



SAPIENZA
UNIVERSITÀ DI ROMA

Multimodal Deep Domain Adaptation

Candidate: Silvia Bucci

Thesis Advisor: Barbara Caputo

Thesis External Advisor:

Mohammad Reza Loghmani

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica
Department of Computer, Control, and Management Engineering

Master of Science in Artificial Intelligence and Robotics

Sapienza University of Rome

A.A. 2017/2018

Multimodal Deep Domain Adaptation

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica
Department of Computer, Control, and Management Engineering (DIAG)
Master of Science in Artificial Intelligence and Robotics

Candidate
Silvia Bucci
1532075

Thesis Advisor
Barbara Caputo

Thesis External Advisor
Mohammad Reza Loghmani

A/A 2017/2018

Ringraziamenti

Ringrazio la mia relatrice, la professoressa Barbara Caputo, per la bellissima opportunità di tesi che mi ha offerto; ringrazio Mohammad per i suoi insegnamenti e per la dedizione con cui mi ha seguito.

Un grande, enorme, infinito GRAZIE va ai miei genitori. Papà, nei tuoi occhi vedo l'amore, nei tuoi occhi mi vedo perfetta, grazie per avermi sempre fatta sentire così, grazie perché quando mi sento persa il tuo sguardo mi fa ritrovare me stessa. Mamma, nei tuoi abbracci trovo tutto quello di cui ho bisogno, il "profumo di mamma" come dicevo da piccola, non hai idea della forza che mi trasmetti con il tuo coraggio, è grazie a te se adesso sono pronta per "correre verso il mondo". Non smetterò mai, mai di ringraziarvi per il vostro amore incondizionato, siete la mia roccia, siete tutto.

Altri due grazie vanno alle mie sorelle, alle mie compagne di gioco preferite da piccole e alle mie compagne di litigate preferite da grandi. La mia dolce Francesca, fragile e sensibile. Grazie, perchè spesso mi sai capire come nessun'altro. Ovunque andrò porterò sempre con me il tuo sguardo in cui vedo un abisso, non smettere MAI di lottare. Nicoletta, grazie per tutte le volte in cui ci sei stata, ne abbiamo passate tante, troppe, in questi cinque anni. Ma al termine di questo viaggio posso dire che ne è valsa la pena, sono felice di aver condiviso questa avventura con te. GRAZIE ai miei quattro pilastri indiscussi, siete il mio tesoro più grande.

Un grazie va a mio nonno che "mi protegge sulla stella più bella che c'è", grazie nonno perchè hai fatto tanto in vita e continui a fare tanto nei miei ricordi. Ringrazio mia nonna che con la sua dolce ingenuità mi fa ricordare il valore delle cose semplici, sei il caffè e latte che mi preparavi nelle mattine d'estate, sei la focaccia che mi aspettava sempre quando venivo a casa tua, sei una boccata d'aria in questo mondo che non aspetta. Ringrazio mia zia che piano piano sto ritrovando, grazie a te e a Gianni per sostenermi, incoraggiarmi e condividere con me momenti importanti, con la speranza che il nostro rapporto continui a crescere sempre di più.

E poi ci sono i cuginetti che, in realtà, sono dei fratelli. Gae, animo gentile, quando siamo insieme sento che mi sai leggere esattamente per quella che sono, ti ringrazio perchè i tuoi occhi premurosi mi convincono che, nonostante tutto, vado bene così. Chiaretta, ti ringrazio per le nostre intense maratone di studio, ti ringrazio per le avventure in giro per il mondo, ti ringrazio per le chiacchierate infinite dalle quali esco sempre arricchita. Grazie ad entrambi perchè fate parte delle mie certezze. Così come la mia bella zietta lottatrice. Nonostante tutte le sfide che hai dovuto affrontare nella tua vita, mi hai sempre trasmesso serenità. Pensando a te mi ricordo che c'è sempre un motivo per cui vale la pena essere felici. Un grazie va anche a mio zio, che, a suo modo, mi è stato vicino.

Mirco, sei il mio migliore amico, il mio complice perfetto, sei la mia metà. Da quando ho te nella mia vita tutto è più bello. Grazie per la dolcezza con cui, anche senza che

io me ne accorgo e forse, a volte, in modi che non comprendo, ti prendi cura di me. Ti ringrazio perchè vuoi la mia felicità, perchè non passa un giorno in cui non mi fai sentire amata, non passa un giorno in cui non fai di tutto per farmi sorridere. Le nostre anime sono talmente vicine che a volte sono spaventata. Perchè se perdo te perdo me. Ti ringrazio perchè ogni giorno, sempre di più, sento che con te ho vinto.

Ringrazio la mia persona, Giulia. Spero di darti indietro almeno la metà di quello che dai tu a me. Grazie per farmi capire le cose con umiltà e sincera bontà, sei il mio porto sicuro. Ringrazio Valeruccia, perchè, seppure così diversa da me, mi ha conquistata. Grazie per la tua spontaneità, per l'affetto che mi dimostri, per i momenti difficili in cui mi sei stata vicina, non li dimenticherò.

E poi c'è la mia Ariù. Grazie, perchè con la tua profondità e la tua intelligenza mi fai sempre vedere le cose da una prospettiva diversa. Sei una persona speciale, che mi fa sentire speciale. Ti ringrazio per la fiducia che hai sempre avuto in me, ti ringrazio per farmi vedere la vita con la tua ironia, sei uno spasso.

Ringrazio le mie amiche di sempre, Grazia Ludovica Giulia e Francesca. Grazie per avermi fatto conoscere il valore dell'amicizia, siamo cresciute insieme, il nostro legame è cresciuto con noi e per me continuate ad essere un punto di riferimento essenziale.

E infine un'enorme grazie va alla vera ricchezza che mi ha lasciato questa laurea. Grazie al gruppo del "c'è", in particolare grazie a Fra che ormai è una star, grazie a Simone per tutti i gelati che abbiamo divorziato dopo le lezioni di probabilità, grazie a Roberto che vedrò sempre come il coniglietto di Galbusera, grazie a Moreno il mio cognatino che solo Dio sa quante ne ha dovute sentire di urla in casa e grazie a Manuel perchè i nostri dialoghi sembrano la trama di un fumetto. Grazie a voi per aver condiviso con me le prime ansie, i primi esami, le prime risate.

Ringrazio Robertino, perchè grazie a lui mi sono rimasti grossomodo sei anni di vita, grazie Cip per aver seguito il mio consiglio sulla pista di pattinaggio, ringrazio Franci (C.) per le costruttive chiacchierate durante le lezioni di Fisica, ringrazio Scarselli per avermi fatto comprare l'utilissimo manuale di Java e infine un grazie va a Bissolino per avermi mostrato come un essere umano possa vivere dentro una BOLLA (cit.).

Poi c'è il gruppo di matti che si sono aggiunti nella seconda parte di questa avventura. Grazie Bea per essere stata sempre al mio fianco nell'ansia, grazie Darietto per essere riuscito a farmi ridere anche nella più buia disperazione (causata da un certo gabbiano), grazie Atif perchè è solo grazie a te se ho tanti bei piccioni in galleria, ringrazio Stefano per avermi voluto conoscere nonostante le coccinelle, grazie Cate per le giornate indimenticabili al mare, un grazie infinito va a Andrea per avermi fatto conoscere Franco, grazie Niki per le torte che ci hai preparato con amore ad ogni compleanno, ringrazio Dani per il simpaticissimo scherzo al ginocchio e un grazie va a Lorenzo per farmi sempre sorridere con la sua acuta ironia.

Ringrazio tutti per le serate a san Lorenzo, per avermi fatto vivere la magia di questi anni. Porterò con me il ricordo di ognuno di voi.

La felicità è reale solo se condivisa.
..perchè alla fine i ringraziamenti sono la parte migliore.

Abstract

Typically a classifier trained on a given dataset (*source domain*) does not perform well if it is tested on data acquired in a different setting (*target domain*). This is the problem that *domain adaptation* (DA) tries to overcome and, while it is a well explored topic in computer vision, it is largely ignored in robotic vision where usually visual classification methods are trained and tested in the same domain. Robots should be able to deal with unknown environments, recognize objects and use them in the correct way, so it is important to explore the domain adaptation scenario also in this context.

The goal of the project is to define a benchmark and a protocol for multi-modal domain adaptation that is valuable for the robot vision community. With this purpose some of the state-of-the-art DA methods are selected: *Deep Adaptation Network* (DAN), *Domain Adversarial Training of Neural Network* (DANN), *Automatic Domain Alignment Layers* (AutoDIAL) and *Adversarial Discriminative Domain Adaptation* (ADDA). Evaluations have been done using different data types: RGB only, depth only and RGB-D over the following datasets, designed for the robotic community: *RGB-D Object Dataset* (ROD), *Web Object Dataset* (WOD), *Autonomous Robot Indoor Dataset* (ARID), *Big Berkeley Instance Recognition Dataset* (BigBIRD) and *Active Vision Dataset*.

Although progresses have been made on the formulation of effective adaptation algorithms and more realistic object datasets are available, the results obtained show that, training a sufficiently good object classifier, especially in the domain adaptation scenario, is still an unsolved problem. Also the best way to combine depth with RGB informations to improve the performance is a point that needs to be investigated more.

Contents

1	Introduction	8
2	Related works	12
3	Domain Adaptation Algorithms	14
3.1	Deep Adaptation Network	14
3.1.1	Multiple Kernel Maximum Mean Discrepancy	14
3.1.2	Model	15
3.1.3	Algorithm	17
3.2	Domain Adversarial Training of Neural Network	19
3.2.1	H-Divergence	19
3.2.2	Model	20
3.3	Automatic Domain Alignment Layers	23
3.3.1	Source and target predictor	23
3.3.2	Domain Alignment Layers	24
3.3.3	Training	24
3.4	Adversarial Discriminative Domain Adaptation	26
3.4.1	Source and target mapping	26
3.4.2	Adversarial losses	27
3.4.3	Adversarial discriminative domain adaptation	27
4	Colorization methods	29
4.1	Surface Normal Colorization	30
4.2	Surface Normal++ Colorization	32
5	Deep cue integration	33
5.1	Fusion Types	33
5.2	High-level Integration	35

6 Experiments	36
6.1 Datasets	36
6.1.1 RGB-D Object Dataset	36
6.1.2 Autonomous Robot Indoor Dataset	37
6.1.3 Web Object Dataset	38
6.1.4 Big Berkeley Instance Recognition Dataset	39
6.1.5 Active Vision Dataset	40
6.2 Experimental Setup	41
6.3 Results and Discussion	46
6.3.1 RGB Only	46
6.3.2 Depth Only	49
6.3.3 RGB-D	57
7 Conclusions	59
Bibliography	61

List of Figures

1.1	Example of a service robot [1, 2].	8
1.2	A coffee mug can have different characteristics [3, 4].	9
1.3	The same object with different illuminations [5].	9
1.4	An example of RGB-D camera with sensors description [11]. . . .	10
3.1	The Deep Adaptation Network architecture for learning transferable features [6].	16
3.2	The Domain Adversarial Training of Neural Network architecture with the deep <i>features extraction</i> (in green), the deep <i>label predictor</i> (in blue) and the <i>gradient reversal layer</i> (in pink) [8].	19
3.3	Automatic Domain Alignment Layers applied on Alexnet. [7] . . .	23
3.4	Phases of the Adversarial Discriminative Domain Adaptation method. Dotted lines denote pinned network parameters [9].	26
4.1	Example of RGB (left) and depth (right) data from the same object [12].	29
4.2	The normal of a point on a surface [34].	30
4.3	Surface normal geometric construction ($n = a \times b$) [35].	30
4.4	RGB image (left), original depth image (center), image of the depth mapped with Surface Normal (right).	31
4.5	Illustration of the pre-processing method steps. (a) Original depth image, (b) after applying recursive median filter, (c) after applying surface normal colorization, (d) application of surface normal colorization after the using of a bilateral filter, (e) after the application of an unsharp mask filter [36].	32
5.1	The low-level integration technique [38].	34
5.2	The middle-level integration technique [38].	34
5.3	The high-level integration technique [38].	34
6.1	Objects from the RGB-D Object Dataset [12].	36

6.2	Detection results in two multi-object scenes [12].	37
6.3	The mobile robot that acquire data from a scene with everyday objects [13].	38
6.4	RGBDToolkit [14]	39
6.5	Side view of the five cameras and the five depth sensors mounted ready to start a data acquisition process [14].	39
6.6	The glass turntable of the data-collection system [14].	40
6.7	Example of scenes of the dataset [15]	41
6.8	Map of the movements of the camera in a scene with all the subset of point chosen for the data recording [15].	41
6.9	From left to right: RGB image, depth image, segmentation mask provided by Big Berkeley Instance Recognition Dataset.	42
6.10	Example of a cropped RGB image on left, example of a cropped depth image on right.	43
6.11	Quantity of images for each distance range in Autonomous Robot Indoor Dataset dataset.	49
6.12	Quantity of images for each object class in ranges: 0-1200/0-1400.	50
6.13	Quantity of images for each object class in ranges: 0-1600/0-1800.	51
6.14	Quantity of images for each object class in ranges: 0-2000/0-2200.	52
6.15	Quantity of images for each object class in ranges: 0-2400/0-max distance.	53
6.16	Accuracies obtained testing the network trained on RGB-D Object Dataset on the eight subsets of Autonomous Robot Indoor Dataset.	54
6.17	Example of the not perfect overlap between RGB and depth images in Big Berkeley Instance Recognition Dataset.	58

List of Tables

6.1	ROD/WOD → ARID RGB experiments of group (1)	47
6.2	ROD/WOD → ARID RGB experiments of group (2)	47
6.3	BigBIRD → Active Vision RGB experiments of group (1)	48
6.4	BigBIRD → Active Vision RGB experiments of group (2)	48
6.5	ROD → ARID Depth experiments of group (1)	55
6.6	ROD → ARID Depth experiments of group (2)	55
6.7	BigBIRD → Active Vision Depth experiments of group (1)	56
6.8	BigBIRD → Active Vision Depth experiments of group (2)	56
6.9	ROD → ARID RGB-D experiments of group (1)	57
7.1	Best improvement reached through domain adaptation algorithms for each RGB only, depth only and RGB-D task. In brackets also the accuracy reached is shown.	60
7.2	Best algorithm for each RGB only and depth only experiment. . .	60
7.3	Group of experiments with which the best result is obtained. . . .	60
7.4	Accuracies obtained using as test set progressive subsets (in terms of camera - object distance) of Autonomous Robot Indoor Dataset. .	60

List of acronyms

AI Artificial Intelligence

CV Computer Vision

SR Service Robots

DA Domain Adaptation

DAN Deep Adaptation Network

DANN Domain Adversarial Training of Neural Network

AutoDIAL Automatic Domain Alignment Layers

ADDA Adversarial Discriminative Domain Adaptation

ROD RGB-D Object Dataset

WOD Web Object Dataset

ARID Autonomous Robot Indoor Dataset

BigBIRD Big Berkeley Instance Recognition Dataset

H Hilbert Space

RKHS Reproducing Kernel Hilbert Space

MMD Maximum Mean Discrepancy

MK-MMD Multiple Kernel Maximum Mean Discrepancy

GRL Gradient Reversal Layer

CNN Convolutional Neural Network

QP Quadratic Program

DA-layers Domain Alignment Layers

GAN Generative Adversarial Network

SVM Support Vector Machine

NLP Natural Language Processing

SGD Stochastic Gradient Descent

Chapter 1

Introduction

Object recognition is the ability to re-identify a previously seen object or acknowledge it as belonging to a specific class of objects. In humans, recognition is performed with little effort even when the object appears in different shapes, colors, texture, or if it is partially occluded or seen from a different perspective. One of the open problems in Artificial Intelligence (AI), and Computer Vision (CV) in particular, is to emulate this skill in artificial systems. Object recognition is particularly important for Service Robots (SR) that aim at assisting humans in their own environments (e.g. houses and offices). These actions are typically performed in indoor environments as houses or offices. A robot should be able to move in these places full of objects and use them in the proper way to perform its task (see Figure 1.1).



