that they are learning different, complementary features. Our class consistency loss aligns the image generators, allowing both domain transfers to influence each other. Finally the self-labeling procedure boost the performance in case of moderate domain shift without hindering it in case of large domain gaps.

3.3 A Hybrid Approach: ADAGE

The ability to generalize across visual domains is crucial for the robustness of visual recognition systems in the wild. Several works have been dedicated to close the gap between a single labeled source domain and a target domain with transductive access to its data. In this section we focus on the wider domain generalization task involving multiple sources and seamlessly extending to unsupervised domain adaptation when unlabeled target samples are available at training time. We propose a hybrid architecture that we name ADAGE: it gracefully maps different source data towards an agnostic visual domain through pixel-adaptation based on a novel incremental architecture, and closes the remaining domain gap through feature adaptation. Both the adaptive processes are guided by adversarial learning. Extensive experiments show remarkable improvements compared to the state of the art.

As we've seen before, Domain Adaptation (DA) is at its core the quest for principled algorithms enabling the generalization of visual recognition methods. Given at least a source domain for training, the goal is to achieve recognition results as good as those achievable on source test data on *any* other target domain, in principle belonging to a different probability distribution, without having prior access to labeled images. Solving this problem will represent a major step towards one of the key goals of computer vision, i.e. having machines able to answer the question 'what do you see?' in the wild; hence, its increased popularity in the community over the last years (see section 2.1 for a review of recent work).

Since its definition [130], the most popular instantiation of the problem has assumed to have access to annotated data from a single source domain and to unlabeled data from a specific target domain. Still, there has been recently a growing interest on how to leverage over multiple source domains when unlabeled target data are available [166], and even more on how to generalize over *any* possible target domain, when it is not possible to access target data of any sort a priori [90]. Intuitively, by leveraging over multiple sources it should be possible to design algorithms able to discard the specific style of each source domain, while capturing the generic content of the visual categories contained in all domains depicting such categories.

Algorithm-wise, the community has attacked the problem with two disjoint strategies. The first is based on end-to-end architectures that minimize both a source classification and a domain shift measure. Specifically, these methods express the domain shift either in terms of difference in the domain statistics [160, 144], or of adversarial domain discrimination [43, 158], or of random walk transition probability among the samples of two domains [59]. All of them aim at aligning the feature representation learned for the domains. The other direction deals with image (instead of feature) transformation. The visual style of a domain, be it source, target or both, is modeled and transferred to images of the different one so that adaptation happens at the input level and standard classification networks can be reused without further internal modifications [128, 13]. While this second solution allows for human-understandable modifications, it should be noted that a visually pleasant transformation may not be what the network needs most to get the best possible adaptation.

In this work we focus on the scenario where multiple source domains are available at training time, with the aim of learning an *agnostic* visual domain as well as the corresponding representation able to capture the intrinsic information carried by all domains while discarding the distinctive style of each individual source. To do so, we propose a hybrid architecture that sits at the intersection of the two algorithmic

approaches outlined above, allowing to get the best of both worlds. Our intuition is that it is possible to reduce the domain shift by acting simultaneously on the image and on the feature space within an adversarial framework. This means dropping the condition of a human-understandable style transfer to the benefit of a new more network-understandable visual language. To this end, we propose an architecture where a novel incremental transformer maps the available images, guided by an adversarial loss, towards an *agnostic*, intermediate visual domain that retains the most important domain invariant information. Such agnostic images are then used as input to a multi-branch feature adaptation block, getting a substantial benefit with respect to separate feature-based and style-transfer based methods. We call our architecture Agnostic Domain Generalization (ADAGE). While in general we do not assume to have access to unlabeled target data, the architecture can be easily extended to the unsupervised multi-source domain adaptation scenario.

We test ADAGE on the domain generalization and unsupervised multi source domain adaptation settings, comparing against recent approaches [89, 176, 166]. In all experiments, for both settings, ADAGE significantly outperforms the state of the art, proving the benefit of a hybrid approach. An ablation study and visualizations of the agnostic domain images complete our experimental study.

3.3.1 Agnostic Domain Generalization

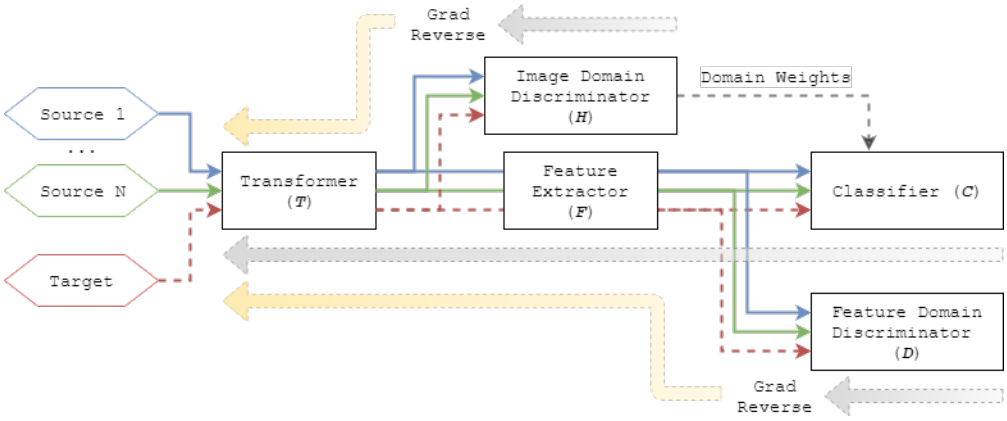


Figure 3.11. A brief description of our architecture: all samples (including target ones, in the DA setting) follow the same path in the network. Firstly they go through the Transformer T that takes as input the original images and outputs the modified ones. Then the updated samples go through the Image Domain Discriminator H and, at the same time, flow through the Feature Extractor F. Once converted to features, the samples progress both into the Classifier C and through the Feature Domain Discriminator D. We use H to get an estimate of domain similarity, which is used to bias the classification loss towards those sources which are more similar to the target. The gradient from H is inverted and flows through T driving image modifications towards domain confusion. Similarly, the gradient from D also inverted, is backpropagated through F and T so that both the feature and the image dedicated blocks benefit from a further push towards the domain agnostic space. The classification gradient travels through the whole network, excluding H and D.

We assume to observe $i = 1 \dots S$ source domains with the ith domain containing N_i labeled instances $\{x_j^i, y_j^i\}_{j=1}^{N_i}$, where x_j^i is the jth input image and $y_j^i \in \{1 \dots M\}$ is the class label. In addition we also have an unlabeled target domain whose data $\{x_j^t\}_{j=1}^{N_t}$ might (DA) or might not (DG) be provided at training time. All the source

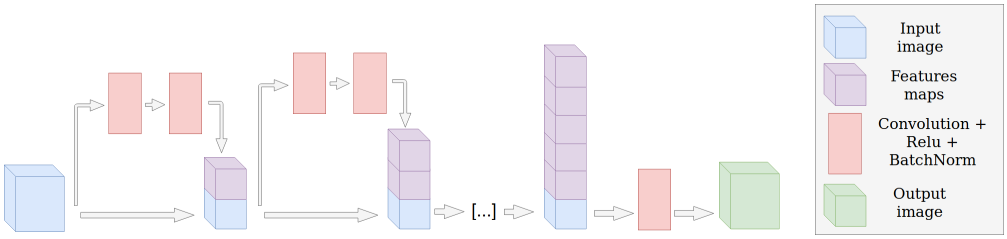


Figure 3.12. Main blocks of our incremental Transformer network. The blue blocks represent the input image data, while the red blocks are a sequence of convolutional + Relu + Batch Normalization layers. The output of the two convolutional blocks are concatenated with the previous inputs, forming a group of image data and features maps that grow along the depth of the network. The number of features increases from 3 (input data) to 256 (final aggregation step), while a last convolutional layer squeezes the features back into 3 channels interpretable as an RGB image (green block).

and target domains share the same label space and overall they have the same conditional distribution, but their marginal distribution is different thus inducing a domain shift.

The goal of ADAGE is to learn a domain agnostic model by adapting both the images and their representation: this is obtained by jointly learning a mapping towards new visual and feature spaces where the domains are confused but the relevant semantic information of the data is maintained. We realize this goal by combining a *feature extractor* F and a *classifier* C sub-networks to three modules: a new *image transformer* T and two *domain discriminators* H, D respectively for images and features. An overview of our deep learning architecture is shown in Figure 3.11.

Image Transformer T modifies the input images to remove their domain-specific style. We defined a new *incremental* structure for this module that exploits the power of layer aggregation [171]: the output of two 3×3 convolutional layers each followed by Relu and Batch Normalization are stacked up with the input and propagated to every subsequent layer (see Figure 3.12). Specifically, the produced feature build up in size resulting in a growing sequence of $\{3, 8, 16, 32, 4, 64, 128\}$ maps, after which a convolution layer brings them down to 3 channels, interpretable as RGB images.

Image Domain Discriminator H receives as input the images produced by T and predicts their domain label. More in details, this module is a multi-class classifier that learns to distinguish among the S source domains in DG, and $S + 1$ in DA (including the target), by minimizing a simple cross-entropy loss $\mathcal{L}_H$. The information provided by this module is used in two ways: to adversarially guide the transformer T to produce images with confused domain identity, and to estimate a similarity measure between the source and the target data when available. The first task is executed through a gradient reversal layer as in [43]. The second is obtained as a byproduct of the domain classifier H by collecting the probability of every source sample in each batch to be recognized as belonging to the target.

Feature Domain Discriminator D is analogous to H but, instead of images, it takes as input their features, performing domain classification by minimizing the cross-entropy loss $\mathcal{L}_D$. Finally, during backpropagation, the inverted gradient regulates the feature extraction process to confuse the domains.

Feature Extractor and Classifier F and C are standard deep learning modules. We built them with the same network structure used in [176] to put them on equal footing. In particular, in the DG setting the classifier learns to distinguish

among the M categories of the sources by minimizing the cross-entropy loss $\mathcal{L}_C$, while for the DA setting it can also provide the classification probability on the target samples $p(x^t) = C(F(T(x_t)))$ that is used to minimize the related entropy loss $\mathcal{L}_e = p(x^t)log(p(x^t))$.

If we indicate with θ the network parameters and we use subscripts to identify the different network modules, we can formally write the overall loss function optimized by ADAGE as:

$$\mathcal{L}(\theta_T, \theta_F, \theta_D, \theta_H, \theta_C) = \sum_{i=1..S,S+1} \sum_{j=1..N^i} \mathcal{L}_C^{j,i}(\theta_T, \theta_F, \theta_C) + \eta \mathcal{L}_e^{j,i=S+1}(\theta_T, \theta_F, \theta_C) - \lambda \mathcal{L}_D^{j,i}(\theta_T, \theta_F, \theta_D) - \gamma \mathcal{L}_H^{j,i}(\theta_T, \theta_H) \ . \tag{3.14}$$

We remark that, as specified by its superscripts, $\mathcal{L}_e^{j,i=S+1}$ is only active in the DA setting, while $\mathcal{L}_D$ and $\mathcal{L}_H$ in the DA case deal with an $\{S+1\}$-multiclass task involving also the target together with the source domains.

As can be noted from (3.14), the number of meta-parameters of our approach is very limited. For λ we use the same rule introduced by [43] that grows the importance of the feature domain discriminator with the training epochs: $\lambda_k = \frac{2}{1+exp(-10k)} - 1$, where $k = \frac{current\_epoch}{total\_epochs}$. We set $\gamma_k = 0.1\lambda_k$ so that only a small portion of the full gradient of the image domain discriminator is backpropagated: in this way we can still get useful similarity measures among the domains while progressively guiding the transformer to make them alike. Finally, the experimental evaluation indicates that ADAGE is robust to the exact choice of η, thus we keep it always fixed to 0.5 just for simplicity.

We conclude this section with some remarks on the characteristics of ADAGE. To our knowledge this is the first method designed to work seamlessly both in the domain generalization and in the unsupervised domain adaptation settings. It is also the first method to introduce an image-level component in a deep learning architecture for domain generalization. Differently from existing GAN-based methods that need a typical alternating training between image adaptation and classification, we train the whole model of ADAGE with a single optimizer. Nonetheless, we are still performing adversarial training, as the gradient originating from our domain discriminators is inverted before reaching our feature extractor and image transformer. Moreover, GAN adaptive approaches aim at transferring the source style to the target data and/or vice-versa [128, 13, 97], while our goal is that of projecting the data of all the available domains to a new agnostic space, where the domain-specific signatures are discarded. Technically we avoid the risk of degenerating all inputs to random noise by priming the network to correctly perform label classification and by starting to confuse the domains only later (see the λ_k update agenda), while still receiving feedback from the classification loss. Finally we underline that the image transformation proposed for ADAGE is not meant to be pleasant to the human eye: its purpose is to start to close the domain gap, instead of fully delegating this task to the features at later stages in the learning process.

3.3.2 Experiments

We tested ADAGE on the DA and DG scenarios always considering the availability of multiple sources. Our framework can easily switch between the two cases with a few key differences. For DG the image H and the feature D domain discriminators deal with S domains, while for DA they need to distinguish among $S+1$ domains including

	Sources	SVHN MNIST-M SYNTH	SVHN MNIST SYNTH	MNIST-M SYNTH MNIST	Avg.
	Target	MNIST	MNIST-M	SVHN	
DG	combine sources	98.7	62.6	69.5	76.9
	MLDG [89]	99.1	61.2	69.7	76.7
	ADAGE Residual	**99.2**	65.8	74.6	79.9
	ADAGE Incremental	99.1	**66.3**	**76.4**	**80.3**
DA	combine sources	98.7	62.6	69.5	76.9
	combine DANN [176]	92.5	65.1	77.6	78.4
	MDAN [176]	97.9	68.7	81.6	82.7
	ADAGE Residual	99.2	87.6	84.1	90.3
	ADAGE Incremental	**99.3**	**88.5**	**86.0**	**91.3**

Table 3.12. Classification accuracy results: experiments with 3 sources.

the target. Moreover, in DA, the unlabeled target data trigger the classification block C to activate the entropy loss and to use the source domain weights provided by the image domain discriminator H. Specifically these weights make sure that our classifier is biased towards the sources more similar to the target.

Datasets and Scenarios We focus on five well known digits datasets: **MNIST** [86], **MNIST-M** [43], **USPS** [41], **SVHN** [117] and (**SYNTH Digits**) (dataset details are in section 2.1.3).

To define the multi source experimental scenarios we follow [166, 176] and reproduce their settings. A first case from [176] involves *three sources* chosen in {MNIST, MNIST-M, SYNTH, SVHN}. Each dataset with the exception of SYNTH, is cyclically used as target. All the images are resized to 28×28 pixels and subsets of 20k and 9k samples are chosen respectively from each source and from the target. A second case from [166] involves *four sources* by adding USPS to the previous dataset group, and focuses on two possible targets, SVHN and MNIST-M. Even in this case the imagesare resized to 28×28 pixels, and 25/9k samples are drawn from each dataset to define the source/target sets[4]. A third case from [46] involves *five sources* and exploits variants of MNIST denoted as $\{M_0, M_{15}, M_{30}, M_{45}, M_{60}, M_{75}\}$. We randomly chose 1000 digit images of ten classes from the original MNIST training set to represent the basic view M_0 with 100 images for each class. The other views are then obtained by rotating the images of 15 degrees in counterclock-wise direction.

For our experiments all the datasets were normalized and zero-centered. In the DG case, the mean and standard deviation of the target for data normalization are calculated batch-by-batch during the testing process. A standard random crop of $90 - 100\%$ of the total image size was applied as data augmentation. The training procedure requires 200 epochs for DA, while for the DG experiments we found beneficial to increase the number of training epochs to 600. For DA we used RmsProp [150] with a lr of $5e^{-4}$, while for DG we used Adam [79] with a lr of $1e-3$. In both cases we step down the lr after 80% of the training.

All the experiments are repeated tree times and we report the average on the obtained classification accuracy results.

Results in Tables 3.12 and 3.13 As a main baseline for the three and

[4]The authors of [166] kindly shared the exact splits used for their experiments. For the three and five sources experiments we considered instead multiple random selections of the samples from the datasets.

