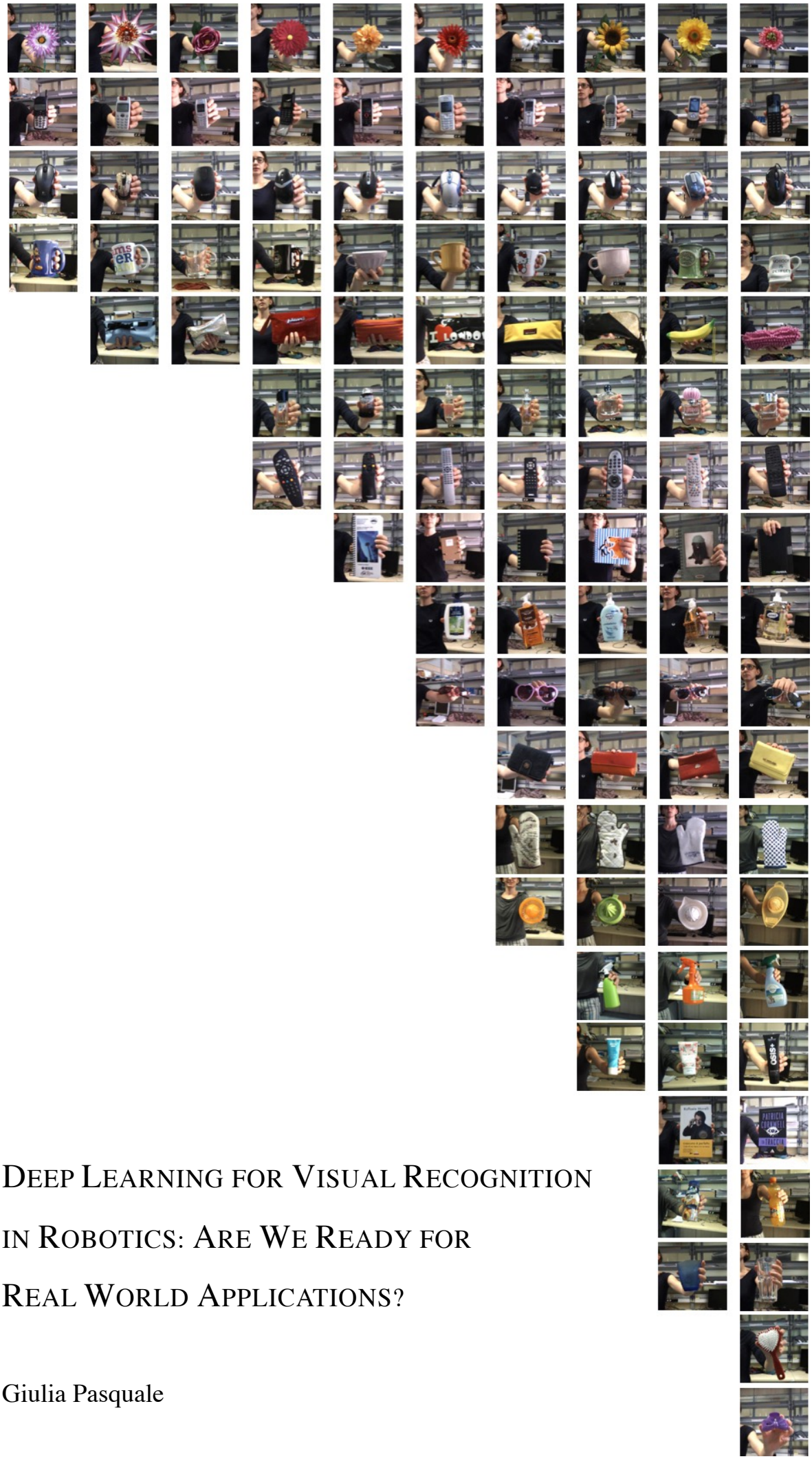

Deep Learning for Visual Recognition in Robotics: Are We Ready for Real World Applications?

Giulia Pasquale

UNIVERSITÀ
DEGLI STUDI
DI GENOVA

ISTITUTO
ITALIANO DI
TECNOLOGIA

DEEP LEARNING FOR VISUAL RECOGNITION IN ROBOTICS: ARE WE READY FOR REAL WORLD APPLICATIONS?

by

Giulia Pasquale

A thesis submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Doctoral Course in Bioengineering and Robotics
XXIX Cycle
Curriculum in Humanoid Robotics

Advisors:
Prof. Lorenzo Natale
Prof. Lorenzo Rosasco

iCub Facility
Istituto Italiano di Tecnologia (IIT)

Laboratory for Computational and Statistical Learning
Istituto Italiano di Tecnologia (IIT)

Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi (DIBRIS)
Università degli Studi di Genova (UNIGE)

April 2017

Abstract

Artificial intelligence has been recently revamped largely because of deep learning methods. Computational vision, specifically object recognition, is the first and perhaps the most clear example where deep learning allowed for unprecedented improvement, reporting ground breaking performances in multiple applications.

Robotics is a main field where the benefits of such progress could be dramatic, since the lack of reliable visual recognition systems is largely considered a major bottle neck for the deployment of robots in everyday life. Indeed, the problem of visual recognition poses remarkable issues, particularly related to real world scenarios [Pinto et al., 2008], which are not easy to tackle in robotic settings. However, deep learning methods did boost the performance levels of artificial systems, such that realizing working solutions for real applications has lately become a feasible goal and there are already promising results. But how close are we to "solving" the problem? What are the major benefits of employing deep learning for robot vision and how can we best leverage them? Are there limitations to this approach, originating new challenges?

To address such questions, in this work we considered a prototypical robotic scenario, where a humanoid robot is taught to recognize different objects by a human through natural interaction. This scenario is representative of several robot vision contexts and is significantly different from classical web-scale image classification problems in that, firstly, there is a reasonably restricted number of objects/categories to discriminate. On the other hand, these are unknown beforehand and the robot is required to learn them from limited examples. Secondly, objects must be recognized reliably in unpredictable configurations and often under wide visual nuisances, which are underrepresented in most computer vision and robotics datasets. We investigated therefore how current deep learning systems, namely deep Convolutional Neural Networks (CNNs), can be best adapted to tackle this scenario.

To this end, we designed a semi-controlled acquisition setting and collected image datasets by recording the robot's view while observing objects shown by the human. The latest of these benchmarks is large in terms of number of objects and categories, and is also segmented into multiple sets of image sequences, representing specific visual transformations like scale,

rotations, and so forth. It provides a unique combination that, to our knowledge, was missing in the literature, allowing for articulated analyses of the invariance and robustness properties of recognition systems within the real world visual experience of a robot.

Provided with a rich and challenging data collection, we performed an extensive evaluation of latest CNN models, simulating increasingly more difficult object categorization and identification tasks, by varying the number of example instances and views, and testing under different conditions. Moreover, we leveraged the structure of our novel dataset to quantify the effects of the presence of specific viewpoint transformations when training and testing CNNs. Notably, this study led us to identify a strategy to improve the invariance properties of the internal representations of CNN models to such transformations. Indeed, by feeding the obtained robust representations to standard classifiers, we could achieve remarkable results in challenging object identification tasks using very few example images. Along the way, we also extended the proposed recognition pipeline to include incrementally a growing number of examples, possibly learning new classes, by updating the classifier in constant time, instead of re-training from scratch.

Overall, this study confirms a huge impact of deep learning on robot vision contexts, while measuring the gap with large-scale web domains. We could devise a fast, reliable and adaptive recognition system, based on invariant deep representations and efficient classifiers, that incredibly boosted the recognition capabilities of two real robotic platforms: the iCub and R1 humanoids. Indeed, by exploiting the proposed pipeline, it is now possible to effectively and quickly teach a high number of novel objects to the robots “on the fly”, in the considered human-robot interaction scenario. Finally, our analysis also points at specific open challenges that need to be addressed for seamless deployment in real world robotic applications.

Contents

Abstract i

1 Introduction 1
- 1.1 Towards Autonomous Humanoids . . . 1
- 1.2 The Problem of Visual Recognition . . . 3
  - 1.2.1 Short "Taxonomy" of Visual Tasks . . . 4
  - 1.2.2 Appearance Variability and the Need of Learning . . . 6
- 1.3 Approaches to Object Recognition . . . 9
  - 1.3.1 Computer Vision . . . 10
  - 1.3.2 Robotics . . . 13
- 1.4 Objectives and Contributions of the Thesis . . . 17
- 1.5 Outline of the Thesis . . . 20
- 1.6 Publication Note . . . 21

2 Machine Learning Setting 23
- 2.1 Supervised Learning Problem . . . 23
  - 2.1.1 Introduction to Statistical Learning Theory . . . 24
  - 2.1.2 Optimal Bayes Classifier and Surrogate Loss Functions . . . 26
  - 2.1.3 The Hypothesis Space . . . 28
  - 2.1.4 Regularization and Model Selection . . . 30
  - 2.1.5 Regularized Least Squares for Classification . . . 31
- 2.2 Stochastic Gradient Descent . . . 32
  - 2.2.1 Gradient Descent . . . 33
  - 2.2.2 Stochastic Gradient Descent . . . 34
  - 2.2.3 Mini-batch Stochastic Gradient Descent . . . 35

3 Deep Convolutional Neural Networks 37
- 3.1 From Pixels to Labels: Building Invariance and Selectivity . . . 37

3.1.1 Local Invariance in Hand-crafted Descriptors . . . 39
3.1.2 In Search of Global Invariance: Dictionary Learning and Pooling . . 39
3.1.3 End-to-end Learning . . . 41
3.2 Basic Architecture . . . 42
3.2.1 Neural Networks . . . 42
3.2.2 Convolutional Neural Networks . . . 44
3.3 The Deep Learning (Re)Evolution . . . 50
3.3.1 AlexNet . . . 51
3.3.2 Beyond AlexNet: ILSVRC 2012-15 . . . 52

4 Learning and Transferring Deep Visual Representations 57
4.1 Backpropagation . . . 57
4.1.1 Sketch of the Algorithm . . . 58
4.1.2 Best Practices . . . 61
4.2 Understanding Deep Image Representations . . . 63
4.2.1 Visual Transformations, Manifolds and Sample Complexity . . . 64
4.2.2 Empirical Investigation of Representation Properties . . . 65
4.3 Transferring Deep Representations . . . 69
4.3.1 Feature Extraction and Kernel Methods . . . 69
4.3.2 Fine-tuning . . . 71

5 The iCubWorld Project 75
5.1 Introduction . . . 75
5.2 Related Work and Project Goal . . . 76
5.2.1 Datasets for Visual Object Recognition . . . 76
5.2.2 iCubWorld . . . 78
5.3 Acquisition Setup . . . 80
5.3.1 Acquisition Scenario and First Releases . . . 80
5.3.2 New Acquisition Setup . . . 82
5.4 iCubWorld28 . . . 84
5.4.1 Datasets for Incremental Learning . . . 84
5.4.2 Dataset Overview . . . 87
5.5 iCubWorld Transformations . . . 88
5.5.1 Datasets for Invariance . . . 88
5.5.2 Dataset Overview . . . 89

6 Are we Done with Object Recognition? The iCub's Perspective. 95

6.1 Introduction . . . 95
6.1.1 Related Work . . . 97
6.2 Methods . . . 99
6.2.1 CNN Architectures . . . 99
6.2.2 Feature Extraction . . . 100
6.2.3 Fine-tuning . . . 100
6.3 Results (Object Categorization) . . . 101
6.3.1 Deep Learning and (the Need for) Knowledge Transfer . . . 101
6.3.2 Do we need more data? . . . 104
6.4 Results (Object Identification) . . . 108
6.4.1 Knowledge Transfer: from Categorization to Identification . . . 109
6.5 Generalization Across Days . . . 110
6.6 Conclusion . . . 113
Appendices . . . 114
6.A Off-the-shelf models and Dataset Bias . . . 114
6.A.1 Image Preprocessing . . . 114
6.A.2 1000-class Categorization Results . . . 115
6.A.3 Viewpoint Biases . . . 115
6.B Model (Pre) Selection . . . 118
6.B.1 Feature Extraction and RLSC . . . 118
6.B.2 Fine-tuning . . . 120
6.C More Experiments on Categorization . . . 126
6.D Comparison with Washington RGB-D . . . 128
6.D.1 Object Categorization Benchmark . . . 129
6.D.2 Object Identification Benchmark . . . 130

7 Object Identification from Few Examples by Improving the Invariance of Deep CNNs 133
7.1 Introduction . . . 133
7.1.1 Methods and Related Work . . . 134
7.2 Evaluating Viewpoint Invariance of Off-the-shelf CNNs . . . 136
7.2.1 Object Categorization . . . 136
7.2.2 Object Identification . . . 140
7.3 Improving Viewpoint Invariance of Off-the-shelf CNNs . . . 143
7.4 One-shot Object Identification on Washington RGB-D . . . 147
7.4.1 Comparison . . . 148
7.4.2 Discussion . . . 150

7.5 Conclusion . . . 151

8 Incremental Learning of New Objects with Fixed Update Time 153
8.1 Introduction . . . 153
8.1.1 Related Work . . . 155
8.2 Dealing with Class Imbalance . . . 156
8.3 Incremental RLSC . . . 160
8.4 Incremental RLSC with Class Extension and Recoding . . . 161
8.4.1 Class Extension . . . 161
8.4.2 Incremental Recoding . . . 162
8.5 Experiments . . . 163
8.5.1 Experimental Protocol . . . 163
8.5.2 Model Selection . . . 165
8.5.3 Results . . . 166
8.6 Conclusion . . . 167

9 Robotic Applications: Putting it All Together 171
9.1 Robot Platforms . . . 171
9.1.1 iCub . . . 172
9.1.2 R1 . . . 173
9.2 Overview . . . 175
9.3 iCubWorld Framework . . . 176
9.3.1 Acquisition . . . 176
9.3.2 Training . . . 180
9.4 On the Fly Object Recognition . . . 180
9.4.1 Representation Extraction . . . 182
9.4.2 Classification . . . 184
9.5 Conclusions . . . 184
Appendices . . . 185
9.A Robots Details . . . 185
9.A.1 iCub . . . 185
9.A.2 R1 . . . 186

10 Future Directions 189

Bibliography 193

List of Figures

1.1 Future scenario . . . 2
1.2 Snapshots from the ON THE FLY RECOGNITION demo . . . 3
1.3 What and Where cortical pathways . . . 4
1.4 Recognition and localization visual tasks . . . 4
1.5 Image classification example . . . 7
1.6 Example of WordNet tree . . . 8

2.1 Example of Feature Map . . . 29

3.1 Selectivity and invariance of image representations . . . 38
3.2 Dictionary Learning pipeline . . . 40
3.3 Linear classifier . . . 42
3.4 Neural Network . . . 43
3.5 From NNs to CNNs . . . 45
3.6 Convolutional Neural Network . . . 48
3.7 The four CNN models employed in this Thesis . . . 53
3.8 ILSVRC Top-5 Error Rate Decrease in latest years . . . 54

4.1 The two knowledge transfer approaches considered in this Thesis . . . 70

5.1 ICUBWORLD acquisition scenario . . . 81
5.2 HELLO ICUBWORLD example images . . . 82
5.3 GROCERIES example images . . . 83
5.4 ICUBWORLD28 example images . . . 87
5.5 ICWT visual transformations . . . 90
5.6 ICWT categories . . . 91
5.7 ICWT semantic variability . . . 92
5.8 ICWT 200 objects . . . 93

6.1 Performance of off-the-shelf networks on ICWT or on ImageNet . . . 102

6.2 Recognition accuracy vs # frames . . . 105
6.3 Recognition accuracy vs # instances . . . 106
6.4 Different training regimes from robot vision and image retrieval settings . . . 107
6.5 Recognition accuracy vs # frames (Identification) . . . 108
6.6 Regognition accuracy vs # frames - Generalization across days . . . 110
6.7 Recognition accuracy vs # instances - Generalization across days . . . 111
6.8 Identification accuracy vs # frames - Generalization across days . . . 112
6.A.1 Performance of off-the-shelf networks tested on ICWT with different object cropping 116
6.A.2 Performance of off-the-shelf networks on ICWT or on ImageNet – 1000 classes . 117
6.A.3 Frame-by-frame predictions of *GoogLeNet* on the *Scale* transformation . . . 118
6.A.4 Frame-by-frame predictions of *GoogLeNet* on *2D Rotation* . . . 118
6.B.1 Model selection for RLSC over the large experiment . . . 120
6.B.2 Model selection for RLSC over the small experiment . . . 121
6.B.3 Model selection for fine-tuning of *CaffeNet* . . . 124
6.B.4 Model selection for fine-tuning of *GoogLeNet* . . . 125
6.C.1 Recognition accuracy vs # frames – 3 objects per category . . . 127
6.C.2 Recognition accuracy vs # instances – 10-class categorization . . . 127
6.C.3 Recognition accuracy vs # instances – 5-class categorization . . . 128

7.1 Generalization performance across transformations - RLSC . . . 137
7.2 Generalization performance across transformations – fine-tuning . . . 138
7.3 Generalization performance across transformations (Identification) – RLSC . . . 140
7.4 Generalization performance across transformations (Identification) – fine-tuning . 141
7.5 Generalization performance across visual transformations – tuned representations 145
7.6 Generalization performance across visual transformations (Identification) – tuned representations . . . 146

8.1 Bayes decision boundaries . . . 158
8.2 Classification accuracy of the proposed method with varying α . . . 165
8.3 Classification accuracy of incremental RLSC and the proposed variant . . . 168

9.1 iCub . . . 172
9.2 R1 . . . 174
9.3 ICUBWORLD acquisition application . . . 177
9.4 Depth-based object localization pipeline . . . 178
9.5 ICUBWORLD directory tree . . . 179
9.6 ON THE FLY OBJECT RECOGNITION application . . . 181
9.A.1 iCub's skin . . . 185

List of Tables

5.1 HELLO ICUBWORLD overview 82
5.2 GROCERIES overview 83
5.3 ICUBWORLD28 overview 86
5.4 ICWT OVERVIEW 91

6.1 Feature extraction layers for the four CNNs considered in this work 99
6.2 Fine-tuning protocols for *CaffeNet* and *GoogLeNet* 101
6.A.1 Preprocessing operations executed on images before feeding the networks 115
6.D.1 Categorization on Washington RGB-D – RLSC 129
6.D.2 Categorization on Washington RGB-D – fine-tuning 129
6.D.3 Identification on Washington RGB-D – RLSC 131
6.D.4 Identification on Washington RGB-D – fine-tuning 131

7.1 One-shot identification on Washington RGB-D 149

8.1 Performance of Naïve RLSC, Rebalanced and Recoding approaches 167

9.1 `caffeCoder` and `GIECoder` comparison 181

Chapter 1

Introduction

In this Chapter, we introduce the problem of visual recognition and its relevance to robotics (Sec. 1.1 and Sec. 1.2). We then consider the main differences between typical computer vision settings and robotic domains, by reviewing the main approaches adopted in both fields to address this problem (Sec. 1.3).

We conclude by motivating this work in light of recent advances in machine learning and providing a short overview of the contributions of this Thesis (Sec. 1.4). Finally, Sec. 1.5 outlines the content of following Chapters and Sec. 1.6 provides a list of publications where the works described in this Thesis originally appeared.

1.1 Towards Autonomous Humanoids

Over the last decades there has been a progressive shift in robotic systems. From industrial robots, supposed to operate in protected and fixed environments like production sites, repeating a limited set of tasks with very high precision and accuracy, we are moving towards service robots, that are expected to act in unconstrained environments and interact with people in everyday life. In this respect, there has been an increasing research interest in cognitive robotics and in the realization of platforms that are in principle able to *learn* and perform a wide range of tasks, rather than being specifically designed for a single specific purpose.

Within this perspective, perception and in particular vision is progressively playing a fundamental role in robotics. Indeed, the majority of tasks, as navigation, reaching for objects, grasping, manipulation, interaction with other agents, critically depends on the ability to visually *localize* and *recognize* in the scene the entities involved. Unfortunately, these are extremely complex tasks to be performed in real world scenarios. As a consequence, the difficulty of providing machines with reliable visual skills is largely considered a main bottle

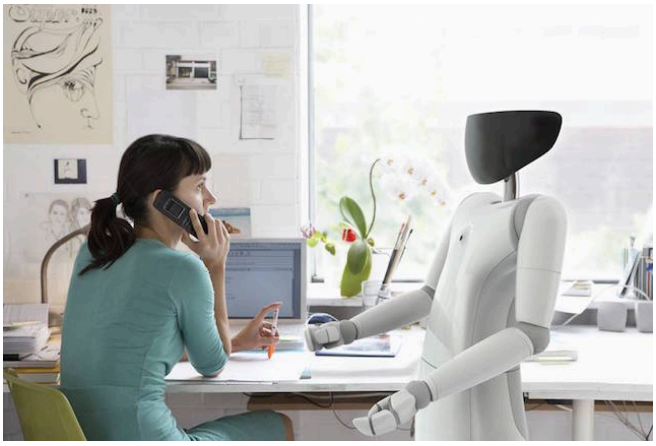

Figure 1.1: **Future scenario:** the R1 robot collaborating with people in a work or home environment.

neck towards the deployment of autonomous agents for concrete applications [Kemp et al., 2007, Corke, 2011].

Specifically, replicating humans' accurate, robust and quickly learnable visual recognition capabilities has always represented a great challenge. Indeed, as it will be described in detail in next Chapters, visually recognizing objects and entities poses remarkable issues, particularly related to the real world context [Pinto et al., 2008], which are not easy to tackle in robotic settings. In this regard, in Sec. 1.3 we will briefly review how this problem has historically been investigated in the computer vision and robotics literature, through an undeniable progress over the last decades that, nevertheless, did not manage to provide sufficiently performing systems as such to be exploited by robot agents to plan their actions.

The recent breakthrough of deep learning methods, on the other hand, is revolutionizing the approach to visual recognition problems, reporting stunning results in multiple applications. These methods incredibly boosted the performance levels and the expectations that we have from artificial systems, such that realizing working solutions for usage in robotics has lately become a feasible goal. But to which extent can we consider the visual recognition problem "solved"? What are the major benefits of employing deep learning in robotics and how to leverage them in order to realize the foreseen scenario in the near future? Are there critical limitations to the deep learning approach, originating new challenges?