Figure 1.1: Example of a service robot [1, 2].

Object recognition presents several challenges. Two object instances of the same category can appear very different from each other (see Figure 1.2). Formally, this concept is referred to as intra-class variability, i.e. diversity between samples of the same class. In addition, pictures of the same object can have a

different appearance due to environmental conditions such as changes in lighting and viewpoint, variations in scale and background, clutter, and occlusion (see Figure 1.3). Usually, in machine learning, the training set is acquired in laboratory under ideal conditions (static lighting, fixed background, same viewpoint, etc.). This gives rise to a set of too "clean" images. As a result, when the classifier is applied to a realistic context, all the problems listed above arise. We can therefore consider training and test sets as if they were two different domains. The discipline that attempts to reduce the discrepancy between images belonging to different domains is Domain Adaptation (DA). Typically, the adaptation takes place between a pair of different domains: a *source domain* and a *target domain*. This project focuses on *unsupervised domain adaptation* so the source domain, with which the classifier is trained, is composed by image-labeled data, while, the target domain, on which the classifier is tested, is made by unlabeled images. The unlabeled condition imposed on the target domain is fundamental because the need to apply domain adaptation algorithms rises mostly from the necessity to deal with unknown environments. This project regards the evaluation of standard state-of-the-art DA algorithms in an unsupervised way on datasets created



Figure 1.2: A coffee mug can have different characteristics [3, 4].



Figure 1.3: The same object with different illuminations [5].

different appearance due to environmental conditions such as changes in lighting and viewpoint, variations in scale and background, clutter, and occlusion (see Figure 1.3). Usually, in machine learning, the training set is acquired in laboratory under ideal conditions (static lighting, fixed background, same viewpoint, etc.). This gives rise to a set of too "clean" images. As a result, when the classifier is applied to a realistic context, all the problems listed above arise. We can therefore consider training and test sets as if they were two different domains. The discipline that attempts to reduce the discrepancy between images belonging to different domains is Domain Adaptation (DA). Typically, the adaptation takes place between a pair of different domains: a *source domain* and a *target domain*. This project focuses on *unsupervised domain adaptation* so the source domain, with which the classifier is trained, is composed by image-labeled data, while, the target domain, on which the classifier is tested, is made by unlabeled images. The unlabeled condition imposed on the target domain is fundamental because the need to apply domain adaptation algorithms rises mostly from the necessity to deal with unknown environments. This project regards the evaluation of standard state-of-the-art DA algorithms in an unsupervised way on datasets created

ad-hoc for service robot functionalities. DA algorithms used can be divided in two groups:

- *Probability distributions alignment* performed minimizing the distance between source and target distributions. This optimization problem can be solved by adding a proper term to the loss function as in Deep Adaptation Network (DAN) [6], or using some additional layers that perform a cross-domain adaptation as in Automatic Domain Alignment Layers (AutoDIAL) [7].
- *Adversarial alignment* accomplished training adversarial networks. The two networks, one trained for standard object classification and the other trained for domain recognition, can be optimized in the same process as in Domain Adversarial Training of Neural Network (DANN) [8] or in more phases as in Adversarial Discriminative Domain Adaptation (ADDA) [9].

With the advent of Microsoft Kinect [10] (see Figure 1.4), RGB-D camera have become increasingly popular. These sensors use range imaging technology

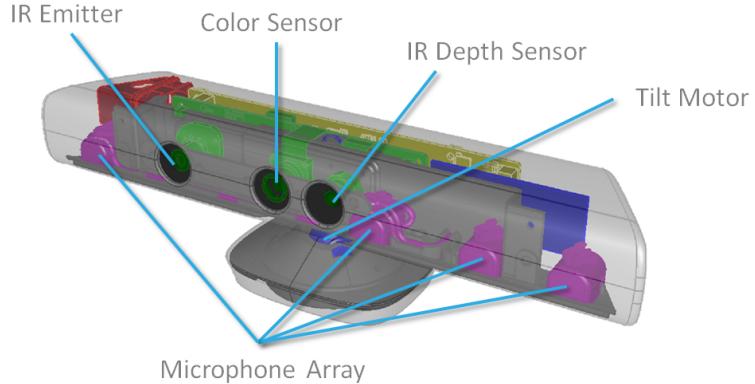


Figure 1.4: An example of RGB-D camera with sensors description [11].

to provide, in addition to the RGB image, geometric information in form of depth images. Depth images can be very valuable in object recognition since depth information is invariant with respect to some environmental conditions, such as changes in lighting and color. Particularly in robotics, where agents move and operate in the environment, depth is a visual information as important as RGB.

The goal of this project is to define a benchmark in the DA scenario that is relevant especially for the robot community. The purpose of this research is to investigate the effectiveness of DA algorithms in function of the particular input modality chosen. The adaptation is performed with respect to RGB, depth and RGB-D input data. It is important to point out that, to exploit the

same networks designed for RGB also with depth informations, it is necessary to simulate a sort of colorization for them. In the case of RGB-D input modality, instead, a completely different approach, described in the next chapters, is applied. The datasets chosen for the experiments are compatible with several robotic environments being composed by objects of every day life with which a robot could be faced with. They are: RGB-D Object Dataset (ROD) [12] Web Object Dataset (WOD) [13], Autonomous Robot Indoor Dataset (ARID) [13], Big Berkeley Instance Recognition Dataset (BigBIRD) [14] and Active Vision Dataset [15]. Analyzing the results, the intention is to identify open issues, specific of robot vision, that current DA approaches fail to solve.

The lower improvements obtained applying DA algorithms on depth images with respect to those obtained on RGB ones leads to the natural conclusion that the algorithms used for this benchmark are more effective on RGB data. Moreover, the use of RGB and depth informations together does not produce the expected results; actually, in some cases, this combination leads to an even worse accuracy with respect to RGB only experiments. So, the optimal method to use depth informations alone, but also the most effective way to exploit depth together with RGB data are still challenges in DA.

The document is composed by a theoretical part followed by an analytical part. In the first one all the algorithms and methods employed for the experiments are described, in the second one the results, with related observations, are shown. In detail, the organization is as follow: in the next chapter some related works are discussed; chapters 3, 4 and 5 show respectively how each DA algorithm works, the colorization methods used and the approach adopted for RGB-D experiments; in chapter 6 there is a presentation of the datasets used together with the commented results obtained from the experiments and, finally, in chapter 7 there is a discussion about the findings of the experiments and the conclusions we have drawn.

Chapter 2

Related works

During the years, a considerable part of the literature, in the context of DA, has focused on the use of linear classifiers [16, 17, 18]. In the last few years, instead, research started to moving towards neural network non-linear representations. For the unsupervised DA (the focus of this work) all the approaches can be divided into two groups. The first one refers to the *instance re-weighting*. It acts by assigning to each source sample a different weight in function of its similarity with the target domain. Once that the re-weighting is performed, the algorithm trains a classifier, using the re-weighted source domain, that should work well for target data. An example is [19] that introduces an unsupervised domain adaptation algorithm for pedestrian detection that exploits deep autoencoders to weight source data. The second group of unsupervised DA is composed by all the algorithms that accomplish *features alignment*. This method acts reducing the distance between the source and target domain feature distributions. It can be performed learning *shallow representation models* [20, 21] or *deep features* [22, 23]. Due to the task-specific variability of the shallow representations, the deep approach is preferred. This last, in turn, can be accomplished in two ways. The first consists in the alignment of source and target distributions either with the insertion of Domain Alignment Layers (DA-layers) in the deep network [7] or adding a proper term to the *loss function* (a common choice is to use the Maximum Mean Discrepancy (MMD) as measure of the dissimilarity between source and target distributions as in [24, 25, 26]). The second way to perform features alignment is maximizing a *domain confusion loss* in such a way that the classifier is not able to distinguish between source and target domains [22, 27].

Object recognition, together with DA, is particularly relevant in robotic applications. One of the main problem in this field is the lack of proper datasets to train robust classifiers that work well in realistic contexts (such as robotic

environments). The idea behind some previous works is the use of data from the World Wide Web as source domain properly adapted to robotic purposes. In particular, [28] and [29] use objects from Google’s 3D Warehouse [30] as source domain. To overcome the differences between web data and real data, a small set of labeled point clouds recorded by mobile robots in realistic environments is added to source domain and domain adaptation algorithms are applied.

Most of the literature both in computer and robotic vision field is focused on RGB images, several domain adaptation benchmarks have been defined using this type of information. Some examples can be found in works in which new algorithms are proposed: in these types of publications a benchmark is necessary to verify the performance of the new domain adaptation algorithm designed (some examples are [6, 8, 7, 9, 31, 25]). Depth, on the other hand, is a kind of information rarely considered in the creation of domain adaptation benchmarks despite this data is particularly relevant in robotics where the aim is to model agents that should be able to move and act in the environment. Even more rare is the use of the RGB-D informations in these scenarios. Both these modalities have been investigated by really few works, as in [32]. For sure, one of the reasons of this gap is in the lack of proper datasets in which RGB and depth informations can be used together.

The contribution of this work is to define a benchmark for the robot vision community using RGB, depth and RGB-D modalities as input data using some state-of-the-art domain adaptation algorithms.

Chapter 3

Domain Adaptation Algorithms

In this chapter we revise into details the domain adaptation algorithms used in this thesis. We explicitly focused on the unsupervised domain adaptation scenario, characterized by a *source* domain $D_s = \{(x_i^s, y_i^s)\}_{i=1}^{n_s}$ with n_s labeled examples, and a *target* domain $D_t = \{(x_j^t)\}_{j=1}^{n_t}$ with n_t unlabeled examples.

3.1 Deep Adaptation Network

Recent studies demonstrate that the features transferability decreases significantly in the higher layers of a deep network with an increment of domain discrepancy. Practically, the features computed in the higher layers depend a lot on the specific dataset and task. The goal of DAN is to increase the transferability in these task-specific layers by generalizing deep convolutional neural networks to the domain adaptation scenario. To reach this goal, the hidden representations of all these task-specific layers are embedded to a Reproducing Kernel Hilbert Space (RKHS) where the mean embeddings of various domain distributions can be matched. Because mean embedding matching is affected by the kernel choices, an optimal multi-kernel selection procedure is designed to reduce further the domain discrepancy.

3.1.1 Multiple Kernel Maximum Mean Discrepancy

The Hilbert Space (H) is the space of the features in which the data are mapped to make them linearly separable in the classification. The RKHS is composed by the inner products of the elements of H and it is defined by a type of kernel. The MMD expresses the "distance" between the images of the source domain and the images of the target domain. This quantity is computed in a

RKHS between the mean embedding (μ_p) of p (probability distribution of the source domain) and the mean embedding (μ_q) of q (probability distribution of the target domain). So the RKHS distance between two probability distributions will be the difference of the square root between two elements of RKHS (two inner products):

$$d^2(p, q) \triangleq \|\mathbf{E}_p[\phi(\mathbf{x}^s)] - \mathbf{E}_q[\phi(\mathbf{x}^t)]\|_{H_k}^2 \quad (3.1)$$

where

$$\mathbf{E}_p[f(\mathbf{x})] = \langle f(\mathbf{x}), \mu_k(p) \rangle \quad \forall f \in H_k$$

$$\mathbf{E}_q[f(\mathbf{x})] = \langle f(\mathbf{x}), \mu_k(q) \rangle \quad \forall f \in H_k$$

Notice that $p = q$ iff $d_k^2(p, q) = 0$. The method is based on the multiple kernel variant of Multiple Kernel Maximum Mean Discrepancy (MK-MMD): the characteristic kernel used to compute the inner product associated to the feature map ϕ ($k(\mathbf{x}^s, \mathbf{x}^t) = \langle \phi(\mathbf{x}^s), \phi(\mathbf{x}^t) \rangle$) is not fixed a priori but it is given as the convex combination of m positive semi-definite kernels $\{k_u\}$:

$$K \triangleq \left\{ k = \sum_{u=1}^m \beta_u k_u : \sum_{u=1}^m \beta_u = 1, \beta_u \geq 0, \forall u \right\}$$

where the positivity constraints on coefficients $\{\beta\}$ are used to ensure that the derived multi-kernel k is characteristic. The kernel selected to compute the mean embeddings of p and q is important to guarantee the test power and low test error.

3.1.2 Model

The starting point of this method is a deep Convolutional Neural Network (CNN) but, just adapting CNN via fine-tuning to the target domain (that has no labels) is difficult and could lead to over-fitting. Therefore the idea is to model a deep adaptation network (DAN) that can take advantage from both source domain (with labeled data) and target domain (with unlabeled data) (see Figure 3.1). In a classical CNN each f_c layer ℓ acquires a nonlinear mapping $\mathbf{h}_i^\ell = f^\ell(\mathbf{W}^\ell \mathbf{h}_i^{\ell-1} + \mathbf{b}^\ell)$ where \mathbf{h}_i^ℓ is the ℓ th layer hidden representation of point \mathbf{x}_i , \mathbf{W}^ℓ and \mathbf{b}^ℓ are the weights and bias of the ℓ th layer, and f^ℓ is the activation function, taking $f^\ell(\mathbf{x}) = \max(\mathbf{0}, \mathbf{x})$ (rectifiers units) for hidden layers or $f^\ell(\mathbf{x}) =$

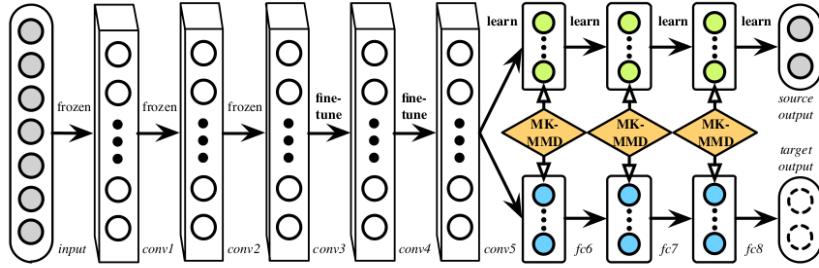


Figure 3.1: The Deep Adaptation Network architecture for learning transferable features [6].

$e^{\mathbf{x}} / \sum_{j=1}^{|\mathbf{x}|} e^{x_j}$ (softmax units) for the output layer. Since $\Theta = \{\mathbf{W}^\ell, \mathbf{b}^\ell\}_{\ell=1}^l$ are the CNN parameters, the empirical risk of CNN will be:

$$\min_{\Theta} \frac{1}{n_a} \sum_{i=1}^{n_a} J(\theta(\mathbf{x}_i^a), y_i^a)$$

in which J is the cross-entropy loss function, and $\theta(\mathbf{x}_i^a)$ represents the condition probability that the CNN will compute a correct prediction y_i^a for the example \mathbf{x}_i^a .

The novelty of DAN is in the contribution of a *MK-MMD-based multi-layer* adaptation regularizer to the CNN risk:

$$\min_{\Theta} \frac{1}{n_a} \sum_{i=1}^{n_a} J(\theta(\mathbf{x}_i^a), y_i^a) + \lambda \sum_{\ell=l_1}^{l_2} d_k^2(D_s^\ell, D_t^\ell) \quad (3.2)$$

where $\lambda > 0$ is a parameter of penalty, l_1 and l_2 are the layers between which the adaptation regularizer acts, $D_*^\ell = \{\mathbf{h}_i^{*\ell}\}$ is the hidden representation of the ℓ th layer for the source and target examples, and $d_k^2(D_s^\ell, D_t^\ell)$ is the MK-MMD among the source and target examples computed in the ℓ th layer representation. The MK-MMD term will lead source and target distributions to become similar under the hidden representations of fully connected layers.