	Sources	SYNTH MNIST MNIST-M USPS	SYNTH MNIST SVHN USPS	Avg.
	Target	SVHN	MNIST-M	
DG	combine sources	73.2	61.9	67.5
	MLDG [89]	68.0	65.6	66.8
	ADAGE Residual	68.2	65.7	66.9
	ADAGE Incremental	**75.8**	**67.0**	**71.4**
DA	combine sources	73.2	61.9	67.5
	combine DANN [166]	68.9	71.6	70.3
	DCTN [166]	77.5	70.9	74.2
	ADAGE Residual	82.3	84.1	83.2
	ADAGE Incremental	**85.3**	**85.3**	**85.3**

Table 3.13. Classification accuracy results: experiments with 4 sources.

	Target	M_0	M_{15}	M_{30}	M_{45}	M_{60}	M_{75}	Avg.
DG	D-MTAE [46]	82.5	96.3	93.4	78.6	94.2	80.5	87.6
	CCSA [114]	84.6	95.6	94.6	82.9	94.8	82.1	89.1
	MMD-AAE [91]	83.7	96.9	95.7	85.2	95.9	81.2	89.8
	CROSS-GRAD [135]	88.3	**98.6**	**98.0**	97.7	**97.7**	91.4	**95.3**
	ADAGE Incremental	**88.8**	97.6	97.5	**97.8**	97.6	**91.9**	95.2

Table 3.14. Domain Generalization accuracy results on experiments with 5 MNIST-rotated sources. For compactness we only indicate the considered target.

	Residual T		Incremental T	
Mode	DG	DA	DG	DA
T	61.73		63.23	
T + E	61.7	56.7	63.2	63.9
T + D	62.1	65.4	62.2	69.9
D	53.0	65.9	53.0	65.9
D + E	53.0	75.1	53.0	75.1
T + H	58.7	61.3	61.4	60.8
T + D + H	59.0	62.5	61.2	68.8
T + E + H	58.7	61.6	61.4	63.9
T + D + E	62.2	82.9	62.2	82.4
T + D + E + H	65.8	87.6	66.3	88.5

Table 3.15. Ablation analysis on the experiment with three sources and target MNIST-M. We turn on and off the different parts of the model: **T**= Transformer, **E**= Entropy, **D**= Feature Domain Discriminator, **H**= Image Domain Discriminator

four sources settings we use the naïve **combine sources** strategy that consists in learning a classifier on all the source data combined together. For a fair comparison we produced these results by keeping only the feature extractor F and the classifier C of our network, while turning off all the adaptive blocks. For DG we benchmark against the meta-learning method **MLDG** presented in [89] using the code provided by the authors and running the experiments on our settings. For DA we report the reference results from previous works. In particular for the three sources experiments the comparison is with the Multisource Domain Adversarial Network **MDAN** [176].

Since this method builds over the DANN algorithm [43] the result obtained with DANN applied on the combination of all the sources (**combine DANN**) is also reported. For the four sources experiments the main comparison is instead with the Deep Cocktail Network (**DCN**) [166], a recent method able to work even with partial class overlap among the sources. We present the accuracy for both the residual and incremental transformer variants of ADAGE and we see that they outperforms all the reference sota baselines in DG and DA, both using three and four sources, with margins of up to 6.84% in DG and of up to 11.07% in DA. Interestingly, using four sources slightly worsens the performances when SVHN is the target: our interpretation is that adding the USPS dataset slightly increases the domain shift between the whole training and test domains, making the adaptation somehow more difficult. Results obtained with the incremental transformer are overall stronger that those obtained with the residual version: we think that this is due to the peculiar structure of the incremental transformer that allows to retain all the expressive capacity of bigger and deeper architectures while keeping the number of parameters low. Indeed the incremental T has only $\frac{1}{3}$ of the parameters with respect to the residual version, thus it is faster in training and allows to better avoid overfitting while mapping the source domain images into a compact agnostic space.

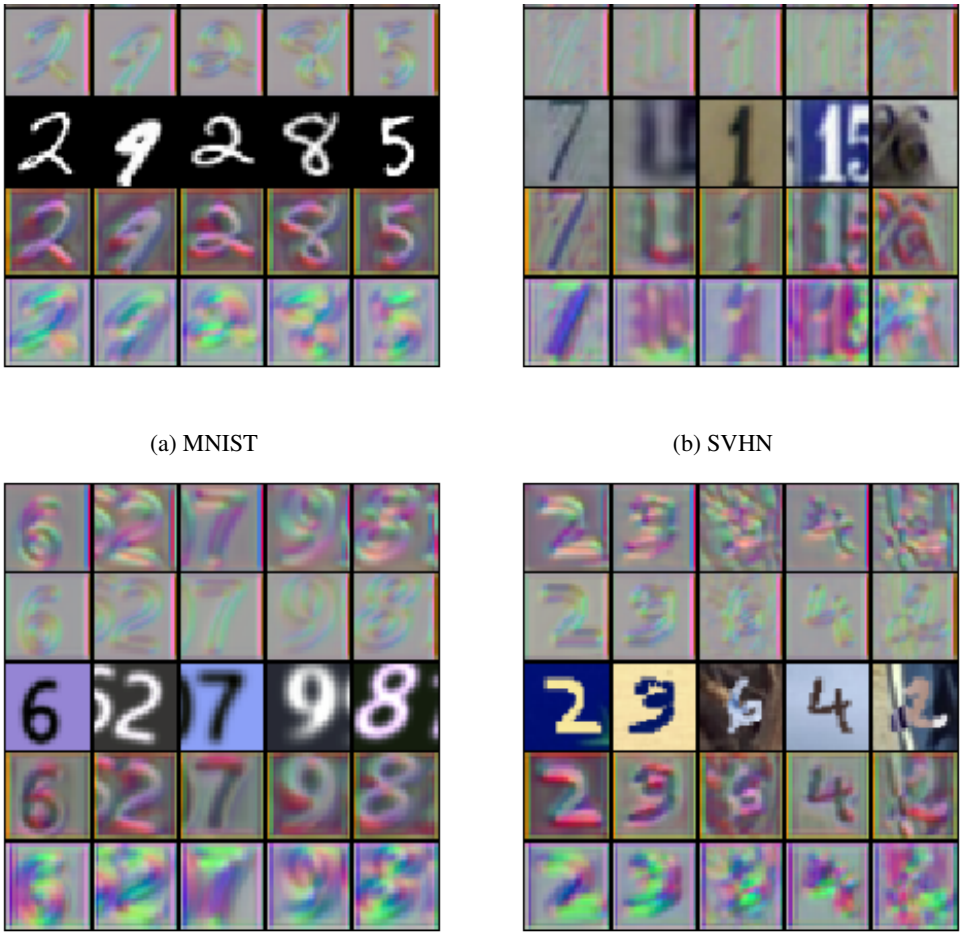
Results in Table 3.14 For the five sources experiments we focus on DG and on the most efficient incremental version of ADAGE. We benchmark against two autoencoder-based DG methods **D-MTAE** and **MMD-AAE** respectively presented in [46] and [91], as well as against the metric-learning **CCSA** method [114] and the very recent **CROSS-GRAD** [135]. The results indicate that ADAGE outperforms three of the four competitors and has results similar to CROSS-GRAD which proposes an adaptive solution based on data augmentation that could potentially be combined with ADAGE.

Further Results Besides evaluating ADAGE on digits images, we tested it also on the **ETH80 object dataset**. We followed [46] focusing on the ETH80-p setting with 5 domains obtained from 5 pitch-rotated views of 8 objects. The images are subsampled to 28×28 and greyscaled. By running ADAGE for DG on this setting, training in turn on four sources and testing on the remaining domain, we get an average accuracy of 94.1%, significantly higher than 87.9% of D-MTAE.

While ADAGE is specifically tailored for the multi-source settings, the reader might wonder how it would behave in the case of a **single source DA with access to unlabeled target data**. As a proof of concept experiment, we tested ADAGE using SVHN as source and MNIST as target. With the same protocol used in our DA experiments, we achieve an 95.7% accuracy, which is on par with the very recent [59] and better than many other competitive methods [70, 128, 179, 131, 134], all scoring an accuracy lower than 91.0%.

Ablation Study Table 3.15 shows the effect of progressively enabling the key components of ADAGE, showcasing the relative importance of each piece. If only the transformer is enabled, but there is no effort to align the domains, the final accuracy is not significantly better than the non-adaptive baseline (62.6%), which shows that simply using a longer network provides only a minimal advantage. A first jump in accuracy appears when the feature domain discriminator D is introduced, but only for DA, as the DG accuracy stays low. The contribution of the image domain discriminator H is negligible by itself and this behavior can be explained considering that we backpropagate only a small part of the H gradient ($\gamma = 0.1\lambda$, see section 3.3.1). However its beneficial effect becomes evident in collaboration with the other network modules: passing from T+D+E to T+D+E+H implies an improvement in accuracy of at least 4% which indicates that the adversarial guidance provided by H on T allows for an image adaptation process complementary to the feature

adaptation one. Using the entropy loss helps a lot, with gains in performance of over 15%. The presence of multiple sources very likely helps in reducing the risk that the entropy loss might mislead the classifier. Note that since the image domain discriminator backpropagates only on the transformer, it is not possible to test any combination containing H but not T.



(a) MNIST (b) SVHN

(c) Synth. Digits (d) MNIST-M

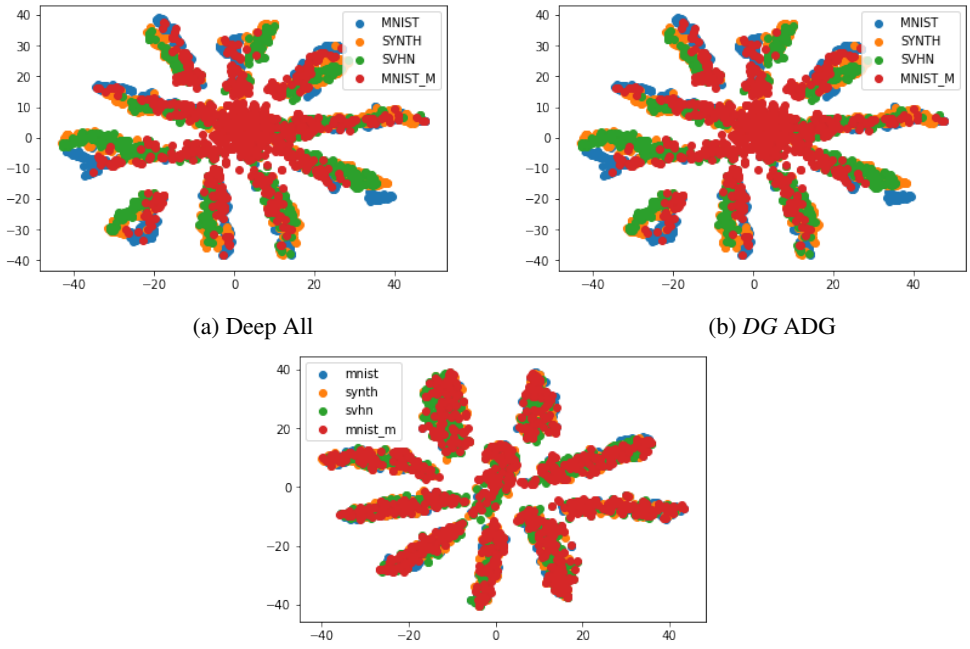
Figure 3.13. Examples of domain-agnostic digits generated by the transformer block in the experiments with three sources and MNIST-M as target. The top two rows show images produced in the DG setting by our residual based (line 1) and incremental based (line 2) transformers. Line 3 shows the original images and in the last two rows we display images produced by the residual (line 4) and incremental transformers in the DA setting. Images transformed with the residual architecture tend to preserve more of the original input. It is worth mentioning that, while the DG images from the target class are more noisy than their equivalent in the DA setting, our transformer does a good job at transforming images that it has never seen before.

Qualitative results While strong numerical results suggest that we are indeed closing the distances between domains, we can improve our understanding by looking at both the images generated for the agnostic domain, and at their embeddings. Figure 3.13 shows the agnostic images generated by the residual and the incremental transformers, in the three source experiment with target MNIST-M, both in the

DA and DG settings. We see that the main effect of T is that of removing the backgrounds and enhancing the edges. We might still be able to distinguish between domains (more or less, depending on the used T) by looking at the style of the digits. We believe this is why we also need domain invariance at the feature level. Fig 3.14 shows the TSNE embedding of features extracted immediately before the final classifier. We see that in the DA setting we completely align the feature spaces of the domains, resulting in a clear per class clustering. In the DG setting the results are less clean, but the clusters are still tighter than those obtained by the combine source baseline.

3.3.3 Conclusions

This section tackles the problem of domain generalization and adaptation when multiple sources are available, proposing the first deep architecture for these settings which jointly performs image and feature adaptation. This makes it possible to learn how to project images into an agnostic visual space, that can be further used for domain alignment in the feature space. Our architecture, ADAGE, achieves impressive results on several benchmarks, outperforming the current state of the art by a significant margin. Future work will further explore alternative architectural choices for performing domain alignment in the feature space, will expand the experimental evaluation to include object-based domain adaptation benchmarks like Office, and will extend ADAGE to the open set multi source domain adaptation and generalization scenarios.



(a) Deep All (b) *DG* ADG

(c) *MDA* ADG

Figure 3.14. This figure shows the TSNE visualization of the class distributions for the three source experiment, keeping MNIST-M as target. The features have been extracted from the penultimate layer. It is interesting to note that while Deep All does a reasonable job at aligning the domains, our method completely aligns each class for every domain.

Chapter 4

Learning to see across modalities

This chapter presents methods which can be used even when classical domain adaption approaches are not applicable; the most common requirement for the use of DA methods is that source and target must share the same classes. Another, more implicit, requirement is that both domains are usually assumed to exist in the same modality. This is not true in many settings. As a real life case study, commonly encountered in robotics, we will focus on RGB-D recognition.

4.1 Synthetic Data: DepthNet

This section presents a study on the use of synthetic data as a proxy for real data, specifically in the context of RGB-D perception. The key intuition behind this approach is that it is possible to gather a much smaller amount of labeled 3D data and exploit it to generate a much larger dataset of 2D images.

Convolutional Neural Networks (CNNs) trained on large scale RGB databases have become the secret sauce in the majority of recent approaches for object categorization from RGB-D data. Thanks to colorization techniques, these methods exploit the filters learned from 2D images to extract meaningful representations in $2.5D$. Still, the perceptual signature of these two kind of images is very different, with the first usually strongly characterized by textures, and the second mostly by silhouettes of objects. Ideally, one would like to have two CNNs, one for RGB and one for depth, each trained on a suitable data collection, able to capture the perceptual properties of each channel for the task at hand. This has not been possible so far, due to the lack of a suitable depth database. This section addresses this issue, proposing to opt for synthetically generated images rather than collecting by hand a $2.5D$ large scale database. While being clearly a proxy for real data, synthetic images allow to trade quality for quantity, making it possible to generate a virtually infinite amount of data. We show that the filters learned from such data collection, using the very same architecture typically used on visual data, learns very different filters, resulting in depth features (a) able to better characterize the different facets of depth images, and (b) complementary with respect to those derived from CNNs pre-trained on 2D datasets. Experiments on two publicly available databases show the power of our approach

4.1.1 Context: RGB-D data

Deep learning has changed the research landscape in visual object recognition over the last few years. Since their spectacular success in recognizing $1,000$ object categories [80], convolutional neural networks have become the new off the shelf state of the art in visual classification. The robot vision community has also attempted to take advantage of the deep learning trend, as the ability of robots to understand what they see reliably is critical for their deployment in the wild. A critical issue when trying to transfer results from computer to robot vision is that robot perception is tightly coupled with robot action. Hence, pure RGB visual recognition is not enough.

The heavy use of $2.5D$ depth sensors on robot platforms has generated a lively research activity on $2.5D$ object recognition from depth maps [82, 141, 24]. Here a strong emerging trend is that of using Convolutional Neural Networks (CNNs) pre-trained over ImageNet [127] by colorizing the depth channel [133]. The approach has proved successful, especially when coupled with fine tuning [36] and/or spatial pooling strategies [172, 25, 26] (for a review of recent work we refer to section 2.2). These results suggest that the filters learned by CNNs from ImageNet are able to capture information also from depth images, regardless of their perceptual difference.

Is this the best we can do? What if one would train from scratch a CNN over a very large scale $2.5D$ object categorization database, wouldn't the filters learned be more suitable for object recognition from depth images? RGB images are perceptually very rich, with generally a strong presence of textured patterns, especially in ImageNet. Features learned from RGB data are most likely focusing on those aspects, while depth images contain more information about the shape and

the silhouette of objects. Unfortunately, as of today a $2.5D$ object categorization database large enough to train a CNN on it does not exist. A likely reason for this is that gathering such data collection is a daunting challenge: capturing the same variability of ImageNet over the same number of object categories would require the coordination of very many laboratories, over an extended period of time.