As we will discuss in Sec. 1.4, this Thesis addresses these questions with an in-depth analysis of the performance of deep learning architectures in typical robotic settings, with the long-term motivation of providing humanoid robots with autonomous visual recognition skills for deployment in real world applications. Towards this end, as a short-term goal, our analysis led us to devise effective solutions to compose a reliable, fast and adaptive visual recognition

Figure 1.2: **Snapshots from the** ON THE FLY RECOGNITION **demo** developed within the Thesis. The robot is able to quickly learn the appearance of novel objects through a natural interaction with humans, recognizing them reliably in unconstrained environments. Details on the demo are provided in Chapter 9; (Left) GPU Technology Conference, Amsterdam; (Right) Maiker Faire, Rome.

system that allows robots to learn to recognize objects "on the fly", by interacting with humans in a natural unconstrained setting. Indeed, as we will see in Chapter 9, the works part of this Thesis have been developed on two humanoid robots, namely the iCub research platform and R1, prototype of general-purpose assistant (represented in Fig. 1.1), whose recognition capabilities have been remarkably advanced thanks to the proposed contributions (see Fig. 1.2).

1.2 The Problem of Visual Recognition

In this Section we introduce the visual recognition problems addressed in the Thesis, by framing them into a brief computational neuroscience and computer vision background.

What and Where. The phenomenon of visual perception in humans involves several elaboration phases taking place in different anatomical parts, starting from the eyes up to multiple cortical areas, with the final goal of extracting meaningful information from the light impinging photoreceptors in the retina. Early studies on primates [Schneider, 1969, Mishkin et al., 1983, Desimone and Ungerleider, 1989] and humans [Ungerleider and Haxby, 1994] suggested that the two tasks of recognition and localization are carried on through two separate cortical pathways, branching off from a common elaboration stage at the level of the primary and secondary visual cortexes. After a first low-level processing, often called *early vision*, the visual signal is projected to (i) the parietal pathway, or "Where", dorsal stream, that elaborates the information necessary to the localization of visual stimuli, and to (ii) the temporal pathway, or

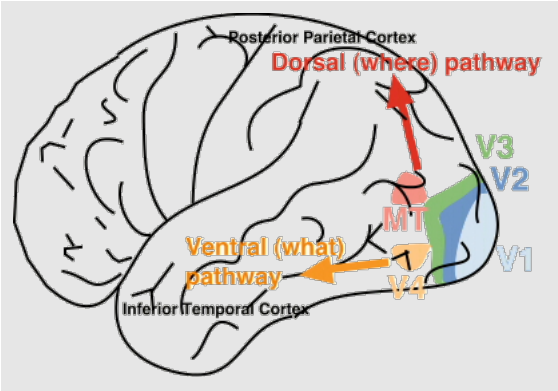

Figure 1.3: **What and Where** cortical pathways, mainly dedicated to, respectively, recognize and localize elements in the visual scene. Image from [a].

[a] http://www.uwosh.edu/departments/psychology/Vreven/Lab/Images/brain.jpg

Figure 1.4: **Overview of the main visual tasks** comprising visual recognition and localization of objects in images.

"What", ventral stream, which instead extracts the information needed to stimuli identification (see Fig. 1.3). Recently, more attention has been paid to possible interactions between the dorsal and ventral streams [Cloutman, 2013, van Polanen and Davare, 2015]. Nevertheless, the dual-stream theory is today well accepted, many computational vision models, addressing the recognition task independently from localization, have been proposed (see, e.g. [Riesenhuber and Poggio, 1999, Serre et al., 2007, Mutch and Lowe, 2008]), and there is evidence that two tasks can be solved separately [Milner and Goodale, 2006].

1.2.1 Short "Taxonomy" of Visual Tasks

Accordingly with biology, the common "taxonomy" of visual tasks in computer and machine vision distinguishes between solving only recognition or also localization of elements (see

Fig. 1.4). In this setting, the visual input is actually an image, a matrix of pixels produced by some digital camera, like the ones mounted on the iCub or R1 humanoids (for details on the robot platforms see Sec. 9.1).

Object Recognition. Object *recognition* is usually referred to as the operation of assigning a (semantic) label to an element depicted in an image. Under the assumption that a predominant part of the image is occupied by a single entity or object, the full image can be associated with a single label and no spatial information is needed (Fig. 1.4, Left). Often this task is also called image – or object – *classification*.

Object recognition consists in two fundamental tasks. The first one is the recognition objects that are known, when these are presented in "new" conditions (e.g., under a different light, viewpoint, or in a different context, etc.). This is usually referred to as object *identification*, or *instance* recognition. Another recognition task is instead object *categorization*; this is the ability to learn the general appearance common to all objects belonging to a specific category, and to associate the correct category label once a new, unseen, object instance belonging to it, is presented. Identification and categorization capabilities are fundamental for humans and robots and in this Thesis we will consider both.

Object Localization. In case the object occupies a small portion of an image, or there are multiple objects in the image, it becomes necessary to specify the spatial region within the image the recognition label refers to. It is therefore required to solve also a *localization* task (Fig. 1.4, Right). There are different flavours of object localization, providing coarser to finer information about objects' occupancy in the image. It is possible for example to just specify a rectangular region (or bounding box) enclosing the recognized object, or to more precisely trace its shape. The first task is usually referred to as object *localization* or *detection*, depending on whether a single or multiple objects are to be (recognized and) localized in an image. The second task instead is called object *segmentation* or also *image/scene parsing*. See Fig. 1.4 for a pictorial of the different tasks.

Localizing and Recognizing Objects in Images. In practice, apart from ad hoc computer vision datasets, images always contain more than a single object and position information is always required, particularly when dealing with visual data recorded from robotic cameras. Indeed, incoming frames in this case represent wide scenes, where objects must be localized before they can be recognized . Taking inspiration from *bottom-up* and *top-down* mechanisms through which humans visually attend specific elements in the visual field [Itti, 2000], we can identify two approaches to localize *and* recognize objects in images.

The most common and largely adopted strategy is to first extract from an image one or more "salient" regions, i.e., candidate Regions Of Interest (ROIs) selected according to some measure of "objectness", which, based on some visual cue (like color), may contain elements to be recognized. Indeed, these salient regions are then processed for recognition. Techniques to improve the recognition output can be finally applied, e.g., discarding regions with low recognition confidence, or aggregating multiple regions based on their predictions (see, e.g., [Girshick et al., 2014, Sermanet et al., 2014] and references therein). Many algorithms have been proposed in the computer vision literature to extract salient regions (see, e.g., [Vaillant et al., 1994, Endres and Hoiem, 2010, Alexe et al., 2012, Uijlings et al., 2013]). These usually apply a processing to the image in order to produce objectness maps by color or edge patterns. In robotics, the real world setting offers a great deal of contextual information, that can be exploited for the detection of salient regions. For instance, the 3D structure of the scene or the temporal correlation between frames can be used to segment the depth map [Pasquale et al., 2016b] or the optical flow [Ciliberto et al., 2011, Ciliberto et al., 2012, Kumar et al., 2015] in order to extract blobs potentially corresponding to objects to be recognized. In these cases, the pipeline is said to be *bottom-up*, since the extracted ROIs and predictions are agnostic with respect to their semantic content.

Another approach consists in exploiting also high-level semantic information (like, e.g., the information coming from a previous bottom-up recognition step, or prior information about the objects to be recognized), to improve, in turn, the localization and recognition performance with *top-down* modulations (see, e.g., the recent work [Shrivastava et al., 2016]).

In this work, we adopt the most common bottom-up strategy and specifically focus on the core task of object recognition. As we will explain in more detail in Chapter 5, we rely on contextual information available in the robotic scenario (specifically, the depth of the scene) in order to isolate a ROI in the image, whose content is then recognized.

1.2.2 Appearance Variability and the Need of Learning

In this Section we introduce how the object recognition problem is commonly defined and treated in machine vision settings. We have presented in Sec. 1.2.1 this task as the problem of assigning a label to an image depicting an object. We have divided between object identification, where the label identifies a specific object instance, and object categorization, where the label identifies a category of objects. In both cases, we are addressing a classification task, i.e., assigning the same label to images belonging to the same class (i.e., depicting the same object instance –identification– or object instances belonging to the same category –categorization–), and a different one otherwise.

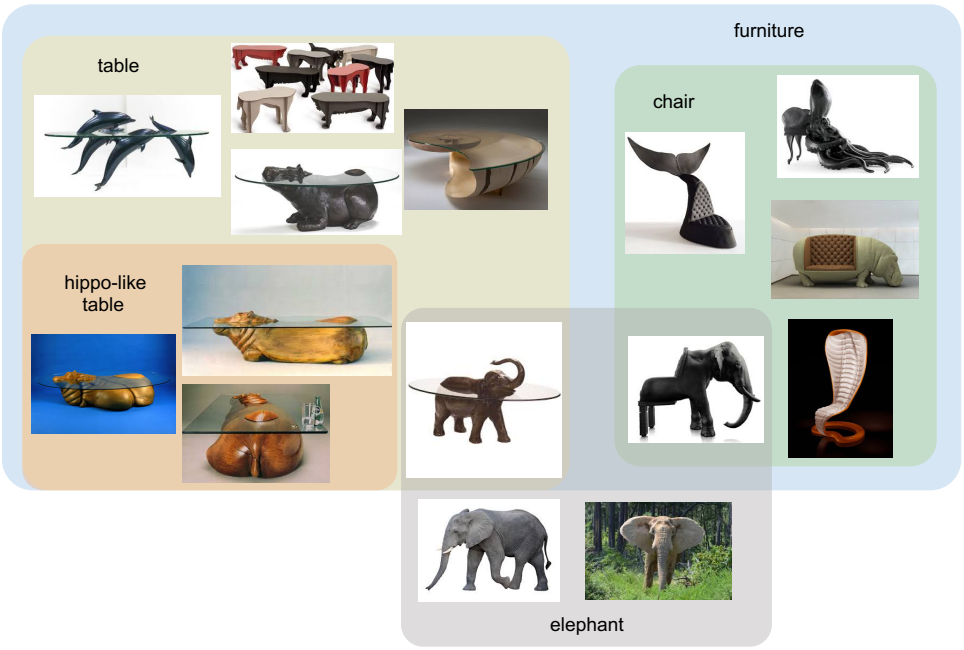

Figure 1.5: **Example of images divided into classes** (*furniture*, *table*, *hippo-like table*, *chair*, *elephant*). While at the instance level (e.g., in the *hippo-like table* class) image variability is in viewpoint and context changes, at the category level the images depict multiple, different, instances, and their variability depends also on how the category is defined.

Appearance Variability. The main challenge involved in this task is related to the extreme variability of objects' appearance in images, which depends on the viewpoint, light conditions, occlusions, and so forth (see, e.g., the different pictures of the *hippo-like table* in Fig. 1.5). Moreover, the appearance of different object instances belonging to a same category is even more different. In particular, this latter variability depends on the definition of category that it is assumed. For instance, *tables* and *chairs* may be considered as a single *furniture* category (in which case the intra-class variability would be higher); on the other hand, each of the two categories may also be further divided into more specific sub-categories (e.g. wooden, plastic, etc...), and so forth. In fact, on the one hand, all object instances belonging to a category must share certain characteristics, which provide them a "similar" visual appearance. On the other hand, however, the definition of object categories generally depends on the problem/context, and it is influenced by assumptions on objects' semantic, psychological, philosophical or functional meaning (e.g., elephant-like chairs or tables in Fig. 1.5 can be considered both as instances of the *elephant* and of the *chair*/*table* category).

On the Definition of Visual Categories. In practice, a solution commonly adopted is to rely

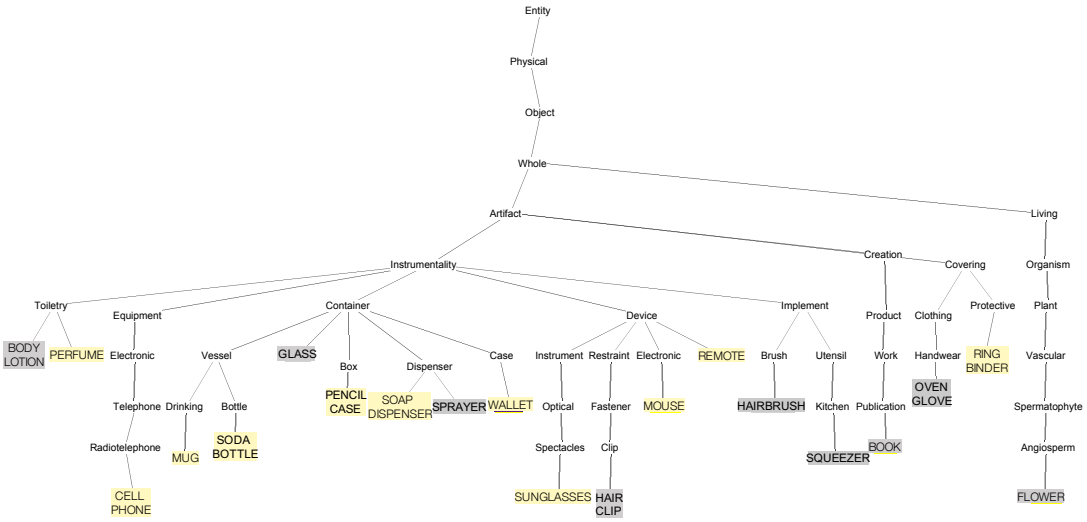

Figure 1.6: **Example of a tree extracted from WordNet's hierarchy.** The 20 leaf synsets are the categories which comprise the ICUBWORLD TRANSFORMATIONS dataset that will be introduced in Sec. 5.5. Yellow categories are also represented in the ImageNet Large-Scale Visual Recognition Challenge classification task.

on standard ontologies as, for instance, the WordNet hierarchy [Miller, 1995][1]. WordNet is a large lexical database for English (but there are versions in many other languages), where nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (called *synonym sets* or *synsets*). Each synset contains a brief definition ("gloss") and short example sentences illustrating the use of the synset members. Word forms with multiple distinct meanings are represented as many distinct synsets, therefore each form-meaning pair is unique and each synset expresses a unique concept. Synsets are interlinked by conceptual relations, in a network of meaningfully related words and concepts. The most frequent relation among synsets is the super-subordinate relation (also called IS-A relation), which links more general synsets (like *furniture*) to increasingly specific ones (like *bed* and *bunkbed*). The part-whole is another common hierarchical relation (which holds e.g. between *chair* and *leg*). Noun hierarchies originate trees. All trees start from the same root node *entity* and end up in several leaf nodes. An example of such a tree is depicted in Fig. 1.6.

The WordNet ontology is commonly used to define object categories in image classification settings. For example, ImageNet is a large-scale dataset of more than 14M images organized accordingly with WordNet's hierarchy. The images have been collected on the web and annotated with object labels and bounding boxes by means of crowdsourcing services

[1] http://wordnet.princeton.edu

like the Amazon Mechanical Turk [2]. The labels are the WordNet identifier (synset ID) of the depicted objects, in such a way that the organization of image categories in ImageNet (more than $\sim$ 20k) closely resembles the one of corresponding synset IDs in WordNet. There are also other datasets following WordNet's organization, like Washington RGB-D [Lai et al., 2011] and the SUN [Xiao et al., 2010] databases.

Image Classification Setting. As it may be evident, the large variability of objects appearance causes the raw pixel values of images belonging to the same class to be possibly very different, and makes it difficult to devise algorithms able to "scratch off" all variations from the images in order to "extract" the identity of the depicted object or category. Moreover, it is desirable to have an algorithm which is in principle able to tackle *any* image classification tasks, independently on the particular objects/categories involved. To this end, as we will extensively see in Chapter 2, this problem has historically been addressed with machine learning methods for classification, like Support Vector Machines (SVM) [Vapnik, 1998] or Regularized Least Squares for Classification (RLSC) [Bishop, 2006] algorithms. In this setting, a set of example images is provided (called *training set*), and the algorithm *learns* from the data to predict the label of new, unseen images. In the common classification scenario (which is also the one considered in this Thesis), each image can belong to one class, a number of classes is considered, for each class a number of examples is provided and the algorithm is trained to predict the class which new images belong to.

For instance, in the annual international competition known as the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC), the object categorization task includes 1000 classes, chosen from WordNet's leaf synsets (considering only nouns). The training set consists in $\sim$ 1.2M images and additional images are collected each year for testing the submitted algorithms. Throughout the years, more tasks have been included in the challenge as object localization and detection (also in videos) and scene parsing (see [Russakovsky et al., 2015] for details). Today, ILSVRC is one of the standard benchmarks for recognition algorithms and, as we will see in more detail in Sec. 5.2, similarly to ImageNet, many other public datasets for visual object recognition are available in the literature.

1.3 Approaches to Object Recognition

We already mentioned in Sec. 1.2.1 one important difference between typical computer vision and robotics settings, namely the availability of contextual information. In this Section we

[2] https://www.mturk.com/mturk/welcome

briefly review how this and other differences historically influenced approaches to object recognition problems in both domains.

Understanding and reproducing humans' ability to recognize objects has always been one of the most investigated topics in neuroscience and computer vision literature. Indeed, vision is undoubtedly the sensory channel most involved in this process and in past decades an extremely rich body of literature addressed the problem of how our brain extracts relevant information from images and how to devise working algorithms to artificially replicate this ability. The first comprehensive computational theories on the cortex and the ventral stream were proposed in the 1980-90s (see [Marr, 1982, Poggio et al., 1985] and [Edelman and Poggio, 1991, Riesenhuber and Poggio, 1999]), while early works in computer vision date back to the 1960s [Hu, 1962, Roberts, 1963].

While the processing of visual information is surely fundamental for recognition, it is not the only mechanism involved. Indeed, it is well understood in cognitive science that object recognition is the result of a multi-modal perceptual process, involving active acquisition of not only visual, but also haptic, tactile and even sound information. Clearly, the robotic setting offers a particularly suitable testbed for these studies. In fact, partially motivated by the remarkable difficulty of devising accurate and robust visual recognition systems for robotic settings (as we will see in Sec. 1.3.1 and, with more detail, in the following Chapters), the exploitation and integration of multiple sensory modalities, in order to improve robots' recognition capabilities, became in past decades a main research line in robotics.

At the same time, active research in computer vision and machine learning progressively advanced visual recognition systems. In recent years, a breakthrough in deep learning boosted their performance to unprecedented levels. Some recognition tasks, before perceived as challenging, have started to be instead considered "solved" (e.g. digit [LeCun et al., 1998] and text recognition, car and pedestrian detection, but also several image classification benchmarks [Griffin et al., 2007, Everingham et al., 2015]), such that deep learning methods have started to be employed also in robotics – with encouraging results.

In Sec. 1.3.1 we provide a brief overview of typical vision-based approaches to the problem of object recognition. We will then consider how the same problem has been addressed in robotics (Sec. 1.3.2). We will see how both field have been recently remarkably influenced by the introduction of deep learning methods.

1.3.1 Computer Vision

In Sec. 1.2.2 we explained how the task of visual object recognition can be seen as a problem of image classification, to be addressed with machine learning methods like SVMs [Vapnik, 1998] or RLSC [Bishop, 2006].

In practice, applying classifiers directly to image pixels is usually not sufficient to discriminate between images representing different objects/categories, since the intra-class variability is way too large. To this end, approaches to extract suited *representations* from images have been subject of active research for decades and have been at the core of the problem of image classification. A "good" image representation (with respect to a certain recognition task) is a function mapping images into vectors that are sensitive to the image content relevant to the task, while being *invariant* to non relevant aspects. For the case of object identification, for instance, usually image representations need to be invariant to objects' rotations, translations, pose, light, background, while being selective for the different objects. While here we briefly review the main approaches proposed in the literature, the problem of representation learning will be treated more in depth in Sec. 3.1.

Hand-designed Local Descriptors. Initial approaches were based on the (manual) design of algorithms to detect local patterns (e.g., small edges, blobs, corners) robustly under different geometric or light transformations. The core idea was to first *encode* a small pixel neighborood into the responses of designed transformed templates (like, e.g., scaled Difference of Gaussians [Marĉelja, 1980] or oriented Gabor filters [Marr and Hildreth, 1980]). Such responses were then *pooled* (e.g., averaged) in order to obtain descriptors of the pattern in the neighborood which were robust to such transformations (like scaling or rotation). Well-known approaches are the Scale Invariant Feature Transform (SIFT) descriptor [Lowe, 2004], the Histogram of Oriented Gradients (HOG) descriptor [Dalal and Triggs, 2005], Haar wavelets [Viola and Jones, 2001] – see [Mikolajczyk and Schmid, 2005] for an overview. So-called *keypoint-matching* techniques were based on (i) the selection of a number of *keypoints* in an image and (ii) the matching of their local descriptors with the ones of the keypoints found in other images. These methods were mostly employed for object identification tasks, since they relied on the robust association of specific reference points.

Dictionary Learning. It became soon evident that in order to tackle more complex tasks like, e.g., object categorization, more powerful descriptors needed to be *learned* from data rather than being manually designed. To this end, one of the most famous approaches proposed was the Bag Of (Visual) Words (BOWs) algorithm [Csurka et al., 2004], which adds a second *coding-pooling* stage on top of local descriptors, where the templates used in the coding stage (that is, the so-called *dictionary*) are learned from data. The BOWs algorithm (that we will see in more detail in Sec. 3.1) takes inspiration from a typical method for document classification. Through these two coding-pooling stages each image is encoded into a vector representative of its global content (that is, a global image representation). Standard classifiers are finally

trained to solve the recognition task in the representation space.

(Bioinspired) Hierarchical Representations. The coding-pooling computational paradigm was actually well known to neuroscientists, since the visual cortex was known to process the visual signal in a very similar way [Riesenhuber and Poggio, 1999]. Indeed, bioinspired computational models of visual recognition – among which a well-known is HMAX [Serre et al., 2007]– were based on the extraction of a global image representation through the composition of multiple coding-pooling stages (where the templates used for coding were learned from data). Similarly to dictionary learning pipelines, standard classifiers were trained afterwards on the extracted image representations to perform the desired recognition task. With a similar inspiration, hierarchical architectures for visual recognition were actually proposed since early 1980s: Fukushima's neocognitron [Fukushima, 1980], alternating layers of (i) convolutions with filters and (ii) spatial pooling/subsampling, can be considered the first multi-layer Convolutional Neural Network (CNN).

End-to-end Learning and Deep Learning Breakthrough. In the same years, [LeCun et al., 1989] proposed for the first time a CNN architecture that, by reducing at each layer the spatial resolution of the image representation, while increasing the number of the employed filters/templates, progressively encoded images into vector representations and finally directly into classification predictions. Exploiting this end-to-end homogeneous structure, LeCun et al. were able to learn the filters in all layers by (Stochastic) Gradient Descent, minimizing the prediction error and back-propagating it though the network. Their claim was that learning the representation with the supervision of the classification task could provide better performance than using hand-designed, or unsupervisedly learned, features. Unfortunately, this training approach did not prove to be very effective in practice, at the time, and therefore BOW-like pipelines continued to be largely employed.