Two significant points that differentiate DAN from previous works are:

- *multi-layer adaptation.* It is not enough to adapt a single layer to reduce sufficiently the dataset bias between source and target domain because there is more than one layer that is not transferable. Furthermore, adapting the representation layers and the classification layer together, we could link the domain discrepancy that characterizes the marginal distribution and the conditional distribution, a critical point for domain adaptation.

- *multi-kernel adaptation.* The choice of the kernel is important: different kernels embed probability distributions in different RKHSs in which different orders of sufficient statistics can be accentuated.

3.1.3 Algorithm

Learning Θ . The computation of best parameters Θ in (3.2) is executed in this way:

- For **MK-MMD** (3.1) the unbiased estimate is used because it can be calculated with linear complexity. In particular $d_k^2(p, q) = \frac{2}{n_s} \sum_{i=1}^{n_s/2} g_k(\mathbf{z}_i)$, where $\mathbf{z}_i \triangleq (\mathbf{x}_{2i-1}^s, \mathbf{x}_{2i}^s, \mathbf{x}_{2i-1}^t, \mathbf{x}_{2i}^t)$ and $g_k(\mathbf{z}_i) \triangleq k(\mathbf{x}_{2i-1}^s, \mathbf{x}_{2i}^s) + k(\mathbf{x}_{2i-1}^t, \mathbf{x}_{2i}^t) - k(\mathbf{x}_{2i-1}^s, \mathbf{x}_{2i}^t) - k(\mathbf{x}_{2i}^s, \mathbf{x}_{2i-1}^t)$. This computation is done with cost $O(n)$. To optimize this part of DAN risk we just need to compute the gradients $\frac{\partial g_k(\mathbf{z}_i^\ell)}{\partial \Theta^\ell}$ for the quad-tuple $\mathbf{z}_i^\ell = (\mathbf{h}_{2i-1}^{s\ell}, \mathbf{h}_{2i}^{s\ell}, \mathbf{h}_{2i-1}^{t\ell}, \mathbf{h}_{2i}^{t\ell})$ of the ℓ th layer hidden representation.
- Similarly to the gradient of MK-MMD, we calculate the corresponding gradient of **CNN risk** $\frac{\partial J(\mathbf{z}_i)}{\partial \Theta^\ell}$, where $J(\mathbf{z}_i) = \sum_{i'} J(\Theta(\mathbf{x}_{i'}^a), y_{i'}^a)$ and $\{(\mathbf{x}_{i'}^a, y_{i'}^a)\}$ correspond to the labeled examples in quad-tuple \mathbf{z}_i .

In summary, to perform mini-batch updates, we calculate the gradient of the objective function (3.2) with respect to the ℓ th layer parameter Θ^ℓ in this way:

$$\nabla_{\Theta^\ell} = \frac{\partial J(\mathbf{z}_i)}{\partial \Theta^\ell} + \lambda \frac{\partial g_k(\mathbf{z}_i^\ell)}{\partial \Theta^\ell} \quad (3.3)$$

Since kernel k is the linear combination of m Gaussian kernels $\{k_u(\mathbf{x}_i, \mathbf{x}_j) = \exp^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\gamma_u}\}$, the gradient $\frac{\partial g_k(\mathbf{z}_i^\ell)}{\partial \Theta^\ell}$ can be quickly solved with the chain rule.

Learning β . For the selection of the optimal kernel parameter β in MK-MMD it should be optimized this term:

$$\max_{k \in K} d_k^2(D_s^\ell, D_t^\ell) \sigma_k^{-2} \quad (3.4)$$

where $\sigma_k^2 = \mathbf{E}_\mathbf{z} g_k^2(\mathbf{z}) - [\mathbf{E}_\mathbf{z} g_k(\mathbf{z})]^2$ is the estimation variance.

Being $\mathbf{d} = (d_1, d_2, \dots, d_m)^T$, each d_u is MMD through kernel k_u . Covariance $\mathbf{Q} = cov(g_k) \in \Re^{m \times m}$ can be calculated in $O(m^2 n)$ cost, i.e. $\mathbf{Q}_{uu'} = \frac{4}{n_s} \sum_{i=1}^{n_s/4} g_{k_u}^\nabla(\bar{\mathbf{z}}_i) g_{k_{u'}}^\nabla(\bar{\mathbf{z}}_i)$ where $\bar{\mathbf{z}}_i \triangleq (\mathbf{z}_{2i-1}, \mathbf{z}_{2i})$ and $g_{k_u}^\nabla(\bar{\mathbf{z}}_i) \triangleq g_{k_u}(\mathbf{z}_{2i-1}) - g_{k_u}(\mathbf{z}_{2i})$. Consequently (3.4) becomes this simpler Quadratic Program (QP),

$$\min_{\mathbf{d}^T \boldsymbol{\beta} = 1, \boldsymbol{\beta} \geq 0} \boldsymbol{\beta}^T (\mathbf{Q} + \varepsilon \mathbf{I}) \boldsymbol{\beta}, \quad (3.5)$$

where $\varepsilon = 10^{-3}$ acts as regularizer to transform the problem in a well-defined way. We can observe that the DAN risk (3.2) is substantially a minimax problem:

$$\min_{\Theta} \max_K d_k^2(D_s^\ell, D_t^\ell) \sigma_k^{-2} \quad (3.6)$$

These two operations act to reach an efficient adaptation for the domain discrepancy, with the purpose of consolidating the transferability of DAN features.

3.2 Domain Adversarial Training of Neural Network

The central point of this method is the implementation of a mapping between source and target domain such that the classifier trained on the source works well also if it is tested on the target. The algorithm focuses on learning features that are both discriminative and domain-invariant. With this in mind two classifiers are optimized:

- the *label classifier* that is the standard class labels predictor.
- the *domain classifier* whose intention is to distinguish between the source and target domains.

The optimal feature mapping is found by *minimizing* the loss function of the label classifier and by *maximizing* the loss function of the domain classifier. The *maximization* then acts *adversarially* to the domain classifier. The innovation of DANN algorithm is in the adding of a Gradient Reversal Layer (GRL) in the classical CNN (Figure 3.2).

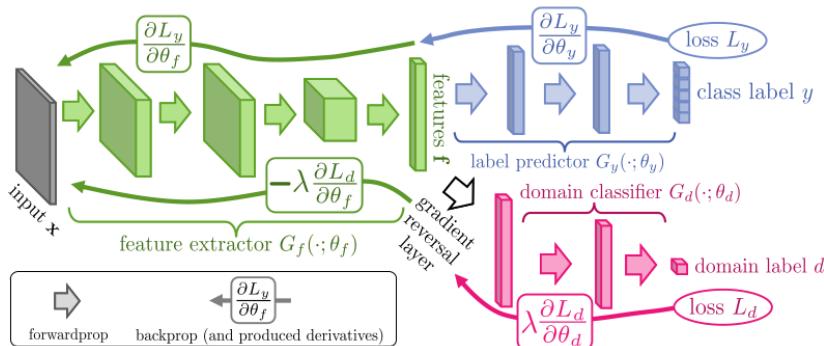


Figure 3.2: The Domain Adversarial Training of Neural Network architecture with the deep *features extraction* (in green), the deep *label predictor* (in blue) and the *gradient reversal layer* (in pink) [8].

3.2.1 H-Divergence

Definition 1 (Ben-David et al., 2006, 2010; Kifer et al., 2004) Given two domain distributions D_S^X and D_T^X over X , and a hypothesis class H (that we assume to be a set of binary classifiers $\eta : X \rightarrow \{0, 1\}$), the H -divergence between D_S^X and D_T^X is

$$d_H(D_S^X, D_T^X) = 2 \sup_{\eta \in H} \left| \Pr_{x \sim D_S^X} [\eta(\mathbf{x}) = 1] - \Pr_{x \sim D_T^X} [\eta(\mathbf{x}) = 1] \right|.$$

This quantity is the ability of the hypothesis class H to recognize between examples produced by D_S^X from examples produced by D_T^X . An important property is that, if the hypothesis class H is symmetric, the *empirical H-divergence* among two examples $S \sim (D_S^X)^n$ and $T \sim (D_T^X)^{n'}$ can be computed in this way:

$$\hat{d}_H(S, T) = 2 \left(1 - \min_{\eta \in H} \left[\frac{1}{n} \sum_{i=1}^n I[\eta(\mathbf{x}_i) = 0] + \frac{1}{n'} \sum_{i=n+1}^{n'} I[\eta(\mathbf{x}_i) = 1] \right] \right) \quad (3.7)$$

in which $I[a]$ is 1 if a is true, and 0 if a is false.

3.2.2 Model

For simplicity we analyze the method in a neural network composed by only one hidden layer which learns a function $G_f : X \rightarrow \mathbb{R}^D$ (with $X = \mathbb{R}^m$) that maps each example in a D -dimensional representation:

$$G_f(\mathbf{x}; \mathbf{W}, \mathbf{b}) = \text{sigm}(\mathbf{W}\mathbf{x} + \mathbf{b}) \quad (3.8)$$

where $\text{sigm}(\mathbf{a}) = \left[\frac{1}{1+\exp(-a_i)} \right]_{i=1}^{|\mathbf{a}|}$. In the same way, the prediction layer learns a function $G_y : \mathbb{R}^D \rightarrow [0, 1]^L$:

$$G_y(G_f(\mathbf{x}; \mathbf{W}, \mathbf{b}); \mathbf{V}, \mathbf{c}) = \text{softmax}(\mathbf{V}G_f(\mathbf{x}; \mathbf{W}, \mathbf{b}) + \mathbf{c})$$

$$\text{softmax}(\mathbf{a}) = \left[\frac{\exp(a_i)}{\sum_{j=1}^{|\mathbf{a}|} \exp(a_j)} \right]_{i=1}^{|\mathbf{a}|}.$$

In these formulations $(\mathbf{W}, \mathbf{b}) \in \mathbb{R}^{D \times m} \times \mathbb{R}^D$ are the parameters of the hidden layer G_f and $(\mathbf{V}, \mathbf{c}) \in \mathbb{R}^{L \times D} \times \mathbb{R}^L$ (with $L = |Y|$) are the parameters of the prediction layer G_y . Given a source sample (\mathbf{x}_i, y_i) the typical classification loss is:

$$\mathcal{L}_y(G_y(G_f(\mathbf{x}; \mathbf{W}, \mathbf{b}); \mathbf{V}, \mathbf{c}), y_i) = \log \frac{1}{G_y(G_f(\mathbf{x}; \mathbf{W}, \mathbf{b}); \mathbf{V}, \mathbf{c})_{y_i}}$$

So during training the optimization problem will be:

$$\min_{\mathbf{W}, \mathbf{b}, \mathbf{V}, \mathbf{c}} \left[\frac{1}{n} \sum_{i=1}^n \mathcal{L}_y^i(G_y(G_f(\mathbf{x}; \mathbf{W}, \mathbf{b}); \mathbf{V}, \mathbf{c}), y_i) + \lambda \cdot R(\mathbf{W}, \mathbf{b}) \right] \quad (3.9)$$

in which n is the number of source examples and $R(\mathbf{W}, \mathbf{b})$ is a regularizer.

The central point of the method is in the addition of a term using the Definition 1 of *H-divergence*, this term is called *domain regularizer*. Consider the output of the hidden layer $G_f(\cdot)$ (3.8) for source example:

$$S(G_f) = \{G_f(\mathbf{x}) | \mathbf{x} \in S\}$$

and, for target example:

$$T(G_f) = \{G_f(\mathbf{x}) | \mathbf{x} \in T\}.$$

Using the empirical *H-divergence* (3.7) (H is a symmetric hypothesis class) between $S(G_f)$ and $T(G_f)$ we have:

$$\hat{d}_H(S(G_f), T(G_f)) = \quad (3.10)$$

$$2 \left(1 - \min_{\eta \in H} \left[\frac{1}{n} \sum_{i=1}^n I[\eta(G_f(\mathbf{x}_i)) = 0] + \frac{1}{n'} \sum_{i=n+1}^N I[\eta(G_f(\mathbf{x}_i)) = 1] \right] \right) \quad (3.11)$$

We estimate the min part of (3.11) with a *domain classifier layer* G_d that learns a function (*logic regressor*) $G_d : \mathbb{R}^D \rightarrow [0, 1]$ with parameters $(\mathbf{u}, z) \in \mathbb{R}^D \times \mathbb{R}$. This classifier gives the probability that an example is from source dataset or target dataset.

$$G_d(G_f(\mathbf{x}); \mathbf{u}, z) = \text{sigm}(\mathbf{u}^T G_f(\mathbf{x}) + z)$$

Its loss is:

$$\mathcal{L}_d(G_d(G_f(\mathbf{x}_i)), d_i) = d_i \log \frac{1}{G_d(G_f(\mathbf{x}_i))} + (1 - d_i) \log \frac{1}{1 - G_d(G_f(\mathbf{x}_i))}$$

in which d_i is the binary *domain label* for the i -th example ($d_i = 0$ if \mathbf{x}_i belong to the source domain, $d_i = 1$ if \mathbf{x}_i belong to the target domain).

We add a domain adaptation term in the equation (3.9) as regularizer term:

$$R(\mathbf{W}, \mathbf{b}) = \max_{\mathbf{u}, z} \left[-\frac{1}{n} \sum_{i=1}^n \mathcal{L}_d^i(\mathbf{W}, \mathbf{b}, \mathbf{u}, z) - \frac{1}{n'} \sum_{i=n+1}^N \mathcal{L}_d^i(\mathbf{W}, \mathbf{b}, \mathbf{u}, z) \right]$$

in which $\mathcal{L}_d^i(\mathbf{W}, \mathbf{b}, \mathbf{u}, z) = \mathcal{L}_d(G_d(G_f(\mathbf{x}_i; \mathbf{W}, \mathbf{b}); \mathbf{u}, z), d_i)$. Since $2(1 - R(\mathbf{W}, \mathbf{b}))$ is a surrogate of $\hat{d}_H(S(G_f), T(G_f))$, this term is the approximation of the H -

divergence equation (3.11).

In summary, we can write the final optimization problem of (3.9) in this way:

$$E(\mathbf{W}, \mathbf{V}, \mathbf{b}, \mathbf{c}, \mathbf{u}, z) = \frac{1}{n} \sum_{i=1}^n \mathcal{L}_y^i(\mathbf{W}, \mathbf{b}, \mathbf{V}, \mathbf{c}) - \lambda \left(\frac{1}{n} \sum_{i=1}^n \mathcal{L}_d^i(\mathbf{W}, \mathbf{b}, \mathbf{u}, z) + \frac{1}{n'} \sum_{i=n+1}^N \mathcal{L}_d^i(\mathbf{W}, \mathbf{b}, \mathbf{u}, z) \right)$$

and the optimal parameters are found solving:

$$(\hat{\mathbf{W}}, \hat{\mathbf{V}}, \hat{\mathbf{b}}, \hat{\mathbf{c}}) = \arg \min_{\mathbf{W}, \mathbf{V}, \mathbf{b}, \mathbf{c}} E(\mathbf{W}, \mathbf{V}, \mathbf{b}, \mathbf{c}, \hat{\mathbf{u}}, \hat{z})$$

$$(\hat{\mathbf{u}}, \hat{z}) = \arg \max_{\mathbf{u}, z} E(\hat{\mathbf{W}}, \hat{\mathbf{V}}, \hat{\mathbf{b}}, \hat{\mathbf{c}}, \mathbf{u}, z)$$

3.3 Automatic Domain Alignment Layers

The technique used in this method aims at aligning the features of the source and target domains to a canonical one with the addition of *DA-layers* in the standard networks. The novelty of this approach is in the self-tuning of the degree of feature alignment in function of the different level of the deep network in which the new layers are embedded. This is the real innovation of this method: the dynamism of domain alignment parameters.