In this chapter we show a different approach: rather than acquiring a $2.5D$ object categorization database, we propose to use synthetic data as a proxy for training a deep learning architecture specialized in learning depth specific features. To this end, we constructed the VANDAL database, a collection of 4.5 million depth images from more than $9,000$ objects, belonging to 319 categories. The depth images are generated starting from $3D$ CAD models, downloaded from the Web, through a protocol developed to extract the maximum information from the models. VANDAL is used as input to train from scratch a deep learning architecture, obtaining a pre-trained model able to act as a depth specific feature extractor. Visualizations of the filters learned by the first layer of the architecture show that the filter we obtain are indeed very different from those learned from ImageNet with the very same convolutional neural network (fig. 4.1). As such, they are able to capture different facets of the perceptual information available from real depth images, more suitable for the recognition task in that domain. We call our pre-trained architecture DepthNet.

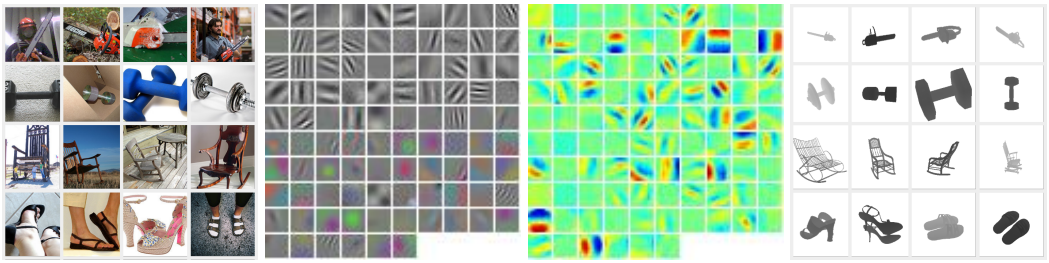


Figure 4.1. Sample images for the classes chainsaw, dumbbell, rocker chair and sandal from ImageNet (a) and VANDAL (d). We show the corresponding filters learned by the very same CNN architecture respectively in (b) and (c) (note that this is colorized for easier viewing). We see that even though the architecture is the same, using $2D$ rather than $2.5D$ images for training leads to learning quite different filters. In (c) some of the features appear to be undefined.

Experimental results on two publicly available databases confirm this: when using only depth, our DepthNet features achieve better performance compared to previous methods based on a CNN pre-trained over ImageNet, without using fine tuning or spatial pooling. The combination of the DepthNet features with the descriptors obtained from the CNN pre-trained over ImageNet, on both depth and RGB images, leads to strong results on the Washington database [82], and to results competitive with fine-tuning and/or sophisticated spatial pooling approaches on the JHUIT database [88]. To the best of our knowledge, this is the first work that uses synthetically generated depth data to train a depth-specific convolutional neural network. All the VANDAL data, the protocol and the software for generating new depth images, as well as the pre-trained DepthNet, is publicly available: https://sites.google.com/site/vandaldepthnet/.

The rest of the section is organized as follows. First, we introduce the VANDAL database, describing its generation protocol and showcasing the obtained depth images (subsection 4.1.2). Subsection 4.1.3 describes the deep architecture used

and subsection 4.1.4 reports our experimental findings. This section concludes with a summary and a discussion on future research.

4.1.2 The VANDAL database

Here we present VANDAL and the protocol followed for its creation. With $4.5M$ synthetic images, it is the largest existing depth database for object recognition. Subsection 4.1.2 describes the criteria used to select the object categories composing the database and the protocol followed to obtain the $3D$ CAD models from Web resources. Subsection 4.1.2 illustrates the procedure used to generate depth images from the $3D$ CAD models.

Figure 4.2. Sample morphs (center, right) generated from an instance model for the category coffee cup (left).

(a) The 319 categories in VANDAL

(b) Some examples from the classes more populated

Figure 4.3. The VANDAL database. On the left, we show a word cloud visualization of the classes in it, based on the numbers of $3D$ models in each category. On the right, we show exemplar models for the six categories more populated: coffee cup, vase, pencil case, computer mouse, table and chair.

Selecting and Generating the $3D$ Models

CNNs trained on ImageNet have been shown to generalize well when used on other object centric datasets. Following this reasoning, we defined a list of object categories as a subset of the ILSVRC2014 list [127], removing by hand all scenery classes, as well as objects without a clear default shape such as clothing items or animals. This resulted in a first list of roughly 480 categories, which was used to query public $3D$ CAD model repositories like $3D$ Warehouse, Yeggi, Archive3D, and many others.

Five volunteers[1] manually downloaded the models, removing all irrelevant items like floor or other supporting surfaces, people standing next to the object and so forth, and running a script to harmonize the size of all models (some of them were originally over 1GB per file). They were also required to create significantly morphed variations of the original $3D$ CAD models, whenever suitable. Fig. 4.2 shows examples of morphed models for the object category coffee cup. Finally, we removed all categories with less than two models, ending up with 319 object categories with an average of 30 models per category, for a total of $9,383$ CAD object models. Fig. 4.3, left, gives a world cloud visualization of the VANDAL dataset, while on the right it shows examples of $3D$ models for the 6 most populated object categories.

From $3D$ Models to 2.5 Depth Images

All depth renderings were created using Blender[2], with a python script fully automating the procedure, and then saved as 8bit grayscale .png files, using the convention that black is close and white is far.

The depth data generation protocol was designed to extract as much information as possible from the available $3D$ CAD models. This concretely means obtaining the greatest possible variability between each rendering. The approach commonly used by real RGB-D datasets consists in fixing the camera at a given angle and then using a turntable to get all possible viewpoints of the object [82, 88]. We tested here a similar approach, but we found out using perceptual hashing[3] that a significant number of object categories had more than 50% nearly identical images.

We defined instead a configuration space consisting of: (a) object distance from the camera, (b) focal length of the camera, (c) camera position on the sphere defined by the distance, and (d) slight ($< 10\%$) random morphs along the axes of the model. Figure 4.4 illustrates the described configuration space. This protocol ensured that almost none of the resulting images were identical. Object distance and focal length were constrained to make sure the object always appears in a recognizable way. We sampled this configuration space with roughly 480 depth images for each model, obtaining a total of 4.5 million images. Preliminary experiments showed

[1] Graduate students from the MARR program at DIAG, Sapienza Rome University.
[2] `www.blender.org`
[3] `http://www.phash.org/`

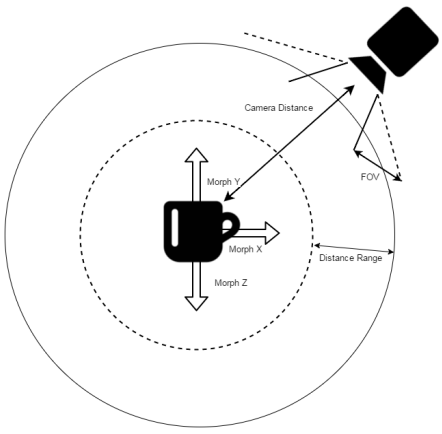


Figure 4.4. Configuration space used for generating renderings in the VANDAL database.

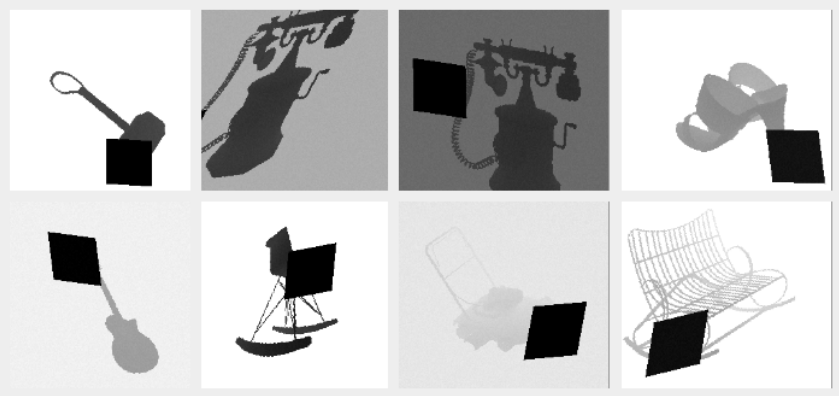


Figure 4.5. Data augmentation samples from various classes (hammer, phone, sandal, guitar, rocker, lawn mower, bench). Note that the contrast brightness variations and noise are hard to visualize on small thumbnails.

that increasing the sampling rate in the configuration space did lead to growing percentages of nearly identical images.

The rendered depth images consist of objects always centered on a white background, as it allows us the maximum freedom to perform various types of data augmentation at training time, which is standard practice when training convolutional neural networks. This is here even more relevant than usual, as synthetically generated data are intrinsically perceptually less informative. The data augmentation methods we used are: image cropping, occlusion (1/4 of the image is randomly occluded to simulate gaps in the sensor scan), contrast/brightness variations (which allows us to train our network to deal with both raw and normalized data), in depth views corresponding to scaling the Z axis and shifting the objects along it, background substitution (substituting the white background with one randomly chosen farther away than the object's center of mass), random uniform noise (as in film grain), and image shearing (a slanting transform). While not all of these data augmentation procedures produce a realistic result, they all contribute as regularizers; we avoided modeling a more complex sensor model noise for efficiency reasons. Figure 4.5 shows some examples of data augmentation images obtained with this protocol.[4]

4.1.3 Learning Deep Depth Filters

Once the VANDAL database has been generated, it is possible to use it to train any kind of convolutional deep architecture. In order to allow for a fair comparison with previous work, we opted for CaffeNet, a slight variation of AlexNet [80]. Although more modern networks have been proposed in the last years [138, 73, 62], it still represents the most popular choice among practitioners, and the most used in robot vision[5]. Its well know architecture consists of 5 convolutional layers, interwoven with pooling, normalization and ReLU layers, plus three fully connected layers. [6]

[4] More in depth information can be found at the project's website: `https://sites.google.com/site/vandaldepthnet/`

[5] Preliminary experiments using the VGG, Inception and Wide Residual networks on the VANDAL database did not give stable results and require further investigation.

[6] CaffeNet differs from AlexNet in the pooling, which is done there before normalization. It usually performs slightly better and has thus gained wide popularity.

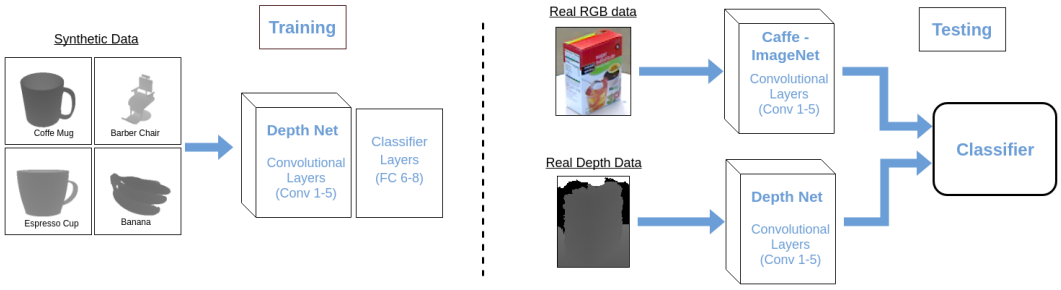


Figure 4.6. DepthNet and our associated RGB-D object classification framework. During training, we learn depth filters from the VANDAL synthetic data (left). During test (right), real RGB and depth data is processed by two distinct CNNs, each specialized over the corresponding modality. The features, derived from the activations of the fifth convolutional layer, are then fed into a cue integration classifier.

Although the standard choice in robot vision is using the output of the seventh activation layer as feature descriptors, several studies in the vision community show that lower layers, like the sixth and the fifth, tend to have higher generalization properties [177]. We followed this trend, and opted for the fifth layer (by vectorization) as deep depth feature descriptor (an ablation study supporting this choice is reported in subsection 4.1.4). We name in the following as **DepthNet** the CaffeNet architecture trained on VANDAL using as output feature the fifth layer, and **Caffe-ImageNet** the same architecture trained over ImageNet.

Once DepthNet has been trained, it can be used as any depth feature descriptor, alone or in conjunction with Caffe-ImageNet for classification of RGB images. We explore this last option, proposing a system for RGB-D object categorization that combines the two feature representations through a multi kernel learning classifier [121]. Figure 4.6 gives an overview of the overall RGB-D classification system. Note that DepthNet can be combined with any other RGB and/or $3D$ point cloud descriptor, and that the integration of the modal representations can be achieved through any other cue integration approach. This underlines the versatility of DepthNet, as opposed to recent work where the depth component was tightly integrated within the proposed overall framework, and as such unusable outside of it [36, 172, 24, 88].

4.1.4 Experiments

We assessed the DepthNet, as well as the associated RGB-D framework of fig. 4.6, on two publicly available databases. Subsection 4.1.4 describes our experimental setup and the databases used in our experiments. Subsection 4.1.5 reports a set of experiments assessing the performance of DepthNet on depth images, compared to Caffe-ImageNet, while in subsection 4.1.5 we assess the performance of the whole RGB-D framework with respect to previous approaches.

Experimental setup

We conducted experiments on the Washington RGB-D [82] and the JHUIT-50 [88] object datasets. The first consists of 41,877 RGB-D images organized into 300 instances divided in 51 classes. Each object instance was positioned on a turntable and captured from three different viewpoints while rotating. Since two consecutive views are extremely similar, only 1 frame out of 5 is used for evaluation purposes. We performed experiments on the object categorization setting, where we

followed the evaluation protocol defined in [82]. The second is a challenging recent dataset that focuses on the problem of fine-grained recognition. It contains 50 object instances, often very similar with each other (e.g. 9 different kinds of screwdrivers). As such, it presents different classification challenges compared to the Washington database.

All experiments, as well as the training of DepthNet, were done using the publicly available Caffe framework [75]. As described above, we obtained DepthNet by training a CaffeNet over the VANDAL database. The network was trained using Stochastic Gradient Descent for 50 epochs. Learning rate started at 0.01 and gamma at 0.5 (halving the learning rate at each step). We used a variable step down policy, where the first step took 25 epochs, the next 25/2, the third 25/4 epochs and so on. These parameters were chosen to make sure that the test loss on the VANDAL test data had stabilized at each learning rate. Weight decay and momentum were left at their standard values of 0.0005 and 0.9. Training and test data was centered by removing the mean pixel.

To assess the quality of the DepthNet features we performed three set of experiments:

1. *Object classification using depth only:* features were extracted with DepthNet and a linear SVM[7] was trained on it. We also examined how the performance varies when extracting from different layers of the network, comparing against a Caffe-ImageNet used for depth classification, as in [133].

2. *Object classification using RGB + Depth:* in this setting we combined our depth features with those extracted from the RGB images using Caffe-ImageNet. While [36] train a fusion network to do this, we simply use an off the shelf Multi Kernel Learning (MKL) classifier [121].

For all experiments we used the training/testing splits originally proposed for each given dataset. For linear SVM, we set C by cross validation. When using MKL, we left the default values of 100 iterations for online and 300 for batch and set p and C by cross validation.

Previous works using Caffe-ImageNet as feature extractor for depth, apply some kind of input preproccessing [133, 36, 172]. While we do compare against the published baselines, we also found that by simply normalizing each image (min to 0 and max to 255), one achieves very competitive results. Also, since our DepthNet is trained on depth data, it does not need any type of preprocessing over the depth images, obtaining strong results over raw data. Because of this, in all experiments reported in the following we only consider raw depth images and normalized depth images.

4.1.5 Assessing the performance of the DepthNet architecture

We present here an ablation study, aiming at understanding the impact of choosing features from the last fully convolutional layer as opposed to the more popular last fully connected layer, and of using normalized depth images instead of raw data. By comparing our results with those obtained by Caffe-ImageNet, we also aim at illustrating up to which point the features learned from VANDAL are different from those derived from ImageNet.