On the contrary, in recent years, the increased data availability – in the form of large web-collected datasets like the ImageNet [Russakovsky et al., 2015] – and computational power – particularly with parallel accelerators as Graphic Processing Units – allowed to train end-to-end CNN architectures with more layers (i.e., deeper) on millions of example images, achieving unexpected good performance and definitely surpassing classical BOW-like object recognition pipelines. Starting from 2012, when [Krizhevsky et al., 2012] proposed the AlexNet model and marked a breakthrough on the reference 1000-class categorization benchmark represented by the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), these architectures attracted a lot of attention in the machine learning and computer vision communities, and the deep learning literature prospered with new CNN models [Sermanet et al., 2014, Simonyan

et al., 2014, Szegedy et al., 2015, He et al., 2016] and new (often commercial) applications.

The key ingredients of their success, i.e., the deep, hierarchical, structure, and the purely data-driven approach, as opposed to so-called "shallow" methods employing only two layers of representation, one of which hand-designed, are subject of active research. Specifically, one of the main reasons for their large employment is that, while typically requiring large-scale datasets to be learned, it has been observed that, once trained (e.g., on ImageNet), these models can be "adapted" to a variety of domains, generalizing particularly well and remarkably improving the state of the art on many challenging visual recognition tasks [Chatfield et al., 2014, Sharif Razavian et al., 2014] (see Sec. 4.3 and references therein). While visual object recognition is the first and perhaps the most clear example where deep learning allowed for unprecedented advances, leading to ground breaking performances and sometimes even surpassing human levels [Netzer et al., 2011, Stallkamp et al., 2012, He et al., 2015, Assael et al., 2016], deep learning already brought improvements in many applications related to machine learning and computer vision, like reinforcement learning [Mnih et al., 2013, Mnih et al., 2015, Silver et al., 2016], speech recognition and natural language processing, autonomous driving, but also finance and trading, recommender systems, medicine, art, etc..

1.3.2 Robotics

While the progress in computer vision and machine learning through last decades has focused on learning representations that allow to recognize objects in images, in robotics this problem has rarely been approached by relying only on objects' images. Even methods using solely visual cues typically exploited the real world context to create 3D object models. In fact, recognition in robotics has often been associated with pose estimation, functional to, e.g., object grasping. Indeed, until recently robots were expected to operate in relatively predictable and constrained working spaces. The goal was hence rather to improve manipulation capabilities by exploiting knowledge of the object's shape, provided, in fact, by a correct object identification.

Today, we foresee robots to operate in unconstrained environments, where the problem of rapidly recognizing objects from the visual input becomes more critical. Robots should heavily rely on this capability, prior to others, in order to understand the scene content, navigate and plan their actions. Moreover, if before visual recognition algorithms were hardly usable in real world robotic settings and the contribution of different sensory modalities was considered one of the main avenues of performance improvements, today this may not exactly by valid anymore. If, on the one hand, it remains true that multi-modal integration is a hallmark of human learning, robotics is a main field where the benefits of deep learning could be dramatic. In the following, we provide a quick overview of typical approaches to object recognition in robotics, including recent works using deep learning.

Visual Approaches

We start by considering works that addressed the problem by relying on visual cues, applying and adapting computer vision methods to the robotic setting.

Object Identification by Keypoint-matching. Inspired by keypoint-matching approaches in computer vision, in robotics many methods were proposed, sharing the common strategy of (i) extracting keypoints from object images and (ii) matching them through, e.g., robust outliers rejection schemes like RANSAC or Hough Voting, to ensure geometric consistency [Fischler and Bolles, 1981]. However, 2D keypoints [Lowe, 2004, Bay et al., 2006] were mostly suited to recognize textured, planar objects [Moreels and Perona, 2007], while 3D keypoints [Steder et al., 2011, Quadros et al., 2012] were often not sufficiently discriminative. With similar inspiration, template matching [Huttenlocher et al., 1993] has also been employed [Hinterstoisser et al., 2010, Hsiao and Hebert, 2014]. However, this latter approach required to exhaustively acquire many viewpoints during training.

Object Identification by Using $3D$ Scans. Other methods relied on the acquisition of 3D scans of objects. These scans could be combined to create a 3D model of the objects [Gordon and Lowe, 2006, Tang et al., 2012, Xie et al., 2013, Singh et al., 2014], that could be then fitted to object images for identification [Taylor and Kleeman, 2003, Ekvall et al., 2005], or used to train classifiers [Lai et al., 2011, Bo et al., 2013]. For instance, [Collet et al., 2009, Collet and Srinivasa, 2010, Collet et al., 2011] use interest points to register a set of images and hence reconstruct a 3D model for the object using a Structure from Motion approach; [Aleotti et al., 2012] perform clustering to derive 3D templates from point clouds; [Choi and Christensen, 2012] use instead edge templates, obtained by projecting polygonal mesh models. Indeed, over the years many algorithm to reconstruct 3D models from multiple views have been proposed (see, e.g, Iterative Closest Point [Huber and Hebert, 2003], Kinect Fusion [Newcombe et al., 2011], ReconstructMe, itSeez3D [Dimashova et al., 2013], CopyMe3D [Sturm et al., 2013] and, recently, [Prankl et al., 2015]). Other methods of similar inspiration work on 3D surfaces rather than image keypoints: these can have global/semi-global [Rusu et al., 2010, Aldoma et al., 2011, Aldoma et al., 2012b, Aldoma et al., 2013] or local approaches [Tombari et al., 2010, Papazov and Burschka, 2011, Aldoma et al., 2012a]. There are also methods merging color and surface cues [Muja et al., 2011, Hinterstoisser et al., 2011].

All these approaches share the same drawback of needing a quite long training phase, to be carried on in a constrained setting in order to ensure a correct model acquisition. Also, they are not suited for object categorization problems.

Dictionary Learning. As anticipated, there are very few works in robotics that apply dictionary learning approaches to perform visual object recognition. Among these, [Filliat, 2007, Botterill et al., 2008] propose the use of a BOW-like representation to perform room identification. BOW approaches have also been adopted to perform object identification based on tactile data [Schneider et al., 2009]. Recently, the performance of different dictionary learning methods was compared, also against a biologically inspired model like HMAX [Serre et al., 2007], in a natural human-robot interaction setting [Ciliberto et al., 2013]. The authors show how learning representation methods can be, in fact, not only faster to train, but also more robust, with respect to classical keypoint matching techniques, when performing object identification on mid/low resolution images affected by visual nuisances (like, e.g., light and viewpoint changes) typically characterizing the view of a robot in an indoor setting.

Beyond Vision

Like in humans, on robots as well visual perception is only one among the possible cues enabling object recognition. Indeed, robots, like humans, are equipped with multiple, progressively more accurate, sensory modalities. A large body of robotics literature addressed therefore the problem of object recognition by integrating modalities other than vision. For instance, haptic and tactile perception [Dahiya et al., 2010, Bartolozzi et al., 2016], proved to be effective to perform object identification tasks in absence of visual input [Schneider et al., 2009, Gorges and Navarro, 2010, Hosoda and Iwase, 2010, Jamali et al., 2016, Vezzani et al., 2016], or to address recognition tasks which are hard to accomplish visually [Chitta et al., 2011], or finally, to complement the visual information [Higy et al., 2016, Sung et al., 2016].

Also, we already anticipated how the real world scenario is a natural source of implicit supervisory signals that robots, like humans, can exploit in the learning process, in the form of, e.g., spatial [Doersch et al., 2015] or temporal [Wang and Gupta, 2015] contextual information. Moreover, it is well known that biological agents use physical interactions with the world as a source of supervision for learning perceptual representations. Active perception is fundamental for babies to acquire cognitive skills: they generate themselves "key" events on which they rely to learn to correlate different modalities (e.g. proprioception, motion, tactile, auditory perception). Such correlation is then used to learn, in turn, to segment and recognize objects and object properties. Similarly to babies, robots can generate actions with a known pattern so that proprioception can be correlated with visual input. Indeed, the role of embodiment in the generation of the sensory information that is then used to learn correlations among perceptual cues in the highly structured natural environment has been largely investigated [Gibson, 1979, Jeannerod, 1997, Natale et al., 2004, Metta et al., 2006b, Fitzpatrick et al., 2008, Leitner et al., 2014, Dansereau et al., 2016]. For instance, in [Sinapov et al., 2014], a robot learns to

recognize objects by performing different behaviors on them, while using vision, proprioception and audio to detect and correlate perceptual changes.

Lately, there are works employing deep Convolutional Neural Networks (CNNs) in this direction. These are also motivated by the fact that these architectures need lots of training data to learn good visual representations and annotation is a costly operation. Self-supervision strategies are therefore welcomed in order to reduce the human intervention. For instance, [Jayaraman and Grauman, 2015, Agrawal et al., 2015] use knowledge of the robot's ego-motion as supervison to correlate visual stimuli and learning a general deep visual representation; [Pinto and Gupta, 2016] devise an autonomous trial-and-error behavior to collect the data for training a deep CNN for the task of grasp detection; in [Pinto et al., 2016] this approach is further refined showing how a robot, by employing four types of physical interactions to collect hundreds of thousands of data-points, is able to learn visual representations effective for object recognition tasks.

Deep Learning for Robotics

Deep learning methods are receiving growing attention in robotics. CNNs trained on ImageNet have been effectively adapted to robotics benchmarks for object recognition, like the Washington RGB-D dataset [Schwarz et al., 2015, Eitel et al., 2015, Held et al., 2016], and place recognition [Sünderhauf et al., 2015, Sünderhauf et al., 2016]. Moreover, the same models have also been adapted to perform other tasks like grasp detection [Lenz et al., 2015, Redmon and Angelova, 2015], affordance prediction [Nguyen et al., 2016] and even tactile material recognition [Baishya and Bäuml, 2016].

While, on the one hand, it is surely true that, as we observed, visual perception is not the only cue enabling object recognition, on the other hand current deep learning methods already proved to be able to successfully tackle typical robotic tasks, like reaching [Zhang et al., 2016, Zhang et al., 2016], grasp detection [Levine et al., 2016b] or manipulation [Levine et al., 2016a, Finn and Levine, 2016], with a strongly vision-based approach. For instance, in [Levine et al., 2016a] the authors replace the separate hand-engineered components for perception, state estimation, and low-level control of a robot arm with a single deep CNNs that maps directly raw image observations into torques at the robot's motors, which is learned by jointly training the perception and control systems end-to-end, with large self-supervisedly acquired data.

1.4 Objectives and Contributions of the Thesis

How Far Can We Go with Deep Learning? As it can be realized, the potential impact of deep learning for robotics is, indeed, remarkable. Specifically, current deep CNNs perform so well on the core task of visual object recognition, that it is natural to ask how well they can do, in robotics, before further perceptual cues/modalities are needed. This investigation is also critical to properly validate further approaches integrating and complementing the visual information with other mechanisms. With this perspective in mind, we have started an effort to isolate and quantify the benefits and limitations, if any, of deep learning approaches to visual object recognition in robot vision contexts.

Benchmarking the Prototypical Visual Experience of a Robot. To this end, we considered a "reference" scenario, where a robot (in our case, the iCub or, more likely, its commercial version R1) is deployed to a user (e.g., in a hospital, a hotel, a shopping centre), where it is required to learn to recognize novel objects, reliably, quickly, and with the natural supervision of humans. As a first step, we designed an acquisition setting, tailored to reflect the prototypical visual experience of the robot, during interaction with a human in the considered scenario. We therefore collected corresponding image datasets by recording the robot's view (in our case, we used the iCub), while it is observing objects shown by a human. In particular, we acquired two datasets of increasing size, namely ICUBWORLD28 and ICUBWORLD TRANSFORMATIONS, aimed at investigating complementary aspects of robotic visual recognition, as described in the following. Indeed, these two releases consistently expand the horizon of ICUBWORLD[3], a data acquisition project whose main goal is to benchmark and progressively develop the visual recognition capabilities of the iCub robot.

Semantic Vs Viewpoint/Context Variability. Provided with faithful benchmarks, we started to analyze the considered context. We first identified a main "gap" marking the difference between robot vision and typical deep learning application fields, related to the task "regime". Indeed, as we anticipated, first, a humanoid robot would not necessarily need to distinguish between a thousand object categories like in ImageNet – at present at least. We would be satisfied keeping categories of the order of, e.g., the object types that can be found in a specific indoor working setting like an office. On the other hand, a robot typically cannot have access to millions of object examples per category, as are available in ImageNet: we are in a different "regime", with fewer categories but also fewer object examples to learn from. This kind of variability is related to categorization problems and is referred to as "semantic", since it comes from how semantic categories are instantiated into different object instances. Second, while

[3] https://robotology.github.io/iCubWorld/

the number of involved objects is typically limited, at the same time, in real world scenarios, each object can be experienced under a remarkable amount of viewpoint/context variability.

A first contribution of this Thesis is therefore to investigate how modern CNNs, trained on web data collections, can be best adapted to this different "regime". To this end, we exploited the iCubWorld Transformations (iCWT) dataset, which we specifically acquired to include several object categories with many instances per category, for testing both categorization and identification tasks. Notably, the dataset is also segmented into multiple sets of sequences, representing objects shown by a human while undergoing isolated visual transformations, like scaling, rotations, light and background changes, hence allowing to test robustness and invariance of recognition systems within a realistic scenario. Indeed, this latter property makes the dataset more suited for robotics than standard computer vision benchmarks such as ImageNet – an argument shared with other recent works as [Borji et al., 2016].

We used iCWT to perform a battery of experiments and explore the robot vision "regime" by simulating increasingly larger recognition tasks with varying number of example objects and views available. We considered a representative pool of recently proposed CNN architectures trained on ImageNet and followed the common trend of adapting these models to other domains either by fine-tuning or by using the activations of their internal layers as features to be fed to classifiers. Interestingly, from this study we observed some unexpected results that, as we will discuss, must be attributed to the intrinsic characteristics of the robot vision domain. To this regard, we further deepened the analysis as explained in the following.

Learning from Few Example Frames. We focused on the critical aspect that the appearance of objects in the real world (and hence in iCWT) is affected by wide ranges of viewpoint transformations. This is particularly critical because not only the robot would be required to recognize objects reliably in almost every configuration, but it should also be able to learn novel objects "on the fly" from *few* example frames. Ensuring to rely on discriminative and invariant image representations is key to achieve satisfying performance in this setting. To this end, we investigated to which extent the performance reported by the considered CNN models in the previous study were impacted by the presence of specific visual nuisances. We care to point out that we could afford such an analysis by leveraging the structure of iCWT, where each visual transformation is isolated from others. We first evaluated the invariance properties of the internal representations of "off-the-shelf" CNN models trained on ImageNet and, then, we quantified how much these properties can be disrupted, but also improved, by fine-tuning the models on image sequences acquired in a real world scenario (like the ones present in iCWT). As a result of this analysis, we were able to devise an affective recognition pipeline which, by employing such improved representations to train Regularized Leas Squared Classifiers

(RLSC), allowed us to perform on the fly object identification from very few example images in the considered human-robot interaction scenario. Moreover, with the same pipeline we also achieved competitive results on the task of one-shot object identification on Washington RGB-D [Lai et al., 2011].

Learning Incrementally. Keeping the same scenario, we finally imagined that the robot would likely encounter the same objects several times, in different situations, and it may be useful to exploit new additional training evidence, when available, in order to reinforce the model of learned objects. To this regard, a main motivation of the ICUBWORLD28 dataset, comprising objects acquired in multiple sessions in different days, is to simulate such a lifelong learning setting. By exploiting this dataset, we started evaluating the benefit of adopting an incremental learning approach, based on training RLSC on top of deep representations, to enrich object models beyond a single training session.

Moreover, within the same incremental setting, we finally considered also the situation where the robot learns to discriminate between a growing number of objects and categories, while accumulating experience about the environment. Indeed, this is another important aspect, since it is not feasible to restrict the robot's knowledge to a set of objects/categories defined beforehand. To this end, we proposed an extension to the recursive RLSC formulation available in the literature, which allows to include in the model not only new examples, but also new classes, in constant update time. By proposing an effective solution to deal with the problem of class imbalance, which typically arises in incremental settings, we could achieve comparable and even higher performance than the batch RLSC counterpart, while being significantly faster.

A Fast and Reliable Visual Recognition System for iCub and R1. To make the scientific contributions of this Thesis readily usable, a set of software tools for visual recognition has been implemented and deployed on the iCub and R1 platforms, remarkably advancing the capabilities of the robots. A first important tool is the ICUBWORLD FRAMEWORK. This firstly comprises the data acquisition application that has been developed to collect the ICUBWORLD dataset releases by interacting with the robot. This application can be exploited by users to acquire in the same way large-scale image collections; indeed, a set of routines to automatize the post-processing of recorded images, with helper functions for training various CNN architectures on the acquired collections, are included in the framework, easing the manual effort of producing datasets and training deep CNNs on them. A second tool is the ON THE FLY RECOGNITION demo, which implements the considered scenario where a human incrementally teaches new objects to the robot "on the fly". With this application, the performance of the proposed recognition pipeline, based on RLSC trained on top of discriminative and invariant

fine-tuned representations, can be tested in a realistic setting.

Are we Done with Object Recognition? The Thesis is concluded with a discussion of the potential developments of the proposed approaches, highlighting remaining challenges and key avenues of improvements, from a pure machine learning perspective but also taking advantage of the robot platform. Indeed, bridging the gap towards the deployment of visual recognition systems for autonomous agents in real applications appears to be an exciting venue for future multidisciplinary research.

1.5 Outline of the Thesis

In **Chapter 2** we introduce the considered supervised machine learning setting, in the framework of the Statistical Learning Theory. This setting is common to all methods employed in the Thesis, from deep Convolutional Neural Networks (CNNs) to Regularized Least Squares Classifiers (RLSC). We then provide notions on Stochastic Gradient Descent (SGD) methods, which can be adopted to address the optimization problems involved by the learning setting.

In **Chapter 3** we introduce the fundamental concepts of learning image representations, through an overview of the approaches historically proposed in the computer vision literature to address this problem. We then explain the architecture of deep CNNs, finally presenting the latest CNN models that, starting from 2012, revolutionized the field. These are also the models used for the evaluations performed in the Thesis.

In **Chapter 4** we describe the algorithm with which these architectures can be trained, namely, by applying SGD and *backpropagating* derivatives through layers. We then discuss some methods recently proposed in the deep learning literature to investigate the properties of the representations learned by such models. These considerations will be useful in Chapter 7, when we will measure the invariance properties of deep CNNs. We finally introduce the idea of *transferring* CNN models trained on ImageNet to other domains, overviewing the literature of works that explore either fine-tuning networks, or using their intermediate representations to train classifiers (like RLSC). These are also the two tools that will be adopted in the experiments presented in following Chapters.

While in Chapters 2, 3, and 4 we provided background material and methods, in the following part we proceed by presenting the contributions of the Thesis.

In **Chapter 5** we present the ICUBWORLD project, justifying and describing the acquisition scenario. We then introduce the two ICUBWORLD28 and ICUBWORLD TRANSFORMATIONS

datasets, published within this Thesis and employed in next Chapters as benchmarks for, respectively, incremental and lifelong leaning (Chapter 8) and viewpoint invariance (Chapter 7).

Chapter 6 sets up the study of the application of modern deep CNNs to the considered robot vision setting, with an extensive evaluation of the best models recently proposed in the literature, fine-tuned with different approaches or used for feature extraction, on a battery of object categorization and identification tasks on ICWT, by varying number of categories, instances, views and tested conditions.

In **Chapter 7** the focus is restricted on the critical role of viewpoint variability, with the investigation and the improvement of the invariance of these models to the different visual transformations present in ICWT. The chapter is concluded with the validation of the proposed object identification pipeline on ICWT and also on the Washington RGB-D dataset.

Chapter 8 reports on the proposed extension of the recursive RLSC, which allows to add new examples and new classes in constant update time, while dealing with class imbalance. Experimental results on MNIST [LeCun et al., 1998], Washington RGB-D and ICUBWORLD28 validate the algorithm.

In **Chapter 9** we first describe the robot platforms on which the works presented within the Thesis have been developed, namely the iCub and R1 robots. Then, we present the ICUBWORLD FRAMEWORK and the ON THE FLY RECOGNITION demo.

Chapter 10 draws conclusions to the contributions reported throughout the Thesis, while proposing future directions for addressing the limitations of current systems.

1.6 Publication Note

Some ideas and figures have appeared previously in the following publications:

Scientific Journals

Pasquale G., Mar T., Ciliberto C., Rosasco L., Natale L., *Enabling depth-driven visual attention on the iCub humanoid robot: instructions for use and new perspectives.*, Frontiers in Robotics and AI, vol. 3(35), 2016.

Pasquale G., Ciliberto C., Odone F., Rosasco L., Natale L., *Are we Done with Object Recognition? The iCub-robot Perspective.* In preparation.

Peer-reviewed Conference Proceedings

Pasquale G., Ciliberto C., Odone F., Rosasco L., Natale L., *Teaching iCub to recognize objects using deep Convolutional Neural Networks*, Proceedings of The 4th Workshop on Machine

Learning for Interactive Systems, 2nd International Conference on Machine Learning (ICML), vol. 43, pp. 21-25, Lille, FR, July 2015.

Pasquale G., Ciliberto C., Rosasco L., Natale L., *Object Identification from Few Examples by Improving the Invariance of a Deep Convolutional Neural Network*, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, KO, October 2016.

Camoriano R.*, **Pasquale G.***, Ciliberto C., Natale L., Rosasco L., Metta G. *Incremental Robot Learning of New Objects with Fixed Update Time*, Proceedings of the IEEE/RSJ International Conference on Robotics and Automation (ICRA), Singapore, May 2016 (to appear).

Chapter 2

Machine Learning Setting

In this Chapter we provide an essential background on the machine learning setting in which the visual recognition problems considered in this Thesis are framed. After a brief introduction to the general setting of supervised learning (Sec. 2.1), we will overview a class of optimization approaches commonly adopted to address learning problems, namely Gradient Descent and Stochastic Gradient Descent methods (Sec. 2.2).

2.1 Supervised Learning Problem

Programming a machine to perform some task requires the knowledge of the "algorithm" that must be executed in order to complete it. However, several tasks cannot be easily decomposed into an ordered sequence of predefined operations, therefore this knowledge is not always available. For instance, referring to Sec. 1.2.2, the large variability of all possible images of two different object categories (in our example, *chairs* and *tables*) makes it difficult to formalize an algorithm to discriminate between the two categories. On the other hand, in these cases it is often relatively easy to collect examples of the desired behavior (i.e., images of chairs and of tables). Indeed, humans are very good at learning by example and by experience. Machine learning is a field of computer science that was born with the aim to make computers learn "by example", as humans can do.

Instead of being programmed with a predefined function, the computer is programmed to *learn* such function from data. This is usually achieved by (i) parametrizing the function into multiple candidate solutions or "hypotheses" and (ii) using empirical evidence (i.e., a dataset of input-output examples) to assign the best values to the function's parameters in order to perform the desired task (an operation known as "training").