3.3.1 Source and target predictor

The typical assumption that is done by the majority of domain adaptation methods is that the domain alignment between source and target can be done using the same predictor. In this work an impossibility theorem is considered, that states the inability of a learner to perform the domain adaptation among distributions characterized by a *covariance shift* without additional hypothesis of dependence among them. The AutoDIAL method implements two different predictors for the source and the target domain starting from two deep neural networks with the same structure and the same weights. Both networks have also the same number of DA-layers in the same positions (Figure 3.3).

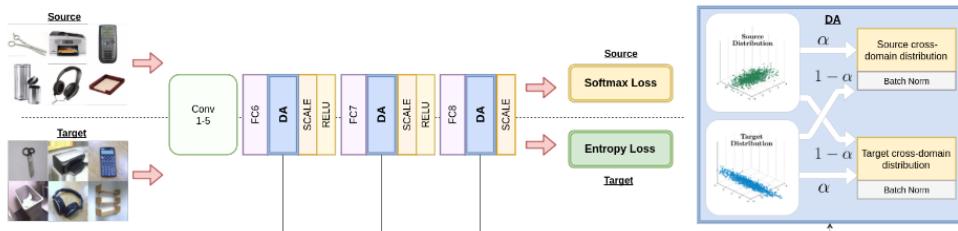


Figure 3.3: Automatic Domain Alignment Layers applied on Alexnet. [7]

Since the source and target distributions are different, the transformations implemented by the adaptation layers to align the two distributions with a reference one are different. Basically the DA-layers are Batch Normalization layers but, instead of using the standard implementation with the first and second-order moment computed on the input distribution, a cross-domain bias is used: the second-order moment is contaminated with the other domain.

3.3.2 Domain Alignment Layers

Let q^s be the probability distribution of source input data x_s and q^t the probability distribution of target input data x_t , let us assume $q_\alpha^{st} = \alpha q^s + (1 - \alpha) q^t$ and $q_\alpha^{ts} = \alpha q^t + (1 - \alpha) q^s$ be cross-domain distributions with the mixed factor $\alpha \in [0.5, 1]$. The outcome of the DA-layers will be, for the source network

$$DA(x_s; \alpha) = \frac{x_s - \mu_{st,\alpha}}{\sqrt{\epsilon + \sigma_{st,\alpha}^2}}$$

and, for the target network

$$DA(x_t; \alpha) = \frac{x_t - \mu_{ts,\alpha}}{\sqrt{\epsilon + \sigma_{ts,\alpha}^2}}$$

with $\epsilon > 0$ a small number useful in case of variance equal to zero, $\mu_{st,\alpha} = E_{x \sim q_\alpha^{st}}[x]$ the mean and $\sigma_{st,\alpha}^2 = Var_{x \sim q_\alpha^{st}}[x]$ the variance of $x \sim q_\alpha^{st}$. Similarly, $\mu_{ts,\alpha} = E_{x \sim q_\alpha^{ts}}[x]$ the mean and $\sigma_{ts,\alpha}^2 = Var_{x \sim q_\alpha^{ts}}[x]$ the variance of $x \sim q_\alpha^{ts}$. It can be noticed that by using $\alpha = 1$ there is an independent alignment of source and target domains: the DA-layers calculate two completely different functions for the source and the target predictors. Using instead $\alpha = 0.5$ the new layers give rise to the same function for the predictors of the two domains: they are transformed likewise ($q_{0.5}^{st} = q_{0.5}^{ts}$) and there is no domain alignment. Thus the choice of the mixing factor is crucial, it is not decided a priori but it is learned in the training process.

3.3.3 Training

The common weights of source and target predictors and the mixing factor related to the DA-layers are computed during the training phase by employing a source labeled dataset and a target unlabeled dataset. We have the following a posterior distribution of the set of parameters θ (weights and mixing factor):

$$\pi(\theta|S, T) \propto \pi(y_s|x_s, T, \theta)\pi(\theta|T, x_s) \quad (3.12)$$

with $x_s = \{x_1^s, \dots, x_n^s\}$ and $y_s = \{y_1^s, \dots, y_n^s\}$ corresponding to data and labels of the source domain. This quantity is maximized to find the best value for θ :

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

In the expression (3.12) two terms can be isolated: $\pi(y_s|x_s, T, \theta)$ the likelihood

of θ w.r.t the source domain and $\pi(\theta|T, x_s)$ the prior term determined by the target domain. Truly each of this two terms depends on both domains thanks to the mixing factor α . The likelihood term can be rewritten in this way (since the data are *i.i.d.* for hypothesis):

$$\pi(y_s|x_s, T, \theta) = \prod_{i=1}^n f_s^\theta(y_i^s; x_i^s)$$

in which $f_s^\theta(y_i^s; x_i^s)$ expresses the probability that the source predictor assigns at the example x_i^s the label y_i^s . The prior distribution is computed as a function of the level of label uncertainty (for an hypothesis θ) when a predictor is used on target samples:

$$h(\theta|T, x_s) = -\frac{1}{m} \sum_{i=1}^m \sum_{y \in Y} f_t^\theta(y; x_i^t) \log f_t^\theta(y; x_i^t)$$

in which $f_t^\theta(y; x_i^t)$ is the probability that the target predictor assigns at the example x_i^t the label y . This term is the empirical entropy of $y|\theta$ conditioned on x from which a prior distribution can be derived:

$$\pi(\theta|T, x_s) \propto \exp(-\lambda h(\theta|T, x_s))$$

with the constraint $\int h(\theta|T, x_s) \pi(\theta|T, x_s) s\theta = \varepsilon$ ($\varepsilon > 0$) expressing how low should be the label uncertainty. The parameter λ is the Lagrange multiplier associated with ε . In conclusion the loss function used during the training process is:

$$L(\theta) = L^s(\theta) + \lambda L^t(\theta) \quad (3.13)$$

where

$$\begin{aligned} L^s(\theta) &= -\frac{1}{n} \sum_{i=1}^n \log f_s^\theta(y_i^s; x_i^s) \\ L^t(\theta) &= -\frac{1}{m} \sum_{i=1}^m \sum_{y \in Y} f_t^\theta(y; x_i^t) \log f_t^\theta(y; x_i^t) \end{aligned}$$

3.4 Adversarial Discriminative Domain Adaptation

The method is based on a Generative Adversarial Network (GAN)s learning where two networks (a generator and a discriminator) are involved: the generator yields images so that confuses the discriminator which attempts to recognize them from real image examples. This mechanism in domain adaptation is used to make sure that the network is not able to distinguish among source and target domain samples. The algorithm implements the following three phases (see Figure 3.4):

- *Pre-training*: a source encoder CNN together with a classifier are trained in the classical way using the labeled source domain.
- *Adversarial Adaptation*: a target encoder CNN is trained in such a way that a discriminator is not able to recognize the domain label of the examples.
- *Testing*: the classifier trained in the first phase is used together with the target mapping learned during the second phase to classify the target examples.

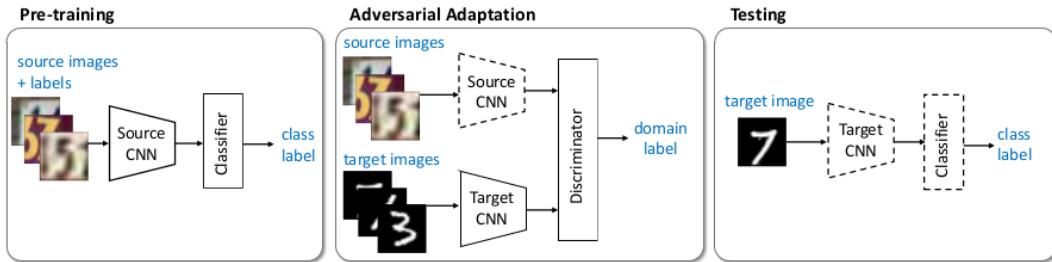


Figure 3.4: Phases of the Adversarial Discriminative Domain Adaptation method. Dotted lines denote pinned network parameters [9].

3.4.1 Source and target mapping

The purpose of this method is to find a target mapping M_t together with a classifier C_t that is able to classify target images also if their labels are unknown (*unsupervised adaptation*). The idea is to learn a source mapping M_s , a source classifier C_s and then compute a good mapping M_t for target domain adapting M_s in such a way that target images can be efficiently classified using the source classifier C_s . The first choice to make is in the parametrization of these two mappings, keeping in mind that the distance between the source and target domains

should be minimized while, at the same time, the category-discriminative target mapping should be preserved. The target mapping M_t is implemented so that it matches, for the architecture, with the source mapping. Moreover, for each layer an equality constraint is imposed:

$$\psi(M_d, M_t) \triangleq \{\psi_{\ell_i}(M_s^{\ell_i}, M_t^{\ell_i})\}_{i \in \{1 \dots n\}}$$

where $M_s^{\ell_i}$ and $M_t^{\ell_i}$ are the i -th layer mapping parameters for the source and target domain respectively. The typical layerwise constraint that can be imposed in a CNN with weight sharing is:

$$\psi_{\ell_i}(M_s^{\ell_i}, M_t^{\ell_i}) = (M_s^{\ell_i} = M_t^{\ell_i})$$

3.4.2 Adversarial losses

After that a parametrization for M_t has been chosen, it is necessary to decide the adversarial loss for the target mapping. To train the generator in a GAN is used the standard loss function but with reversed labels:

$$\mathcal{L}_{adv_M}(\mathbf{X}_s, \mathbf{X}_t, D) = -\mathbb{E}_{\mathbf{x}_t \sim \mathbf{X}_t}[\log D(M_t(\mathbf{x}_t))]$$

We refer to this function with the expression *GAN loss function*. It can be noticed that with this formula only the target mapping M_t is learned, the source mapping remains fixed. This is the typical approach of the GAN method in which the distribution of real images is fixed and the distribution of the generate images is learned in order to match it.

3.4.3 Adversarial discriminative domain adaptation

We are able now to revisit the three phases of the ADDA algorithm in a more technical way. The first phase, in which a source mapping M_s and a source classifier C (C_s) are learned, is performed in a standard way, with this classical supervised loss:

$$\min_{M_s, C} \mathcal{L}_{cls}(\mathbf{X}_s, Y_s) = -\mathbb{E}_{(x_s, y_s) \sim (\mathbf{X}_s, Y_s)} \sum_{k=1}^K \mathbb{1}_{[k=y_s]} \log C(M_s(\mathbf{x}_s))$$

where \mathbb{E} is the expected value of the source examples and K is the number of class categories.

The second phase, in which a target mapping M_t and a domain discriminator D are learned, is performed alternately optimizing two loss functions:

$$\min_D \mathcal{L}_{adv_D}(\mathbf{X}_s, \mathbf{X}_t, M_s, M_t) = -\mathbb{E}_{\mathbf{x}_s \sim \mathbf{X}_s}[\log D(M_s(\mathbf{x}_s))] - \mathbb{E}_{\mathbf{x}_t \sim \mathbf{X}_t}[\log(1 - D(M_t(\mathbf{x}_t)))] \quad (3.14)$$

$$\min_{M_s, M_t} \mathcal{L}_{adv_M}(\mathbf{X}_s, \mathbf{X}_t, D) = -\mathbb{E}_{\mathbf{x}_t \sim \mathbf{X}_t}[\log D(M_t(\mathbf{x}_t))] \quad (3.15)$$

Since has been opted to let M_s fixed during the second phase, the loss functions (3.14) and (3.15) are minimized without a revisiting of the pre-training phase.

Chapter 4

Colorization methods

In this project the domain adaptation algorithms are applied using both RGB and depth informations as input data. It has been demonstrated that the depth produces additional informations to the standard RGB modality in the object recognition field: RGB provides texture, color and aspect informations while depth gives geometrical informations of the object shape that are invariant to the light conditions (see Figure 4.1). While RGB modality is composed by nature of three channels and it can be directly feed in a CNN, depth informations needs some transformations to be mapped in the three-channel input of a network. This mapping is done to take advantage from the features of pre-trained CNNs but also to match the two modalities and investigate the performance of the RGB-D input characteristics.

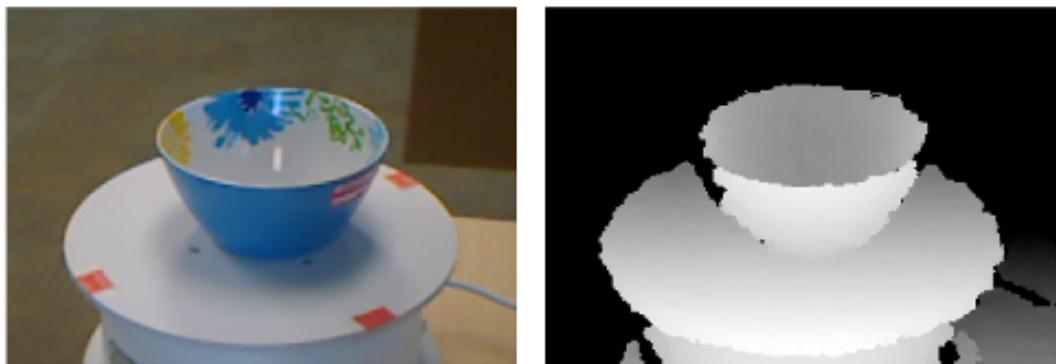


Figure 4.1: Example of RGB (left) and depth (right) data from the same object [12].

4.1 Surface Normal Colorization

Normal maps are usually standard RGB images where the Red Green and Blue components coincide with the X, Y, and Z coordinates of the surface normal. By definition a normal is a line or vector perpendicular to an object. In the three-dimensional space (our case of study) the normal is called *surface normal* [33]. The surface normal of a point P belonging to an object is a vector perpendicular to the plane tangent to the surface in that point (see Figure 4.2).

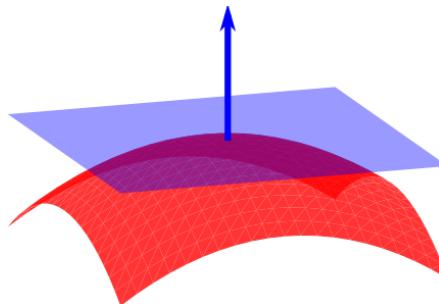


Figure 4.2: The normal of a point on a surface [34].

For each pixel of the depth image the surface normal is computed in this way: in the horizontal direction (x-axis) and in the vertical direction (y-axis) the gradients are computed to obtain two 3D vectors $a = [1, 0, \frac{\partial z}{\partial x}]^T$ and $b = [0, 1, \frac{\partial z}{\partial y}]^T$ towards the z-axis; the surface normal is computed with the cross product of a and b resulting in the vector $n = [-\frac{\partial z}{\partial x}, -\frac{\partial z}{\partial y}, 1]$ (see Figure 4.3).

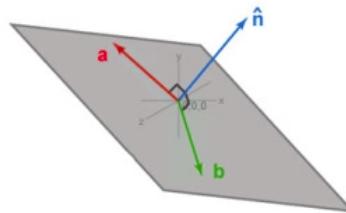


Figure 4.3: Surface normal geometric construction ($n = a \times b$) [35].

Once that the surface normal vector is computed, it is normalized using the Euclidean norm and then each of the three values of n are mapped to the corresponding RGB channel in this way:

$$x \in [-1, 1] \rightarrow \text{Red} \in [0, 255]$$

$$y \in [-1, 1] \rightarrow \text{Green} \in [0, 255]$$

$$z \in [0, 1] \rightarrow \text{Blue} \in [128, 255]$$

An example of the final resulting depth image can be seen in Figure 4.4. We can notice that the surface normal colorization method capture properly the structural informations of the object.



Figure 4.4: RGB image (left), original depth image (center), image of the depth mapped with Surface Normal (right).

4.2 Surface Normal++ Colorization

To overcome the lack of informations that characterize most of the images of the datasets used in this project, for some experiment turned out to be useful to adopt a different pre-processing for the images called *surface normal++* [36].