Figure 4.7 shows results obtained on the Washington database, with normalized and raw depth data, using as features the activations of the fifth pooling layer (pool5),

[7] Liblinear: http://www.csie.ntu.edu.tw/ cjlin/liblinear/

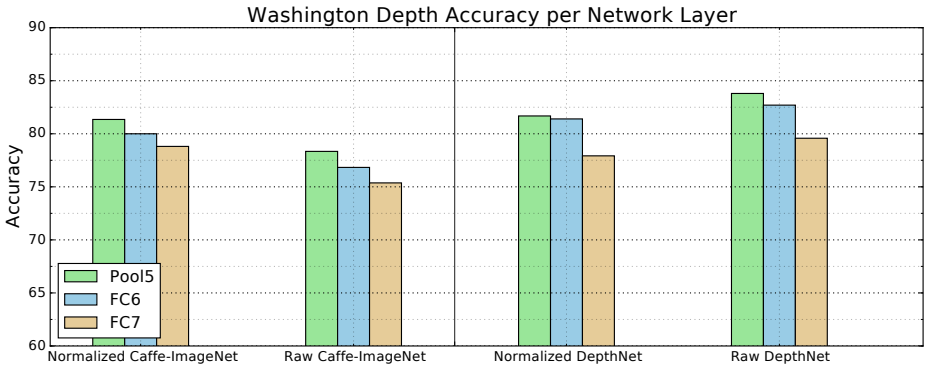


Figure 4.7. Accuracy obtained by DepthNet and Caffe-ImageNet over the Washington database, using as features pool5, FC6 and FC7. Results are reported for raw and normalized depth images.

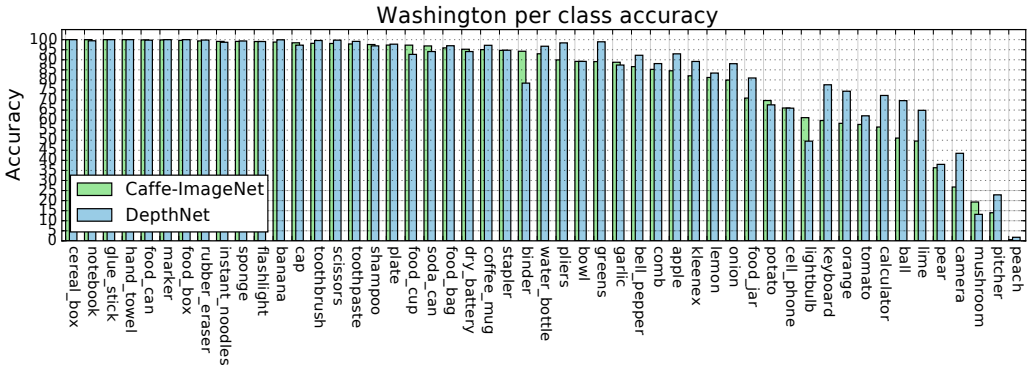


Figure 4.8. Accuracy per class on the Washington dataset, depth images. Classes sorted by the Caffe-ImageNet accuracies.

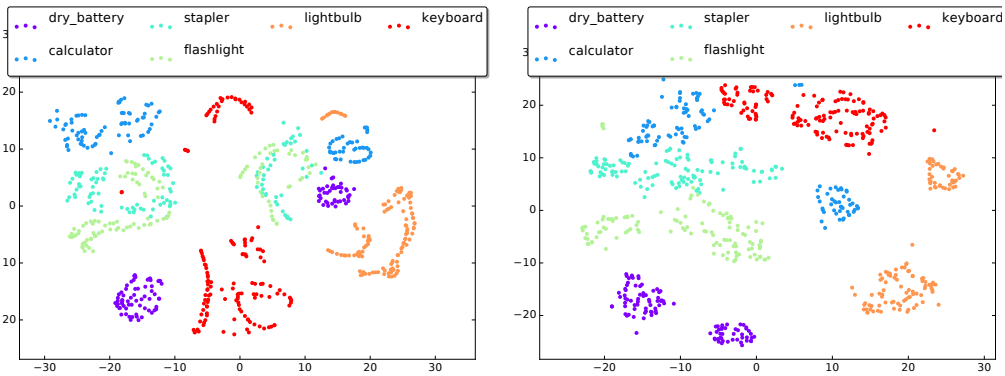
of the sixth fully connected layer (FC6), and of the seventh fully connected layer (FC7). Note that this last set of activations is the standard choice in the literature. We see that for all settings, pool5 achieves the best performance, followed by FC6 and FC7. This seems to confirm recent findings on RGB data [177], indicating that pool5 activations offer stronger generalization capabilities when used as features, compared to the more popular FC7.

The best performance is obtained by DepthNet, pool5 activations over raw depth data, with a 83.8% accuracy. DepthNet achieves also better results compared to Caffe-ImageNet over normalized data. To get a better feeling of how performance varies when using DepthNet or Caffe-ImageNet, we plotted the per-class accuracies obtained using pool5 and raw depth data. We sorted them in descending order according to the Caffe-ImageNet scores (fig. 4.8).

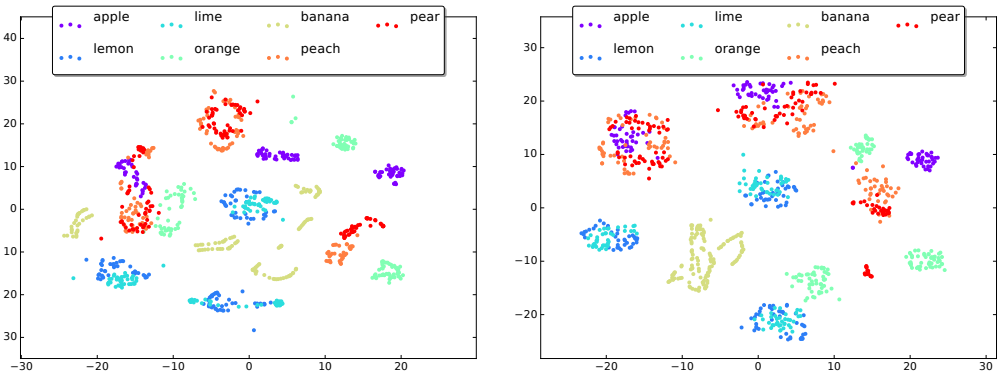
While there seems to be a bulk of objects where both features perform well (left), DepthNet seems to have an advantage over challenging objects like apple, onion, ball, lime and orange (right), where the round shape tends to be more informative than the specific object texture. This trend is confirmed also when performing a t-SNE visualization [104] of all the Washington classes belonging to the high-level categories 'fruit' and 'device' (fig. 4.9). We see that in general the DepthNet features tend to cluster tightly the single categories while at the same time separating them

very well. For some classes like dry battery and banana, the difference between the two representations is very marked. This does not imply that DepthNet features are always better than those computed by Caffe-ImageNet. Fig. 4.8 shows that CaffeNet features obtain a significantly better performance compared to DepthNet over the classes binder and mushroom, to name a few.

The features learned by the two networks seem to focus on different perceptual aspects of the images. This is likely due to the different set of samples used during training, and the consequent different learned filters (fig. 4.1).



(a) Device classes as seen by Caffe-ImageNet (left) and DepthNet (right)



(b) Fruit classes as seen by Caffe-ImageNet (left) and DepthNet (right)

Figure 4.9. t-SNE visualizations for the categories device (top) and fruit (bottom).

From these figures we can draw the following conclusions: (a) DepthNet provides the overall stronger descriptor for depth images, regardless of the activation layer chosen and the presence or not of preprocessing on the input depth data; (b) the features derived by the two networks tend to capture different features of the data, and as such are complementary. As we will show in the next subsection, this last point leads to very strong results when combining the two with a principled cue integration algorithm.

Assessing the performance of the RGB-D architecture

In this subsection we present experiments on RGB-D data, from both the Washington and JHUIT databases, assessing the performance of our DepthNet-based framework of fig. 4.6 against previous approaches. Table 4.1 shows in the top row our results, followed by results obtained by Caffe-ImageNet using the

pool5 activations as features, as well as results from the recent literature based on convolutional neural networks.

First, we see that the results in the RGB column stresses once more the strength of the pool5 activations as features: they achieve the best performance without any form of fine tuning, spatial pooling or sophisticated non-linear learning, as done instead in other approaches [36, 172, 24]. Second, DepthNet on raw depth data achieves the best performance among CNN-based approaches with or without fine tuning like [133, 36], but it is surpassed by approaches encoding explicitly spatial information through pooling strategies, and/or by using a more advanced classifier than a linear SVM, as we did. We would like to stress that we did not incorporate any of those strategies in our framework on purpose, to better assess the sheer power of training a given convolutional architecture on perceptually different databases. Still, nothing prevents in future work the merging of DepthNet with the best practices in spatial pooling and non-linear classifiers, with a very probable further increase in performance.

Lastly, we see that in spite of the lack of such powerful tools, our framework achieves the best performance on RGB-D data. This clearly underlines that the representations learned by DepthNet are both powerful and able to extract different nuances from the data than Caffe-ImageNet. Rather than the actual overall accuracy reported here in the table, we believe this is the breakthrough result we offer to the community in this work.

Method:	RGB	Depth Mapping	Depth Raw	RGB-D
DepthNet RGB-D Framework	$\mathbf{88.49 \pm 1.8}$	81.68 ± 2.2	$\mathbf{83.8 \pm 2.0}$	$\mathbf{92.25 \pm 1.3}$
Caffe-ImageNet Pool5	$\mathbf{88.49 \pm 1.8}$	81.11 ± 2	78.35 ± 2.5	90.79 ± 1.2
Caffe-ImageNet FC7 finetuning [36]	84.1 ± 2.7	83.8 ± 2.7	–	91.3 ± 1.4
Caffe-ImageNet FC7 [133]	83.1 ± 2.0	–	–	89.4 ± 1.3
CNN only [24]	82.7 ± 1.2	78.1 ± 1.3	–	87.5 ± 1.1
CNN + FisherKernel + SPM [24]	86.8 ± 2.2	$\mathbf{85.8 \pm 2.3}$	–	91.2 ± 1.5
CNN + Hypercube Pyramid + EM [172]	87.6 ± 2.2	85.0 ± 2.1	–	91.4 ± 1.4
CNN-SPM-RNN+CT [26]	85.2 ± 1.2	83.6 ± 2.3	–	90.7 ± 1.1
CNN-RNN+CT [25]	81.8 ± 1.9	77.7 ± 1.4	–	87.2 ± 1.1
CNN-RNN [141]	80.8 ± 4.2	78.9 ± 3.8	–	86.8 ± 3.3

Table 4.1. Comparison of our DepthNet framework with previous work on the Washington database. With *depth mapping* we mean all types of depth preprocessing used in the literature.

Experiments over the JHUIT database confirms the findings obtained over the Washington collection (table 4.2). Here our RGB-D framework obtains the second best result, with the state of the art achieved by the proposers of the database with a non CNN-based approach. Note that this database focuses over the fine-grained classification problem, as opposed to object categorization as explored in the experiments above. While the results reported in Table 4.2 on Caffe-ImageNet using FC7 seem to indicate that the choice of using pool5 remains valid, the explicit encoding of local information is very important for this kind of tasks [174, 3]. We are inclined to attribute to this the superior performance of [88]; future work incorporating spatial pooling in our framework, as well as further experiments on the object identification task in the Washington database and on other RGB-D data collections will explore this issue.

4.1.6 Conclusions

In this section we focused on object classification from depth images using convolutional neural networks. We argued that, as effective as the filters learned

Method:	RGB	Depth Mapp.	Depth Raw	RGB-D
DepthNet Pool5	–	**54.37**	**55.0**	**90.3**
Caffe-ImageNet Pool5	88.05	53.6	38.9	89.6
Caffe-ImageNet FC7 [133]	82.08	47.87	26.11	83.6
CSHOT + Color pooling + MultiScale Filters [88]	–	–	–	**91.2**
HMP [88]	81.4	41.1	–	74.6

Table 4.2. Comparison of our DepthNet framework with previous work on the JHUIT database. As only one split is defined, we do not report std.

from ImageNet are, the perceptual features of $2.5D$ images are different, and that it would be desirable to have deep architectures able to capture them.

To this purpose, we created VANDAL, the first depth image database synthetically generated, and we showed experimentally that the features derived from such data, using the very same CaffeNet architecture widely used over ImageNet, are stronger while at the same time complementary to them. This result, together with the public release of the database, the trained architecture and the protocol for generating new depth synthetic images, is the contribution of this work.

We see this work as the very beginning of a long research thread. By its very nature, DepthNet could be plugged into all previous work using CNNs pre-trained over ImageNet for extracting depth features. It might substitute that module, or it might complement it; the open issue is when this will prove beneficial in terms of spatial pooling approaches, learning methods and classification problems. A second issue we plan to investigate is the impact of the deep architecture over the filters learned from VANDAL.

While in this work we chose on purpose to not deviate from CaffeNet, it is not clear that this architecture, which was heavily optimized over ImageNet, is able to exploit at best our synthetic depth database. While preliminary investigations with existing architectures have not been satisfactory, we believe that architecture surgery might lead to better results. Furthermore, including the $3D$ models from [164] might greatly enhance the generality of the DepthNet.

Finally, we believe that the possibility to use synthetic data as a proxy for real images opens up a wide array of possibilities: for instance, given prior knowledge about the classification task of interest, would it be possible to generate on the fly a task specific synthetic database, containing the object categories of interest under very similar imaging conditions, and train and end-to-end deep network on it? How would performance change compared to the use of network activations as done today? Future work will focus on these issues.

4.2 Transfer Learning across modalities: DE^2CO

While Domain Adaptation methods perform extremely well, they have certain limitations; the main one being the need for source and the target to share the same categories. There have been some works attempting to [17] work around this issue, but they still assumes some kind of semantic overlap. On the other hand, when there are sufficient labeled samples from the target, one can usually perform transfer learning via finetuning [169]. The limitation to this approach is that we assume that the source and target exist in the same modality (i.e. RGB images). In this section we will focus on a method which allows us to perform transfer learning across modalities, with no assumption being made on the class labels.

The ability to classify objects is fundamental for robots. Besides knowledge about their visual appearance, captured by the RGB channel, robots heavily need also depth information to make sense of the world. While the use of deep networks on RGB robot images has benefited from the plethora of results obtained on databases like ImageNet, using convnets on depth images requires mapping them into three dimensional channels. This transfer learning procedure makes them processable by pre-trained deep architectures. Current mappings are based on heuristic assumptions over pre-processing steps and on what depth properties should be most preserved, resulting often in cumbersome data visualizations, and in sub-optimal performance in terms of generality and recognition results.

Here we take an alternative route and we attempt instead to *learn* an optimal colorization mapping for any given pre-trained architecture, using as training data a reference RGB-D database. We propose a deep network architecture, exploiting the residual paradigm, that learns how to map depth data to three channel images. A qualitative analysis of the images obtained with this approach clearly indicates that learning the optimal mapping preserves the richness of depth information better than current hand-crafted approaches.

Experiments on the Washington, JHUIT-50 and BigBIRD public benchmark databases, using CaffeNet, VGG-16, GoogleNet, and ResNet50 clearly showcase the power of our approach, with gains in performance of up to 16% compared to state of the art competitors on the depth channel only, leading to top performances when dealing with RGB-D data.

4.2.1 Motivation

Robots need to recognize what they see around them to be able to act and interact with it. Recognition must be carried out in the RGB domain, capturing mostly the visual appearance of things related to their reflectance properties, as well as in the depth domain, providing information about the shape and silhouette of objects and supporting both recognition and interaction with items. The current mainstream state of the art approaches for object recognition are based on Convolutional Neural Networks (CNNs, [85]), which use end-to-end architectures achieving feature learning and classification at the same time. Some notable advantages of these networks are their ability to reach much higher accuracies on basically any visual recognition problem, compared to what would be achievable with heuristic methods; their being domain-independent, and their conceptual simplicity. Despite these advantages, they also present some limitations, such as high computational cost, long training time and the demand for large datasets, among others.

This last issue has so far proved crucial in the attempts to leverage over the spectacular success of CNNs over RGB-based object categorization [148, 80] in the

depth domain. Being CNNs data-hungry algorithms, the availability of very large scale annotated data collections is crucial for their success, and architectures trained over ImageNet [32] are the cornerstone of the vast majority of CNN-based recognition methods. Besides the notable exception of [20], the mainstream approach for using CNNs on depth-based object classification has been through transfer learning, in the form of a mapping able to make the depth input channel compatible with the data distribution expected by RGB architectures.

Following recent efforts in transfer learning [49, 33, 170] that made it possible to use depth data with CNN pre-trained on a database of a different modality, several authors proposed hand-crafted mappings to colorize depth data, obtaining impressive improvements in classification over the Washington [81] database, that has become the golden reference benchmark in this field [133, 36].