More precisely, we can formalize the problem as that of finding an underlying input-output relation $f(x) \sim y$ between two quantities x and y, which allows to predict, for any new input

x_{new}, its correct corresponding output $f(x_{new}) \sim y_{true}$. In this work we consider the setting of *supervised* learning, where, to learn the function, a set (called *training set*) of n input-output pairs is available. Each of these pairs is an example of the relation we want to learn, i.e., $S = \{x_i, y_i\}_{i=1}^n$, each x_i being a point for which we are provided the true output y_i. Other settings as *unsupervised* or *semi-supervised* learning consider situations where only some or none of the correct outputs of the training points are known.

In the following, we briefly introduce the supervised learning problem in the framework of the Statistical Learning Theory (STL) for classification. We follow an intuitive approach taking inspiration from [Rosasco, 2016] and refer to [Vapnik, 1998, Steinwart and Christmann, 2008] for a more formal and detailed introduction to STL.

2.1.1 Introduction to Statistical Learning Theory

We assume that the inputs of our problem belong to an input space $\mathcal{X}$ and the outputs to an output space $\mathcal{Y}$. The space $\mathcal{X}$ naturally depends on the domain of the problem. Here we consider the most common situation where $\mathcal{X} \subseteq \mathbb{R}^d$, i.e., the inputs are mapped into d-dimensional vectors in the Euclidean space. The space $\mathcal{Y}$ can be defined according to the task: e.g., $\mathcal{Y} \subseteq \mathbb{R}$ in regression, $\mathcal{Y} = \{-1, 1\}$ or $\mathcal{Y} = \{0, 1\}$ binary classification, $\mathcal{Y} = \{1, \dots, T\}$ in multiclass classification (where T is the number of classes to discriminate), and so forth.

The joint space $\mathcal{X} \times \mathcal{Y}$ is the data space, where the points are distributed according to a probability distribution (which is unknown) $\rho(x, y)$, which models the input-output relation that we aim to learn (generally affected by different sources of uncertainty). Considering that, for the Bayes rule, $\rho(x, y) = \rho_{\mathcal{X}}(x)\rho(y|x)$, the conditional distribution $\rho(y|x)$ accounts for the noise in the input-output relation and the marginal distribution $\rho_{\mathcal{X}}(x)$ models the distribution of the input points. In the classical setting of supervised learning, the examples in S are assumed to be sampled independently from ρ.

In order to estimate the "best" input-output relation, we can start fixing a *loss function* $\ell : \mathcal{Y} \times \mathcal{Y} \to [0, \infty)$, as a (point-wise) measure of the error $\ell(y, f(x))$ we incur in when predicting $f(x)$ in place of y. The "best" input-output relation hence will be a *target function* $f^* : \mathcal{X} \to \mathcal{Y}$, minimizing the *expected error* (or *expected risk*):

$$f^* = \inf_{f \in \mathbb{H}} \mathcal{E}(f) \tag{2.1}$$

$$\mathcal{E}(f) = \mathbb{E}[\ell(y, f(x))] = \int_{\mathcal{X} \times \mathcal{Y}} \ell(y, f(x))\rho(x, y)dxdy, \tag{2.2}$$

which can be seen as a measure of the error on all possible data. Note that $\mathbb{H}$ is the *hypothesis space*, that is, the space of functions on which we search f. This space should be such that computations are feasible and, at the same time, it should be *rich* enough with respect to the complexity of the problem, which is not known a-priori.

Unfortunately, the target function cannot be computed using the formulation above, since ρ is unknown and only the points in S are available. To this end, a well-established strategy is to minimize the *empirical error* (or *empirical risk*):

$$\hat{\mathcal{E}}(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f(x_i)), \tag{2.3}$$

as a proxy for the expected error $\mathcal{E}(f)$. The goal of a learning algorithm therefore is to use a training set S to find an estimator $\hat{f}$ for f^*, which behaves *similarly*. In this case, the algorithm is said to have good *generalization* properties, since it learned, from S, to correctly predict all possible new samples. Note that $\hat{f}$ is a random variable (it depends on the training set S, which is randomly sampled from ρ). Therefore, the goal of a learning algorithm is to minimize $\mathbb{E}_S[\mathcal{E}(\hat{f}) - \mathcal{E}(f^*)]$.

In other words, a good learning algorithm should be able to describe well the data and at the same time be stable with respect to noise and sampling. Indeed, the examples in S are only a part of the whole data distribution, which may contain the true trend we aim to learn, but also noise. Hence, an estimator $\hat{f}$ varying a lot depending on the sample S, would *overfit* the training set and typically would be unstable, exhibiting very low or zero error on the provided training examples, but yet a large error on future data (i.e., it would fail to *generalize*).

It is then evident that, in practice, computing

$$\hat{f} = \inf_{f \in \mathbb{H}} \hat{\mathcal{E}}(f) \tag{2.4}$$

cannot provide the desired estimator, since it leads to overfit the training set. To this end, *regularization* terms are usually introduced, in order to control the trade-off between data-fitting and stability. In machine learning, *regularizing* indicates the general practice to avoid being too "faithful" to the training set, ensuring stability and hence improving the generalization properties of the estimator.

A large class of methods, sometimes called Regularization Networks, employs the so called Tikhonov regularization scheme:

$$\hat{f} = \inf_{f \in \mathbb{H}} \Big(\hat{\mathcal{E}}(f) + \lambda R(f) \Big) \tag{2.5}$$

based on the idea of adding a term $R(f)$ (called *regularizer*), aimed to contrast the minimization of the empirical risk, ultimately controlling the stability of the solution. The parameter $\lambda > 0$ is the regularization parameter and balances the mixing of the two terms. It can be shown [Steinwart and Christmann, 2008, Shawe-Taylor and Cristianini, 2004a] that, under mild assumptions on the distribution ρ, it is possible for $\hat{f}$ to converge in probability to the ideal f^* as the number of training points n grows indefinitely. Many different classes of methods can be framed into this general formulation, by choosing a loss function, a kind of regularization term and, of course, fixing the hypothesis space.

2.1.2 Optimal Bayes Classifier and Surrogate Loss Functions

Many different types of loss functions have been proposed in the literature, depending on the task to be learned and the output space $\mathcal{Y}$ (e.g., classification, regression, etc...). As explained in Sec. 1.2, in this work we focus on classification problems. In this regard, a very natural choice for measuring the accuracy of a classifier is simply to count the number of prediction errors. This corresponds to using the so called misclassification loss or 0-1 *loss*. Referring for example to a binary classification problem, we can consider $\mathcal{Y} = \{-1, 1\}$ and define this loss as $\ell(y, f(x)) = \mathbf{1}(y - f(x))$ (where $\mathbf{1}(\cdot)$ is the indicator function). In this case, the problem in (2.1)-(2.2) becomes:

$$b^* = \inf_{f \in \mathbb{H}} \int_{\mathcal{X} \times \{-1,1\}} \mathbf{1}(y - f(x)) d\rho(x, y). \tag{2.6}$$

The solution b^* can be shown to satisfy the equation

$$b^*(x) = \begin{cases} 1 & \text{if } \rho(1|x) > \rho(-1|x) \\ -1 & \text{otherwise} \end{cases} \tag{2.7}$$

for all $x \in \mathcal{X}$ and it is known as the *optimal Bayes classifier*.

Unfortunately, estimating $\rho(y|x)$ is unfeasible because it requires very large training datasets. To this end, we have seen that, in practice, an estimate of b^* can be computed by minimizing the *regularized empirical risk* (Eq. (2.5)).

However, the 0-1 loss is a non-convex function and, as we will see in more detail in Sec. 2.2, this makes the optimization involved in Eq. (2.5) very hard. For this reason, convex relaxations of the 0-1 loss (known as *surrogate* losses) are often used instead. The most common, for instance, are:

- the *hinge loss* $\ell(y, f(x)) = \|1 - yf(x)\|_+ = max(1 - yf(x), 0)$, used in Support Vector Machines (SVMs);
- the *logistic loss* $\ell(y, f(x)) = \log(1 + e^{-yf(x)})$ used in Logistic Regression (LR);
- the *square loss* $\ell(y, f(x)) = (y - f(x))^2$ used in the Regularized Least Squares for Classification (RLSC) algorithm.

Indeed, it has been shown [Steinwart and Christmann, 2008, Bartlett et al., 2006] that, while simplifying the optimization problem in Eq. (2.5), adopting *surrogate* losses in place of the 0-1 loss allows to asymptotically recover the optimal Bayes classifier. This is easy to see for example for the square loss: for any $f : \mathcal{X} \to \mathbb{R}$, we have that

$$\begin{aligned} \int (y - f(x))^2 d\rho(x, y) &= \int \int (y - f(x))^2 d\rho(y|x) d\rho(x) \\ &= \int \left[(1 - f(x))^2 \rho(1|x) + (f(x) + 1)^2 \rho(-1|x) \right] d\rho(x), \end{aligned}$$

which implies that the solution of the least square expected risk minimization

$$f^* = \inf_{f \in \mathbb{H}} \int_{\mathcal{X} \times \{-1,1\}} (y - f(x))^2 d\rho(x, y) \tag{2.8}$$

satisfies

$$f^*(x) = 2\rho(1|x) - 1 = \rho(1|x) - \rho(-1|x) \tag{2.9}$$

for all $x \in \mathcal{X}$. Therefore $f^*(x) > 0$ if and only if $\rho(1|x) > \rho(-1|x)$ and the optimal Bayes classifier can be recovered by considering $sign(f^*(x))$ as a prediction rule. A very similar reasoning can be applied to the logistic loss, by showing that, in this case, $f^*(x) = \log\left(\frac{\rho(1|x)}{\rho(-1|x)}\right)$.

Extension to Multiclass Problems. The extension to multiclass problems is generally tackled by employing as many binary classifiers as the number of classes to discriminate, in a *one-vs-all* schema where each classifier is trained to discriminate its class against all the others [Rifkin and Klautau, 2004]. In this setting, $\mathcal{Y} = \{1, \ldots, T\}$ where T is the number of classes to discriminate.

The label $y \in \{1, \ldots, T\}$ can be encoded into a vector $e_y \in \{0, 1\}^T \subseteq \mathbb{R}^T$ of the canonical basis $\{e_1, \ldots, e_T\}$ of $\mathbb{R}^T$, i.e., a vector whose y-th coordinate is equal to 1 and the remaining 0, also known as the one-hot encoding of the true label y. The above losses can still be used, by replacing y with e_y and considering $f : \mathcal{X} \to \mathbb{R}^T$ a vector-valued function to be learned, such that the prediction rule provides $\underset{t=1,\ldots,T}{\operatorname{argmax}} f(x)^t \sim y$ (where z^t denotes the t-th entry of the vector z).

To this regard, note that the multiclass approach can also be applied to binary classification problems by setting $T = 2$, where classes have labels $y \in \{1, 2\}$ and encoded labels are $e_1 = [1, 0]^\top$ and $e_2 = [0, 1]^\top$. This would lead to train two distinct classifiers, choosing the predicted class as the $argmax$ of their scores. The two approaches are clearly equivalent, since the optimal Bayes classifier corresponds respectively to the inequalities $\rho(1|x) > \rho(-1|x)$ or $\rho(1|x) > \rho(2|x)$.

The Cross-Entropy Loss. A special note on the logistic loss is in order, since this, and in particular its multiclass extension, the *cross-entropy loss*, are the most commonly adopted in Neural Networks for classification.

It is possible to show that, if we adopt the alternative output space $\mathcal{Y} = \{0, 1\}$, the logistic loss can be expressed as the *binary cross-entropy loss*:

$$\ell(y, \sigma(f(x))) = -y \log(\sigma(f(x))) - (1 - y) \log(1 - \sigma(f(x)))$$

with

$$\sigma(f(x)) = \frac{1}{1 + e^{-f(x)}}$$

the so called *sigmoid* function, normalizing the output scores such that $\sigma(f(x)) = \rho(1|x)$ and $1 - \sigma(f(x)) = \rho(0|x)$. Indeed, it can also be shown that the binary cross-entropy loss minimizes the Kullback-Leibler divergence [Bishop, 2006] between the true and the predicted output class probability distributions, respectively y and $\sigma(f(x))$. The multiclass extension is:

$$\ell(e_y, \sigma(f(x))) = -\sum_{t=1}^{T} e_y^t \log(\sigma(f(x))^t) = -\log(\sigma(f(x))^y),$$

with

$$\sigma(f(x)) = \frac{e^{f(x)}}{\sum_{t=1}^{T} e^{f(x)^t}}$$

where $e_y \in \{0,1\}^T$ is the one-hot encoding of the true label y. The prediction rule provides $\underset{t=1,\dots,T}{\operatorname{argmax}}\, \sigma(f(x))^t \sim y$. The non linear function $\sigma(\cdot)$ employed here is called *softmax* and, like the sigmoid, it normalizes the output scores so that they are positive quantities summing to one and can be interpreted as class probabilities.

2.1.3 The Hypothesis Space

Before proceeding with the discussion of regularization terms, it is necessary to spend some words on the hypothesis space $\mathbb{H}$. Indeed, regularization acts on the functions $f \in \mathbb{H}$ by preventing them from following the points in the training set S "too faithfully". On the one hand, these functions must be "rich" enough to model the phenomenon we are interested in; on the other hand, the larger is the modeling *capacity* of the hypothesis space, the higher is the risk of overfitting and the need for regularization. Usually, these functions are parametrized by a set of parameters θ, i.e., $f(x) = f_\theta(x)$ and the regularization acts on the values of θ.

Linear Models. The simplest space is the space of linear functions, where $\theta = w \in \mathbb{R}^d$ and $\mathbb{H}$ is defined as:

$$\mathbb{H} = \{f_w : \mathbb{R}^d \to \mathbb{R} \ : \ \exists w \in \ \mathbb{R}^d \text{ such that } f_w(x) = w^\top x, \ \forall x \in \mathbb{R}^d\}.$$

In this way, each function $f_w \in \mathbb{H}$ is uniquely defined by a vector w, and the optimization in Eq. (2.5) becomes:

$$\hat{w} = \underset{w \in \mathbb{R}^d}{\arg\min}\, \hat{\mathcal{E}}(w) + \lambda R(w) \tag{2.10}$$

$$\hat{\mathcal{E}}(w) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f_w(x_i)) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, w^\top x_i) \tag{2.11}$$

where it can be noticed that the regularization acts on the values w.

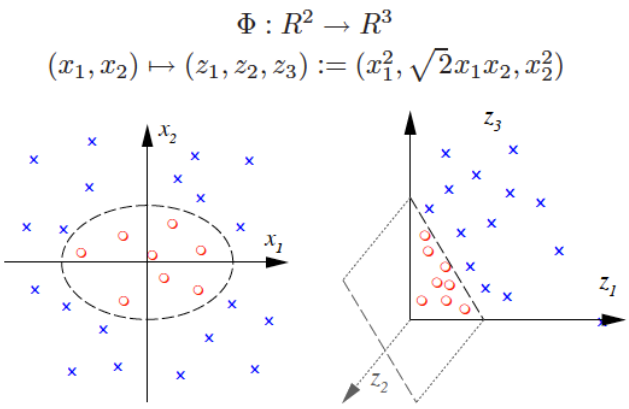

Figure 2.1: **Example of Feature Map** from $\mathbb{R}^2$ to $\mathbb{R}^3$. The classification problem becomes linearly separable in the (higher dimensional) feature space. Picture from [Rosasco, 2016].

Note that this model can be easily extended to multiclass problems by denoting $\theta = W \in \mathbb{R}^{d\times T}$, such that:

$$\mathbb{H} = \{f_W : \mathbb{R}^d \to \mathbb{R}^T \ : \ \exists W \in \ \mathbb{R}^{d\times T} \text{ such that } f_W(x) = W^\top x, \ \forall x \in \mathbb{R}^d\}.$$

Feature Maps. In order to model non linear functions, a well-established approach is to leverage the same formulation as Eq. (2.10)-(2.11), but "replacing" any point $x \in \mathbb{R}^d$ with $\Phi(x) \in \mathbb{R}^p$, where $\Phi : \mathcal{X} \to \mathcal{Z}$ is a suitable function mapping the input space $\mathcal{X}$ into a new space $\mathcal{Z}$, usually called *feature space*. The idea behind this approach is that classes which are not easily separable by a linear model, may become easily separable by a *linear model in the feature space*, since the model may be non linear anymore in the original input space (see Fig. 2.1 for a pictorial representation of this concept). Therefore, by properly defining $\Phi(x)$, the inputs can be mapped into a new space (typically, an Hilbert space, i.e., a space provided with a scalar product) where solving the learning task may be easier.

Kernel Methods. Unfortunately, defining a suitable mapping is generally difficult for most problems. To this regard, it has been shown that mapping points in higher dimensions can generally help making them separable [Bishop, 2006]. One of the most studied implications of this idea is its extension to infinite-dimensional feature spaces.

To this regard, it is possible to demonstrate (*Representer Theorem*) that the solution of the problem (2.10) can always be expressed as $w_S = \sum_{i=1}^n x_i c_i$, where $x_1, \ldots, x_n$ are the points in the training set and $c = \{c_1, \ldots, c_n\}$ a set of coefficients to be learned (in place of w_S). In this formulation, it can be observed that $f(x) = w_S^\top x = \sum_{i=1}^n x_i^\top x c_i$ depends on the input

points only through inner products $x_i^\top x$. Hence, with a similar reasoning followed to introduce feature maps, we can think of replacing these products with $K(x_i, x)$, where $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a Positive Definite (PD), symmetric, function, called *kernel*. In fact, it can be shown (*Mercer's Theorem*) that every kernel has one (in fact many) associated feature maps in a corresponding space $\mathcal{Z}$ (called a *Reproducing Kernel Hilbert Space*), such that $K(x_i, x) = \langle \Phi(x_i), \Phi(x) \rangle_F$. The advantage of this formulation is that, in this way, the mapping $\Phi(x)$ does not have to be explicitly computed, and we can operate in an infinite-dimensional feature space $\mathcal{Z}$ just through the kernel K.

By choosing a suitable non linear kernel it is then possible to derive "kernelized" versions of Regularization Networks, induced by the different loss functions (i.e., kernel RLS, kernel SVM, and so forth). For more details, we refer to [Shawe-Taylor and Cristianini, 2004b, Murphy, 2012].

Often, feature maps and kernels are employed jointly, by first mapping the inputs into a feature space, where the classification problem is hopefully simplified; then, since such mapping may be not sufficient to make the classes linearly separable, a kernel method is employed. As we mentioned in Sec. 1.3.1 and we will see more in detail in Sec. 3.1, in past decades a large body of computer vision and machine learning literature studied the problem of learning good image *representations*, which can be seen as the problem of learning to map images into suitable feature spaces.

2.1.4 Regularization and Model Selection

Once the hypothesis space is fixed, it is possible to define the type of regularization to be applied to the parameters θ characterizing the functions $f_\theta \in \mathbb{H}$. Referring to Eq. (2.10)-(2.11) for instance, $f_w(x) = w^\top x$ and the choice $R(f) = R(w) = \|w\|_2^2$ (or $\|W\|_F^2$, where we denote by $\|\cdot\|_F^2$ the squared Frobenius norm of a matrix) leads to the well known $L2$ regularization. The term $R(w) = \|w\|_2^2$ prevents the solutions of the optimization problem (2.10) from having too large components (that would cause $f_w(x)$ to be unstable with respect to little variations in the input).

Another common regularization strategy is to use $R(w) = \|w\|_1$ (or $\|W\|_1$), known as $L1$ or LASSO regularization. This term enforces sparse solutions and hence provides a regularization effect through variable selection. A combination of $L1$ and $L2$ regularization is often employed, which in the literature is referred to as Elastic Nets. Each of these regularization terms is added to the empirical risk and is weighted by a *regularization parameter* (λ in Eq.(2.10)).

Commonly, λ is not the only regularization parameter involved: other parameters typically regulate the complexity of the form of the function f_θ to be learned. Classical examples are

kernel parameters (e.g., the variance of the Gaussian kernel, or the degree of the polynomial kernel) or finally –as we will see– the number of units and layers in a Neural Network. In fact, many different forms of regularization exist, some of which will be considered in Sec. 4.1.2 with specific reference to Neural Networks.

Cross-validation. In general, a machine learning algorithm can involve therefore multiple regularization parameters (also called *hyper-parameters*), the tuning of which is maybe the most critical aspect in order to properly apply the algorithm. This is often indicated as the problem of *model selection* and one of the most common approaches to address it is by *cross-validation*.

Cross-validation is an empirical procedure to search the best value for a regularization parameter. Usually, the range where to search is fixed according to some previous knowledge on the data. The training set S is then split into two separate parts and, for each value in the selected range:

1. the regularization parameter (in our example, λ) is fixed to that value and Eq. (2.10) is optimized with respect to w (that is, the model is trained) on the first dataset part;
2. the learned model is tested on the second dataset part.

The two parts are often called respectively *training* and *validation* sets. Finally, the model corresponding to the regularization parameter value providing the lowest error on the validation set is selected. Of course, this method (also known as *holdout* cross-validation) can be extended to tune multiple regularization parameters jointly.

Several variations of this basic procedure exist. For example, in the so called *k-fold* cross-validation, for each value of the regularization parameter, the training set S is split into multiple ($k > 2$) parts, where $k - 1$ parts are used for training and the remaining one for validating, repeating the procedure k times so that all parts are tested on, and finally averaging the validation error on them. This procedure is generally chosen to obtain a more robust estimate for the error associated with each value of the regularization parameter. When the training set is particularly small, the *leave-one-out* cross-validation can also be employed, which is a variation of the k-fold cross-validation where the validation set is composed by only a single point.

2.1.5 Regularized Least Squares for Classification

We conclude the Section by presenting the Regularized Least Squares for Classification (RLSC) algorithm, a well established learning method that will also be frequently employed in the experiments reported in this Thesis (Chapters 6, 7 and 8).

As anticipated, the RLSC algorithm uses the square loss (Sec. 2.1.2). In the following, for simplicity we present its linear version (see Sec. 2.1.3), although kernelized versions (employing, specifically, a Gaussian kernel) will be used in the experiments in Chapters 6 and 7. We adopt $L2$ regularization (Sec. 2.1.4).

Considering a multiclass setting where T classes are to be discriminated, and assuming a linear model $f(x) = W^\top x$, with W a matrix in $\mathbb{R}^{d \times T}$, the problem addressed by RLSC is finding:

$$\widehat{W} = \underset{W \in \mathbb{R}^{d \times T}}{\arg\min} \|Y - XW\|_F^2 + \lambda \|W\|_F^2 \tag{2.12}$$

given a training set $\{x_i, y_i\}_{i=1}^n$, with inputs $x_i \in \mathcal{X} = \mathbb{R}^d$ and labels $y_i \in \{1, \ldots, T\}$. The matrices $X \in \mathbb{R}^{n \times d}$ and $Y \in \mathbb{R}^{n \times T}$ are such that their i-th rows correspond respectively to $x_i \in \mathbb{R}^d$ and $e_{y_i} \in \mathbb{R}^T$.