The steps of image pre-processing are the following:

- *Recursive median filter* proposed by (Lai et al., 2011). For each missing depth value of a depth image non-missing values in its neighbourhood are taken in consideration by a median filter that is recursively applied to fill the "holes" and to minimize the blurring.
- *Borders underlined*. When a median filter is applied border problems may appear. To overcome this issue a border replication technique is used.
- *Bilateral filter*. The task of this filter is to reduce the noise presents on a depth image preserving borders and increasing the smoothing.
- *Surface Normal*. For each pixel is applied the surface normal colorization explained in the previous section.
- *Unsharp mask filter*. When the bilateral filter is applied, despite trying to find a compromise between preserving borders and increasing the smoothing, some details are lost. To limit the damages an unsharp mask filter can be used. It increments contrast between borders and other high-frequency elements.

In Figure 4.5 it can be seen an example of a depth image generated after the pre-processing described.

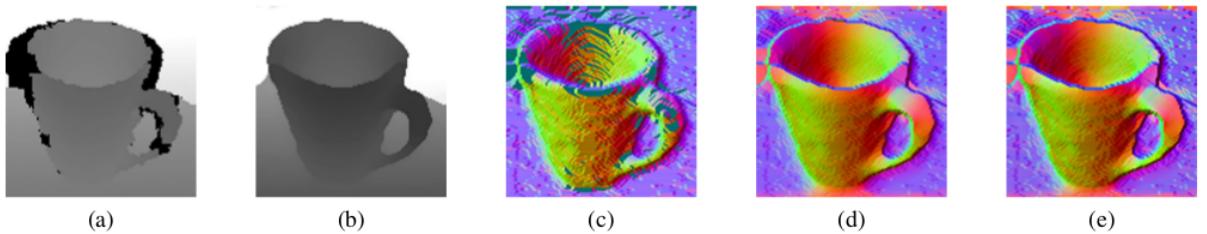


Figure 4.5: Illustration of the pre-processing method steps. (a) Original depth image, (b) after applying recursive median filter, (c) after applying surface normal colorization, (d) application of surface normal colorization after the using of a bilateral filter, (e) after the application of an unsharp mask filter [36].

Chapter 5

Deep cue integration

Due to the basic intuition under which a model trained with more features reaches a better performance, in the object recognition scenario the use of cue integration is becoming an increasingly popular choice. Particularly in this area characterized by a great intraclass variability, only one type of feature could not be enough to create a robust classifier that overcomes this issue: in many cases RGB information only could not be sufficient for the correct classification of an object whose shape, given by depth informations, could instead be decisive for the recognition.

Classification with Multiple Cue Integration.

Let $\{x_i, y_i\}_{i=1}^N$ be N samples of training set in which $x_i \in X$ is the input data (for instance an image) and $y_i \in Y$ is the label; let $\phi^j : X \rightarrow R^{D_j}$ be a features extractor from a set of F functions ($j = 1, \dots, F$) with D_j the dimension of the j -th feature. The intention is to learn a classifier $f : X \rightarrow Y$ that is able to classify the new sample obtained by the integration of all the F features extracted.

5.1 Fusion Types

In this section there will be a brief description of the three most popular techniques by which cue integration can be applied [37]:

- ***Low-level Integration:***

the new samples are formed with the features extracted directly from data. It is also known as *pre-mapping fusion* in fact the combination of data is done before any type of feature mapping using the information provided directly by sensors. Then, these new samples are put into a supervised learning algorithm to find the best parameters of the classifier (see Figure 5.1).

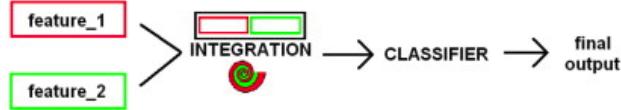


Figure 5.1: The low-level integration technique [38].

- ***Middle-level Integration:***

it is a more complex technique with respect to the other fusion approaches. In this case the integration is done in a middle level together with the mapping, then the resulted features are used for the final classifier (see Figure 5.2).

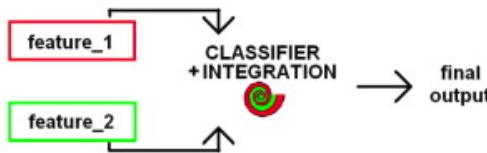


Figure 5.2: The middle-level integration technique [38].

- ***High-level Integration:***

for each feature a classifier is trained independently, from each of them the confidence scores are extracted and combined to have the new samples. This type of integration can be seen as a two-layer scheme: with the first layer the confidence score for each feature is obtained using different learning algorithms; with the second layer these confidence scores are combined and used to train, for example, a linear Support Vector Machine (SVM) (see Figure 5.3).

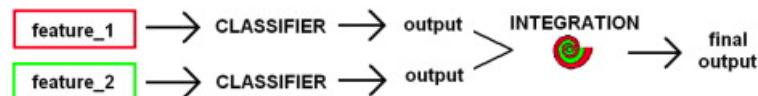


Figure 5.3: The high-level integration technique [38].

5.2 High-level Integration

In the previous section a short overview has been made on the most popular approaches with which cue integration can be implemented. For the purpose of the project, now we focus on the last one: the *high-level integration*.

Suppose that $\{\mathbf{x}_i, z_i\}_{i=1}^N$ is the training set, F is the number of cues chosen and ϕ^j is the j -th feature mapping. A classifier is trained for each input modality and then, to combine all the features together, for each modality the confidence scores $s^j(\mathbf{x})$ associated to the specific classifier is taken. Finally these confidence scores are put together for the final classification:

$$s(\mathbf{x}, z) = \sum_{j=1}^F \beta_z^j s^j(\mathbf{x}) \quad (5.1)$$

where $s^j(\mathbf{x}) = \mathbf{w}_z^j \cdot \phi^j(\mathbf{x})$ with \mathbf{w}_z^j computed independently for each cue. The weights β_z^j determine how much the j -th classifier should affect the final one, this quantity is optimized jointly considering all the classifiers involved (look at [37] for more details). The high-level cue integration technique is introduced in this project with the aim of combine RGB and depth channels to evaluate the performance of this multi-modal data type in the domain adaptation scenario as in [32]. It is exploited following this sequence:

- One network, properly modified to perform domain adaptation, is trained independently for RGB and depth input modalities in such a way that the best weights \mathbf{w}_z^j (with $j = 1, 2$) are found for both data type.
- The activation values (that represent the confidence score $s^j(\mathbf{x})$ in (5.1)) of the last layer before the classification (for example the output of the layer fc7 in *AlexNet*) are taken from the two trained network. For each image, of both source and target domain, two feature vectors (one for RGB modality, one for depth modality) are captured. Subsequently they are concatenated having in the end one "big" feature vector for each image (that intrinsically has both RGB and depth informations).
- The new concatenated features of source domain are used to train the linear SVM and the new concatenated features of target domain are used to test it for evaluate the adaptation.

Chapter 6

Experiments

6.1 Datasets

All the unsupervised adaptation transfer tasks are conducted on five publicly-available datasets: ROD, ARID, WOD, BigBIRD and Active Vision Dataset.

6.1.1 RGB-D Object Dataset

ROD is composed by objects of everyday life that can be found in places where robots should be able to operate (home, office, ...) (see Figure 6.1 for some examples).



Figure 6.1: Objects from the RGB-D Object Dataset [12].

The categories of objects involved are 51, a subset of the 1000 in ImageNet [39]. In particular, the dataset has visual and depth informations about 300 objects each belonging to one of these 51 categories (as result one category has from three to fourteen instances). The images of the objects have been captured by an RGB-D camera that together get color and depth informations at 640x480 pixels of resolution; each pixel has actually four channels: red, blue, green and

depth. The camera has been located one meter away from a turntable that rotates with a fixed speed on itself. Registration took place with the camera mounted at 30° , 45° and 60° above the horizon; for each object, data have been recorded at 20 Hz during a complete revolution at each of the three heights. The dataset also provides 8 natural scenes of ordinary indoor places in which the objects of the RGB-D dataset have been disseminated, the scenes were recorded using the RGB-D camera at an height almost equal to the level of the eyes of a human being. Together with these 8 natural scene, also the ground truth bounding boxes are provided (see Figure 6.2).

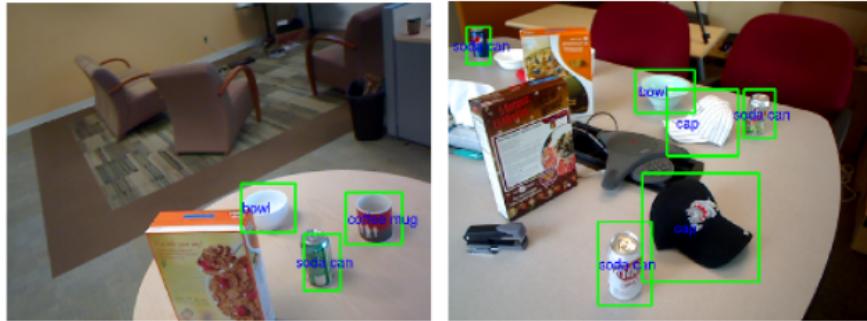


Figure 6.2: Detection results in two multi-object scenes [12].

Extracting the images from these scene you will have images of the objects in different ways: from different points of view, at a different distance from the camera or with partial occlusion.

6.1.2 Autonomous Robot Indoor Dataset

ARID coincides in number and typology with the 51 categories of ROD described in the previous paragraph. This two datasets can be seen as complementary: the same objects are collected in a restricted setting (in this case we refer only to the raw ROD without the natural scenes) and in a real-life environment (ARID). The real-life context, with which a robot should be able to deal, is obtained acquiring the images with an RGB-D camera attached on a mobile robot that navigates in a typical human environment (see Figure 6.3).

Hence, despite ROD that is characterized by an assigned camera-object distance, a fixed background and no changes in light setting, ARID includes all these real-world traits: alteration in the objects illumination, dynamic viewpoint, clutter, occlusion, partial view, change in the object scale and background variation. The total number of objects involved is 153: 3 instances for each category. To guarantee the natural variation in the lighting of the objects, data have been



Figure 6.3: The mobile robot that acquire data from a scene with everyday objects [13].

acquired in 10 sessions not in the same day and in different hours. Each session has the duration of one hour in which the mobile robot loops over four established waypoints. The environment has been prepared with 30-31 objects that are relocated every two patrolling loops to assure the variation of both the object view and the camera-object distance. Every time that the mobile robot reaches a waypoint the RGB-D camera scans the scene using the pan-tilt unit to do an horizontal movement. The camera collects RGB and depth images at 30 Hz with a resolution of 640x480 pixels.

6.1.3 Web Object Dataset

WOD is composed by images taken from the web through searches query made on Google, Yahoo, Bing and Flickr. The best images among those resulting from the researches are selected using a method designed by Massouh et al. [40] that use both visual and Natural Language Processing (NLP) informations to eliminate most of the noise (the remainder is removed by hand). The objects that constitute the dataset belong to the same 51 categories of ROD and ARID but, unlike them, WOD has a much bigger number of instances (each image potentially contains a different object) and he does not have depth informations of the objects.

6.1.4 Big Berkeley Instance Recognition Dataset

BigBIRD was designed with the aim of investigate object recognition only at instance level. It is composed by 100 3D object instances each different from the others but without any type of classification at category level. It's an high quality dataset both for the large number of images per object (with RGB and depth informations) and for the high-resolution that characterizes them. The images were captured using 5 high-resolution cameras (12.2 MP) and 5 depth sensors mounted using a RGBDToolkit (see Figure 6.4) in 5 different heights and positions (see Figure 6.5).



Figure 6.4: RGBDToolkit [14]



Figure 6.5: Side view of the five cameras and the five depth sensors mounted ready to start a data acquisition process [14].

Each object has been placed on the glass turntable to which the cameras, together with four lights, points (see Figure 6.6).

The four lights are located at the bottom, at the back wall, at the front corners and at the back corners. It is also important to point out the presence of

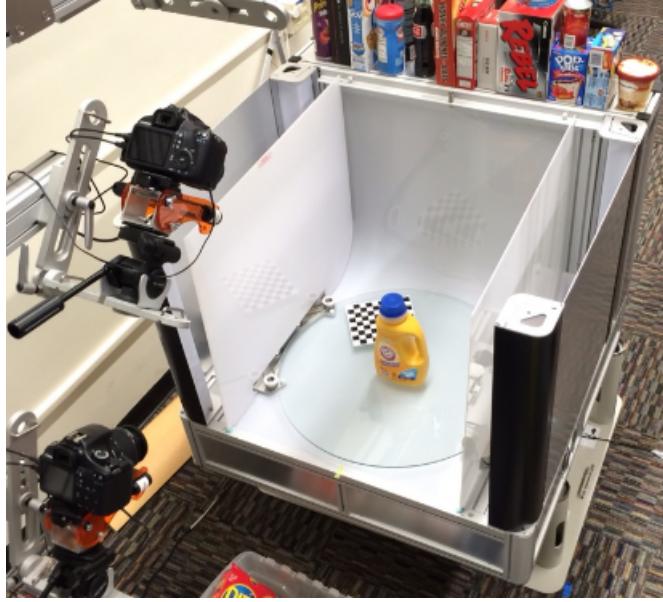


Figure 6.6: The glass turntable of the data-collection system [14].

a chessboard that improves data calibration between RGB and depth informations. One by one the objects have been placed on the turntable which rotated in units of 3 degree until done a complete revolution and, for each angle, all five cameras acquire data. In total 600 RGB images and 600 point clouds have been acquired for each object ($360^\circ / 3^\circ = 120$, $120 \times 5 = 600$). Subsequently a segmentation mask of the objects, for each view, was produced. The entire process of acquisition of the data took less than 6 minutes for object almost without human contribution.

6.1.5 Active Vision Dataset

Active Vision Dataset was built starting from the BigBIRD dataset described above: 33 object instances similar to those of BigBIRD are included in the scenes. The number of scenes recorded is 9 (see Figure 6.7 for examples) but some of them are captured twice with some small changes (displacing objects that are typically moved by people as books, chairs, . . .) for a total of 17 scans.

The goal is to resemble the motion of a robot that moves within daily environments such as office, kitchen, living room, ecc.. The scenes are recorded within these type of rooms with a Kinect v2. For each scene a set of points (58-201) is chosen, once the robot has reached one of these points, it will rotate the camera on itself. It was chosen to record an image every 30 degree in each point to not have an excessive number of images for scene (see Figure 6.8). The scans have been labeled resulting in 3000 (on average) 2D bounding box for each scan.



Figure 6.7: Example of scenes of the dataset [15]

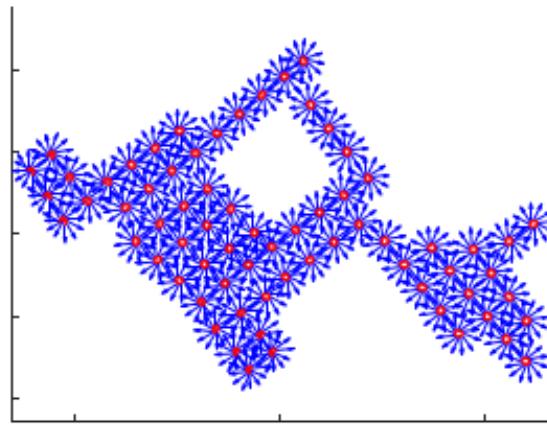


Figure 6.8: Map of the movements of the camera in a scene with all the subset of point chosen for the data recording [15].