We argue that this strategy is sub-optimal. By hand-crafting the mapping for the depth data colorization, one has to make strong assumptions on what information, and up to which extent, should be preserved in the transfer learning towards the RGB modality. While some choices might be valid for some classes of problems and settings, it is questionable whether the family of algorithms based on this approach can provide results combining high recognition accuracies with robustness across different settings and databases. Inspired by recent works on colorization of gray-scale photographs [72, 83, 27], we tackle the problem by exploiting the power of end-to-end convolutional networks, proposing a deep depth colorization architecture able to learn the optimal transfer learning from depth to RGB for any given pre-trained convnet.

Our deep colorization network takes advantage of the residual approach [63], learning how to map between the two modalities by leveraging over a reference database (Figure 4.10, top), for any given architecture. After this training stage, the colorization network can be added on top of its reference pre-trained architecture, for any object classification task (Figure 4.10, bottom). We call our network $(DE)^2CO$: DEep DEpth COlorization.

We assess the performance of $(DE)^2CO$ in several ways. A first qualitative analysis, comparing the colorized depth images obtained by $(DE)^2CO$ and by other state of the art hand-crafted approaches, gives intuitive insights on the advantages brought by learning the mapping as opposed to choosing it, over several databases. We further deepen this analysis with an experimental evaluation of our and other existing transfer learning methods on the depth channel only, using four different deep architectures and three different public databases, with and without fine-tuning. Finally, we tackle the RGB-D object recognition problem, combining $(DE)^2CO$ with off-the shelf state of the art RGB deep networks, benchmarking it against the current state of the art in the field. For all these experiments, results clearly support the value of our algorithm.

All the $(DE)^2CO$ modules, for all architectures employed, are available at https://github.com/fmcarlucci/de2co.

4.2.2 Colorization of Depth Images

Although depth and RGB are modalities with significant differences, they also share enough similarities (edges, gradients, shapes) to make it plausible that convolutional filters learned from RGB data could be re-used effectively for representing colorized depth images. The approach currently adopted in the literature consists of designing ad-hoc colorization algorithms, as revised in section 2.2. We refer to these kind of approaches as *hand-crafted depth colorization*. Specifically, we choose ColorJet [36], SurfaceNormals [11] and *SurfaceNormals++* [1] as baselines against

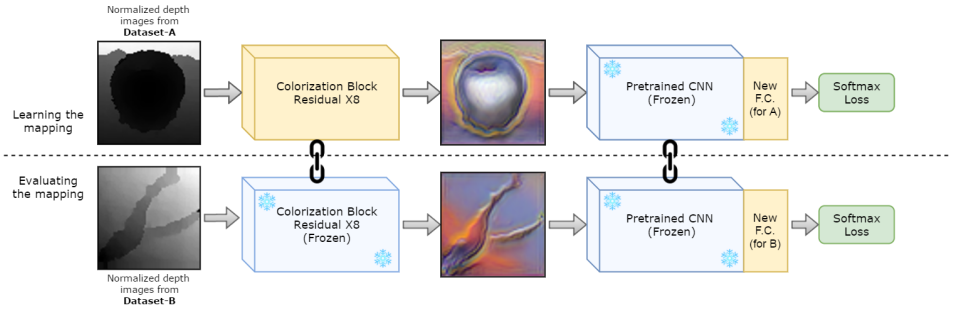


Figure 4.10. The $(DE)^2CO$ pipeline consists of two phases. First, we learn the mapping, from depth to color, maximizing the discrimination capabilities of a network pre trained on ImageNet. In this step the network is frozen and we are only learning the mapping and the final layer. We then evaluate the colorization on a *different* depth dataset: here we also freeze the colorization network and only train a new final layer for the testbed dataset.

which we will assess our data driven approach because of their popularity and effectiveness.

In the rest of the section we first briefly summarize ColorJet (subsection 4.2.2), SurfaceNormals and SurfaceNormals++ (subsection 4.2.2). We then describe our deep approach to depth colorization (subsection 4.2.2). To the best of our knowledge, $(DE)^2CO$ is the first deep colorization architecture applied successfully to depth images.

Hand-Crafted Depth Colorization: ColorJet

ColorJet works by assigning different colors to different depth values. The original depth map is firstly normalized between 0-255 values. Then the colorization works by mapping the lowest value to the blue channel and the highest value to the red channel. The value in the middle is mapped to green and the intermediate values are arranged accordingly [36]. The resulting image exploits the full RGB spectrum, with the intent of leveraging at best the filters learned by deep networks trained on very large scale RGB datasets like ImageNet. Although simple, the approach gave very strong results when tested on the Washington database, and when deployed on a robot platform.

Still, ColorJet was not designed to create realistic looking RGB images for the objects depicted in the original depth data (Figure 4.12, bottom row). This raises the question whether this mapping, although more effective than other methods presented in the literature, might be sub-optimal. In subsection 4.2.2 we will show that by fully embracing the end-to-end philosophy at the core of deep learning, it is indeed possible to achieve significantly higher recognition performances while at the same time producing more realistic colorized images.

Hand-Crafted Depth Colorization: SurfaceNormals(++)

The SurfaceNormals mapping has been often used to convert depth images to RGB [11, 163, 36]. The process is straightforward: for each pixel in the original image the corresponding surface normal is computed as a normalized $3D$ vector, which is then treated as an RGB color. Due to the inherent noisiness of the depth channel, such a direct conversion results in noisy images in the color space. To address this issue, the mapping we call *SurfaceNormals++* was introduced by Aakerberg [1]: first,

a recursive median filter is used to reconstruct missing depth values, subsequently a bilateral filter smooths the image to reduce noise, while preserving edges. Next, surface normals are computed for each pixel in the depth image. Finally the image is sharpened using the unsharp mask filter, to increase contrast around edges and other high-frequency components.

Deep Depth Colorization: (DE)^{2}CO

(DE)^{2}CO consists of feeding the depth maps, normalized into grayscale images, to a *colorization network* linked to a standard CNN architecture, pre-trained on ImageNet.

Given the success of deep colorization networks from grayscale images, we first tested existing architectures in this context [175]. Extensive experiments showed that while the visual appearance of the colorized images was very good, the recognition performances obtained when combining such network with pre-trained RGB architectures was not competitive. Inspired by the generator network in [12], we propose here a *residual* convolutional architecture (Figure 4.11). By design [63], this architecture is robust and allows for deeper training. This is helpful here, as (DE)^{2}CO requires stacking together two networks, which, even for not very deep architectures, might lead to vanishing gradient issues. Furthermore, residual blocks works at pixel level [12] helping to preserve locality.

Our architecture works as follows: the $1x228x228$ input depth map, reduced to $64x57x57$ size by a conv&pool layer, passes through a sequence of 8 residual blocks, composed by two small convolutions with batch normalization layers and leakyRelu [103] as non linearities. The last residual block output is convolved by a three features convolution to form the 3 channels image output. Its resolution is brought back to 228x228 by a *deconvolution* (upsampling) layer.

Our whole system for object recognition in the depth domain using deep networks pre-trained over RGB images can be summarized as follows: the entire network, composed by (DE)^{2}CO and the classification network of choice, is trained on an annotated reference depth image dataset. The weights of the chosen classification network are kept frozen in their pre-trained state, as the only layer that needs to be retrained is the last fully connected layer connected to the softmax layer. Meanwhile, the weights of (DE)^{2}CO are updated until convergence.

After this step, the depth colorization network has learned the mapping that maximizes the classification accuracy on the reference training dataset. It can now be used to colorize *any* depth image, from any data collection. Figure 4.12, top rows, shows exemplar images colorized with (DE)^{2}CO trained over different reference databases, in combination with two different architectures (CaffeNet, an implementation variant of AlexNet, and VGG-16 [139]).

We see that, compared to the images obtained with ColorJet and SurfaceNormal++, our colorization technique emphasizes the objects contours and their salient features while flatting the object background, while the other methods introduce either high frequency noise (SurfaceNormals++) or emphasize background gradient instead of focusing mainly on the objects (ColorJet). In the next subsection we will show how this qualitative advantage translates also into a numerical advantage, i.e. how learning (DE)^{2}CO on one dataset and performing depth-based object recognition on another leads to a very significant increase in performance on several settings, compared to hand-crafted color mappings.

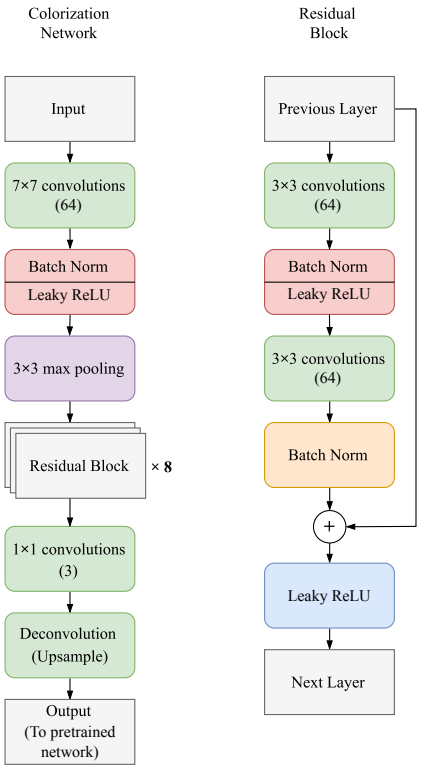


Figure 4.11. Overview of the $(DE)^2CO$ colorization network. On the left, we show the overall architecture; on the right, we show details of the residual block.

4.2.3 Experiments

We evaluated our colorization scheme on three main settings: an ablation study of how different depth mappings perform when the network weights are kept frozen (subsection 4.2.3), a comparison of depth performance with network finetuning (subsection 4.2.3) and finally an assessment of $(DE)^2CO$ when used in RGB-D object recognition tasks (subsection 4.2.3). Before reporting on our findings, we illustrate the baselines we used (subsection 4.2.3). The datasets we use (Washington [81], JHUIT-50 [87] and BigBIRD [140]) are described in section 2.2.3.

Experimental Setup

Hand-crafted Mappings According to previous works [36, 1], the two most effective mappings are *ColorJet* [36] and *SurfaceNormals* [11, 1]. For ColorJet we normalized the data between 0 and 255 and then applied the mapping using the OpenCV libraries[8]. For the SurfaceNormals mapping we considered two versions: the straightforward conversion of the depthmap to surface normals [11] and the enhanced version *SurfaceNormals++* [1] which uses extensive pre-processing and generally performs better[9].

[8] "COLORMAP_JET" from *http://opencv.org/*
[9] The authors graciously gave us their code for our experiments.

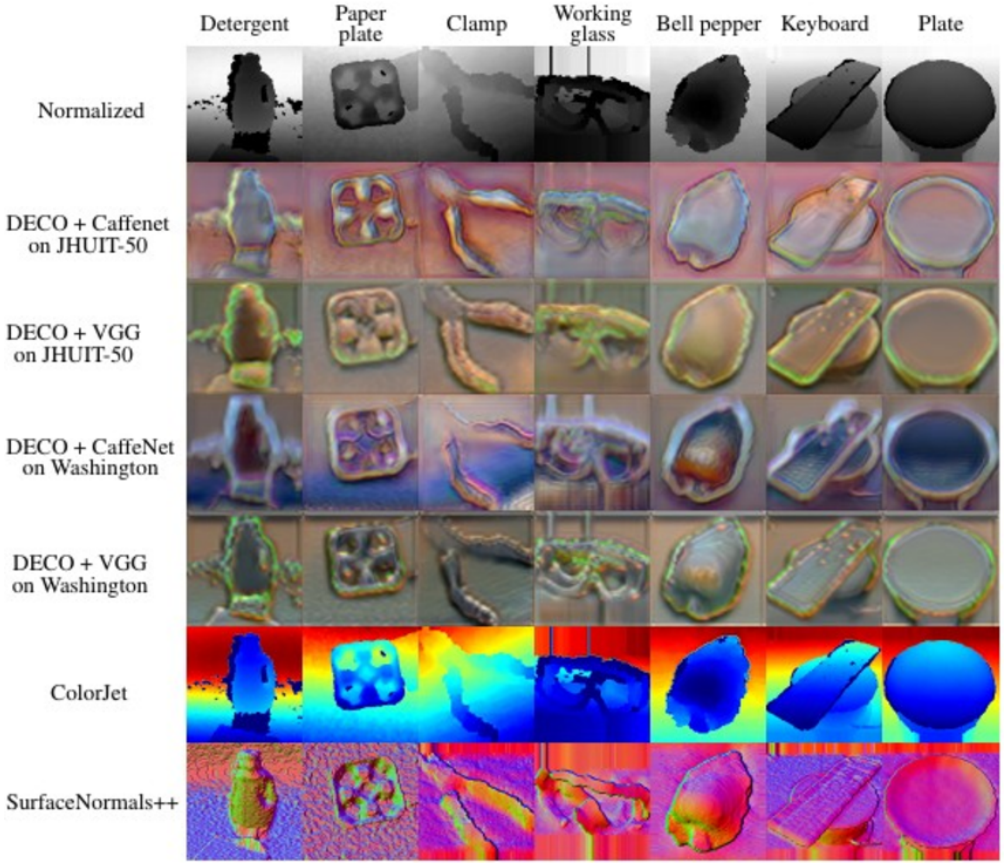


Figure 4.12. (DE)^{2}CO colorizations applied on different objects, taken from [87, 81, 140]. Top row shows the depth maps mapped to grayscale. From the second to the fourth row, we show the corresponding (DE)^{2}CO colorizations learned on different settings. Fifth row shows *ColorJet* views [36], while the last row shows the surface normals mapping. [1] *SurfaceNormals++*. These images showcase (DE)^{2}CO's ability to emphasize the object's shape and to capture its salient features.

Ablation Study

In this setting we compared our (DE)^{2}CO method against hand crafted mappings, using pre-trained networks as feature extractors and only retraining the last classification layer. We did this on the three datasets described in section 2.2.3, over four architectures: CaffeNet (a slight variant of the AlexNet [80]), VGG16 [139] and GoogleNet [148] were chosen because of their popularity within the robot vision community. We also tested the recent ResNet50 [63], which although not currently very used in the robotics domain, has some promising properties. In all cases we considered models pretrained on ImageNet [32], which we retrieved from Caffe's *Model Zoo*[10].

Training (DE)^{2}CO means minimizing the multinomial logistic loss of a network trained on RGB images. This means that our network is attached between the depth images and the pre-trained network, of which we freeze the weights of all but the last layer, which are relearned from scratch (see Figure 4.10). We trained each network-dataset combination for 50 epochs using the Nesterov solver [116] and 0.007

[10] *https://github.com/BVLC/caffe/wiki/Model-Zoo*

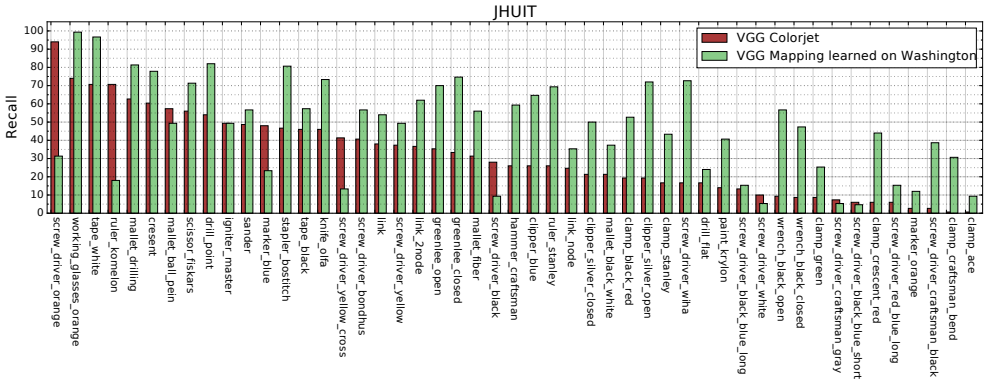


Figure 4.13. Per class recall on JHUIT-50, using VGG, with $(DE)^2CO$ learned from Washington. Recalls per class are sorted in decreasing order, according to ColorJet performance. In this setting, $(DE)^2CO$, while generally performing better, seems to focus on different perceptual properties and is thus, compared with the baseline, better at some classes rather than others.

starting learning rate (which is stepped down after 45%). During this phase, we used the whole *source* datasets, leaving aside only 10% of the samples for validation purposes.