It can be shown that the solution to Eq. (2.12) can be obtained in closed form:

$$\widehat{W} = (X^\top X + \lambda I_d)^{-1} X^\top Y, \tag{2.13}$$

where I_d is the $d \times d$ identity matrix (see for instance [Boyd and Vandenberghe, 2004]).

According to the prediction rule introduced in Sec. 2.1.2, a given $x_{new} \in \mathbb{R}^d$ can therefore be classified according to

$$\hat{f}(x_{new}) = \underset{i=1,\ldots,T}{\arg\max} \ \hat{f}(x_{new})_i = \underset{i=1,\ldots,T}{\arg\max} \ (\widehat{W}^{(i)})^\top x_{new}, \tag{2.14}$$

with $\widehat{W}^{(i)} \in \mathbb{R}^d$ denoting the i-th column of $\widehat{W}$.

2.2 Stochastic Gradient Descent

In the previous Section we introduced the general supervised learning setting within the framework of the Statistical Learning Theory, concluding with the definition of an optimization problem, which must be solved in order to learn the model parameters from the examples in the training set. We reported the example of the RLSC algorithm, where this optimization problem can be solved in closed form. Unfortunately, in most cases the solution of an optimization problem cannot be obtained in closed form and iterative algorithms must be adopted. To this end, in this Section we introduce a class of methods, which goes under the name of Gradient Descent (GD) and Stochastic Gradient Descent (SGD), widely employed to solve the optimization problems involved by learning algorithms.

For simplicity, here we consider a linear model for binary classification. However, most considerations apply similarly to multiclass problems. Moreover, in Sec. 4.1, we will see how the same algorithms can be applied also to more complex and non linear functions (namely, Neural Networks and Convolutional Neural Networks).

2.2.1 Gradient Descent

We start by rewriting Eq. (2.10) in this form:

$$\hat{w} = \arg\min_{w \in \mathbb{R}^d} F(w) \qquad \text{where} \qquad F(w) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, w^\top x_i) + \lambda R(w) \tag{2.15}$$

and $F : \mathbb{R}^d \to \mathbb{R}$ is the sum of the empirical error and the regularizer. In most cases, the solution of this optimization problem cannot be obtained in closed form by solving $\nabla F_w(w) = 0$ (with $\nabla_w F(w)$ the gradient or the subgradient of F with respect to w). For instance, this is possible for RLS, but not for logistic regression. Therefore, iterative optimization methods, such as *Gradient Descent* (GD), must be adopted. In GD, at the first iteration the solution is initialized to a point $w_t = w_0$ and, then, the following steps are taken:

$$w_{t+1} = w_t - \gamma_t \nabla_w F(w_t) \quad t = 1, \ldots, T \tag{2.16}$$

$$\nabla_w F(w_t) = \frac{1}{n} \sum_{i=1}^{n} \nabla_w \ell(y_i, w_t^\top x_i) + \lambda \nabla_w R(w_t) \tag{2.17}$$

where $\nabla_w F(w_t)$ is the gradient of F with respect to w computed at w_t and γ_t is a positive constant (or a sequence) called the step size or *learning rate*. The idea is that we start from an initial point and progressively move in the direction opposite to the gradient of the function (which is the steepest direction pointing towards the minimum). The term *descent* comes from the fact that, for a suitable choice of the step size γ_t, it can be shown that if F is a convex function, $F(w_t) \geq F(w_{t+1}) \; \forall w_t$. Indeed, if F is a convex function, choosing the step size appropriately ensures the iterations to converge to a global minimum w_S from any initial point [Boyd and Vandenberghe, 2004]. The relative change in the iterates $||w_t - w_{t-1}||$ and/or in the quantity to be minimized $||F(w_t) - F(w_{t-1})||$ is often used as a stopping criterion.

The step size γ_t is a key parameter: if it's too small, the iterations will take too long to converge, but if it's too large, the iterations may diverge. For example, for convex, differentiable and Lipschitz functions, a suitable choice for the step size can be shown to be $\gamma = 1/L$, where L is the Lipschitz constant of the gradient (less or equal than the biggest eigenvalue of the Hessian matrix for all w). In all other cases, strategies in order to choose the step size appropriately must be devised (see, e.g., [Boyd and Vandenberghe, 2004] and references therein).

Note on non-convex optimization and deep learning. In case the function F is non-convex, the iterations may converge to critical points that are not a global minimizer (e.g. local minima, saddle points) or simply diverge, depending on the initialization point and many other factors as the shape of the function. While convex analysis has been throughly studied in the literature, the analysis of non-convex optimization has rather been overlooked in the past decades.

As we will see in Chapter 3, recently large Convolutional Neural Network (CNN) models empirically proved to work very well on large-scale image classification problems, but the theoretical foundations of their effectiveness are unclear. Indeed, the function to be minimized by GD when employing deep CNNs is non-convex. For this reason, the study of non-convex optimization has been recently attracting the attention of the deep learning literature and is an active field of research.

2.2.2 Stochastic Gradient Descent

This class of techniques derives from the basic idea of GD by considering that the objective function F is the summation of n different components (see Eq. (2.15)), each depending only on a single training point, and so is the gradient (Eq. (2.17)). The key concept of *Stochastic Gradient Descent* (SGD) is hence to use, at each iteration t, only one component, among these ones, to update w_t (instead of averaging the gradient of F on all the training set). If we denote the index of a point sampled from S at iteration t by $i_t \in \{1, \dots, n\}$, we have:

$$w_{t+1} = w_t - \gamma_t G_{i_t}(w_t) \quad t = 1, \dots, T \tag{2.18}$$

$$G_{i_t}(w_t) = \nabla_w \ell(y_{i_t}, w_t^\top x_{i_t}) + \lambda \nabla_w R(w_t) \tag{2.19}$$

The term *stochastic* in the name of the methods comes from the fact that $G_{i_t}(w_t)$ can be seen as a stochastic estimate of the gradient $\nabla_w F(w_t)$. If the objective function F is convex, it can be shown [Nemirovskii et al., 1983] that SGD still converges to a global minimum, albeit with slower rates than batch GD. The recent interest in these methods is motivated by the fact that, for non-convex functions, SGD empirically proved to work much better in practice than batch counterparts. In particular, when the training set is large, SGD provides an approach to update the solution more frequently (instead of having to visit all points at each iteration), which demonstrated to work well in many applications [Bottou, 2012].

The sequence i_t of example indices can be chosen deterministically or stochastically. Some popular choices for instance are:

1. picking points in S without replacement until all points have been visited,
2. repeating the first procedure multiple times, every time maintaining the same order in visiting the points,
3. repeating the first procedure multiple times, every time changing the visiting order,
4. sampling points in S with replacement according to a uniform distribution.

Note that in the SGD setting, the step size always needs to be chosen as a sequence γ_t going to zero. For example, for strictly convex and differentiable functions, the choice $\gamma_t = \frac{1}{t}$ can be shown to suffice [Nemirovskii et al., 1983].

2.2.3 Mini-batch Stochastic Gradient Descent

One largely employed variation of SGD is the *mini-batch* SGD, a method which can be seen as "in between" GD and SGD. This method maintains the advantage of SGD, namely, frequent solution updates. However, at the same time, at each iteration an average of the gradient over multiple points is used, instead of the gradient computed in a single point. Conversely, this procedure can be seen also as a batch GD which considers, at each iteration, a small group of points rather than the whole trainining set S. In this way, the stochastic estimate of the gradient is made more robust, but the solution is updated rather frequently. Denoting by $b_t = \{(i_t)_j\}$, for $j = 1, \dots, m$ the set of indices of a *mini-batch* of m points sampled from S at iteration t, we have:

$$w_{t+1} = w_t - \gamma_t G_{b_t}(w_t) \quad t = 1, \dots, T \tag{2.20}$$

$$G_{b_t}(w_t) = \frac{1}{m} \sum_{j=1}^{m} \nabla_w \ell(y_{(i_t)_j}, w_t^\top x_{(i_t)_j}) + \lambda \nabla_w R(w_t) \tag{2.21}$$

where the different choices described for SGD regarding the sampling of the points i_t can be adopted. Generally, in mini-batch SGD every iteration t the gradient of m points is hence averaged, and the number of iterations needed to visit all n points in S is called *epoch* (one epoch corresponds to $\lceil \frac{n}{m} \rceil$ iterations).

Variations on the Solution Update. It has been largely shown in the literature that some modifications of the basic SGD update rule can in practice speed up convergence. In the following, we denote for convenience the gradient at iteration t by $\mathbf{g_t} = G_{i_t}(w_t)$ (or $\mathbf{g_t} = G_{b_t}(w_t)$ in mini-batch SGD). Different approaches to replace $\Delta w_t = -\gamma_t \mathbf{g_t}$ with more effective update quantities have been proposed and are largely employed when training deep Neural Network models.

A popular approach is the **Momentum** update [Rumelhart et al., 1988]. This method introduces an intermediate variable $\mathbf{v_t} = \alpha \mathbf{v_{t-1}} - \gamma_t \mathbf{g_t}$ where $\mathbf{v_0} = \mathbf{0}$ and $\alpha \in [0, 1]$. The solution is then updated simply as $\Delta w_t = \mathbf{v_t}$. The name of the method comes from a physics interpretation where the loss is seen as a hill and its negative gradient, $\mathbf{g}$, is interpreted as the force felt by a particle at height w on such hill, acting on its velocity, rather than on its position directly. Indeed, $\mathbf{g} = m\frac{d\mathbf{v}}{dt}$ (if we suppose m the mass of the particle). The parameter α is called *momentum* (in the physical view it actually would correspond to the coefficient of friction damping velocity and reducing the kinetic energy of the system). In general, momentum enforces steps towards consistent directions of the gradient.

While momentum modulates the update with a running estimate of the first moment of the gradient (its mean), a number of methods have also been developed that modulate the update

using the second moment:

- **Adagrad** update [Duchi et al., 2011]. This method uses an intermediate variable $\mathbf{r_t} = \mathbf{r_{t-1}} + \mathbf{g_t} \odot \mathbf{g_t}$ (where $\odot$ is element-wise multiplication) of sum of squared gradients, in order to modulate the update as follows: $\Delta w_t = -\frac{\gamma_t}{\sqrt{\mathbf{r_t}+\delta}}\mathbf{g_t}$, where δ is a small number preventing division by zero.
- **RMSProp** update [Tieleman and Hinton, 2012]. This method uses a running mean of the second moment $\mathbf{r_t} = \rho\mathbf{r_{t-1}} + (\mathbf{1} - \rho)\mathbf{g_t} \odot \mathbf{g_t}$ (where ρ is usually set to be slightly lower than 1, i.e., 0.99).
- **Adam** update [Kingma and Ba, 2015]. This method estimates both the first and second running moments and can be seen as a combination of RMSProp with Momentum.

All these methods have an equalizing effect on parameter updates, i.e., parameters that see large gradients will take smaller steps, while parameters that see very low gradients will take larger steps.

Chapter 3

Deep Convolutional Neural Networks

In this Chapter we provide an essential background on deep Neural Networks and in particular on Convolutional Neural Networks (CNNs), hierarchical architectures that are at the core of modern deep learning methods.

Since the focus of this Thesis is on machine vision applications, we start by introducing the problem of learning image representations, through a short overview of some fundamental approaches that have been proposed in the literature in past decades (Sec. 3.1). Building on the setting introduced in Chapter 2, we then describe the basic CNN architecture (Sec. 3.2), providing an intuitive explanation for its main building blocks and its effectiveness in image representation learning. Finally, we present four well-established CNN models that, in latest years, have proved to be remarkably effective in several vision problems (Sec. 3.3). These are also the models that will be employed in the experiments reported in the Thesis (Chapters 6, 7).

3.1 From Pixels to Labels: Building Invariance and Selectivity

Building on previous Chapters, we can define the multiclass image classification problem as the problem of finding

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f_\theta(X_i)) + \lambda R(\theta) \tag{3.1}$$

where $f_\theta(X)$ is a function characterized by parameters generally denoted with θ, and acting, in this case, on an image $X \in \mathbb{R}^{c \times r \times 3}$, i.e., belonging to the space of all possible RGB (i.e., 3-channel) images of c columns and r rows. The training set S is composed in this case by n images X_i, each one associated to a label $y_i \in R^T$ (expressed, e.g., as the one-hot encoding of the name of the object class represented in the image).

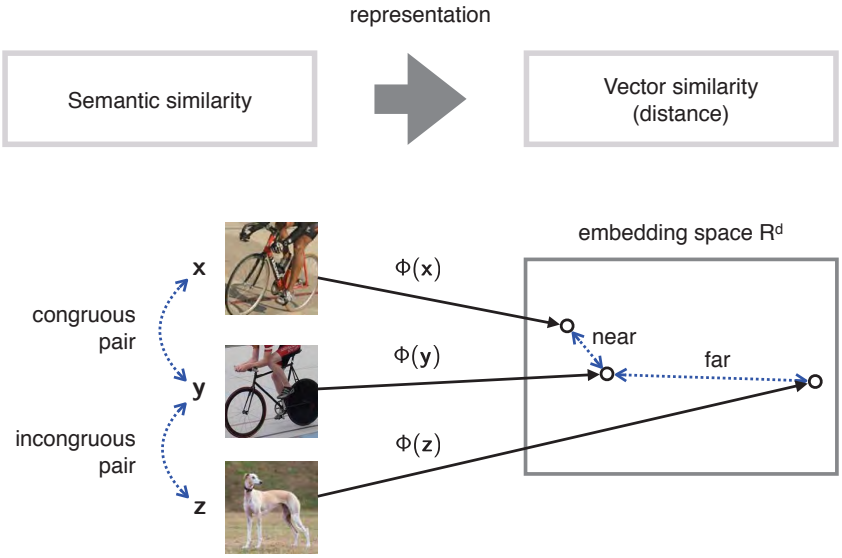

Figure 3.1: **Pictorial representation of the concept of selectivity and invariance of an image representation.** Picture adapted from A. Vedaldi's lecture at the "2015 International Computer Vision Summer School".

Applying a linear model to this problem would lead to have a function $f_W(\tilde{X}) = W\tilde{X}$, where the parameters θ would be a weight matrix $W \in \mathbb{R}^{T\times(c*r*3)}$, T being the number of classes and $c * r * 3$ the length of the input vector $\tilde{X}$, where we denote in this way the "vectorized" version of the image X (obtained, e.g., by concatenating the image rows, one channel after the other).

As will be clarified in the following, while this model is relatively simple, it is too limited for tackling complex problems as the classification of natural images. We already pointed out that raw pixel values can largely vary depending on objects' pose, occlusions, background, illumination, and so forth. In categorization tasks also different object instances should be mapped into the same class. There is therefore such a high degree of intra-class variability at the level of raw images, that linear classifiers are definitely not likely to succeed in separating the different classes – apart from particularly simple cases. To this regard, in Sec. 2.1.3 we explained that a way to "enrich" linear models is to map the input into a *feature space*, where the classification task can be simplified. We also mentioned that, in computer vision, feature maps are also commonly indicated as image *representations*.

In particular, in the image classification scenario, a suitable feature space is a space where each image is mapped into a representation that is sensitive only to the image content which is relevant for the classification task, while being *invariant* to irrelevant aspects. More specifically, an image representation can be considered "good" for a classification task if it is *invariant*

to class-preserving image transformations, while being *selective* for the different classes. In Fig. 3.1 a pictorial representation of this idea is provided.

Since, in general, it is not possible to know beforehand which content will be relevant for a particular classification task, image representations, like classifiers, must be learned from data and from examples. In fact, the image classification problem typically boils down to the problem of learning good image representations.

The two fundamental steps in the construction of an image representation which is (i) selective for a specific pattern (let's say an edge, or a part of an object, inside a particular image region) and (ii) invariant to some visual transformation of the pattern (let's say rotation) are:

1. The detection of the pattern under all possible configurations of the considered transformation (in this case, the detection of the pattern in all possible orientations). This step is also referred to as *coding*, because it usually employs a number of *templates* or *filters*, each tuned to a different pattern configuration and *encodes* the input (the considered image region in this example) into their responses (e.g., by multiplying pixel-wise each template with the image region).
2. The aggregation of such responses, by applying a pooling operator (e.g., the average or maximum).

While we refer to [Anselmi et al., 2015, Anselmi et al., 2016, Poggio and Anselmi, 2016] for a formal explanation of this concept, in the following we review how different approaches proposed in the literature can be reconnected to this paradigm and highlight how, through the progressive addition of more coding-pooling *layers*, they progressively led to recent deep learning methods.

3.1.1 Local Invariance in Hand-crafted Descriptors

As anticipated in Sec. 1.3.1, SIFT or HOG-like descriptors [Lowe, 2004, Ke and Sukthankar, 2004, Mikolajczyk and Schmid, 2004, Bay et al., 2006, Dalal and Triggs, 2005]), can be shown to be based on these two stages of computation (see, e.g., [Mahendran and Vedaldi, 2015]) in order to robustly describe small patterns, irrespective of local transformations (as rotations, scaling or translations). These descriptors are usually referred to be *hand-crafted* since the filters in the coding step are manually engineered. While the manual filter design is possible if the patterns to be detected are simple (edges, corners, blobs), the same approach is not feasible to detect more specific and extended patterns (like, e.g., object parts).

3.1.2 In Search of Global Invariance: Dictionary Learning and Pooling

A variety of methods was then proposed in the literature, which can be seen as iterating the two above steps of coding-pooling, in order to detect more complex patterns, invariant to wider

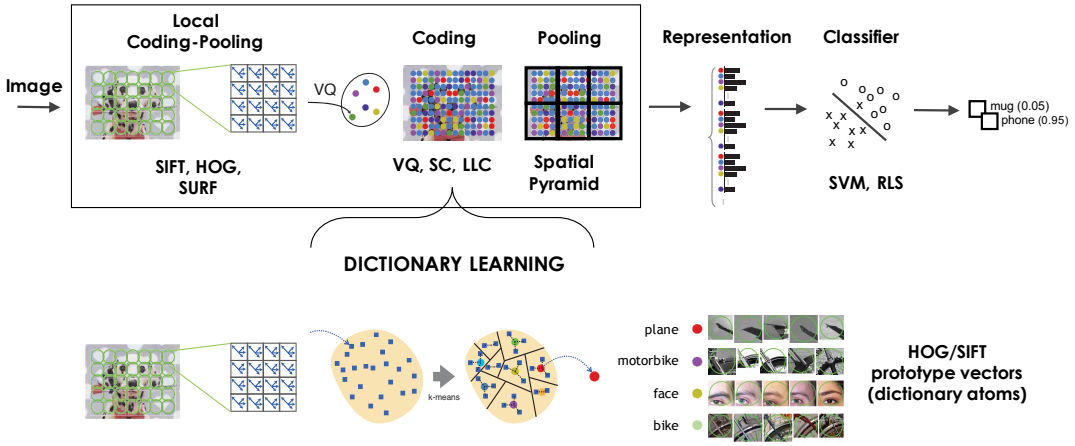

Figure 3.2: **Main steps involved in dictionary learning pipelines.** Picture adapted from A. Vedaldi's lecture at the "2015 International Computer Vision Summer School".

transformations.

In these pipelines, at the first stage, local descriptors like SIFTs are usually extracted either in a dense grid [Fei-Fei and Perona, 2005] or only from some selected keypoints [Jegou et al., 2010, Sánchez et al., 2013].

The second encoding is instead learned from data. To this end, the core idea of *dictionary learning* methods is to use a subset of local descriptors (extracted from a number of example images) to *learn* the most "meaningful" descriptors for the considered problem. These can be seen as "representation prototypes" of specific patterns (like parts of the most frequent objects in the images). The term *dictionary* comes from the inspiration from the Bag Of Words (BOW) algorithm for text classification, since the learned representation prototypes can be interpreted as the *words* of a *dictionary* of terms frequently appearing in the texts (that is, in the images) to be classified.

Once the dictionary is learned, it is then used as a basis to *encode* the local descriptors extracted from any new image. These are *mapped* on the dictionary codebook and become therefore just linear combinations of prototypes (words). The encoded descriptors are finally spatially pooled (e.g., averaged) over the full image. The result is that, in this way, each image is associated to a vector representation, indicator of its semantic content (like the word count of the text). Once images are encoded into their vector representations, standard classifiers are trained (and tested) in the representation space.

Several strategies have been proposed in the literature to learn the dictionary. Some of them are unsupervised and are based on the computation of higher order statistics on the pool of local descriptors (see, e.g., Super Vectors [Zhou et al., 2010], VLAD encoding [Jegou et al., 2010], Fisher Vectors [Sánchez et al., 2013]). Others are based instead on the minimization of

the reconstruction error of the original descriptors, constrained to (most often) some sparsity criterium (see, e.g., hard/soft Vector Quantization [Philbin et al., 2008], Sparse Coding [Lee et al., 2006, Yang et al., 2009] and Locally-constrained Linear Coding [Wang et al., 2010]). We refer to [Jarrett et al., 2009, Boureau et al., 2010a, Chatfield et al., 2011] for comprehensive overviews. Interestingly, [Boureau et al., 2010a] show that exploiting the prediction error of the classification task as a supervisory signal to learn the dictionary (in a similar way deep networks learn filters' weights with backpropagation) does improve performance.

Several works in the literature also proposed different pooling approaches (see, e.g., [Boureau et al., 2010a, Boureau et al., 2010b, Boureau et al., 2011, Russakovsky et al., 2012, Jia et al., 2012, Fanello et al., 2014]). In particular, it was observed that, instead of pooling the encoded descriptors over the full image, pooling among smaller regions and then concatenating the pooled descriptors (a strategy known as Spatial Pyramid Matching [Lazebnik et al., 2006, Yang et al., 2009]) allows to maintain a "minimal" spatial structure in the final image representation, that is indeed critical to recognition tasks.

In Fig. 3.2 a pictorial representation of the main steps involved in a typical dictionary learning pipeline is reported.

3.1.3 End-to-end Learning

One major limitation of dictionary learning approaches resides in the fact that only one representation layer is actually learned from data. Another relevant point is that, in all these approaches, the representation learning stage is completely separated from the learning of the classification task: a feature map is first learned, and then the mapped images are used to train a classifier. A different approach is to jointly solve the problem of learning the feature map and the classification problem. This can be achieved by, e.g., changing the hypothesis space by acting on the form of the function $f_\theta(X)$. An intuition for this approach is that the joint optimization of the feature map and the classification task may lead to learn a more effective image representation.

The first example of such a system was proposed by LeCun et al. [LeCun et al., 1989, LeCun et al., 1998]. The authors managed to express a multi-layer, hierarchical, image processing pipeline in a recursive form, which could be optimized by applying SGD (see Sec. 2.2). This approach is often referred to as *end-to-end* learning, since it learns a continuous mapping from the image X to the final prediction $f_\theta(X)$, as opposed to the dictionary learning approaches previously mentioned.