6.2 Experimental Setup

The experiments were conducted using the four deep domain adaptation networks described in chapter 3: DAN [6], DANN [8], AutoDIAL [7], ADDA [9]. The evaluation is performed using multimodal input data: RGB only, depth only and RGB-D on the adaptation tasks ROD \rightarrow ARID, WOD \rightarrow ARID and BigBIRD \rightarrow Active Vision. The domain adaptation algorithms have been implemented modifying properly two of the most popular deep network architectures: AlexNet [41] and ResNet-50 [42]. In both cases the networks are pre-trained on ImageNet [43] and then fine-tuned on the specific dataset. Depth images have been colorized either with Surface Normal, in the case of depth only input modality, or with Surface Normal++, for the RGB-D data (both colorization methods are described in chapter 4); the resulted images, now mapped on three channels, can be fed into domain adaptation networks as if they were RGB images.

Datasets. In order to have datasets belonging to the same adaptation task

approximately of the same size, from ROD 41877 images have been taken in RGB and depth formats, from ARID 40713 and from WOD 50547 (for this last dataset only in RGB format, the reason is explained in the previous section). It is noteworthy that for the RGB-D experiments, to reach a better performance, only a subset of ARID composed by 36050 images has been used: depth images with more than 75% of null pixels have been removed. For BigBIRD and Active Vision, with the aim of have the same amount of images for both datasets with a balanced distribution over all the object classes, a particular selection was done. First of all the evaluation set of BigBIRD, composed by only a subset of the whole dataset, has been selected with the aim of avoiding the presence of too similar images that could compromise the training with overfitting. Applying the same selection policy used for the evaluation set of Washington, for each object one frame every 6 degree of rotation, from each of the 5 cameras, is taken obtaining a total of 300 images per class ($360^\circ/6^\circ = 60$, $60 \times 5 = 300$). Then, since the images have a lot of noise (the chessboard, the glass turntable, etc.) it has been decided to make a crop focused on the object. From the official website you can download the RGB images, the depth images and the segmentation masks (an example in Figure 6.9), the crop is done using the segmentation mask to infer border informations about the object.

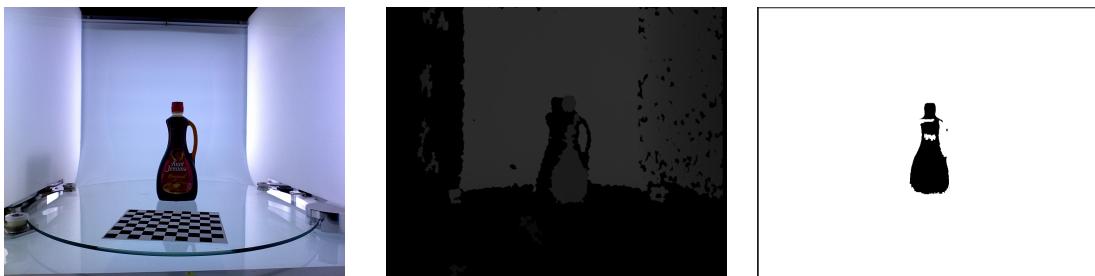


Figure 6.9: From left to right: RGB image, depth image, segmentation mask provided by Big Berkeley Instance Recognition Dataset.

For all the images (in both RGB and depth formats) it was decided to have a bounding box that wrap the object in such a way that there are 25 pixel from the borders of it to the edges of the box (see Figure 6.10 for an example of both formats).

Since some objects appear very rarely in Active Vision scenes, to have a dataset composed by a sufficient quantity of images for each object class (300, to be consistent with BigBIRD), the number of categories in these two datasets has been reduced from 33 to 30. The objects removed are: *expo marker red* (221 images), *softsoap clear* (286 images) and *red cup* (12 images). For Active Vision



Figure 6.10: Example of a cropped RGB image on left, example of a cropped depth image on right.

the 300 images were selected randomly among all those available in each category. After these steps both datasets become composed by 9000 images in either RGB and depth formats.

Preprocessing. It was done a different preprocessing of the images in function of the starting network used: for AlexNet both RGB and depth images are first scaled to 256x256 pixels and then randomly cropped at 227x227 pixels; for ResNet-50 it was necessary to do the preprocessing of RGB and depth images in a specular way without any random component since the weights computed using this basic network are then used also for RGB-D experiments. In this case the images are scaled directly to 224x224 pixels that is the size of the input data layer of ResNet-50.

DAN setting. For AlexNet the MK-MMD adaptation regularizer is added to the CNN risk for *fc7-fc8* layers; in ResNet-50 it is added for *average pool* and *fc1000*. For all the experiments the parameters related to the learning rate policy have been set up as proposed by the authors: policy *inverse*¹, with gamma 0.001, power 0.75 and the back propagation method used to train the network is Stochastic Gradient Descent (SGD) with momentum 0.9.

DANN setting. The GRL is added after *fc7* in AlexNet and after *pool5* in ResNet-50. Also for DANN the learning rate policy parameters have been set up as proposed by the authors: policy *inverse*, with gamma 0.001, power 0.75 and the back propagation method used to train the network is SGD with momentum

¹

learning_rate = base_lr · (1 + γ · current_iteration)^{-power}

0.9.

AutoDIAL setting. The Domain Alignment layers are inserted in *fc6-fc7-fc8* for AlexNet and only in *fc1000* for ResNet-50. The learning rate policy has been set up also in this case as the authors: policy *step*², gamma 0.1, step size 85% of the total number of epochs, weight decay 5×10^{-4} and the back propagation method used to train the network is SGD with momentum 0.9.

ADDA setting. In this case only ResNet-50 has been used as base model, the algorithm is applied on this network with the same setting proposed by the authors. In particular, for the second phase of the algorithm (the *Adversarial Adaptation*), the layers up to block *conv4* of the target model have been fine-tuned. The *discriminator* is composed by three fully connected layer of size 1024, 2048 and 3072 every succeed by ReLUs and there is one fully connected layer for the final outcome. The policy of the learning rate is *fixed*³ and the back propagation method used to train the network is SGD with momentum 0.9.

Training. To have a starting point for the evaluation of the results obtained from the domain adaptation tasks a baseline for each input modality is also provided. It consists of the results obtained training a network with source domain and testing it on target domain without any kind of adaptation. The set of adaptation experiments can be divided into two macro groups: (1) whole target dataset in both train and test phases, (2) two different subsets of target dataset for train and test phases. For group (1) the size of target datasets have been shown in the paragraph *Datasets* of this section: 40713 → ARID, 9000 → Active Vision. For group (2) a subdivision of them was made. Since ARID is provided with 3 instances for each object class, the dataset has been divided into three different train/test splits using for each test set a different instance of the same object. The size of the resulted splits obviously is variable: Split1 → 26188/14525, Split2 → 27685/13028, Split3 → 27553/13160. Active Vision is subdivided according to another policy but also in this case the splits created are three. Each split has the same size: 7200 images for train and 1800 images for test. The subdivision is done,

²

learning_rate = base_lr · $\gamma^{\frac{\text{current_iteration}}{\text{size_step}}}$

³

learning_rate = base_lr

for the test set, taking 60 images for each object class ($60 \times 30 = 1800$) and, for the train set, the rest of the images have been taken ($(300 - 60) \times 30 = 7200$). Obviously the images chosen for the test sets are different for each split.

6.3 Results and Discussion

The experiments have been conducted trying several base learning rates, different batch sizes and seldom also trying different configurations of the learning rate multipliers within the networks. Below are shown only the parameter settings related to the best results obtained. For all the experiments the mean file of ImageNet dataset has been used.

6.3.1 RGB Only

Tables 6.1 and 6.2 show the best results for the adaptation tasks ROD/WOD → ARID relatively for the group of experiment (1) and (2) (the subdivision of the experiments is explained in the previous section in paragraph *Training* of Section 6.2) as well, Tables 6.3 and 6.4 report the best results for the adaptation task BigBIRD → Active Vision. The parameter setting which led to the best results for the shift ROD/WOD → ARID in both groups of experiments is the following:

- DAN

AlexNet → lr⁴: 0.0001, bs⁵: 64, epochs: 30, MK-MMD loss weight: 1

ResNet-50 → lr: 0.001, bs: 128, epochs: 30, MK-MMD loss weight: 0.3

- DANN

AlexNet → lr: 0.0001, bs: 128, epochs: 30, Domain Classifier loss weight: 0.1

ResNet-50 → lr: 0.001, bs: 64, epochs: 30, Domain Classifier loss weight: 0.1

- AutoDIAL

AlexNet → lr: 0.0001, bs: 256, epochs: 30, Target loss weight (λ in (3.13)): 0.1

ResNet-50 → lr: 0.001, bs: 256, epochs: 30, Target loss weight (λ in (3.13)): 0.4

- ADDA

ResNet-50 → lr: 0.001, bs: 128, epochs: 30, Mapping & Adversarial loss weight: 1

We can notice that, with AlexNet as basic network, AutoDIAL outperforms all comparison methods; using ResNet-50, instead, the best algorithm is DANN although also the other algorithms obtained good results. Contrary to expectations, only the half of the second group of experiments has a worse performance with respect to the first group although in the second group the adaptation and the test phases have been done with a different set of images. In this second group of

Method	AlexNet		ResnNet-50
	ROD → ARID	ROD → ARID	WOD → ARID
Source only [13]	0.291	0.337	0.388
DAN	0.34	0.429	-
DANN	0.329	0.459	0.582
AutoDIAL	0.378	0.442	-
ADDA	-	0.422	-

Table 6.1: ROD/WOD → ARID RGB experiments of group (1).

Method	AlexNet		ResnNet-50
	ROD → ARID	ROD → ARID	WOD → ARID
Source only [13]	0.291	0.337	0.388
DAN	0.349	0.466	-
DANN	0.356	0.439	0.527
AutoDIAL	0.299	0.467	-
ADDA	-	0.357	-

Table 6.2: ROD/WOD → ARID RGB experiments of group (2).

experiments it can be notice that the best performance using AlexNet is obtained with DANN, instead, using ResNet-50 AutoDIAL reaches the best accuracy. It is interesting to note that the highest level of accuracy, for ResNet-50, with the second group of experiments is reached. It was chosen to perform the adaptation for WOD → ARID using only the algorithm which led to the best result in the shift ROD → ARID: DANN. We can notice an evident improvement of the accuracy obtained using WOD as source dataset. This is motivated by the fact that taking the images from the web gives rise to a much more realistic dataset with respect to ROD in which the images are acquired in a quite unrealistic settings.

For the shift BigBIRD → Active Vision the best set of parameters is the following:

- DAN

AlexNet → lr: 0.001, bs: 128, epochs: 30, MK-MMD loss weight: 0.7

ResNet-50 → lr: 0.001, bs: 64, epochs: 30, MK-MMD loss weight: 0.7

- DANN

AlexNet → lr: 0.001, bs: 64, epochs: 30, Domain Classifier loss weight: 0.1

ResNet-50 → lr: 0.001, bs: 128, epochs: 30, Domain Classifier loss weight: 0.1

- AutoDIAL

⁴base learning rate

⁵batch size

Method	AlexNet	ResnNet-50
	BigBIRD → Active Vision	BigBIRD → Active Vision
Source only	0.264	0.361
DAN	0.459	0.497
DANN	0.473	0.537
AutoDIAL	0.521	0.512
ADDA	-	0.512

Table 6.3: BigBIRD → Active Vision RGB experiments of group (1).

Method	AlexNet	ResnNet-50
	BigBIRD → Active Vision	BigBIRD → Active Vision
Source only	0.264	0.361
DAN	0.344	0.518
DANN	0.499	0.542
AutoDIAL	0.558	0.498
ADDA	-	0.572

Table 6.4: BigBIRD → Active Vision RGB experiments of group (2).

AlexNet → lr: 0.001, bs: 256, epochs: 30, Target loss weight (λ in (3.13)):

0.1

ResNet-50 → lr: 0.001, bs: 256, epochs: 30, Target loss weight (λ in (3.13)):

0.1

- ADDA

ResNet-50 → lr: 0.001, bs: 128, epochs: 30, Mapping & Adversarial loss weight: 1

We can observe that, for the first group of experiments, the best algorithms, for both AlexNet and ResNet-50 networks, are the same as for the task ROD → ARID, but here AutoDIAL and DANN remain the best algorithms also for the second group of experiments. As for the previous adaptation shift, paradoxically, the performances increase in the second group of experiments for almost all the algorithms. Probably this is due to the fact that, using less images from the target domain during the adaptation, a more general classifier is produced.

6.3.2 Depth Only

As said in the previous section, a baseline is obtained testing the target domain on a network trained with source domain in absence of adaptation. About the baseline for the task ROD → ARID it is interesting to do experiments based on the distance of the objects from the camera for two reasons: (i) RGB-D cameras have specific ranges of distance within which they provide reliable results, (ii) ROD dataset contains only images with objects in about a meter from the camera so, the classifier trained on this dataset should be able to recognize better the images of ARID in which the objects are within one meter from the observer. With this in mind, eight subsets of images from ARID have been grouped. The ranges considered are: 0 - 1200 mm, 0 - 1400 mm, 0 - 1600 mm, 0 - 1800 mm, 0 - 2000 mm, 0 - 2200 mm, 0 - 2400 mm, whole dataset. Figure 6.11 shows the histogram of the quantity of images for each range of subdivision and, for completeness, Figure 6.12, 6.13, 6.14, 6.15 report the distribution of the objects in function of their category for each specific range.

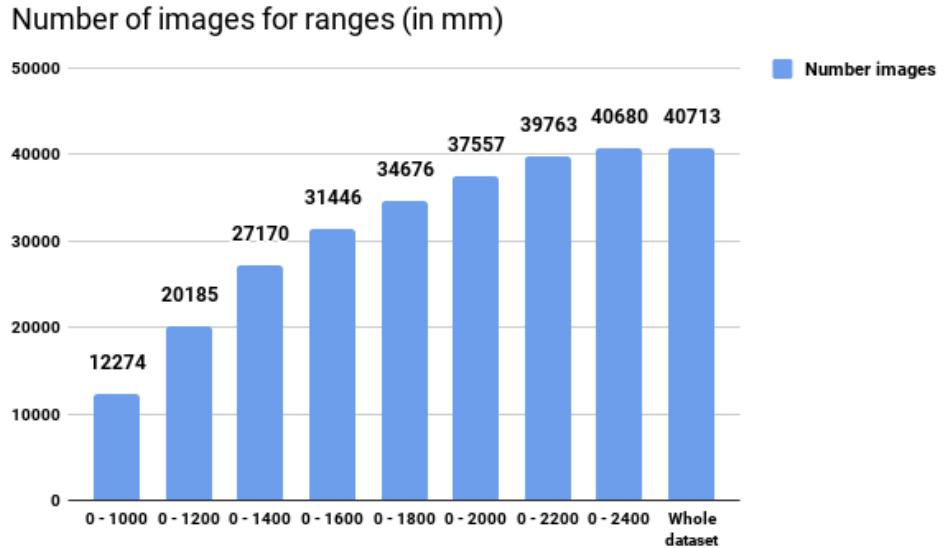
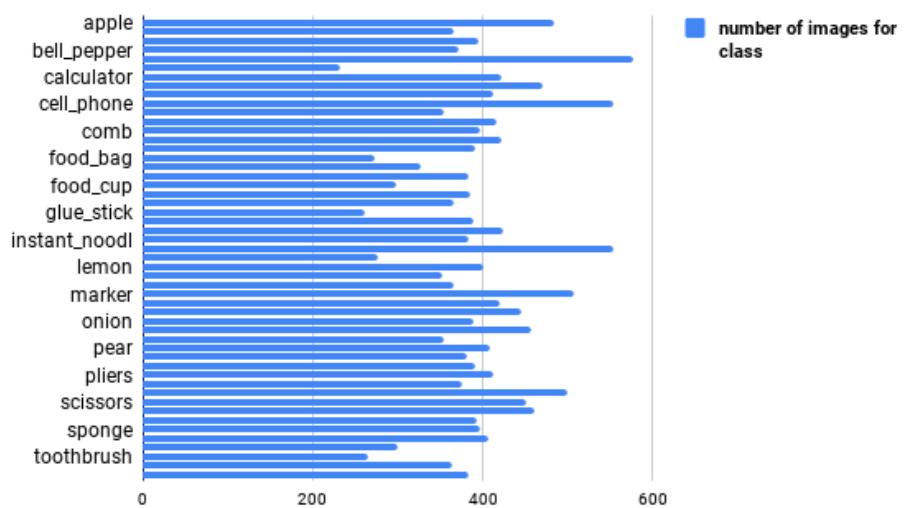


Figure 6.11: Quantity of images for each distance range in Autonomous Robot Indoor Dataset dataset.