When the dataset on which we train the colorizer is different from the test one, we simply retrain the new final layer (freezing all the rest) for the new classes.

Effectively we are using the pre-trained networks as feature extractors, as done in [133, 36, 172] and many others; for a performance analysis in the case of network finetuning we refer to paragraph 4.2.3. In this setting we used the Nesterov (for Washington and JHUIT-50) and ADAM (for BigBIRD) solvers. As we were only training the last fully connected layer, we learned a small handful of parameters with a very low risk of overfitting.

Table 4.4 reports the results from the ablation while Figure 4.13 focuses on the class recall for a specific experiment. For every architecture, we report the results obtained using ColorJet, SurfaceNormals (plain and enhanced) and $(DE)^2CO$ learned on a reference database between Washington or JHUIT-50, and $(DE)^2CO$ learned on the combination of Washington and JHUIT-50. For the CaffeNet and VGG networks we also present results on simple grayscale images. We attempted also to learn $(DE)^2CO$ from BigBIRD alone, and in combination with one (or both) of the other two databases. Results on BigBIRD only were disappointing, and results with/without adding it to the other two databases did not change the overall performance. We interpret this result as caused by the relatively small variability of objects in BigBIRD with respect to depth, and for sake of readability we decided to omit them in this work.

We see that, for all architectures and for all reference databases, $(DE)^2CO$ achieves higher results. The difference goes from +1.7%, obtained with CaffeNet on the Washington database, to the best of +16.8% for VGG16 on JHUIT-50. JHUIT-50 is the testbed database where, regardless of the chosen architecture, $(DE)^2CO$ achieves the strongest gains in performance compared to hand crafted mappings. Washington is, for all architectures, the database where hand crafted mappings perform best, with the combination Washington to CaffeNet being the most favorable to the shallow mapping.

On average it appears the CaffeNet is the architecture that performs best on this datasets; still, it should be noted that we are using here all architectures as

Network	Time (ms)
CaffeNet	695
VGG	1335
GoogleNet	1610
ResNet-50	1078
(DE)^{2}CO colorizer	400

Network	Time (s)
CaffeNet	1.87
(DE)^{2}CO + CaffeNet	1.23
VGG	2.91
(DE)^{2}CO + VGG	2.16

Table 4.3. Left: forward-backward time for 50 iterations, as by *caffe time*. Right: feature extraction times for 100 images; note that using (DE)^{2}CO actually speeds up the procedure. We explain this by noting that (DE)^{2}CO uses single channel images and thus needs to transfer only $\frac{1}{3}$ of the data from memory to the GPU - clearly the bottleneck here.

feature extractors rather than as classifiers. On this type of tasks, both ResNet and GoogLeNet-like networks are known to perform worse than CaffeNet [7], hence our results are consistent with what reported in the literature. In Table 4.5 we report a second ablation study performed on the width and depth of (DE)^{2}CO architecture. Starting from the standard (DE)^{2}CO made of 8 residual blocks with 64 filters for each convolutional layer (which we found to be the best all-around architecture), we perform additional experiments by doubling and halving the number of residual blocks and the number of filters in each convolutional layer. As it can be seen, the (DE)^{2}CO architecture is quite robust but can be potentially finetuned to each target dataset to further increase performance.

In table 4.3 we report runtimes for the considered networks. As the results show, while (DE)^{2}CO requires some extra computation time, in real life this is actually offset by the fact that only $\frac{1}{3}$ of the data is being moved to the GPU.

Method:	Washington [81]	JHUIT-50 [87]	BigBIRD Reduced [140]
VGG16 on Grayscale	74.9	33.7	22
VGG16 on ColorJet	75.2	35.3	19.9
VGG16 on SurfaceNormals	75.3	30.8	16.8
VGG16 on SurfaceNormals++	77.3	35.8	11.5
VGG16 (DE)^{2}CO learned on Washington	**79.6**	**52.7**	22.8
VGG16 (DE)^{2}CO learned on JHUIT-50	78.1	51.2	23.7
CaffeNet on Grayscale	76.6	44.6	22.9
CaffeNet on ColorJet	78.8	45.0	22.7
CaffeNet on SurfaceNormals	79.3	38.3	18.9
CaffeNet on SurfaceNormals++	81.4	44.8	14.0
CaffeNet (DE)^{2}CO learned on Washington	**83.1**	53.1	**28.6**
CaffeNet (DE)^{2}CO learned on JHUIT-50	79.1	**57.5**	25.2
GoogleNet on ColorJet	73.5	40.0	21.8
GoogleNet on SurfaceNormals	72.9	36.5	18.4
GoogleNet on SurfaceNormals++	**76.7**	41.5	13.9
GoogleNet (DE)^{2}CO learned on Washington	–	**51.9**	**25.2**
GoogleNet (DE)^{2}CO learned on JHUIT-50	76.6	–	24.4
ResNet50 on ColorJet	75.1	38.9	18.7
ResNet50 on SurfaceNormals	77.4	33.2	16.5
ResNet50 on SurfaceNormals++	**79.6**	45.4	13.8
ResNet50 (DE)^{2}CO learned on Washington	–	**45.5**	23.9
ResNet50 (DE)^{2}CO learned on JHUIT-50	76.4	–	**24.7**

Table 4.4. Object classification experiments in the depth domain, comparing (DE)^{2}CO and hand crafted mappings, using 5 pre-trained networks as feature extractors. Best results for each network-dataset combination are in **bold**, overall best in **red bold**. Extensive experiments were performed on VGG and Caffenet, while GoogleNet and ResNet act as reference.

filters/blocks	4 blocks	8 blocks	16 blocks
32 filters	56.5	52.8	57.1
64 filters	56.8	53.1	53.6
128 filters	53.1	53.9	53.3

Table 4.5. (DE)^{2}CO ablation study, learned on Washington, tested on JHUIT-50. Grid search optimization over width and depth of generator architecture shows improved results.

Finetuning

In our finetuning experiments we focused on the best performing network from the ablation, the CaffeNet (which is also used by current competitors [36, 1]), to see up to which degree the network could exploit a given mapping. The protocol was quite simple: all layers were free to move equally, the starting learning rate was 0.001 (with step down after 45%) and the solver was *SGD*. Training went on for 90 epochs for the Washington and JHUIT-50 datasets and 30 eps. for BigBIRD (a longer training was detrimental for all settings). To ensure a fair comparison with the static mapping methods, the (DE)^{2}CO part of the network was kept frozen during finetuning.

Results are reported in Table 4.6. We see that here the gap between hand-crafted and learned colorization methods is reduced (very likely the network is compensating existing weaknesses). *SurfaceNormals++* performs pretty well on Washington, but less so on the other two datasets (it's actually the worse on BigBIRD). Surprisingly, the simple grayscale conversion is the one that performs best on BigBIRD, but lags clearly behind on all other settings. (DE)^{2}CO on the other hand, performs comparably to the best mapping on every single setting **and** has a 5.9% lead on JHUIT-50; we argue that it is always possible to find a shallow mapping that performs very well on a specific dataset, but there are no guarantees it can generalize.

RGB-D

While this section focuses on how to best perform recognition in the depth modality using convnets, we wanted to provide a reference value for RGB-D object classification using (DE)^{2}CO on the depth channel. To classify RGB images we follow [1] and use a pretrained VGG16 which we finetuned on the target dataset (using the previously defined protocol). RGB-D classification is then performed, without further learning, by computing the weighted average (weight factor α was cross-validated) of the *fc8* layers from the RGB and Depth networks and simply selecting the most likely class (the one with the highest activations). This cue integration scheme can be seen as one of the simplest, off-the-shelf algorithm for doing classifications using two different modalities [153]. We excluded BigBIRD from this setting, due to lack of competing works to compare with.

Results are reported in Tables 4.7-4.8. We see that (DE)^{2}CO produces results on par or slightly superior to the current state of the art, even while using an extremely simple feature fusion method. This is remarkable, as competitors like [1, 36] use instead sophisticated, deep learning based cue integration methods. Hence, our ability to compete in this setting is all due to the (DE)^{2}CO colorization mapping, clearly more powerful than the other baselines.

It is worth stressing that, in spite of the different cue integration and depth mapping approaches compared in Tables 4.7-4.8, convnet results on RGB are already very high, hence in this setting the advantage brought by a superior performance on

the depth channel tends to be covered. Still, on Washington we achieve the second best result, and on JHUIT-50 we get the new state of the art.

Method:	Washington [81]	JHUIT-50 [87]	BigBIRD Reduced [140]
CaffeNet on Grayscale	82.7 ± 2.1	53.7	**29.6**
CaffeNet on ColorJet	83.8 ± 2.7	54.1	25.4
CaffeNet on SurfaceNormals++	**84.5** ± 2.9	55.9	17.0
CaffeNet $(DE)^2CO$ learned on Washington	84.0 ± 2.0	60.0	–
CaffeNet $(DE)^2CO$ learned on JHUIT-50	82.3 ± 2.3	**62.0**	–
CaffeNet $(DE)^2CO$ learned on Washington + JHUIT-50	84.0 ± 2.3	61.8	28.0

Table 4.6. CaffeNet finetuning using different colorization techniques.

4.2.4 Conclusions

This section presented a framework for learning deep colorization mappings. Our architecture follows the residual philosophy, learning how to map depth data to RGB images for a given pre-trained convolutional neural network. By using our $(DE)^2CO$ algorithm, as opposed to the hand-crafted colorization mappings commonly used in the literature, we obtained a significant jump in performance over three different benchmark databases, using four different popular deep networks pre trained over ImageNet. The visualization of the obtained colorized images further confirms how our algorithm is able to capture the rich informative content and the different facets of depth data. All the deep depth mappings presented in this section are available at https://github.com/fmcarlucci/de2co.

Future work will further investigate the effectiveness and generality of our approach, testing it on other RGB-D classification and detection problems, with various fine-tuning strategies and on several deep networks, pre-trained over different RGB databases, and in combination with RGB convnet with more advanced multimodal fusion approaches.

Method:	RGB	Depth	RGB-D
FusionNet [36]	84.1 ± 2.7	83.8 ± 2.7	91.3 ± 1.4
CNN + Fisher [92]	**90.8** ± 1.6	81.8 ± 2.4	**93.8** ± 0.9
DepthNet [20]	88.4 ± 1.8	83.8 ± 2.0	92.2 ± 1.3
CIMDL [163]	87.3 ± 1.6	84.2 ± 1.7	92.4 ± 1.8
FusionNet enhanced [1]	89.5 ± 1.9	**84.5** ± 2.9	93.5 ± 1.1
$(DE)^2CO$	89.5 ± 1.6	84.0 ± 2.3	93.6 ± 0.9

Table 4.7. Selected results on Washington RGB-D

Method:	RGB	Depth	RGB-D
DepthNet [20]	88.0	55.0	90.3
Beyond Pooling [87]	–	–	91.2
FusionNet enhanced [1]	**94.7**	56.0	95.3
$(DE)^2CO$	**94.7**	**61.8**	**95.7**

Table 4.8. Selected results on JHUIT-50

Chapter 5

Conclusions

This chapter summarizes the main results and contributions presented in this thesis, discusses the open issues and sketches possible future directions of research.

5.1 Summary

A quick overview of the current state-of-the-art categorization methods shows us that all deep learning approaches based on a large amount of samples reach impressive results on difficult datasets [80, 63]. However, most of them provide very few guarantees when only a small amount of training samples is available, or more in general, if there is a mismatch between the training and the testing distribution [155, 124].

The purpose of this thesis has been to understand how to best transfer what we know to a new domain, where data is scarce. We have shown that there are multiple ways of addressing this, depending on the availability of auxiliary data or pretrained models. More specifically, we tackled the issue from a conventional point of view, as an unsupervised domain adaptation problem, in chapter 3, and proposed multiple solutions: we showed that we can reduce the domain gap by aligning the feature distribution using an ad-hoc layer (AutoDIAL, 3.1) or by using GANs to project the images in the style of the other domain (3.2). In section 3.3 we showed a complex approach that, by aligning both features and images, reaches even better performance.

In the second part we presented methods which look at the problem from a different angle: how should we proceed when there is no *source* domain to leverage? In section 4.1 we use $3D$ CAD models to generate simulated depth views as a proxy for real data. A CNN is trained on this synthetic data and then evaluated on real RGB-D datasets; experiments confirm this to be a viable option.

A general purpose transfer learning method is presented in section 4.2. The intuition being that it is possible to learn a non linear transformation of the target image dataset that maps it as the most discriminative input for a network pretrained in a different modality. The method was successfully evaluated on the recognition task for Depth images, by mapping them to RGB; but its prerequisite-free structure means it could also be applied to other tasks (such as semantic segmentation or detection) and modalities.

In conclusion our work demonstrated that, by properly exploiting pre-existing knowledge, it is possible to train effective deep learning models even when data is scarce. For this purpose we presented a number of alternative solutions, ranging from classical domain adaptation approaches to innovative, across modalities, transfer learning techniques, each suitable for different learning conditions.

5.2 Open Issues

All the proposed algorithmic solutions for transfer learning and domain adaptation have been presented together with an analysis of their properties, discussing which is the best setting to apply them and evaluating their limits. We briefly describe in the following a few aspects of this work that might be somehow improved and remain relevant for future work.

Regarding methods which transform the input images(sections 3.3, 3.2) a strong limitation must be noted: to date, no generative approach has successfully managed to fully capture the nuances and the complexities of real life datasets. Until a breakthrough in this field (perhaps an improvements in GANs) solves the issue, algorithms which depends on it will always be limited to working on strongly structured data, such as faces or digits. Part of the problem is likely due to the size of these real life datasets: it is extremely hard to learn how to generate an object when you can only see 10 samples of it (i.e. Office [130]). The solution to this might lie, as has been done for object recognition [170], in the use of pretrained models which are finetuned for the specific task. To the best of our knowledge, to date, no one has had success with this idea on a GAN; as we use pretrained networks for classification, we should pretrain our GANs on large scale dataset and then finetune them on the smaller, target, dataset.

All of our proposed DA methods are designed to work in the unsupervised domain adaptation setting, with the assumption that no labels are available for the target. Clearly this hypothesis does not always hold and it is likely that, if target labels are available, a much better alignment between domains can be achieved. Future work should investigate how to best adapt our approaches to this semi-supervised setting. Likewise, our DA methods could be adapted to work on the recent problem of *open set* [17] domain adaptation, a more realistic scenario where only a few categories of interest are shared between source and target .

Our work on synthetic depth images, (section 4.1) could be significantly improved by trying to reduce the huge domain gap that exists between our virtual images and the real ones. Firstly, we could take into account and accurately model sensor noise during the rendering process. Furthermore our objects are simply floating in mid-air, which is unlike anything the robot could encounter in real life. The final domain shift could be eliminated by the use of GANs (similarly to SBADA-GAN) to make the synthetic images even more lifelike.

Our $(DE)^2CO$ (section 4.2) framework for transfer learning has been evaluated on the task of depth based object recognition. There are no specific requirements in the algorithm which limit its applicability to this setting, and it would be very interesting to see how much we can transfer to a different task (detection or semantic segmentation, for example) or a different modality. Multispectral cameras in particular seem a very interesting candidate: they are used for many things, from automated produce inspection, to artwork analysis and mine detection, but all relative datasets are quite small in size. Potentially, much better performances could be obtained if we could leverage the wealth of data we have in the RGB modality.

Appendix A

Domain Adaptation

A.1 AutoDIAL

A.1.1 DA-layers formulas

We rewrite Eq. 3.1 to make sample indexes and domain dependency explicit:

$$y_i = \texttt{DA}(x_i, d_i; \alpha) = \frac{x_i - \mu_{d_i,\alpha}}{\sqrt{\epsilon + \sigma^2_{d_i,\alpha}}}\,. \tag{A.1}$$

Using this notation, the global batch statistics are

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i\,, \quad \sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2\,, \tag{A.2}$$

while the domain-specific statistics are

$$\mu_d = \frac{1}{n_d}\sum_{i=1}^{n} 1_{d=d_i} x_i\,, \quad \sigma^2_d = \frac{1}{n_d}\sum_{i=1}^{n} 1_{d=d_i}(x_i - \mu_d)^2\,, \tag{A.3}$$

and the α-mixed statistics are

$$\begin{aligned}
\mu_{d,\alpha} &= \alpha\mu_d + (1-\alpha)\mu\,, \\
\sigma^2_{d,\alpha} &= \frac{\alpha}{n_d}\sum_{i=1}^{n} 1_{d=d_i}(x_i - \mu_{d,\alpha})^2 + \frac{1-\alpha}{n}\sum_{i=1}^{n}(x_i - \mu_{d,\alpha})^2 \\
&= \alpha\sigma^2_d + (1-\alpha)\sigma^2 + \alpha(1-\alpha)(\mu - \mu_d)^2\,,
\end{aligned} \tag{A.4}$$

where n and n_d are, respectively, the total number of samples in the batch and the number of samples of domain d in the batch.