In the following, we will express such mapping as $f_\theta(X) = CNN_\theta(X)$, where θ will be a set of weight matrices and $CNN(\cdot)$ is what is called a deep (Convolutional) Neural Network.

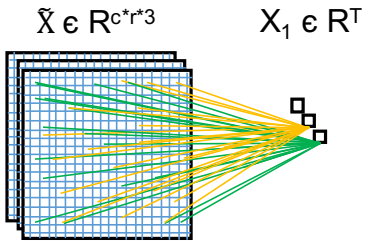

Figure 3.3: **Multiclass linear classifier** acting on an unrolled RGB image $\tilde{X}$ and producing a vector X_1 of T output scores. Each score is the result of a weighted sum of all pixels, passed through the $softmax$ non linearity if the cross-entropy loss is employed.

3.2 Basic Architecture

In Sec. 2.1 we introduced the general supervised learning setting as the problem of learning, from data, the parameters of a specified model; in Sec. 2.2 we sketched how SGD approaches can be applied to address this problem. We did not specified a particular choice of a model, considering for simplicity linear ones. However, we have seen at the beginning of Sec. 3.1 that this simple choice is not sufficient for image classification settings, and we reviewed the principal methods proposed in the literature to map images into good feature spaces. We finally introduced Convolutional Neural Networks (CNNs) as models that approach the image classification problem by learning a continuous mapping from images to labels, through the definition of a suitable hierarchical architecture. In this Section we describe such architecture, while in Sec. 4.1 we will describe how its parameters can be learned through a particular application of SGD also known as the *backpropagation* algorithm.

We refer to [Goodfellow et al., 2016] for a more detailed and comprehensive introduction on deep learning and deep CNNs. We also refer to [Srinivas et al., 2016] for a useful brief overview on the topic.

3.2.1 Neural Networks

We start by considering the multiclass classifier with cross-entropy loss introduced in Sec. 2.1.2. In this case, the input-output relation between the image and the predicted class scores provides $\sigma(f_W(\tilde{X})) = \sigma(W\tilde{X})$, with $\sigma(\cdot)$ the $softmax$ function and $W \in \mathbb{R}^{T\times(c*r*3)}$. In Fig. 3.3 we report a pictorial representation of such model, which is often called single-layer *perceptron* [Rosenblatt, 1958, Bishop, 2006] or *Neural Nework* (NN). Indeed, we called the output X_1, as opposed to multi-layer architectures for which we will see that the output will be denoted by X_L, L being the number of layers. We adopt the usual convention of NNs where the rows of the matrix W are represented, with different color, as connections between the corresponding

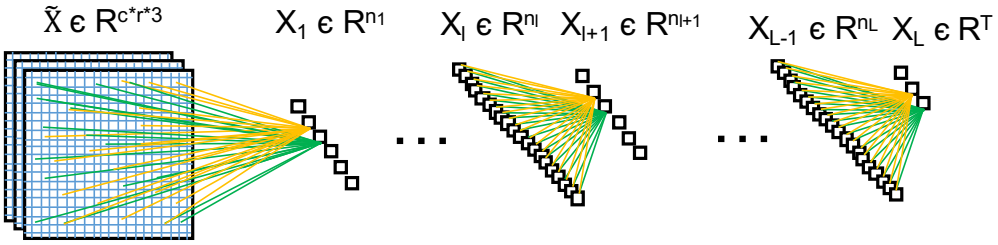

Figure 3.4: **Deep Neural Network** acting on an unrolled RGB image $\tilde{X}$ and producing a vector X_L of T output scores. At each layer, each unit outputs the weighted sum of all units in the previous layer (of all pixels at first layer), passed through a non linear function. The $softmax$ non linearity is used in the last layer if the cross-entropy loss is employed.

inputs and output unit (in this case, each output unit receives connections from all pixels in the image). Note that we represented $\tilde{X}$ with the shape of a 3-dimensional array even if in the formulation it is unrolled and treated like a 1-dimensional array.

Biological Inspiration. It is useful to note that this model has a biological interpretation where the unit of computation (e.g., one output unit in Fig. 3.3) can be seen as modeling the basic operations carried on by a neuron in the cortex [McCulloch and Pitts, 1943]. This unit computes the weighted sum of its inputs and passes it through a non linear function. Similarly, a neuron receives inputs from a pool of other neurons through synaptic connections located in a specific part of the cell called *dentrites*; it sums them up, weighting each input depending on the strength and kind of the synaptic connections (inhibitory –negative weight– or excitatory –positive weight–), a computation which has been proved to happen in another part of the cell called *soma*; finally, it sends out the output to other neurons, through a third part of the cell called *axon*. Since then neurons communicate through *spikes* and the transmitted signal is encoded in their firing frequency, which is a positive quantity, the non linearity $\sigma(\cdot)$ can be interpreted as suppressing the negative part –eventually saturating to some value, as it happens in biological neurons.

Since, as we pointed out, natural image classification tasks are unlikely to be linearly separable problems, a direct extension of this single-layer model (often indicated as "shallow") can be built by keeping a biological inspiration and concatenating multiple layers of neurons, where the output of one layer becomes the input of the following one. In this way, the global input-output relation takes the form of a composed function, where each block is built on top of the previous one, in a hierarchy of neuronal layers that retrace the connections between lower and higher cortical areas in the brain.

The resulting model is called multi-layer perceptron or Neural Network (NN) and was first proposed in the 1960s [Rosenblatt, 1958]. An example of such architecture, comprising L layers, is sketched in Fig. 3.4. Each layer is described by the following formulas (where we suppress the dependency of the current layer on previous one for simplicity, i.e. $X_{l+1}(X_l) := X_{l+1}$):

$$\begin{cases} X_1 & = f_{NL}(U_1) \\ U_1 & = W_1 \tilde{X} \\ W_1 & \in \mathbb{R}^{n_1 \times (c*r*3)} \end{cases} \quad \begin{cases} X_{l+1} & = f_{NL}(U_{l+1}) \\ U_{l+1} & = W_{l+1} X_l \\ W_{l+1} & \in \mathbb{R}^{n_{l+1} \times n_l} \\ l & = 1, \ldots, L-2 \end{cases} \quad \begin{cases} X_L & = \sigma(U_L) \\ U_L & = W_L X_{L-1} \\ W_L & \in \mathbb{R}^{n_L \times n_{L-1}} \end{cases} \tag{3.2}$$

where we note that $\sigma(\cdot)$, the non linear function used in the output layer (X_L), is the $softmax$, while the non linearity used in the so called *hidden layers* (X_1 to X_{L-1}), indicated as $f_{NL}(\cdot)$, is a different non linear function, some common examples being the hyperbolic tangent $f_{NL}(z) = \frac{1-e^z}{1+e^z}$ or the sigmoid $f_{NL}(z) = \frac{1}{1+e^{-z}}$ or the Rectifying Linear Unit (ReLU) $f_{NL}(z) = max(z, 0)$. Recently, the ReLU is the preferred option for training deep networks, because it avoids saturation issues and it allows for faster differentiation, thus simplifying backpropagation of derivatives (for details refer to [Krizhevsky et al., 2012, He et al., 2015]).

Note that we did not introduce biases into the linear models in order to keep the equations simple, also for the derivation of the backpropagation algorithm that will be reported in Sec. 4.1. The introduction of biases is usually done in order to account for uncentered data, but has little impact in the model formulation and optimization (since the bias does not have to be affected by regularization).

This model provides $X_L(\tilde{X}) := NN_\theta(\tilde{X})$, where $NN_\theta(\cdot)$ is a composed function and the parameters θ are the union of all weight matrices plus the biases, i.e. $\theta = \{W_1, \ldots, W_L\}$, and $L2$ regularization usually acts on the weight matrices. Other important parameters which must be assigned in order to completely specify a NN model are (i) the number of layers, L, and (ii) the number of units in each layer, $n_1, \ldots, n_L$. These have a critical impact on the *capacity* of the model and, therefore, on the risk of overfitting the training set, and in general are chosen with cross-validation.

3.2.2 Convolutional Neural Networks

The model described so far does not assume any spatial structure in the input vector. However, since we are actually considering 3-dimensional signals with a quite meaningful spatial distribution, it is reasonable to account for this in the definition of a proper image classifier. Moreover, when applied to images, hierarchical architectures as the one described in Fig. 3.4, also called *vanilla* NNs, are characterized by such an extremely large number of parameters

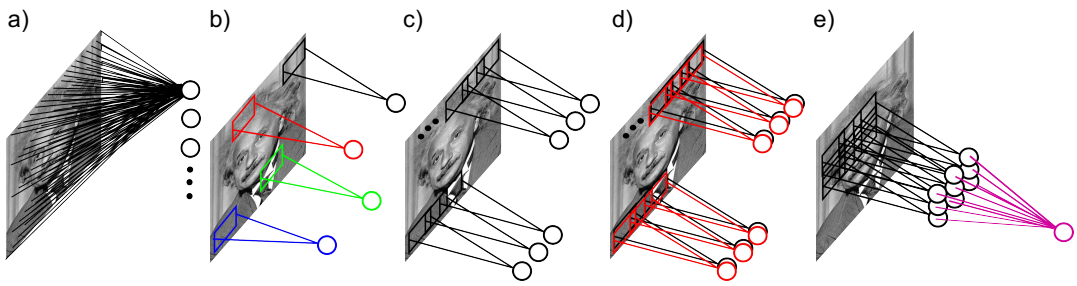

Figure 3.5: **From NNs to CNNs**: a) a classical fully connected layer of a vanilla NN where each unit is connected to all pixels in the image; b) the weights of each unit are constrained to be *local*; c) units share the same weights and in fact the layer now convolves the image with this small weight matrix; d) more, different filters are employed, to detect multiple patterns; e) a subsequent spatial pooling layer aggregates units' responses in image regions, in order to achieve invariance to translations within the region. Picture adapted from Yann LeCun.

that their training is often computationally unfeasible. The main problem of this kind of model is that each neuron is connected to *all* pixels in the image. Since the first studies in computer vision and neuroscience [Hubel and Wiesel, 1962, Fukushima, 1980], it became evident that neither effective algorithms can be developed with this approach, nor computations in the visual cortex are organized in this way. Convolutional Neural Networks (CNNs) account for both limitations and provide effective architectures for image classification, while adopting bio-inspired computational principles.

Convolution Layers

Convolution layers are a particular kind of NN layers where the connectivity of each unit is constrained to be *local* and *shared*. To better explain this idea, refer to Fig. 3.5: in a classical "fully connected" layer of a vanilla NN, each unit is connected to all pixels in the image (Fig. 3.5 (a)). However, this approach is not suitable to deal with images for at least two reasons: first, usually interesting patterns are local rather than global; second, an interesting pattern must be correctly detected at every possible image location. Hence, employing NNs requires to learn an explosion of parameters, which may be unnecessary. Indeed, a much more reasonable approach is to focus each unit to "look" only at a restricted image region, detecting possible interesting patterns *within that region*. This can be achieved by constraining the weights of units to be non-zero only in a pixel neighbourhood (Fig. 3.5 (b)). Then, the weights of all units can be further constrained to be equal (Fig. 3.5 (c)), such that a same pattern can be detected at every possible position. It is easy to see that the operation performed by the layer is now equivalent to convolving the image with the small matrix of non-zero weights (which has

become the equivalent of a *filter*). Moreover, since a remarkable number of parameters has been saved, it is finally possible to employ multiple filters to detect different patterns – each at every spatial location (Fig. 3.5 (d)).

In image processing and computer vision, convolving an image with a set of filters (often called a "bank" of filters) is a very common operation. Filtering is one way to build *feature maps*, i.e., topographic maps of *features* extracted from the image, which encode relevant information and discard non relevant aspects – information relevance of course being dependent on the task. Note that the use of the same term adopted in machine learning when mapping inputs into feature spaces is not by chance, indeed, they are exactly the same thing.

Biological Inspiration. Neurophysiological studies has shown that connections between neurons in the visual cortex are topologically organized (see [Fishman, 1997] for an historical overview). In particular, each neuron in the primary visual cortex (V1) receives inputs from a pool of photoreceptors that are located in a small area on the retina, and they are spatially ordered so that neurons that are close one to the other receive input from adjacent portions of the retina. This ordered mapping from photoreceptors in the retina to neurons in V1 forms a so called *retinotopic* map of neurons, each one "seeing" a specific location of the visual field (called *receptive field* (RF)). Retinotopic maps in V1 are only an example of topographic organization of neuron populations in the nervous system. Indeed, visual information is not the only sensory input that is processed in a predefined spatial arrangement: tonotopic maps have been found in the auditory cortex and somatotopic maps in the the primary somatosensory cortex [Killackey et al., 1995, Kaltenbach et al., 1992]. The computation in V1 is further organized such that, for each location in the retina, there exist neurons tuned to different and specific visual stimuli –like small edges differently oriented [Hubel and Wiesel, 1959, Hubel and Wiesel, 1962]. In other words, there exist multiple retinotopic maps in V1, each one composed of neurons, called *simple cells*, tuned to the same specific visual stimulus. This is the basic mechanism that enables us to perceive every different stimulus in every location in the visual field. The corresponding implementation of a retinotopic map of neurons, each one with the same RF and receiving *local* connections from a small neighborood of photoreceptors, is provided by the operation of *convolution* of a filter (modeling the RF) with an image.

Pooling Layers

Feature maps are topographical representations of the relevant content in an image. As we mentioned, what matters in recognition is the information about "what" is in the image, irrespective of its spatial location. Hence in order to build suitable mappings for image classification, this *distributed* information has to be spatially aggregated into *global* representations. To this end,

spatial pooling layers are introduced into CNNs and interleaved with convolution ones.

Each unit in a pooling layer aggregates the output of units that are located in its neighborood in the previous convolution layer, by computing, e.g., their average or their maximum (see Fig. 3.5 (e) for a pictorial representation). In this way, the output of the pooling unit will be the same, for any position of the stimulus that has been detected by units in the previous layer. In other words, the unit response will be *selective* for the stimuluts, while being *invariant* to its position within the pooled spatial region. We refer to [Boureau et al., 2010a, Boureau et al., 2010b, Boureau et al., 2011] for an in-depth analysis of pooling, providing also arguments in favour of the use of max rather than $average$ pooling in hierarchical architectures like CNNs.

Biological Inspiration. Like simple cells, pooling cells have also been found in V1 and are called *complex cells*. The response of a complex cell has been measured to be the same when varying the location of a visual stimulus within its RF, i.e., it exhibits *invariance* to the exact position of the stimulus within its RF.

Hierarchical Architecture

Deep CNNs are finally built by ay alternating (i) convolutional and (ii) pooling layers (with subsampling), in order to progressively increase (i) the complexity of the encoded stimuli and (ii) the spatial invariance of feature maps. More precisely, usually, in CNNs the spatial resolution and the size of feature maps decreases, from lower to higher layers, until eventually the size of the feature map is reduced to just a single pixel. Reducing the spatial resolution of the signal allows to save memory and computations, such that it is then possible to progressively increase the number of filters employed at each layer, detecting in this way more different patterns. As a result, the full image is encoded into a single global vector representation (of dimension given by the number of filters/channels in the last layer). Note that the role of non linearities is fundamental and, usually, element-wise non linearities (as the ones mentioned for vanilla NNs) are interleaved between convolution and pooling. Without them, indeed, the whole architecture could be reduced to a single linear layer.

At this point, such image representation can be fed to a fully connected layer, which will output classification scores. Of course, multiple fully-connected layers can be concatenated. However, their efficacy is debated, compared with the increase in the number of parameters they involve. To this regard, modern deep architectures are progressively reducing the number of fully-connected layers employed at the end of convolution layers.

An example of CNN is represented in Fig. 3.6 and in the following we provide some simple equations for its convolutional and pooling layers. The underscript l indicates the layer, as before. The element in row x and column y of slice (or, channel) z is denoted by $X_l(x, y, z)$.

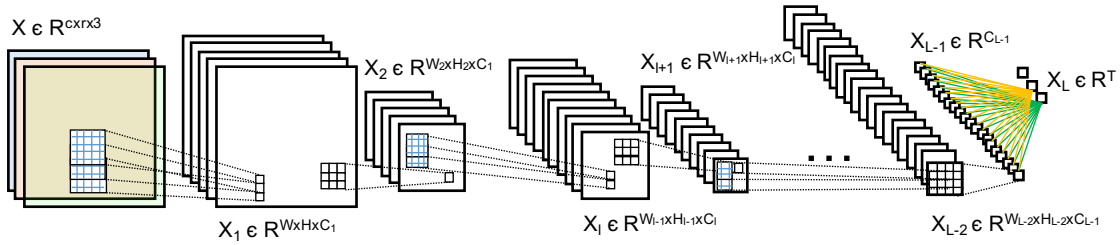

Figure 3.6: Convolutional Neural Network acting on an RGB image X and producing a vector X_L of T output scores. Convolution and pooling layers (with subsampling) alternate, in order to progressively (i) increase the specificity and (ii) decrease the size and spatial resolution of the feature maps, until a global vector representation is extracted and passed to a fully-connected layer.

Each convolutional layer is characterized by a number C_l of filters, indicated as $W_{l,z}$, for $z = 1, \ldots, C_l$, such filters are 3-dimensional arrays, usually all with same spatial support $S_l \times S_l$ and same number of channels as the input feature map X_{l-1} (i.e., $W_{l,z} \in \mathbb{R}^{S_l \times S_l \times C_{l-1}}$). The convolution between a filter and a feature map is therefore a 3-dimensional convolution, where the third dimension of the two is the same, and can be alternatively expressed as the sum of 2-dimensional convolutions, between each slice in the feature map and the corresponding slice in the filter, as in the following formulation:

$$X_l(x,y,z) = \begin{cases} conv: & f_{NL}(U_l(x,y,z)) \\ & U_l(x,y,z) = \sum_{c=1}^{C_{l-1}} (\sum_{u,v \in [-S_l, S_l]} W_{l,z}(u,v,c) X_{l-1}(x-u, y-v, c))) \\ pool: & \mathrm{f}_{\underset{u,v \in [-P_l, P_l]}{\mathrm{POOL}}} X_{l-1}(x+u, y+v, z) \end{cases} \tag{3.3}$$

where $f_{POOL} = avg$ or $f_{POOL} = max$ and $P_l \times P_l$ is the spatial support of pooling at layer l. Border effects in the convolution implementation should be accounted (usually this is done by zero-padding the signal). We omitted the biases for the sake of simplicity, but note that in general they are included into the formulation of convolution layers. Also, we did not include in the equations the –yet fundamental– subsampling operation: usually, this is also included in the convolution operation, in the form of a stride greater than one.

According to this formulation, the output is $X_L(X) = CNN_\theta(X)$, where $CNN_\theta(\cdot)$ is again a composed function, provided by an alternation of convolution and pooling layers, followed by one (possibly more) fully-connected layer (which have been defined for NNs).

The parameters θ in this case are the union of all convolution filters

$$\theta = \{W_{1,1}, \ldots, W_{1,C_1}, \ldots, W_{L_{CONV},1}, \ldots, W_{L_{CONV},C_{L_{CONV}}}\}$$

supposing to have L_{CONV} convolution layers, plus the weights W_{FC} (and possibly biases) of the fully-connected layer.

Computational Efficiency. To provide an idea of the practical implications of the computational advantages provided by the convolution-pooling approach with respect to the fully connected approach of vanilla NNs, we can consider as an example a relatively small 256×256 RGB image and a 100-class classification task. We can quickly compute that a simple CNN with a quite standard architecture, stacking, e.g., 4 convolutional layers with increasing number of filters (let's suppose from 32 in the first layer to $64, 128, 256$ in the following ones), all with a 3×3 spatial support, and alternating with pooling layers on a 4×4 neighbourhood (these are all common settings for CNNs), would need $\sim 400k$ parameters for $\sim 2.5m$ units. On the other hand, just the first fully connected layer of a vanilla NN would need $256 \times 256 \times 3 \times 32 \sim 6.3m$ parameters for only 32 units.

Biological Inspiration. We pointed out the close similarity between convolution and pooling layers in CNNs with retinotopic neuronal layers of simple and complex cells in primates' visual cortex. In fact, topographical maps of simple-complex cells have been found from lower to higher cortical areas in the ventral stream designated to visual recognition (see [Yamins and DiCarlo, 2016] and references therein [Hubel and Wiesel, 1962, Logothetis et al., 1995, Malach et al., 2002, Hung et al., 2005, Rust and DiCarlo, 2010]). In these maps, the receptive fields of simple cells are tuned to progressively more complex and extended visual stimuli, from edges, to simple patterns, to specific object parts, and so forth, playing the role of *templates* that encode more and more selective and abstract representations. At the same time, interleaved layers of complex cells introduce an increasing degree of spatial invariance to wider visual transformations [Hung et al., 2005, Rust and DiCarlo, 2010]. It has been also found that invariant responses are produced by complex cells by pooling responses of groups of simple cells tuned to transformed versions of the same stimulus.

Based on these observations, computational models of the ventral stream were proposed in the literature [Serre et al., 2007, Mutch and Lowe, 2008], that iterated convolution-pooling layers multiple times in order to progressively build more invariant and selective representations – in fact resembling the structure of deep CNNs. Taking inspiration from biology, these models, among which a well known example is *HMAX* [Serre et al., 2007], explicitly introduced representation invariance by employing transformed templates (e.g., rotated versions of the same filter). To this regard, a recent theory by Poggio et al. [Poggio and Anselmi, 2016]

hypothesizes that the main computational goal of the ventral stream is to compute neural representations of images that are invariant to the visual transformations commonly experienced. The authors also point out interesting analogies with the computations that are carried out in modern CNNs.

In further support of this analogy, there has been recent experimental evidence that latest CNN architectures ([Krizhevsky et al., 2012, Zeiler and Fergus, 2014]), which we will see in the next Section, not only can match and surpass human performance on challenging recognition tasks [He et al., 2015], but can also predict well neural population responses in the top two layers of the ventral visual hierarchy (V4 and Inferior Temporal cortex) – even without being constrained to match any neural data [Yamins et al., 2014, Cadieu et al., 2014]. In addition to yielding greatly improved models of visual cortex, the authors suggest that these results may also indicate that a process of biological performance optimization may have shaped neural mechanisms [Yamins and DiCarlo, 2016]. Indeed, we will also see in Sec. 4.2.2 that recent works exploring visualization techniques to analyze the properties of the representations learned at the different layers of deep CNNs, found also in these artificial models units tuned to increasingly more complex shapes, terminating with receptive fields covering almost the full image and selective for specific objects and object categories [Zeiler and Fergus, 2014, Yosinski et al., 2014b, Simonyan et al., 2014].