Testing a network (AlexNet), trained with ROD, on each of the eight subsets, the accuracy decreases with the increase of distance range (see Figure 6.16). As predicted, the best accuracy is obtained using as test dataset the images of ARID in which the images are within a meter of distance from the camera. This is a limitation of ROD dataset due to a too static recording. Although it is not the best, to be consistent with RGB experiments, as baseline is taken the result

Object range 0-1200 mm - Accuracy: 16,33



Object range 0-1400 mm - Accuracy: 14,20

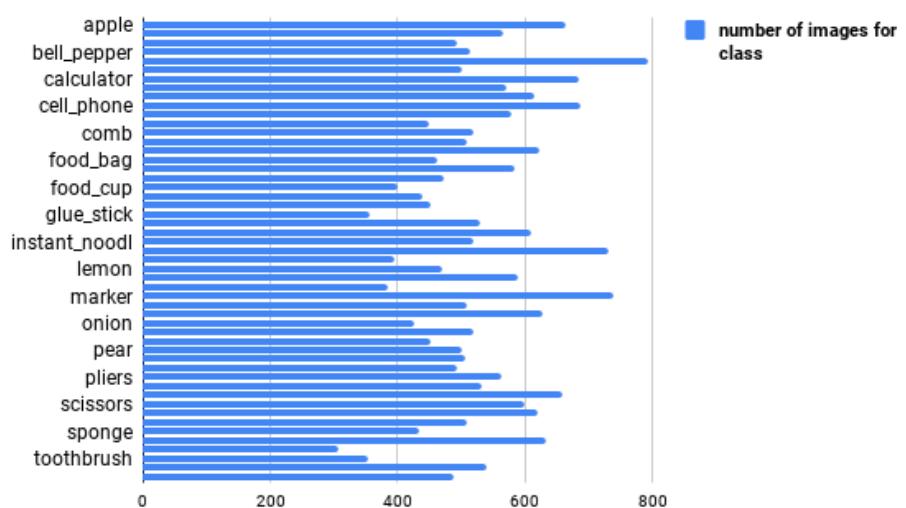


Figure 6.12: Quantity of images for each object class in ranges: 0-1200/0-1400.

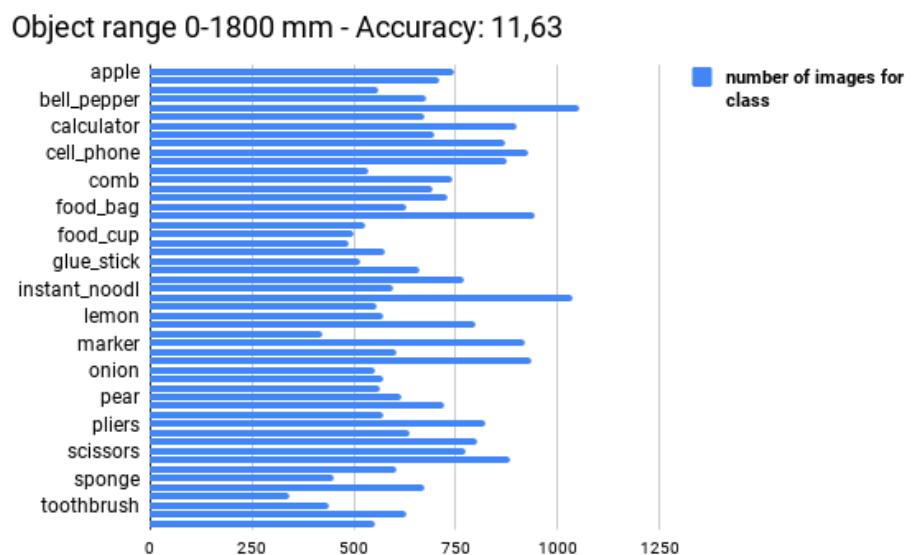
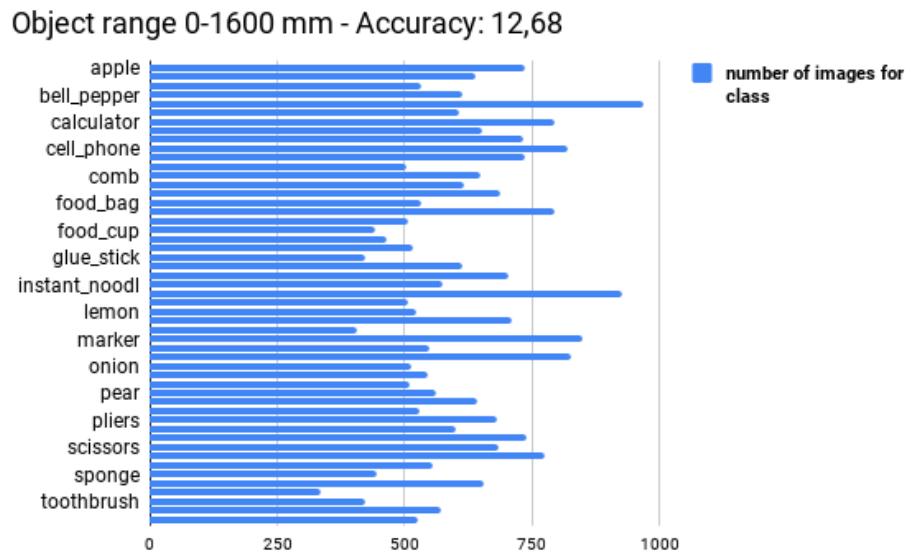


Figure 6.13: Quantity of images for each object class in ranges: 0-1600/0-1800.

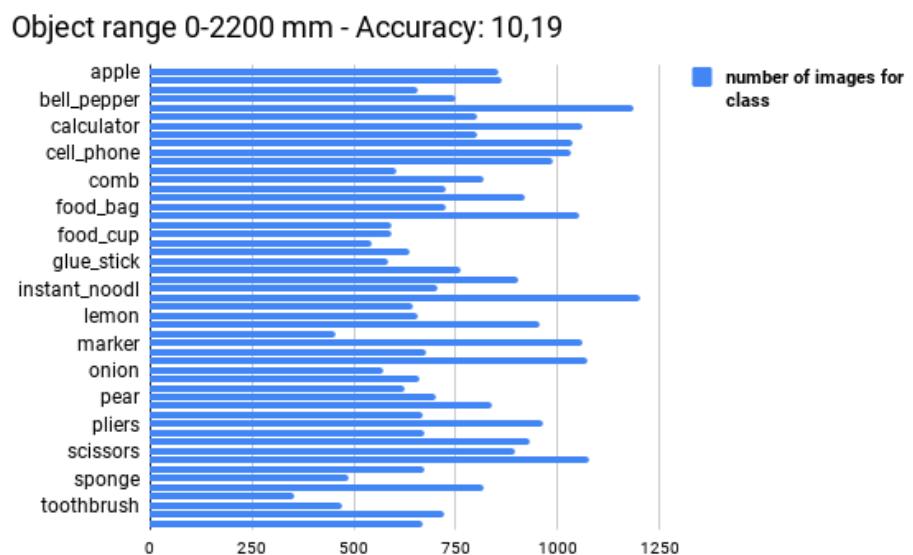
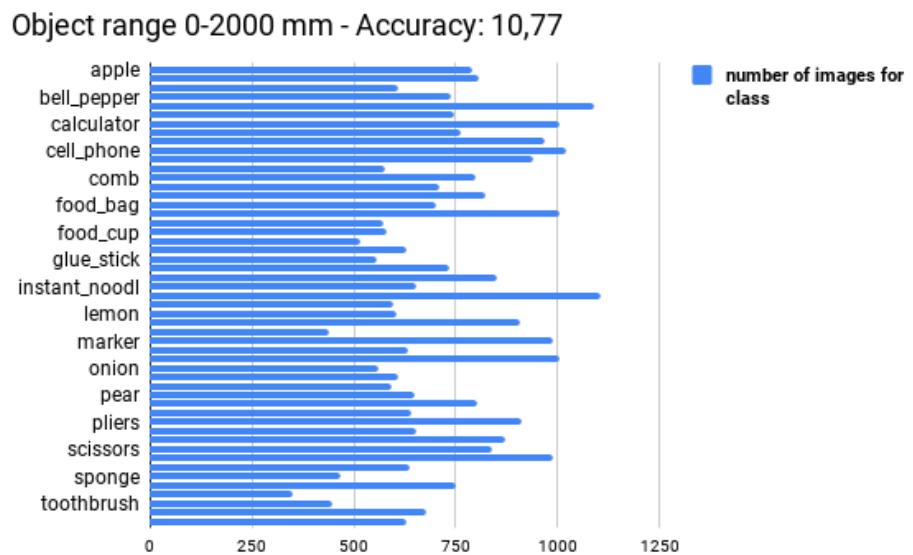


Figure 6.14: Quantity of images for each object class in ranges: 0-2000/0-2200.

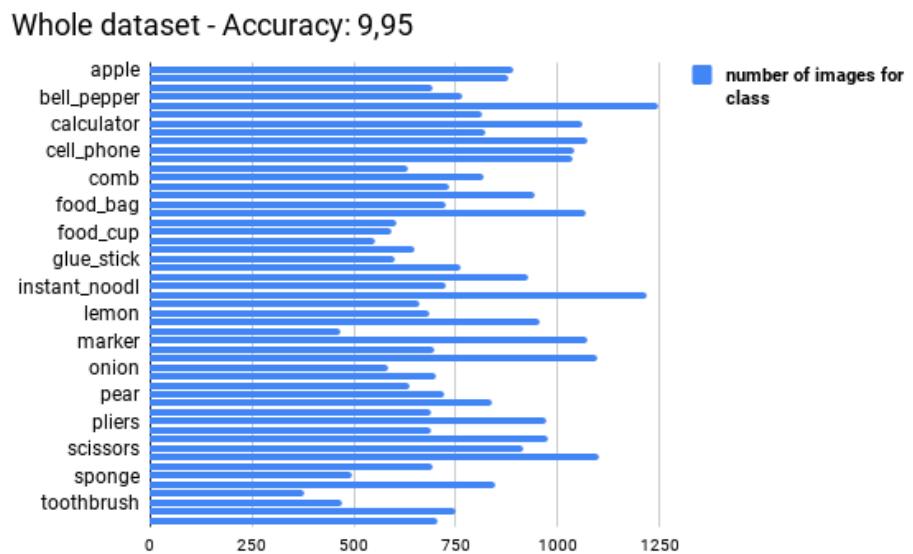
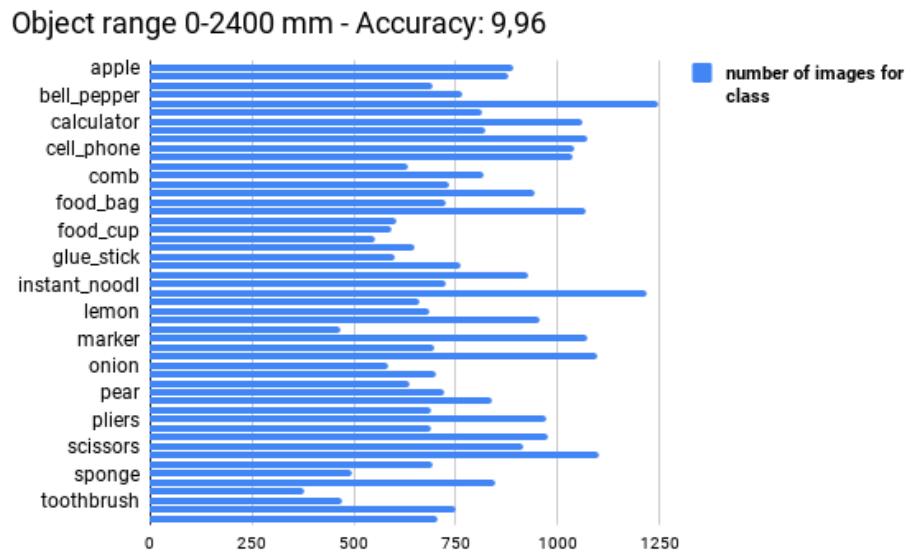


Figure 6.15: Quantity of images for each object class in ranges: 0-2400/0-max distance.

obtained with the whole ARID dataset.

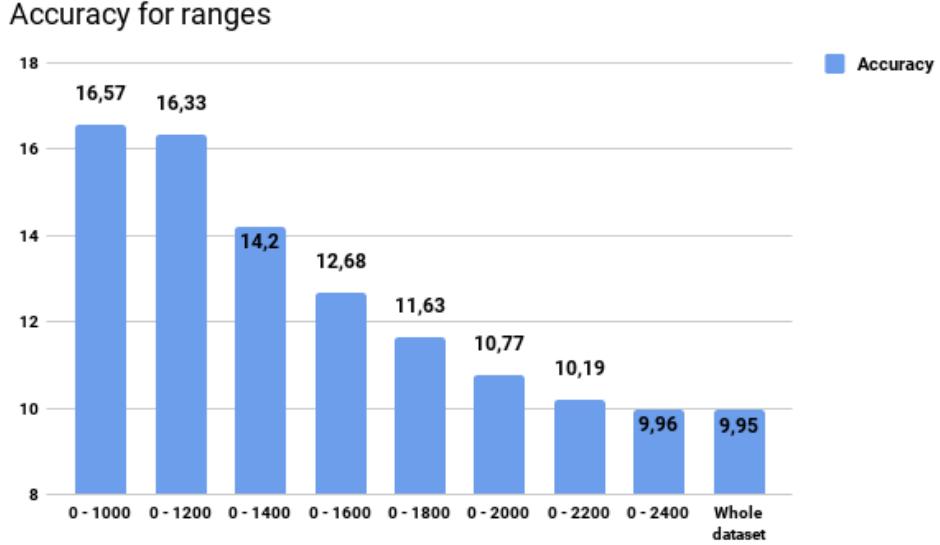


Figure 6.16: Accuracies obtained testing the network trained on RGB-D Object Dataset on the eight subsets of Autonomous Robot Indoor Dataset.

Now we are ready to discuss the results obtained applying domain adaptation algorithms to colorized depth input data. Table 6.5 and 6.6 show the unsupervised adaptation results of the transfer task ROD → ARID for both (1) and (2) groups of experiments. The parameter setting which led to these results, in both groups of experiments, is the following:

- DAN

AlexNet → lr: 0.001, bs: 64, epochs: 30, MK-MMD loss weight: 1

ResNet-50 → lr: 0.001, bs: 64, epochs: 30, MK-MMD loss weight: 0.3

- DANN

AlexNet → lr: 0.001, bs: 64, epochs: 30, Domain Classifier loss weight: 0.1

ResNet-50 → lr: 0.001, bs: 64, epochs: 30, Domain Classifier loss weight: 0.1

- AutoDIAL

AlexNet → lr: 0.0001, bs: 256, epochs: 30, Target loss weight (λ in (3.13)): 0.4

ResNet-50 → lr: 0.001, bs: 256, epochs: 30, Target loss weight (λ in (3.13)): 0.4

- ADDA

ResNet-50 → lr: 0.001, bs: 128, epochs: 30, Mapping & Adversarial loss weight: 1

Method	AlexNet	ResnNet-50
	ROD → ARID	ROD → ARID
Source only	0.10	0.112
DAN	0.144	0.203
DANN	0.148	0.182
AutoDIAL	0.172	0.157
ADDA	-	0.228

Table 6.5: ROD → ARID Depth experiments of group (1).