A.1.2 Results on the SVHN – MNIST benchmark

In this section we report the results we obtain in the SVHN [117] to MNIST [86] transfer benchmark. We follow the experimental protocol in [44], using all SVHN images as the source domain and all MNIST images as the target domain, and compare with the following baselines: CORAL [143]; the Deep Adaptation Networks (DAN) [99]; the Domain-Adversarial Neural Network (DANN) in [44]; the Deep

Method	Accuracy
CORAL [93]	63.1
DAN [99]	71.1
DANN [44]	73.9
DRCN [47]	82.0
DSN [14]	82.7
ATN [131]	86.2
AutoDIAL	**90.3**

Table A.1. Results on the SVHN to MNIST benchmark.

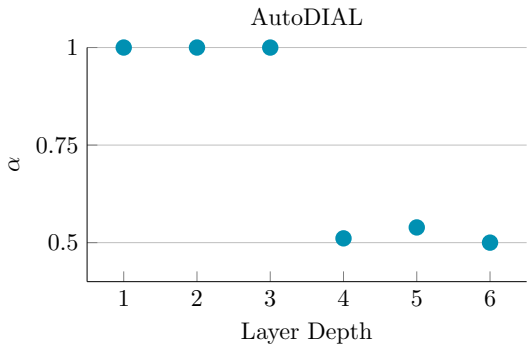


Figure A.1. α parameters learned on the SVHN – MNIST dataset, plotted as a function of layer depth.

Reconstruction Classification Network (DRCN) in [47]; the Domain Separation Networks (DSN) in [14]; the Asymmetric Tri-training Network (ATN) in [131].

As in all baselines, we adopt the network architecture in [42], adding DA-layers after each layer with parameters. Training is performed from scratch, using the same meta-parameters as for AlexNet (see section 3.1.2), with the following exceptions: initial learning rate $l_0 = 0.01$; 25 epochs; learning rate schedule defined by $l_p = l_0/(1 + \gamma p)^\beta$, where $\gamma = 10$, $\beta = 0.75$ and p is the learning progress linearly increasing from 0 to 1.

As shown in Table A.1, we set the new state of the art on this benchmark. It is worth of note that AutoDIAL also outperforms the methods, such as ATN and DSN, which expand the capacity of the original network by adding numerous learnable parameters, while only employing a single extra learnable parameter in each DA-layer . The α parameters learned by AutoDIAL on this dataset are plotted in Fig. A.1. Similarly to the case of AlexNet and Inception-BN on the Office-31 dataset, the network learns higher values of α in the bottom of the network and lower values of α in the top. In this case, however, we observe a steeper transition from 1 to 0.5, which interestingly corresponds with the transition from convolutional to fully-connected layers in the network.

A.1.3 Feature distributions

In this section we study the distributions of a set of randomly sampled features from different layers of AutoDIAL – Inception-BN, learned on the Amazon–DSLR task of the Office 31 dataset. In Fig. A.2 we compare the histograms of these features,

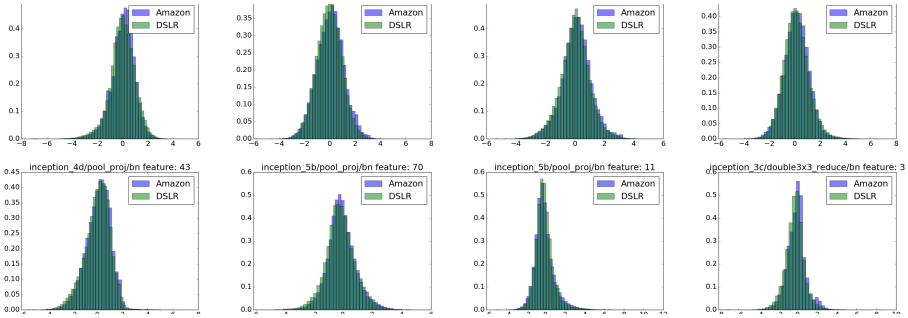


Figure A.2. Distributions of randomly sampled source/target features from different layers of AutoDIAL – Inception-BN learned on the Amazon–DSLR task of the Office 31 dataset (best viewed on screen).

computed on the whole source and target sets and taken *after* the DA-layers . The plots clearly show the aligning effect of our DA-layers , as most histograms are very closely matching. It is also interesting to note how the alignment effect seems to be mostly independent of the particular shape the distributions might take.

A.2 ADAGE

A.2.1 Transformers architectures

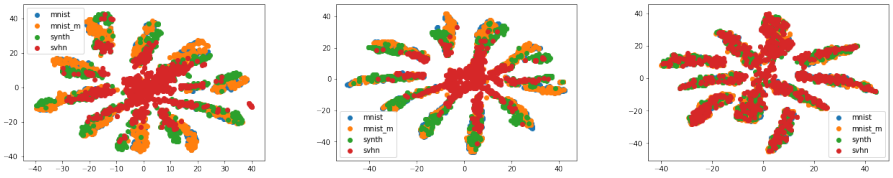
We have proposed two different types of architecture for the Transformer network: a residual version that follows the trend of [13] [128] [179] and an incremental version loosely inspired by [171]. The residual Transformer A.5 uses a sequence of residual blocks made of two $3x3$ convolutional layers with 64 filters each to calculate the residual of the input (an initial convolution brings the input data to 64 channels); this residual is then summed to the residual block input and fed to the next block in a full residual fashion. A total of 4 blocks have been used, after which a final convolutional layer brings the data back to three RGB channels. We finally use a last residual operation by summing up this output to the original input image: we found this operation beneficial to the stability of the algorithm. The incremental Transformer 2 instead slowly build up the feature maps size by concatenating the output of two 3x3 convolutional layers to its input. The feature maps size grows following this sequence: $3-8-16-32-4-64-128$ after which they are bring down to 3 RGB channels. No residual w.r.t the input image is used for the incremental version. In both residual and incremental architectures, each convolutional layer is always followed up by a Relu non linearity and by a Batch Normalization layer but in the last convolutional layer, where it is beneficial to skip the Batch Normalization layer in order to the output image not being limited by the normalization procedure.

A.2.2 Training details

During the experiments we found in some cases that the Transformer have been so successful in confusing the domains that the features domain discriminators cannot properly distinguish them, failing into provide a meaningful loss to the architecture. In those cases, the features domain discriminator produce an abnormal high loss and let the full architecture diverge. We found a simple trick that is able to remove the most cases of instability: whenever the features domain discriminator loss exceed a threshold value, stopping the backpropagation of this loss in this iteration will avoid the training collapse. We empirically found that a threshold value of 3.0 works fine for every case, and we didn't found any performance difference, both accuracy-wise and image quality-wise, between training with and without this trick.

(a) MNIST (b) MNIST-M (c) Synth. Digits (d) SVHN

Figure A.3. Examples of domain-agnostic digits generated by transformer block in the three source experiments with SVHN as target. Top two rows show images produced in the DG setting by residual based (line 1) and incremental based (line 2) transformers. Line 3 shows the original images and in the last two rows we display images produced by the residual (line 4) and incremental transformers in the DA setting. As in the case with MNIST-M as target, images transformed with the residual architecture tend to preserve more of the original input.



(a) Deep All (b) *DG* ADG (c) *MDA* ADG

Figure A.4. TSNE visualization of the classification features. Here *SVHN* is the target

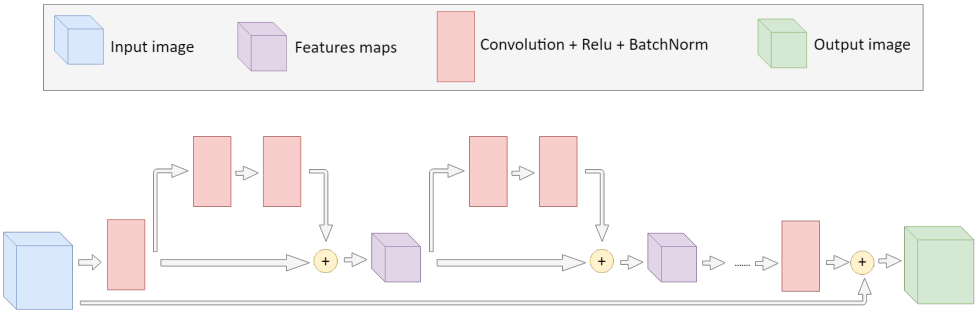


Figure A.5. Main blocks of our residual Transformer network. The blue block represent the input image data, while the red blocks are a sequence of Convolutional + Relu + Batch Normalization layers. The output of the two convolutional blocks are summed with the previous input. The number of kernels is fixed at 64 for the whole architecture until the last convolutional layer that brings the features back into 3 RGB channels (green block).

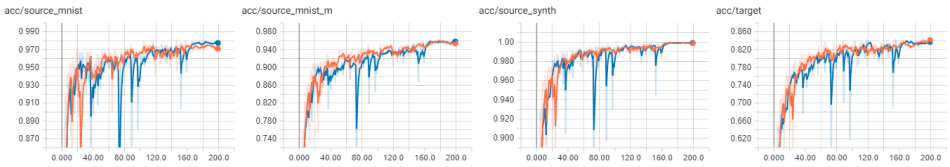


Figure A.6. Accuracy plots of our ADAGE residual transformer training on SVHN as target, red and blue are two separate runs. The three plots on the left show the accuracy on the **sources**, the one on right accuracy on the **target**. Note how there is a strong correlation between the performance on the source and on the target

	Sources	MNIST-M SYNTH	MNIST SYNTH	SYNTH MNIST	Avg.
	Target	MNIST	MNIST-M	SVHN	
DG	combine sources	98.7	62.6	69.5	76.9
	combine sources**?**	92.8	56.1	81.4	76.8
	MLDG **?**	99.1	61.2	69.7	76.7
	ADAGE Residual	**99.2**	65.8	74.6	79.9
	ADAGE Incremental	99.1	**66.3**	**76.4**	**80.3**
DA	combine sources	98.7	62.6	69.5	76.9
	combine sources**?**	92.8	56.1	81.4	76.8
	best single DANN **?**	96.7	59.1	81.8	79.2
	combine DANN **?**	92.5	65.1	77.6	78.4
	MDAN **?**	97.9	68.7	81.6	82.7
	ADAGE Residual	99.2	87.6	84.1	90.3
	ADAGE Incremental	**99.3**	**88.5**	**86.0**	**91.3**

	Sources	MNIST MNIST-M USPS	MNIST SVHN USPS	Avg.
	Target	SVHN	MNIST-M	
DG	combine sources	73.2	61.9	67.5
	combine sources**?**	64.6	60.7	62.7
	MLDG **?**	68.0	65.6	66.8
	ADAGE Residual	68.2	65.7	66.9
	ADAGE Incremental	**75.8**	**67.0**	**71.4**
DA	combine sources	73.2	61.9	67.5
	combine sources**?**	64.6	60.7	62.7
	separate DANN av.**?**	61.4	71.1	66.3
	combine DANN **?**	68.9	71.6	70.3
	DCTN **?**	77.5	70.9	74.2
	ADAGE Residual	82.3	84.1	83.2
	ADAGE Incremental	**85.3**	**85.3**	**85.3**

Table A.2. Extended edition of the classification accuracy results. *On the left*: experiments with 3 sources. *On the right*: experiments with 4 sources. Note that we report multiple versions of the *combine source* baseline: we tried our best to replicate the base training protocol and network, but we could not fully replicate the published results with our own implementation. Still, the average results of the two versions of combine source do not differ excessively, especially in the three sources scenario (left).

Bibliography

[1] A. Aakerberg and K. Nasrollahi. Depth value pre-processing for accurate transfer learning based rgb-d object recognition. In *IJCAI*, 2017.

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

[3] A. Angelova and P. M. Long. Benchmarking large-scale fine-grained categorization. In *Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on*, pages 532–539. IEEE, 2014.

[4] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):898–916, 2011.

[5] U. Asif, M. Bennamoun, and F. Sohel. Efficient rgb-d object categorization using cascaded ensembles of randomized decision trees. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 1295–1302. IEEE, 2015.

[6] U. Asif, M. Bennamoun, and F. A. Sohel. Rgb-d object recognition and grasp detection using hierarchical cascaded forests. *IEEE Transactions on Robotics*, 2017.

[7] H. Azizpour, A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson. Factors of transferability for a generic convnet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(9):1790–1802, 2016.

[8] S. Ben-David, T. Lu, T. Luu, and D. Pál. Impossibility theorems for domain adaptation. In *AISTATS*, 2010.

[9] G. Blanchard, G. Lee, and C. Scott. Generalizing from several related classification tasks to a new unlabeled sample. In *Advances in neural information processing systems*, pages 2178–2186, 2011.

[10] M. Blum, J. T. Springenberg, J. Wülfing, and M. A. Riedmiller. A learned feature descriptor for object recognition in rgb-d data. In *ICRA*, pages 1298–1303, 2012.

[11] L. Bo, X. Ren, and D. Fox. Unsupervised feature learning for RGB-D based object recognition. In *13th International Symposium on Experimental Robotics, ISER*, 2012.

[12] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. *arXiv preprint arXiv:1612.05424*, 2016.

[13] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan. Unsupervised pixel-level domain adaptation with gans. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

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

[15] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan. Domain Separation Networks. In *Neural Information Processing Systems (NIPS)*, 2016.

[16] L. Bruzzone and M. Marconcini. Domain adaptation problems: A dasvm classification technique and a circular validation strategy. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(5):770–787, 2010.

[17] P. P. Busto and J. Gall. Open set domain adaptation. In *ICCV*, pages 754–763, 2017.

[18] F. M. Carlucci, L. Porzi, B. Caputo, E. Ricci, and S. Rota Bulò. Autodial: Automatic domain alignment layers. In *International Conference on Computer Vision (ICCV)*, 2017.

[19] F. M. Carlucci, L. Porzi, B. Caputo, E. Ricci, and S. Rota Bulò. Just dial: domain alignment layers for unsupervised domain adaptation. In *International Conference on Image Analysis and Processing*, 2017.

[20] F. M. Carlucci, P. Russo, and B. Caputo. A deep representation for depth images from synthetic data. In *ICRA*, 2017.

[21] F. M. Carlucci, P. Russo, and B. Caputo. (de)2co: Deep depth colorization. *IEEE Robotics and Automation Letters*, 2018.

[22] F. M. Carlucci, P. Russo, T. Tommasi, and B. Caputo. Agnostic domain generalization. *arXiv preprint arXiv:1808.01102*, 2018.

[23] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.

[24] Y. Cheng, R. Cai, X. Zhao, and K. Huang. Convolutional fisher kernels for rgb-d object recognition. In *3D Vision (3DV), 2015 International Conference on*, pages 135–143. IEEE, 2015.

[25] Y. Cheng, X. Zhao, K. Huang, and T. Tan. Semi-supervised learning for rgb-d object recognition. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pages 2377–2382. IEEE, 2014.

[26] Y. Cheng, X. Zhao, K. Huang, and T. Tan. Semi-supervised learning and feature evaluation for rgb-d object recognition. *Computer Vision and Image Understanding*, 139:149–160, 2015.

[27] Z. Cheng, Q. Yang, and B. Sheng. Deep colorization. In *ICCV*, 2015.

[28] F. Chollet. keras. https://github.com/fchollet/keras, 2017.

[29] W.-S. Chu, F. De la Torre, and J. F. Cohn. Selective transfer machine for personalized facial action unit detection. In *CVPR*, 2013.

[30] K. Crammer, M. Kearns, and J. Wortman. Learning from multiple sources. *J. Mach. Learn. Res.*, 9:1757–1774, June 2008.

[31] H. Daume III. Frustratingly easy domain adaptation. In *Annual Meeting of the Association of Computational Linguistics (ACL)*, 2007.

[32] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[33] 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.

[34] L. Duan, I. W. Tsang, D. Xu, and T.-S. Chua. Domain adaptation from multiple sources via auxiliary classifiers. In *International Conference on Machine Learning (ICML)*, 2009.