On the other hand, it must be noted that, differently from biological models like *HMAX*, the filters in modern deep CNNs are not constrained to be transformed versions of each other, but, as we will see in Sec. 4.1, are learned with the only goal of minimizing the recognition error. Therefore, these architectures are not invariant "by construction" to any visual transformation but translation (because of convolutions), and their invariance properties instead strongly depend on the data on which the filters are learned. As we will see in the following Section, it is for this reason that one key ingredient of the success of modern deep CNNs lies in the variability and size of the data corpora on which they are trained. To this regard, in Sec. 4.2, we will see some recent work investigating the invariance properties of deep CNNs and, in Chapter 7, we will address a similar study.

3.3 The Deep Learning (Re)Evolution

The first CNN architectures learned end-to-end were proposed by LeCun et al. [LeCun et al., 1989, LeCun et al., 1998] and are known as *LeNets*. At the time, they already renewed the interest in Deep Learning (DL) models for visual recognition, which had been mostly neglected in the literature due to the scarce empirical results provided by vanilla NNs. The interest in these models remained however pretty limited, until the groundbreaking work of [Krizhevsky

et al., 2012], who were able to train a very large CNN (known as the "AlexNet") and achieve astonishing performance on the challenging ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 [Russakovsky et al., 2015]. Since then, lots of architectural variations have been proposed, providing better and better performance on the ILSVRC. In the following, we briefly describe AlexNet and some well-established CNN models, also adopted in the experiments part of this Thesis.

3.3.1 AlexNet

To introduce this architecture, we make use of the following notation for the "prototypical" operations/layers of a CNN (recalling what we saw in Sec. 3.2):

- **Convolution layer (C)**: local convolution with respect to a bank of (learned) linear filters.
- **Spatial Downsampling**: usually included in the above operation by using strided convolutions.
- **Element-wise Non Linearity (nl)**: such as sigmoid functions [Bishop, 2006] or, more recently, Rectifying Linear Units [He et al., 2015].
- **Spatial Pooling layer (P)**: aggregates filter responses in a single component, for instance by taking the maximum value observed (max-pooling).
- **Fully-connected layer (FC)**: classical Neural Network layer where each unit outputs a weighted sum of all units in previous layer.

Often, also a normalization (n) of unit outputs is included. For instance, the Local Contrast Normalization adopted in [Jarrett et al., 2009], inspired by neuroscience evidence [Pinto et al., 2008], suppresses the output of each unit by a quantity proportional to the sum of the outputs of the other units at the same spatial position. The idea is to enforce competition between features at the same location in different feature maps.

With this "convention", we can represent AlexNet [Krizhevsky et al., 2012] as:

$$\begin{aligned} X \rightarrow & [C, nl, n, P] \rightarrow [C, nl, n, P] \rightarrow [C, nl] \rightarrow [C, nl] \rightarrow [C, nl, P] \rightarrow \dots \\ \dots \rightarrow & [FC, nl, drop] \rightarrow [FC, nl, drop] \rightarrow [FC, softmax] \rightarrow e_y \end{aligned}$$

where we kept the notation of Chapter 3 and X is the image and e_y is the one-hot encoding of its label (Fig. 3.7, Left). Each layer is characterized by the design parameters that we saw in Sec. 3.2.2, in such a way that the output of the last pooling layer is a vector (that is, a number of feature maps with a single pixel of spatial resolution). A certain number of fully-connected layers follows, in order to produce the final label. Here $drop$ is the Dropout regularization technique (that will be introduced in Sec. 4.1.2), adopted in the two fully-connected layers

which involve most of the parameters (in this case, ~ 53M over a total of ~ 60M). The model was trained with the standard backpropagation algorithm for ~ 90 epochs of mini-batch SGD on the ILSVRC 2012 training set. It finally reported a $\sim 17\%$ top-5 error rate on the object categorization task, by far surpassing the best "shallow" pipeline (based on Fisher Vectors) that achieved $\sim 26\%$. We refer to the original publication [Krizhevsky et al., 2012] for more details on the architecture, training and performance.

Note that the AlexNet architecture was not very different from the CNNs proposed in [LeCun et al., 1998]. It is now well-accepted that the larger (i) training set and (ii) model size (with training made feasible by empowered GPUs) plays a key role for the effectiveness of this approach. To this regard, we note that model selection is a critical, open, problem when dealing with deep CNNs: while the "AlexNet formula" proved to learn well on ImageNet, opening the way to many other variations, as we will see in the following, replicating the same approach with smaller models and datasets may not lead to as good results. Indeed, as we will show in Sec. 4.3, this limitation recently motivated the adoption of transfer learning approaches to "re-use" the representations learned by large CNNs on ImageNet, for other tasks and datasets, rather than learning new models from scratch.

3.3.2 Beyond AlexNet: ILSVRC 2012-15

Since the breakthrough of AlexNet in 2012, large attention was paid to deep CNNs in the machine learning and computer vision communities. In the following years, other CNN models, evolved from AlexNet, further improved the state of the art on the ILSVRC. For the studies carried on in this Thesis, we selected architectures that achieved best performance on the challenge between 2012 and 2015 (or smaller versions of such architectures).

In fact, the great enthusiasm raised by the stunning performances of deep CNNs progressively motivated the authors of the various architectures to publicly release not only the implementation of their winning models, but also the model weights themselves, as they have been learned on the ILSVRC 2012 training set. Today, there is an increasing trend among research groups to release the CNN models that they devised and trained, and a lot of models learned on the different visual tasks are publicly available. Accordingly, Deep Learning frameworks are progressively wide spreading, which, reducing the coding effort required to produce implementations of the above models on dedicated hardware like GPUs, consequently facilitate their sharing. In particular, in this Thesis we used the CAFFE [Jia et al., 2014] framework, one of the first and most efficient libraries to implement and train deep CNNs.

Below, we summarize the key features of the CNN architectures considered in the Thesis, providing also details on the models implemented in CAFFE and trained on the ILSVRC. In Fig. 3.7 we report their pictorial representations and in Fig. 3.8 their performance on the

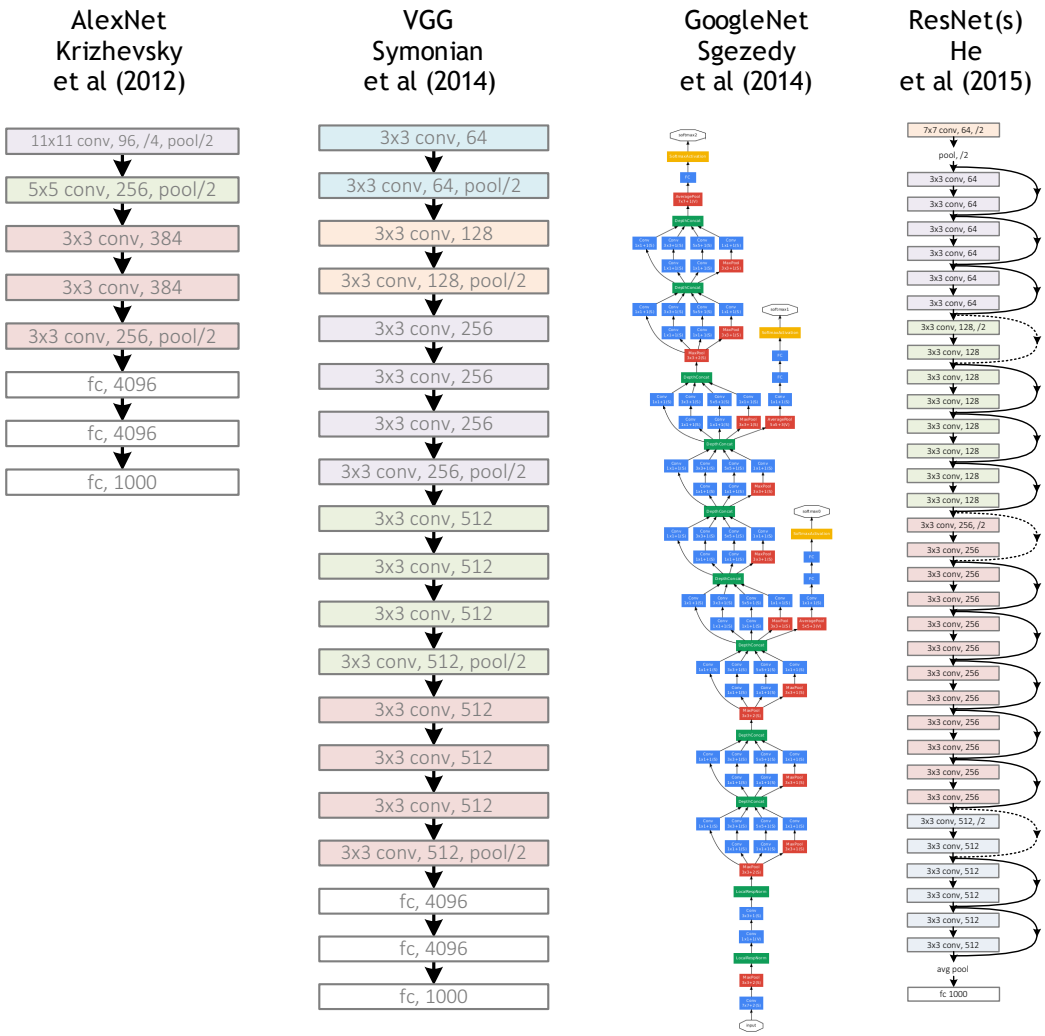

Figure 3.7: **The four CNN models employed in this Thesis** (best viewed in electronic form). Network pictures from K. He's ICML "Tutorial on Deep Residual Networks"[a] and from [Szegedy et al., 2015].

[a] http://kaiminghe.com/icml16tutorial/index.html

ILSVRC challenges throughout the years. Note that the CAFFE implementations that we used slightly differ from the original models proposed by the authors and usually report slightly different error rates on the ILSVRC benchmark: we refer to models' pages on CAFFE website for the specific (minor) differences of each architecture.

CaffeNet[1] A small variation of the AlexNet model, winner of ILSVRC 2012 [Krizhevsky et al., 2012]. As we saw, it concatenates 5C + 3FC layers comprising $\sim$ 60M parameters.

VGG-16[2] Second classified at ILSVRC 2014 [Simonyan et al., 2014]. It maintains the general

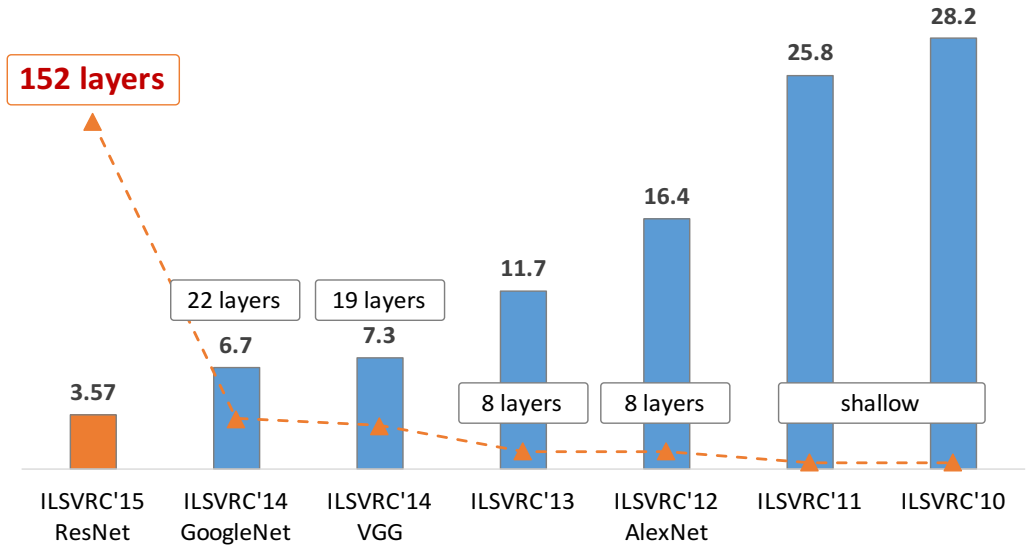

Figure 3.8: **ILSVRC Top-5 Error Rate Decrease of CNN models in latest years** (and depth increase). Picture from K. He's ICML "Tutorial on Deep Residual Networks"[a].

[a] http://kaiminghe.com/icml16tutorial/index.html

structure of AlexNet, but with 13C + 3FC layers comprising $\sim$ 140M parameters. It first proposes an efficient computational pattern employing only 3×3 convolutions and 2×2 pooling. On the other hand, it is relatively expensive to train in terms of memory and time because of the large FC layers (especially the first one, $fc6$).

GoogLeNet[3] Winner of ILSVRC 2014 [Szegedy et al., 2015]. It diverges from previous architectures in that it concatenates so called *inception* modules, which are an evolution of the *Network In Network* computational structure [Lin et al., 2013], based on the concept of empowering convolution layers by sliding, over the input, a non linear "micro" Neural Network, instead of the simpler linear filter. Moreover, it replaces the FC layers on top of the last convolution layer with an average pooling layer, leaving only a single FC layer with Dropout before the final *softmax*. Overall, the adoption of these tricks allows to reduce the number of parameters to only $\sim$ 4M, into a relatively high number of 22 layer. In order to facilitate the gradient flow through the layers during training, two *auxiliary classifiers* extract network predictions at intermediate steps, whose errors are summed up and backpropagated with the main network output.

ResNet-50[4] Stacking 50 layers with $\sim$ 20M parameters, it is a smaller version of the ILSVRC 2015 winner architecture proposed by [He et al., 2016], that was composed of 152 layers. The name of this kind of architectures is short for *residual networks*, due to the presence

of *skipping connections* between non consecutive layers. The idea is that, instead of learning an usual input-output relationship $y = f(x)$, the $residual$ between the input and the output $y = r(x) + x$ is learned instead. This trick is motivated by the fact that, when f has a compositional form, the second formulation can facilitate the gradient backpropagation through layers. They also adopt Batch Normalization (see Sec. 4.1.2) and, as in *GoogLeNet*, employ a single FC layer at the end, without Dropout.

Chapter 4

Learning and Transferring Deep Visual Representations

In the previous Chapter we introduced deep Convolutional Neural Networks (CNNs), architectures that can directly map images into labels and that can be trained end-to-end. We described the basic architecture of a CNN and presented some recent models well-known in the literature for their impressive performance.

In this Chapter, we complete the introduction to deep CNNs by describing the optimization algorithm that can be used to train them (Sec. 4.1). We then provide a short overview of why and how these models have been recently widely employed in a variety of applications: first, we review some works that, motivated by the stunning performances reported by these architectures, tried to understand the properties of the representations that they can learn from data (Sec. 4.2); finally, we present the main approaches recently proposed in the literature to exploit the powerful representations learned by these models on web data collections in other different visual tasks and domains (Sec.4.3). This is relevant for the Thesis since in Chapters 6 and 7 we will investigate the possibility of adopting these approaches as well in robotic settings.

4.1 Backpropagation

In this Section we describe how the SGD algorithm presented in Sec. 2.2 can be applied when the function to be optimized is a function of the form $f_\theta(X) = CNN_\theta(X)$, and the parameters θ to be learned are convolution filters and weight matrices. We will consider the mini-batch SGD setting, since this is the most commonly adopted when training modern CNNs. However, the computations we will develop can be adopted also in SGD or batch GD.

4.1.1 Sketch of the Algorithm

We start considering the setting of Eq. (2.20) and (2.21), where we replace the linear model with the CNN model:

$$G_{b_t}(\theta_t) = \frac{1}{m}\sum_{j=1}^{m} \nabla_\theta E_{(i_t)_j} + \lambda \nabla_\theta R(\theta_t) \tag{4.1}$$

$$\nabla_\theta E_{(i_t)_j} = \nabla_\theta \ell(y_{(i_t)_j}, CNN_\theta(X_{(i_t)_j})) \tag{4.2}$$

For the sake of brevity, we will not compute $\nabla_\theta R(\theta_t)$, but we will focus on $\nabla_\theta E_{(i_t)_j}$. We just clarify that usually $L2$ regularization is adopted, also called *weight decay*. Indeed, $\nabla_w \|w\|_2^2 = 2w$ and hence, when updating a parameter w in SGD, if this is affected by $L2$ regularization, we also decay it linearly. As we will see later on in this Section, other forms of regularization are also employed when training deep CNNs.

In order to proceed with SGD, we then need to compute the derivative of the prediction error with respect to model's parameters for a given training example (Eq. (4.1)). In the following, we will show how this computation can be developed for hierarchical structures like CNNs. For simplicity, we will suppress the underscript indicating the sample index in the training set:

$$\nabla_\theta E = \nabla_\theta \ell(y, CNN_\theta(X)).$$

Since $CNN_\theta(X)$ is the result of composing L layers of functions, each characterized by parameters that need to be optimized by SGD, the core idea of the *backpropagation* algorithm is to apply the *chain rule* when differentiating the error E with respect to some parameter affecting the function at some layer l. The name of the procedure comes from the fact that, as we will observe, the derivative of E with respect to parameters at layer l (that we can denote as θ_l) depends on the derivatives of E with respect to parameters in all layers higher than l (which are "external" in the function composition). Therefore, while, in order to compute the output prediction $CNN_\theta(X)$ given an image X, we need to propagate the signal *feedforward* in the architecture (by computing the output of layer 1, then feeding this quantity to the next layer in order to compute the output of layer 2, then 3, and so forth), in order to compute $\frac{\partial E}{\partial \theta_l}$ we need first to compute $\frac{\partial E}{\partial \theta_L}$, then by feeding this quantity to the previous layer $L-1$ we can obtain $\frac{\partial E}{\partial \theta_{L-1}}$, and so forth, propagating the signal *backward* through the network.

The idea of backpropagating derivatives was first proposed in [Werbos, 1974] for vanilla NNs and then, later on, it was applied also to CNNs [Rumelhart et al., 1988, LeCun et al., 1989, LeCun et al., 1998]. Indeed, the core idea of backpropagating derivatives in hierarchical architectures in order to optimize their parameters through GD or SGD can be applied to every compositional function that can be expressed as a Directed Acyclic Graph (DAG) [Goodfellow

et al., 2016], where feedforward computation provides predictions and backward computation provides gradients of prediction errors with respect to parameters.

In order to describe the algorithm for CNNs, we proceed by specifying the parameters θ_l with respect to which we want to differentiate, which depend on the layer kind. We start by considering fully-connected layers. These are the final layers in a CNN architecture and therefore the most "external" and the first that we need to differentiate when backpropagating derivatives. We hence derive the backpropagation formulas for a vanilla NN first. Then, we will show how these formulas can be adapted to convolutional and pooling layers.

Neural Networks

We consider a Neural Network as defined in Eq. (3.2) and compute the derivative of the prediction error with respect to the generic component in the i^{th} row and j^{th} column of the weight matrix W_l, denoted by $\theta_l = W_l^{i,j}$. We start applying the chain rule as follows:

$$\nabla_{W_l^{i,j}} E = \frac{\partial E}{\partial U_l^i} \frac{\partial U_l^i}{\partial W_l^{i,j}} = s_l^i X_{l-1}^j \tag{4.3}$$

$$\frac{\partial E}{\partial U_l^i} := s_l^i = \frac{\partial E}{\partial X_l^i} \frac{\partial X_l^i}{\partial U_l^i} = \frac{\partial E}{\partial X_l^i} f'_{NL}(U_l^i) \tag{4.4}$$

where we denote by v^i the i^{th} component of vector v. If we start with $l = L$ we obtain:

$$s_L^i = \frac{\partial E}{\partial U_L^i} = \frac{\partial E}{\partial X_L^i} \frac{\partial X_L^i}{\partial U_L^i} == -(y^i - X_L^i)$$

Indeed, by replacing in Eq. (3.2) the $cross\text{-}entropy$ loss and explicitating the $softmax$ normalization we obtain:

$$\frac{\partial E}{\partial U_L^i} = -\frac{\partial}{\partial U_L^i} \left(\log \left(\frac{e^{U_L^{t_{true}}}}{\sum_{h=1}^{T} e^{U_L^h}} \right) \right) = \begin{cases} i \neq t_{true} & : -(-X_L^i) \\ i = t_{true} & : -(1 - X_L^i) \end{cases} \tag{4.5}$$

For previous layers, we can proceed by backpropagating s_l^i as follows:

$$s_{l-1}^i := \frac{\partial E}{\partial U_{l-1}^i} = \sum_{c=1}^{n_l} \frac{\partial E}{\partial U_l^c} \frac{\partial U_l^c}{\partial U_{l-1}^i} = \sum_{c=1}^{n_l} s_l^c \frac{\partial U_l^c}{\partial X_{l-1}^i} \frac{\partial X_{l-1}^i}{\partial U_{l-1}^i} = f'_{NL}(U_{l-1}^i) \sum_{c=1}^{n_l} s_l^c W_l^{c,i} \tag{4.6}$$

The complete algorithm can be implemented as in Alg. 1, where the setting is the one of mini-batch SGD described in Eq. (2.20), (2.21), and (4.1), (4.2). The procedure to operatively

obtain the gradient of the error with respect to any network's weight is sketched in 1 and can be summarized as in the following. Once we sampled a mini-batch of m examples from the training set S and computed their predictions, we start differentiating the prediction error on this mini-batch with respect to each network weight. According to Eq. (4.1), we can differentiate the prediction error of each sample and then sum up the gradients over all samples in the mini-batch. In order to differentiate the prediction error of a sample with respect to each network weight, we apply backpropagation. Once we are done, we can update each weight by subtracting the sum of the computed gradients.

Algorithm 1 Mini-batch SGD applied to a Neural Network through $backpropagation$.

weight initialization
repeat
1. sample a mini-batch of m images $b_t = \{(X_{(i_t)_k}, y_{(i_t)_k})\} := \{(X_k, y_k)\}, k = 1, \ldots, m$
2. compute $k = 1, \ldots, m$ predictions X_{L_k} (FEEDFORWARD PASS)
3. init gradients to zero and accumulate over the batch (BACKWARD PASS)
   for $k = 1, \ldots, m$ **do**
   $y := y_k$
   $X_L := X_{L_k}$
   $s_L = -(y - X_L)$
   $\nabla_{w_L^{i,j}} E = \nabla_{w_L^{i,j}} E + \frac{1}{m} s_L^i X_{L-1}^j \quad i = 1, \ldots, n_l, \quad j = 1, \ldots, n_{l-1}$
   for $l = L - 1, \ldots, 1$ **do**
   $\nabla_{w_l^{i,j}} E = \nabla_{w_l^{i,j}} E + \frac{1}{m} s_l^i X_{l-1}^j \quad i = 1, \ldots, n_l, \quad j = 1, \ldots, n_{l-1}$
   $s_{l-1}^i = f'_{NL}(U_{l-1}^i) \sum_{c=1}^{n_l} s_l^c W_l^{c,i} \quad i = 1, \ldots, n_{l-1}$
   end for
   end for
4. compute the update $w_l^{i,j} = w_l^{i,j} - \gamma \nabla_{w_l^{i,j}} E$

until convergence

Convolutional Neural Networks

When applying backpropagation to CNNs, it is necessary to adapt the computation of the gradients, but the general structure of Alg. 1 remains. Therefore, we will provide the modified backward formulas for convolution and pooling layers.