Method	AlexNet	ResnNet-50
	ROD → ARID	ROD → ARID
Source only	0.10	0.112
DAN	0.144	0.207
DANN	0.145	0.19
AutoDIAL	0.147	0.158
ADDA	-	0.209

Table 6.6: ROD → ARID Depth experiments of group (2).

Referring to the set (1) of experiments, with AlexNet as starting net, we can observe that AutoDIAL outperforms the other methods. Instead, the best result using ResNet-50 is reached with ADDA. The primate of these two algorithms persists also for the second group of experiments although the highest accuracies are reached with the first one. Tables 6.7 and 6.8 show the best results obtained using depth informations as input data for the domain adaptation algorithms applied on the task BigBIRD → Active Vision. For this domain shift the set of parameters is the following:

- DAN

AlexNet → lr: 0.001, bs: 64, epochs: 30, MK-MMD loss weight: 1

ResNet-50 → lr: 0.001, bs: 64, epochs: 30, MK-MMD loss weight: 0.7

- DANN

AlexNet → lr: 0.0001, bs: 64, epochs: 30, Domain Classifier loss weight: 0.1

ResNet-50 → lr: 0.001, bs: 128, epochs: 30, Domain Classifier loss weight: 0.1

- AutoDIAL

AlexNet → lr: 0.001, bs: 256, epochs: 30, Target loss weight (λ in (3.13)): 0.1

ResNet-50 → lr: 0.001, bs: 256, epochs: 30, Target loss weight (λ in (3.13)):

0.4

- ADDA

ResNet-50 → lr: 0.001, bs: 64, epochs: 30, Mapping & Adversarial loss

Method	AlexNet		ResnNet-50	
	BigBIRD → Active Vision			
Source only	0.041		0.055	
DAN	0.073		0.11	
DANN	0.067		0.094	
AutoDIAL	0.089		0.078	
ADDA	-		0.097	

Table 6.7: BigBIRD → Active Vision Depth experiments of group (1).

Method	AlexNet		ResnNet-50	
	BigBIRD → Active Vision			
Source only	0.041		0.055	
DAN	0.08		0.102	
DANN	0.07		0.098	
AutoDIAL	0.084		0.101	
ADDA	-		0.101	

Table 6.8: BigBIRD → Active Vision Depth experiments of group (2).

weight: 1

Also in this case, for AlexNet, the best results are obtained using AutoDIAL in both groups of experiments. With ResNet-50 as starting net, instead, the best accuracy is reached with DAN. For depth, as we had expected also for RGB, the highest value is reached by the first group of experiments. Being depth informations less representative of the objects with respect to RGB data, obviously it is obtained a higher result using this last information. It is interesting now to see what happens if both inputs are used in the adaptation process.

6.3.3 RGB-D

For the RGB-D experiments the high-level cue integration described in section 5.2 has been used. This approach is a starting point to see which is the best method to combine various image channels: in the deep domain adaptation scenario there are many other possible approaches to perform this combination, for example connecting both types of images (RGB and depth) at the CNN network level as [44] or [45]. The reason of the insertion of these experiments in the benchmark is to assess which is the improvement obtained adding informations, for instance about the shape, in an RGB object description. For RGB-D trials only experiments in which the whole target domain appears in both adaptation and testing phases are performed (group (1)). It was chosen to apply two domain adaptation algorithms for the task ROD → ARID using ResNet-50 as starting net: DAN and DANN (this last is the best for RGB experiments). It is good to remind that, for this set of experiments, depth images are colorized using the Surface Normal ++ technique and, for ARID dataset, only the depth images in which the number of non null pixels is greater than 75% are considered. Table 6.9 reports the results.

Method	ResnNet-50
	ROD → ARID
Source only	0.316
DAN	0.439
DANN	0.459

Table 6.9: ROD → ARID RGB-D experiments of group (1).

From the numbers in the table it can be notice that this type of combination of RGB and depth informations not produces improvements with respect to the results obtained with RGB only channel: the performance with DAN decreases by 0.01%, instead, with DANN, it remains the same. This could happen when the simplest way to combine data is used: the fact that the SVM replaces the last fully connected layer in ResNet-50 led to a lower accuracy. For this reason it is important to consider a different baseline to evaluate the adaptation improvements (the one reported in Table 6.9). As it can be notice the results for the adaptation task BigBIRD → Active Vision are missing. This is due to the not perfect overlap between RGB and depth images. The segmentation masks used for the crops (see paragraph *Datasets* of Section 6.2) are not well calibrated with depth images resulting in a not precise object bounding box. Figure 6.17 shows an explicit example in which the size of the cropped image is the same but in the



Figure 6.17: Example of the not perfect overlap between RGB and depth images in Big Berkeley Instance Recognition Dataset.

depth one the object is not well centered.

Chapter 7

Conclusions

To summarize, the following conclusions were reached: (i) Table 7.1 shows that the DA methods do not work on depth with the same effectiveness with which they work on RGB data, (ii) always from Table 7.1 it can be seen that the high level integration method used to combine RGB and depth produces a good improvement with respect to its baseline, but not with respect the accuracy obtained with the same experiment using only RGB as input data. (i) and (ii) prove that how deal with depth informations is still an open problem that needs specific research efforts to be used properly. (iii) AutoDIAL outperforms in every experiment (except one) which has AlexNet as starting network (see the first two columns of Table 7.2), (iv) DANN outperforms for the greatest part of the experiments which have ResNet-50 as starting network (see Table 7.2), (v) the accuracy obtained applying an adaptation algorithm on the task WOD → ARID is higher with respect to the one obtained applying the same algorithm to the shift ROD → ARID, (vi) the group of experiments that reaches an higher accuracy, in the case of RGB input data, is the one in which two different subsets of the target dataset are used for the adaptation and testing phases (group (2)) (see first row of Table 7.3), instead, (vii) for all experiments with only depth as input data the best accuracy is reached with the group that uses the same dataset for both adaptation and testing phases (group (1)) (see second row Table 7.3), (viii) the model trained with ROD depth images produces a better performance if it is tested on the ARID depth images in which the objects are within a meter from the camera (see Table 7.4).

In this work it has been presented a new benchmark valubles for the robot vision community in the object recognition field. In particular, the core of its contribution is in the investigation of domain adaptation with multimodal input data. The novelty is mostly in the use of depth: a lot of domain adaptation

studies were conducted on RGB images but not on depth and even less on RGB combined with depth. Some off-the-shelf unsupervised domain adaptation algorithms, together with datasets suited for robotic purposes, have been used. The experiments conducted shows that domain alignment, especially in the depth space, is still an open problem that needs more research efforts to be used properly.

Group of experiments	AlexNet		ResNet-50	
	ROD → ARID	BB → AV	ROD → ARID	BB → AV
RGB (1)	+8.7% (37.8%)	+25.7% (52.1%)	+12.2% (45.9%)	+17.6% (53.7%)
RGB (2)	+6.5% (35.6%)	+29.4% (55.8%)	+13% (46.7%)	+18.1% (54.2%)
Depth (1)	+7.2% (17.2%)	+4.8% (8.9%)	+11.6% (22.8%)	+5.5% (11%)
Depth (2)	+4.7% (14.7%)	+4.3% (8.4%)	+9.7% (20.9%)	+4.7% (10.2%)
RGB-D				+14.3% (45.9%)

Table 7.1: Best improvement reached through domain adaptation algorithms for each RGB only, depth only and RGB-D task. In brackets also the accuracy reached is shown.

Group of experiments	AlexNet		ResNet-50	
	ROD → ARID	BB → AV	ROD → ARID	BB → AV
RGB (1)	AutoDIAL	AutoDIAL	DANN	DANN
RGB (2)	DANN	AutoDIAL	AutoDIAL	DANN
Depth (1)	AutoDIAL	AutoDIAL	ADDA	DAN
Depth (2)	AutoDIAL	AutoDIAL	ADDA	DAN

Table 7.2: Best algorithm for each RGB only and depth only experiment.

Input modality	AlexNet		ResNet-50	
	ROD → ARID	BB → AV	ROD → ARID	BB → AV
RGB	(1)	(2)	(2)	(2)
Depth	(1)	(1)	(1)	(1)

Table 7.3: Group of experiments with which the best result is obtained.

Range (mm)	0-1000	0-1200	0-1400	0-1600	0-1800	0-2000	0-2200	0-2400	All
Accuracy	0.166	0.163	0.142	0.127	0.116	0.108	0.102	0.0996	0.0995

Table 7.4: Accuracies obtained using as test set progressive subsets (in terms of camera - object distance) of Autonomous Robot Indoor Dataset.

Bibliography

- [1] URL: <https://laughingsquid.com/honda-unveils-latest-incredibly-sophisticated-asimo-humanoid-robot/>.
- [2] URL: <http://www.nextme.it/tecnologia/robotica/1351-asimo-il-primo-robot-umanoide-10-anni-dopo-le-nuove-sfide-di-honda>.
- [3] URL: <https://www.irenteverything.com/rentals/black-coffee-cup/>.
- [4] URL: <https://www.target.com/p/coupe-mint-coffee-mug-room-essentials-153/-/A-17471568>.
- [5] URL: <https://www.flickr.com/photos/jacobrtabor/8119035527>.
- [6] M. Long and J. Wang. “Learning transferable features with deep adaptation networks”. In: *ICML* (2015). arXiv: 1502.02791.
- [7] F.M. Carlucci, L. Porzi, B. Caputo, E. Ricci, and S. Rota Bulo. “Auto-dial: Automatic domain alignment ‘ layers’”. In: *International Conference on Computer Vision (ICCV)* (2017). arXiv: 1505.07818.
- [8] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. “Domain-adversarial training of neural networks”. In: (2015). arXiv: 1505.07818.
- [9] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. “Adversarial discriminative domain adaptation”. In: *WAT* (2016). arXiv: 1505.07818.
- [10] Zhengyou Zhang. “Microsoft kinect sensor and its effect”. In: *IEEE multi-media* 19.2 (2012), pp. 4–10.
- [11] URL: <http://talkingaboutme.tistory.com/84>.
- [12] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. “A large-scale hierarchical multi-view RGB-D object dataset.” In: *ICRA*. IEEE, 2011, pp. 1817–1824. URL: <http://dblp.uni-trier.de/db/conf/icra/icra2011.html#LaiBRF11>.

- [13] R.L. Mohammad, B. Caputo, and V. Markus. “Recognizing Objects In-the-wild: Where Do We Stand?” In: (2017). arXiv: 1709.05862.
- [14] A. Singh, J. Sha, K.S. Narayan, T. Achim, and P. Abbeel. “BigBIRD: A Large-Scale 3D Database of Object Instances”. In: *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)* (2014), pp. 509–516.
- [15] P. Ammirato, P. Poirson, E. Park, and A. Berg. “A Dataset for Developing and Benchmarking Active Vision”. In: *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)* (2017).
- [16] John Blitzer, Ryan McDonald, and Fernando Pereira. “Domain adaptation with structural correspondence learning”. In: *Proceedings of the 2006 conference on empirical methods in natural language processing*. Association for Computational Linguistics. 2006, pp. 120–128.
- [17] Lorenzo Bruzzone and Mattia Marconcini. “Domain adaptation problems: A DASVM classification technique and a circular validation strategy”. In: *IEEE transactions on pattern analysis and machine intelligence* 32.5 (2010), pp. 770–787.
- [18] Pascal Germain, Amaury Habrard, François Laviolette, and Emilie Morvant. “A PAC-Bayesian approach for domain adaptation with specialization to linear classifiers”. In: *International Conference on Machine Learning*. 2013, pp. 738–746.
- [19] Xingyu Zeng, Wanli Ouyang, Meng Wang, and Xiaogang Wang. “Deep learning of scene-specific classifier for pedestrian detection”. In: *European Conference on Computer Vision*. Springer. 2014, pp. 472–487.
- [20] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. “Geodesic flow kernel for unsupervised domain adaptation”. In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE. 2012, pp. 2066–2073.
- [21] Mingsheng Long, Guiguang Ding, Jianmin Wang, Jiaguang Sun, Yuchen Guo, and S Yu Philip. “Transfer sparse coding for robust image representation”. In: *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE. 2013, pp. 407–414.
- [22] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. “Simultaneous deep transfer across domains and tasks”. In: *Computer Vision (ICCV), 2015 IEEE International Conference on*. IEEE. 2015, pp. 4068–4076.

- [23] Yaroslav Ganin and Victor Lempitsky. “Unsupervised domain adaptation by backpropagation”. In: *arXiv preprint arXiv:1409.7495* (2014).
- [24] Joaquin Quiñonero-Candela, Masashi Sugiyama, Anton Schwaighofer, and N Lawrence. *Covariate shift and local learning by distribution matching*. 2008.
- [25] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. “Unsupervised domain adaptation with residual transfer networks”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 136–144.
- [26] Baochen Sun, Jiashi Feng, and Kate Saenko. “Return of Frustringly Easy Domain Adaptation.” In: *AAAI*. Vol. 6. 7. 2016, p. 8.
- [27] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. “Generative adversarial nets”. In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.
- [28] Kevin Lai and Dieter Fox. “3D laser scan classification using web data and domain adaptation.” In: *Robotics: Science and Systems*. Vol. 2. 2009.
- [29] Kevin Lai and Dieter Fox. “Object recognition in 3D point clouds using web data and domain adaptation”. In: *The International Journal of Robotics Research* 29.8 (2010), pp. 1019–1037.
- [30] Google. *3D Warehouse*. 2008. URL: <http://sketchup.google.com/3dwarehouse/>.
- [31] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. “Deep domain confusion: Maximizing for domain invariance”. In: *arXiv preprint arXiv:1412.3474* (2014).
- [32] Novi Patricia, Fabio M Carlucci, and Barbara Caputo. “Deep Depth Domain Adaptation: A Case Study”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 2645–2650.
- [33] Wikipedia. *Normal Mapping*. 2008. URL: https://en.wikipedia.org/wiki/Normal_mapping.
- [34] URL: [https://en.wikipedia.org/wiki/Normal_\(geometry\)#/media/File:Surface_normal_illustration.svg](https://en.wikipedia.org/wiki/Normal_(geometry)#/media/File:Surface_normal_illustration.svg).
- [35] URL: <http://willould.co.uk/relighting-part-02-maths/>.

- [36] Andreas Aakerberg, Kamal Nasrollahi, and T Heder. “Depth value pre-processing for accurate transfer learning based rgb-d object recognition”. In: *Under Review-IJCCI* (2017).
- [37] Jie Luo. “Open-ended Learning of Visual and Multi-modal Patterns”. In: (2011).
- [38] URL: <https://www.sciencedirect.com/science/article/pii/S0167865508000974>.
- [39] Jia Deng, Wei Dong, Richard Socher, Li-jia Li, Kai Li, and Li Fei-fei. “Imagenet: A large-scale hierarchical image database”. In: *In CVPR*. 2009.
- [40] N. Massouh, F. Babiloni, T. Tommasi, J. Young, N. Hawes, and B. Caputo. “Learning Deep Visual Object Models From Noisy Web Data: How to Make it Work”. In: *ArXiv e-prints* (Feb. 2017). eprint: 1702.08513.
- [41] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.
- [42] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.
- [43] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. “Imagenet: A large-scale hierarchical image database”. In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE. 2009, pp. 248–255.
- [44] Andreas Eitel, Jost Tobias Springenberg, Luciano Spinello, Martin Riedmiller, and Wolfram Burgard. “Multimodal deep learning for robust RGB-D object recognition”. In: *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE. 2015, pp. 681–687.
- [45] Judy Hoffman, Saurabh Gupta, Jian Leong, Sergio Guadarrama, and Trevor Darrell. “Cross-modal adaptation for RGB-D detection”. In: *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE. 2016, pp. 5032–5039.