[35] L. Duan, D. Xu, and I. W. Tsang. Domain adaptation from multiple sources: A domain-dependent regularization approach. *IEEE Transactions on Neural Networks and Learning Systems*, 23(3):504–518, March 2012.

[36] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard. Multimodal deep learning for robust rgb-d object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 681–687. IEEE, 2015.

[37] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. An evaluation of the rgb-d slam system. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1691–1696. IEEE, 2012.

[38] S. R. Fanello, C. Ciliberto, L. Natale, and G. Metta. Weakly supervised strategies for natural object recognition in robotics. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4223–4229. IEEE, 2013.

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

[40] G. French, M. Mackiewicz, and M. Fisher. Self-ensembling for visual domain adaptation. In *Blind Submission, International Conference on Learning Representations (ICLR)*, 2018.

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

[42] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, 2015.

[43] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.*, 17(1):2096–2030, 2016.

[44] 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.

[45] M. Ghifary, D. Balduzzi, W. B. Kleijn, and M. Zhang. Scatter component analysis: A unified framework for domain adaptation and domain generalization. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (1):1–1, 2017.

[46] M. Ghifary, W. B. Kleijn, M. Zhang, and D. Balduzzi. Domain generalization for object recognition with multi-task autoencoders. In *International Conference on Computer Vision, (ICCV)*, 2015.

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

[48] M. Ghifary, W. B. Kleijn, M. Zhang, D. Balduzzi, and W. Li. Deep reconstruction-classification networks for unsupervised domain adaptation. In *European Conference on Computer Vision (ECCV)*, 2016.

[49] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

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

[51] B. Gong, Y. Shi, F. Sha, and K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *CVPR*, 2012.

[52] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[53] R. Gopalan, R. Li, and R. Chellappa. Domain adaptation for object recognition: An unsupervised approach. In *International Conference on Computer Vision (ICCV)*, 2011.

[54] Y. Grandvalet and Y. Bengio. Semi-supervised learning by entropy minimization. In *NIPS*, 2004.

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

[56] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Learning rich features from rgb-d images for object detection and segmentation. In *European Conference on Computer Vision*, pages 345–360. Springer, 2014.

[57] S. Gupta, J. Hoffman, and J. Malik. Cross modal distillation for supervision transfer. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[58] A. Habrard, J.-P. Peyrache, and M. Sebban. Iterative self-labeling domain adaptation for linear structured image classification. *International Journal on Artificial Intelligence Tools*, 22(5), 2013.

[59] P. Haeusser, T. Frerix, A. Mordvintsev, and D. Cremers. Associative domain adaptation. In *International Conference on Computer Vision (ICCV)*, 2017.

[60] J. Han, L. Shao, D. Xu, and J. Shotton. Enhanced computer vision with microsoft kinect sensor: A review. *IEEE transactions on cybernetics*, 43(5):1318–1334, 2013.

[61] A. Handa, T. Whelan, J. McDonald, and A. J. Davison. A benchmark for rgb-d visual odometry, 3d reconstruction and slam. In *Robotics and automation (ICRA), 2014 IEEE international conference on*, pages 1524–1531. IEEE, 2014.

[62] 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.

[63] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[64] A. Hermans, G. Floros, and B. Leibe. Dense 3d semantic mapping of indoor scenes from rgb-d images. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 2631–2638. IEEE, 2014.

[65] J.-J. Hernandez-Lopez, A.-L. Quintanilla-Olvera, J.-L. López-Ramírez, F.-J. Rangel-Butanda, M.-A. Ibarra-Manzano, and D.-L. Almanza-Ojeda. Detecting objects using color and depth segmentation with kinect sensor. *Procedia Technology*, 3:196–204, 2012.

[66] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[67] J. Hoffman, S. Gupta, and T. Darrell. Learning with side information through modality hallucination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 826–834, 2016.

[68] J. Hoffman, S. Gupta, J. Leong, S. Guadarrama, and T. Darrell. Cross-modal adaptation for rgb-d detection. In *International Conference in Robotics and Automation (ICRA)*, 2016.

[69] J. Hoffman, B. Kulis, T. Darrell, and K. Saenko. Discovering latent domains for multisource domain adaptation. In *European Conference on Computer Vision (ECCV)*, 2012.

[70] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. A. Efros, and T. Darrell. Cycada: Cycle-consistent adversarial domain adaptation. *arXiv preprint arXiv:1711.03213*, 2017.

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

[72] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2016)*, 35(4), 2016.

[73] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[74] I.-H. Jhuo, D. Liu, D. T. Lee, and S.-F. Chang. Robust visual domain adaptation with low-rank reconstruction. In *Computer Vision and Pattern Recognition (CVPR)*, 2012.

[75] 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 *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[76] C. Keskin, F. Kıraç, Y. E. Kara, and L. Akarun. Real time hand pose estimation using depth sensors. In *Consumer depth cameras for computer vision*, pages 119–137. Springer, 2013.

[77] A. Khosla, T. Zhou, T. Malisiewicz, A. A. Efros, and A. Torralba. Undoing the damage of dataset bias. In *European Conference on Computer Vision*, pages 158–171. Springer, 2012.

[78] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim. Learning to discover cross-domain relations with generative adversarial networks. In *International Conference on Machine Learning, ICML*, pages 1857–1865, 2017.

[79] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[80] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[81] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1817–1824. IEEE, 2011.

[82] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1817–1824. IEEE, 2011.

[83] G. Larsson, M. Maire, and G. Shakhnarovich. Learning representations for automatic colorization. In *ECCV*, 2016.

[84] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, Dec 1989.

[85] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.

[86] 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.

[87] C. Li, A. Reiter, and G. D. Hager. Beyond spatial pooling: Fine-grained representation learning in multiple domains. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4913–4922, 2015.

[88] C. Li, A. Reiter, and G. D. Hager. Beyond spatial pooling: fine-grained representation learning in multiple domains. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4913–4922, 2015.

[89] D. Li, Y. Yang, Y. Song, and T. M. Hospedales. Learning to generalize: Meta-learning for domain generalization. In *Conference of the Association for the Advancement of Artificial Intelligence (AAAI)*, 2018.

[90] D. Li, Y. Yang, Y. Z. Song, and T. M. Hospedales. Deeper, broader and artier domain generalization. In *International Conference on Computer Vision (ICCV)*, 2017.

[91] H. Li, S. Jialin Pan, S. Wang, and A. C. Kot. Domain generalization with adversarial feature learning. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[92] W. Li, Z. Cao, Y. Xiao, and Z. Fang. Hybrid rgb-d object recognition using convolutional neural network and fisher vector. In *Chinese Automation Congress (CAC), 2015*, pages 506–511. IEEE, 2015.

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

[94] H. Liu, M. Shao, and Y. Fu. Structure-preserved multi-source domain adaptation. In *ICDM*, pages 1059–1064. IEEE, 2016.

[95] M.-Y. Liu, T. Breuel, and J. Kautz. Unsupervised image-to-image translation networks. In *Neural Information Processing Systems (NIPS)*, 2017.

[96] M.-Y. Liu and O. Tuzel. Coupled generative adversarial networks. In *Advances in neural information processing systems*, pages 469–477, 2016.

[97] M.-Y. Liu and O. Tuzel. Coupled generative adversarial networks. In *Advances in neural information processing systems (NIPS)*, 2016.

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

[99] M. Long and J. Wang. Learning transferable features with deep adaptation networks. In *ICML*, 2015.

[100] M. Long, J. Wang, and M. I. Jordan. Deep transfer learning with joint adaptation networks. *arXiv preprint arXiv:1605.06636*, 2016.

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

[102] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[103] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

[104] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[105] M. Mancini, S. R. Bulo, B. Caputo, and E. Ricci. Robust place categorization with deep domain generalization. *IEEE Robotics and Automation Letters*, 2018.

[106] M. Mancini, H. Karaoguz, E. Ricci, P. Jensfelt, and B. Caputo. Kitting in the wild through online domain adaptation. *IROS*, 2018.

[107] M. Mancini, L. Porzi, S. R. Bulò, B. Caputo, and E. Ricci. Boosting domain adaptation by discovering latent domains. *CVPR*, 2018.

[108] Y. Mansour, M. Mohri, and A. Rostamizadeh. Domain adaptation with multiple sources. In *Neural Information Processing Systems (NIPS)*, 2009.

[109] X. Mao, Q. Li, H. Xie, R. Y. Lau, and Z. Wang. Multi-class generative adversarial networks with the l2 loss function. *arXiv preprint arXiv:1611.04076*, 2016.

[110] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 922–928. IEEE, 2015.

[111] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

[112] B. Moiseev, A. Konev, A. Chigorin, and A. Konushin. Evaluation of traffic sign recognition methods trained on synthetically generated data. In *International Conference on Advanced Concepts for Intelligent Vision Systems (ACIVS)*, 2013.

[113] E. Morvant. Domain adaptation of weighted majority votes via perturbed variation-based self-labeling. *Pattern Recognition Letters*, 51:37–43, 2015.

[114] S. Motiian, M. Piccirilli, D. A. Adjeroh, and G. Doretto. Unified deep supervised domain adaptation and generalization. In *International Conference on Computer Vision (ICCV)*, 2017.

[115] K. Muandet, D. Balduzzi, and B. Schölkopf. Domain generalization via invariant feature representation. In *International Conference on Machine Learning, ICML*, 2013.

[116] Y. Nesterov. A method of solving a convex programming problem with convergence rate o (1/k2). In *Soviet Mathematics Doklady*, volume 27, pages 372–376, 1983.

[117] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 5, 2011.

[118] L. Niu, Q. Tang, A. Veeraraghavan, and A. Sabharwal. Learning from noisy web data with category-level supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7689–7698, 2018.

[119] A. Odena, C. Olah, and J. Shlens. Conditional image synthesis with auxiliary classifier GANs. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 70, 2017.

[120] T. J. O'Neill. Normal discrimination with unclassified observations. *Jounral of the American Statistical Association*, 73(364):821–826, 1978.

[121] F. Orabona, L. Jie, and B. Caputo. Online-batch strongly convex multi kernel learning. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 787–794. IEEE, 2010.

[122] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2011.

[123] J. Papon and M. Schoeler. Semantic pose using deep networks trained on synthetic rgb-d. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 774–782, 2015.

[124] F. Perronnin, J. Sénchez, and Y. L. Xerox. Large-scale image categorization with explicit data embedding. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2297–2304. IEEE, 2010.

[125] U. Rafi, J. Gall, and B. Leibe. A semantic occlusion model for human pose estimation from a single depth image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 67–74, 2015.

[126] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[127] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[128] P. Russo, F. M. Carlucci, T. Tommasi, and B. Caputo. From source to target and back: symmetric bi-directional adaptive gan. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[129] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *ECCV*, 2010.

[130] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *European Conference on Computer Vision, (ECCV)*, 2010.

[131] K. Saito, Y. Ushiku, and T. Harada. Asymmetric tri-training for unsupervised domain adaptation. In *International Conference on Machine Learning, (ICML)*, 2017.

[132] S. Sankaranarayanan, Y. Balaji, C. D. Castillo, and R. Chellappa. Generate to adapt: Aligning domains using generative adversarial networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[133] M. Schwarz, H. Schulz, and S. Behnke. Rgb-d object recognition and pose estimation based on pre-trained convolutional neural network features. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 1329–1335. IEEE, 2015.

[134] O. Sener, H. O. Song, A. Saxena, and S. Savarese. Learning transferrable representations for unsupervised domain adaptation. In *Advances in Neural Information Processing Systems*, pages 2110–2118, 2016.

[135] S. Shankar, V. Piratla, S. Chakrabarti, S. Chaudhuri, P. Jyothi, and S. Sarawagi. Generalizing across domains via cross-gradient training. In *International Conference on Learning Representations (ICLR)*, 2018.

[136] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014.

[137] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. *Preprint arXiv:1612.07828*, 2016.

[138] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[139] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015.

[140] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel. Bigbird: A large-scale 3d database of object instances. In *ICRA*, 2014.

[141] R. Socher, B. Huval, B. Bath, C. D. Manning, and A. Y. Ng. Convolutional-recursive deep learning for 3d object classification. In *Advances in neural information processing systems*, pages 656–664, 2012.

[142] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. The german traffic sign recognition benchmark: a multi-class classification competition. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 1453–1460. IEEE, 2011.

[143] B. Sun, J. Feng, and K. Saenko. Return of frustratingly easy domain adaptation. In *AAAI*, 2016.

[144] B. Sun, J. Feng, and K. Saenko. Return of frustratingly easy domain adaptation. In *Conference of the Association for the Advancement of Artificial Intelligence (AAAI)*, 2016.

[145] B. Sun and K. Saenko. Deep coral: Correlation alignment for deep domain adaptation. *arXiv preprint arXiv:1607.01719*, 2016.

[146] L. Sun, C. Zhao, and R. Stolkin. Weakly-supervised dcnn for rgb-d object recognition in real-world applications which lack large-scale annotated training data. *arXiv preprint arXiv:1703.06370*, 2017.

[147] Q. Sun, R. Chattopadhyay, S. Panchanathan, and J. Ye. A two-stage weighting framework for multi-source domain adaptation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 505–513. 2011.

[148] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[149] Y. Taigman, A. Polyak, and L. Wolf. Unsupervised cross-domain image generation. In *International Conference on Learning Representations (ICLR)*, 2017.

[150] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. In *COURSERA: Neural Networks for Machine Learning*, 2012.

[151] T. Tommasi. Learning to learn by exploiting prior knowledge. 2013.

[152] T. Tommasi, M. Lanzi, P. Russo, and B. Caputo. Learning the roots of visual domain shift. *arXiv preprint arXiv:1607.06144*, 2016.

[153] T. Tommasi, F. Orabona, and B. Caputo. Discriminative cue integration for medical image annotation. *Pattern Recognition Letters*, 29(15):1996–2002, 2008.

[154] T. Tommasi and T. Tuytelaars. A testbed for cross-dataset analysis. In *ECCV*, 2014.

[155] A. Torralba and A. A. Efros. Unbiased look at dataset bias. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1521–1528. IEEE, 2011.

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

[157] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko. Simultaneous deep transfer across domains and tasks. In *International Conference in Computer Vision (ICCV)*, 2015.

[158] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko. Adversarial discriminative domain adaptation. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[159] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

[160] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

[161] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proc. CVPR*, pages 5018–5027, 2017.

[162] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *Trans. Img. Proc.*, 13(4):600–612, Apr. 2004.

[163] Z. Wang, R. Lin, J. Lu, J. Feng, et al. Correlated and individual multi-modal deep learning for rgb-d object recognition. *arXiv preprint arXiv:1604.01655*, 2016.

[164] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[165] J. Xie, W. Hu, S.-C. Zhu, and Y. N. Wu. Learning sparse frame models for natural image patterns. *IJCV*, 114(2-3):91–112, 2015.

[166] R. Xu, Z. Chen, W. Zuo, J. Yan, and L. Lin. Deep cocktail network: Multi-source unsupervised domain adaptation with category shift. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

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

[168] J. Yang, R. Yan, and A. G. Hauptmann. Cross-domain video concept detection using adaptive svms. In *Proceedings of the 15th ACM international conference on Multimedia*, pages 188–197. ACM, 2007.

[169] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *NIPS*, 2014.

[170] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *NIPS*, 2014.

[171] F. Yu, D. Wang, E. Shelhamer, and T. Darrell. Deep layer aggregation. *arXiv preprint arXiv:1707.06484*, 2017.

[172] H. F. Zaki, F. Shafait, and A. Mian. Convolutional hypercube pyramid for accurate rgb-d object category and instance recognition. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 1685–1692. IEEE, 2016.

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

[174] N. Zhang, R. Farrell, and T. Darrell. Pose pooling kernels for sub-category recognition. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3665–3672. IEEE, 2012.

[175] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. *ECCV*, 2016.

[176] H. Zhao, S. Zhang, G. Wu, J. ao P. Costeira, J. M. F. Moura, and G. J. Gordon. Multiple source domain adaptation with adversarial learning. In *Workshop of the International Conference on Learning Representations (ICLR-W)*, 2018.

[177] L. Zheng, Y. Zhao, S. Wang, J. Wang, and Q. Tian. Good practice in cnn feature transfer. *arXiv preprint arXiv:1604.00133*, 2016.

[178] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, pages 487–495, 2014.

[179] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *International Conference on Computer Vision (ICCV)*, 2017.

[180] X. Zhu. Semi-supervised learning literature survey. 2005.