The only parameters that we need to update are in convolution layers (since the pooling layers have no learnable parameters). Specifically, we are interested in computing the update of each weight of the generic z^{th} slice of the j^{th} filter. In the following, we will consider Eq. (3.3), slightly changing notation to a one more practical to deal with partial derivatives: we

will denote $W_{l,j}(u,v,z) := W_l^{j,z}(u,v)$ and, accordingly, $X_l(x,y,z) = X_l^z(x,y)$. Therefore the backpropagation of the derivatives through the generic layer l will be:

$$s_{l-1}^j(x,y) = \begin{cases} conv: & f'_{NL}(U_{l-1}^j(x,y)) \sum_{c=1}^{C_l} \sum_{u,v\in[-S_l,S_l]} s_l^c(x+u,y+v) W_l^{c,j}(-u,-v) \\ pool: & f'_{NL}(U_{l-1}^j(x,y)) s_l^j(\lfloor \frac{x}{P_l} \rfloor, \lfloor \frac{y}{P_l} \rfloor) \end{cases} \tag{4.7}$$

The parameter update for the convolutional layer will be:

$$\nabla_{W_l^{j,z}} E(u,v) = \sum_{x,y} s_l^j(x,y) X_{l-1}^z(x+u,y+v) \quad u,v \in [-S_l, S_l] \tag{4.8}$$

Note that, once we derive backward formulas for a layer kind, backpropagating derivatives through it is straightforward and can be done similarly to a standard feedforward pass, just changing the input and using the layer's backward formula instead of forward. In fact, this is how backpropagation is implemented in modern deep learning frameworks like Caffe [Jia et al., 2014], where for each layer both the forward and the backward interface is defined.

Note also that, in order to backpropagate the gradient, all unit activations from the forward pass must be kept in memory. This is an important aspect, which in practice prevents from using batch GD (or too large mini-batch sizes in SGD) for training large networks.

4.1.2 Best Practices

We conclude our review on deep CNNs and their training with some practical remarks.

Image Pre-processing. Subtracting the mean image of the training set from each image that is fed to the network –both at training and testing time– is a common practice, which has been proved to be beneficial for training [LeCun et al., 1998]. Another critical step is feeding the network with images of a specific size, usually determined by the architecture (images are usually cropped or resized in order to meet this requirement). Indeed, the architecture of CNNs is such that, as we have seen, recursive pooling and subsampling operations produce, at a certain layer, a 1×1 feature map, which can be given as input to fully-connected layers. However, this happens only for a specific input size.

Dataset Pre-processing. Shuffling the training set, in order to provide enough variability in each batch, is important for most settings and is standard practice. This is particularly critical in robotic settings, where the images are frames acquired online and their order in the training set may follow their temporal acquisition. In Sec. 2.2 we listed some possibilities that can be adopted when implementing SGD algorithms in order to sample the training

examples. In practice, data is used always more than once and multiple epochs of SGD are run. There is empirical evidence that randomly shuffling the training set at each epoch provides better convergence (i.e., the third choice in Sec. 2.2). However, also the second choice (i.e., maintaining the same sampling order at each epoch, cycling on the training set) is commonly adopted. Indeed, when dealing with large training sets, images are usually saved on disk into high throughput formats like in a single Lightning Memory-Mapped Database (LMDB), and re-shuffling them at each epoch may cause a remarkable computational slowdown.

Dataset Augmentation. Modern deep CNNs are characterized by millions of parameters and need lots (thousands to millions) of training examples in order to learn to generalize without overfitting. Since acquiring, and especially annotating, so many images requires a remarkable effort, it is common to artificially enrich the training set variability by applying simple transformations to the images. Usually this procedure is performed at runtime, when pre-processing the image before feeding it to the network, to avoid storage overheads. Common data augmentation approaches include geometric transformations, as random extraction of crops and horizontal reflections, or relighting transformations, altering the intensities of RGB channels in order to reproduce natural illumination changes. See [Krizhevsky et al., 2012, He et al., 2016] for some examples.

Regularization Techniques. We mentioned that $L2$ regularization is added to the empirical risk when training deep CNNs with backpropagation. Usually, a single value λ is chosen by cross-validation for all network weights (but it is possible in principle to choose different values for, e.g., specific layers).

Other forms of regularization often employed in the fully-connected layers (which are the most rich in parameters and hence the most exposed to the risk to overfitting) are Dropout [Hinton et al., 2012, Wager et al., 2013, Srivastava et al., 2014] or DropConnect [Wan et al., 2013]. These techniques introduce stochasticity in the forward pass of the network, in order to prevent co-adaptation between units. Specifically, in the forward passes during training a random fraction p of units is suppressed, simulating the training of multiple independent models. At test time, all units are multiplied by p, simulating the averaging over the ensemble of trained models.

Recently, Batch Normalization [Ioffe and Szegedy, 2015] proved to be effective and became standard practice in training deep networks. The motivation behind this technique is to address the problem known as *(internal) covariate shift*, namely the change in the distribution of network activations at each layer, due to the change of network parameters, during training. The authors introduce "normalization stages" between layers, where each unit activation is

normalized (independently) by subtracting the mean, and dividing by the variance, of all the activations of that unit over a mini-batch. To account for this normalization during the training, they also introduce a rescaling layer after each BN layer, whose parameters are optimized by backpropagation, such that the model can possibly choose to ignore BN for some layers.

Model Selection. There are many architectural choices that need to be specified when defining a CNN model, which go beyond the parameters that can be optimized by SGD. As we mentioned, the number of layers, the number and size of filters in each convolutional layer, the spatial support of each pooling layer, are only some of them. The training protocol must also be completely defined, in terms of number of epochs, learning rate (possibly layer-specific and usually decaying, with some policy that needs to be decided), the amount of regularization, and so forth. All these hyper-parameters must be assigned by cross-validation, as described in Sec. 2.1, by searching in a space where plausible ranges have been established (usually, based on some knowledge of the problem). Manual, grid or random search can be implemented, depending also on the dimension of the search space (see [Bergstra and Bengio, 2012] for possible advantages of random versus grid search).

Regarding, in particular, the number of epochs, during SGD the model is usually tested on the validation set (and saved) at every epoch, such that the model at the epoch with the lowest validation loss can be finally chosen. To this regard, Polyak averaging can be adopted [Polyak and Juditsky, 1992], considering averaged weight values over a number of epochs rather than the values at the very last iteration. Multiple SGD trials are performed when possible, in order to account for the stochasticity of weight initialization (and other sources of randomness as Dropout).

4.2 Understanding Deep Image Representations

In latest years, there has been an increasing interest towards the reasons behind the incredible performances provided by the CNN models presented in Sec. 3.3. The fact that the filters of these models are not constrained but rather fully learned from data, motivated a number of works in the literature to "look" inside the layers of these hierarchical architectures, in order to understand the invariance and selectivity properties of the intermediate representations conveyed by the networks. This interest is largely motivated also by the fact that, as we will see in the next Section, the representations learned by the presented CNN models proved to be effective not only, as expected, on the dataset on which they were trained (namely, ILSVRC or similar large-scale databases) but, surprisingly, also on a large variety of other visual tasks and datasets.

In the following, we introduce some useful notions related to the concepts of hierarchical representation and invariance (Sec. 4.2.1). We then provide an overview of some approaches that have been proposed in the literature to measure this property for deep CNNs (Sec. 4.2.2). The concepts introduced in this Section will be used in Chapter 7.

4.2.1 Visual Transformations, Manifolds and Sample Complexity

In [DiCarlo and Cox, 2007, DiCarlo et al., 2012], the authors build on the similarity between the structure of deep CNNs and computational models of visual recognition in the cortex and offer a useful geometrical perspective that may help visualizing and understanding the concept of representation selectivity and invariance. The global response of a population/layer of neurons to an image representing a certain object from a particular view is seen as a vector/point (we called it a *representation*), in a space with dimensionality defined by the number of neurons in the layer. Images of the object undergoing different identity-preserving visual transformations (as scaling, rotations, translations, and so forth), will produce different patterns of population activities, that will correspond to different response vectors (or image representations). We can think of these response vectors as points, defining a low-dimensional manifold in this high dimensional space. At the input level, the manifolds originated by different transforming objects will be "tangled" together. The computational goal of the ventral stream, and of modern CNNs as well, can hence be seen as transforming such input visual representations in order to map them into an output space where the manifolds of the different objects are clearly untangled.

We recall that in Sec. 3.2.2 we have seen the same idea from a machine learning point of view. Indeed, there we presented the problem of separating all possible images that contain one particular object from images of other objects as the problem of finding a suitable mapping, encoding images into a feature space where it was possible to "easily" separate the object class (in fact, the object manifold) from other classes, by positioning decision boundaries as, e.g., hyperplanes. In a bio-inspired model like a deep CNN, one can think of a decision boundary as a higher-level neuron (e.g., in the very last layer of a network) that predicts the object identity by thresholding a weighted sum of the neuron responses at the lower level [DiCarlo and Cox, 2007, DiCarlo et al., 2012].

Invariance and Sample Complexity. Within this perspective, if the goal is object *recognition*, at some layer the manifolds of the different objects must become not only separated, but also *collapsed*. In other words, the images will be encoded into representations that will be *invariant* to the visual transformations non relevant to recognize the object, while being *selective* for it. Such representations will have discarded information like the object pose, light, or position in

the image, while retaining the information necessary to predict its identity.

To this end, it has been recently shown [Anselmi et al., 2015, Anselmi et al., 2016] that relying on invariant representations reduces the sample complexity of the associated learning problem: by mapping all images of an object into, ideally, a single point in a feature space, the number of example images necessary to learn to predict the object's identity can be reduced to, ideally, just one. It was not by chance, therefore, that it was observed that neurons at higher stages of processing in the cortex tend to maintain their selectivity for objects across changes in view [Rust and DiCarlo, 2010]. Learning to recognize objects from few (often, just one) example images is a hallmark of human learning, and the fact that object manifolds are gradually untangled through nonlinear selectivity and invariance computations applied at each stage of the ventral pathway well explains this ability [DiCarlo et al., 2012, Poggio and Anselmi, 2016].

Relevance for Real World Visual Recognition. The above ideas found confirmation in the work of [Pinto et al., 2008, Pinto et al., 2011], where the authors addressed the problem of understanding the challenges involved in "real world" object recognition. To this end, they simulated an – apparently simple – real world visual recognition task, involving a few objects synthetically rendered under different visual transformations (like scaling, rotations, background change). Interestingly, they found that *HMAX*-like bioinspired models [Mutch and Lowe, 2008], explicitly conveying invariant representations, outperformed, on this kind of tasks, other BOW-like computer vision pipelines that had instead proved better on natural image classification datasets like Caltech 101 [Fei-Fei et al., 2007]. Building on this result, they pointed out the critical role of visual nuisances typically affecting real world recognition problems and advocated the adoption of suited benchmarks to investigate their individual effect (in contrast to largely employed "photographic-based" sets of images in the wild).

The effectiveness of *HMAX*-like models was also confirmed in a real world robotic setting involving the iCub robot [Ciliberto et al., 2013]. In particular, these findings motivated our contributions in Chapters 5 and 7, were, by means of deep CNNs trained to be robust to real world visual nuisances, we will build a pipeline that is proved to be able to recognize objects from few example images in challenging robotic scenarios.

4.2.2 Empirical Investigation of Representation Properties

Given, on the one hand, the critical importance of invariance in real world object recognition and, on the other hand, the stunning performance reported by modern CNNs in challenging large-scale natural image classification tasks, it is natural to ask whether the internal representations of such models are invariant. In the following, we report some relevant recent work

addressing this investigation with different approaches.

Visualization Techniques

A first approach to qualitatively investigate the properties of the internal representations of CNNs is to *visualize* them. To this end, the simplest solution is to just plot the unit responses at the different layers when an image is fed to the network. Although trivial, this strategy allows to have an idea of the sensitivity of the different units in the layers. For instance, it may be particularly interesting to visualize how the unit responses change in real time, while the network receives a stream of frames from a video sequence containing, e.g., an object undergoing some visual transformation. Indeed, all approaches that we will see in the following consider population responses at different layers and, in order to quantify the invariance of the network's intermediate representations, measure how these change when image sequences of this kind are fed to the network.

Another visualization approach consists in finding ways to "project" the population or unit responses back to the image domain. Backprojecting a population response can provide an idea of the relevant image content that has been retained by the image representation at that layer (see, e.g., [Mahendran and Vedaldi, 2015, Dosovitskiy and Brox, 2016]). Backprojecting a unit response instead can provide an idea of the receptive field of the unit at the image level, to find, e.g., the patterns to which the unit is tuned, as which objects or object parts (see [Zeiler and Fergus, 2014] or [Szegedy et al., 2014, Simonyan et al., 2014, Nguyen et al., 2015]). All these visualization methods are based on the idea of maximizing the unit response with respect to the input images. To this end, a common technique is to backpropagate the derivative of the unit response with respect to the image, in a very similar way the classical backpropgation algorithm backpropagates the derivative of the prediction error with respect to model weights. A useful overview of different techniques is provided in [Yosinski et al., 2014b].

These works confirm, at a qualitative level, the fact that units in deep CNNs trained on large datasets as ILSVRC are tuned to progressively more complex and "abstract" patterns when increasing the layer throughout the architecture, starting from Gabor-like receptive fields at lower layers and ending up in higher layers with units covering the whole image and selectively responding to particular object classes. While qualitative considerations on representation invariance could be drawn also with these techniques (see, e.g., [Mahendran and Vedaldi, 2015]), in the following we provide an overview of works that instead tried to *quantify* the degree of invariance of the network's internal representations at the different layers.

Measuring Invariance

Approaches. The invariance of an image representation to a certain visual transformation can be either evaluated "directly", i.e., by taking measurements over the representation itself, or "indirectly", by evaluating the performance of a classifier that uses that representation. In both cases, the setting is usually one where images representing different objects that undergo the considered transformation are mapped into the representation space, and the structure of the resulting manifolds is investigated. In the first case, measurements can be applied to quantify the shape of the manifolds, to evaluate for instance if images of a same object are clustered, how much the different objects/clusters are separated, and so forth. In the second case, the relationship between invariance and sample complexity is exploited (see Sec. 4.2.1), and the classification accuracy of classifiers trained in the representation space, on few example images per object (ideally, one) is used as an indirect measure of the invariance of the representation. To this regard, we recall that, if the representation is invariant to the considered transformation, all images of the same object will be mapped into vectors similar to the one(s) used for training (but different objects in general will be mapped into different vectors) and therefore objects will be recognized with high accuracy.

As we will discuss more in detail in Sec. 5.5.1, in the literature datasets that provide image sequences representing objects undergoing isolated visual transformations are limited. For this reason, most works investigating invariance adopt computer imagery and synthetically render the objects undergoing the transformations to be studied. Other works limit the analysis to geometric transformations digitally applied to images. Works investigating invariance on "real world" sequences are rare and in Chapter 7 we will provide a contribution in this direction.

In the following, we provide some examples of the different approaches.

Literature Overview. In [Lenc and Vedaldi, 2015], the authors consider geometric image transformations, like scaling and in-plane rotations, to quantify the *equivariance* properties of AlexNet's layers by learning the (linear) relationship that predicts the change in a layer's output given a certain image transformation. Within this perspective, they define invariance as a special case where such change is small or null. They also consider the interesting problem of evaluating to which extent representations learned by training multiple instances of CNN architectures on the same or different data are in fact *equivalent* although being characterized by different parametrizations. To this end, they adopt an approach based on the idea of assessing the drop in performance of "Frankenstein" architectures where some layers of a model have been "replaced" with others from different models.

In [Zeiler and Fergus, 2014], the invariance of the different layers of a CNN model similar to AlexNet (also trained on ILSVRC) is measured by simply taking the Euclidean distance

between representations extracted from transformed images, and a reference image. They consider again geometric transformations applied to natural images, like translations, in-plane rotations and scaling, by varying degrees, and find that, while small transformations can have a dramatic effect in the first layer, much less impact is measured at the second to last layer, being this latter nicely equivariant and quasilinear for translation and scaling. Indeed, the find the network output being almost invariant to translation and scaling, but, in general, not to rotation.

Differently from these works, the following studies generate visual transformations either synthetically, by rendering of 3D CAD models [Peng et al., 2015, Aubry and Russell, 2015], or by acquiring datasets for the purpose, as it is done in [Borji et al., 2016] (and as we will do with iCubWorld Transformations in Sec. 5.5).

In [Goodfellow et al., 2009] a battery of empirical tests is proposed to measure the change of visual representations (they consider stacked autoencoders or deep belief networks) to different image transformations. Interestingly, to our knowledge, this work is among the few which use natural videos of transforming objects for their evaluation (but unfortunately they do not release the collected dataset).

The work of [Aubry and Russell, 2015] may be interpreted also as a visualization technique, in that the authors apply a PCA-based approach to project the output of layers into a low-dimensional space and visualize it. They render 3D CAD models to generate image sequences with varying object type, style, color (which they consider "intrinsic" factors), and viewpoint, scale, light changes (that is, "extrinsic" factors), and exploit the control over such transformations to measure their effect on the projected layer responses.

In [Peng et al., 2015] the authors exploit computer generated imagery to measure the invariance of CNN layers to object textures, colors, background, pose and shape. With respect to previous works like [Goodfellow et al., 2009, Lenc and Vedaldi, 2015, Aubry and Russell, 2015, Bakry et al., 2015], they follow the same approach of [Pinto et al., 2011] and measure invariance "indirectly", by observing the performance of classifiers trained on image representations extracted as the output of CNN layers. In this Thesis, we follow the same approach to measure the invariance of the considered deep CNNs.

Recently, [Bakry et al., 2015], building on the work of [DiCarlo and Cox, 2007], tried to get insights about how tangled/untangled the manifolds of different object instances are, in the internal layers of CNNs trained on ILSVRC. Their analysis focuses on 3D viewpoint changes. In fact, they use as data sources the Washington RGB-D [Lai et al., 2011] and the Pascal 3D+ [Xiang et al., 2014] datasets, which contain mainly this kind of variation. They employ different methods based on linear classifiers, k-Nearest Neighbors (k-NN) and direct measurements on the manifold structures, to investigate whether the viewpoint manifolds of object instances are collapsed or preserved throughout network layers. Similarly to [Zeiler and

Fergus, 2014], they find that the representation conveyed by CNNs trained on ILSVRC tries to separate the viewpoint manifolds of different instances, while preserving – and not collapsing – them, up to the very last layer, which finally enforces view invariance.

4.3 Transferring Deep Representations

In Sec. 3.3, we have presented some well-known models of deep CNN that in latest years remarkably advanced the state of the art of large-scale image classification, starting with the ImageNet benchmark. We have also pointed out how these architectures are characterized by millions of parameters and therefore require tons of examples to be successfully trained. In Sec. 4.1, we have formalized their training algorithm and in Sec. 4.1.2 we reported on some useful procedures widely adopted in practice to make the learning converge on these large-scale datasets. While these achievements are surely remarkable, it is natural to ask what is the applicability of these methods to smaller settings, where the training data may be less abundant and often also the recognition task is characterized by fewer classes.

In the following, we will see that recent empirical evidence has shown that, while scaling down these architectures and training them on smaller tasks is unlikely to provide particularly good results (also because of model selection issues), on the other hand, impressive performance is achieved by "transferring" the representations learned by these models on large web data corpora, to new, potentially smaller, domains. Different approaches to implement this strategy have been investigated in the literature. In this Section we review two of the most well-established methods, based on the idea of re-using the learned representation (Sec. 4.3.1) or fine-tuning it on the new domain (Sec. 4.3.2). These are also the two methods that will be empirically assessed in our experiments in Chapters 6 and 7.

4.3.1 Feature Extraction and Kernel Methods

In Sec. 4.2.2 we have reviewed several works that visualize and analyze the intermediate representations extracted from the internal layers of CNN models, specifically the ones introduced in Sec. 3.3. These works show that there are units tuned to more specific and more abstract image patterns from one layer to the following, starting with topographic maps encoding low-level local features, and ending up with vector representations encoding task- or domain-specific features, indicative of the global image content like objects and scene elements.

Within this perspective, there has been a number of works that explored the possibility of applying such intermediate image representations also to other datasets and tasks, beyond the one where they have been learned. The idea is to use a previously trained CNN model as a "black-box" feature extractor, which is fed with an image and provides its representation

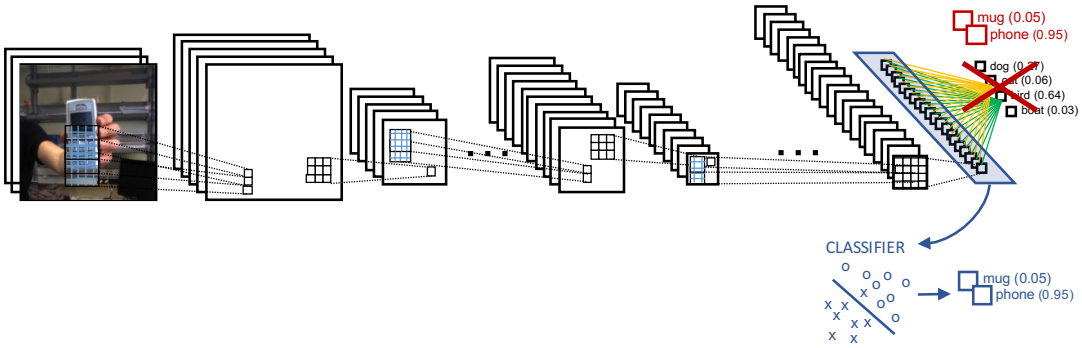

Figure 4.1: Example of a Convolutional Neural Network (Sec. 3.2.2) and of the two knowledge transfer approaches considered in this Thesis. **(Blue pipeline) feature extraction**: in this case the response of one of the layers is used as image representation to be fed to a classifier like the RLSC (see Sec. 4.3.1), which is trained on the new task. **(Red pipeline) fine-tuning**: in this case the network is trained end-to-end on the new task, by replacing the final layer and using the original model as a "warm-restart" (see Sec. 4.3.2).

as the output of one of the intermediate layers. Once the images from the new domain are encoded into such representations, standard classifiers, like RLSC or SVMs, are trained in the feature space for the new task (similarly to what is done in dictionary learning). This strategy, depicted in Fig. 4.1 (Blue pipeline), can be interpreted as using the original CNN model as is, until the layer at which the representation is extracted, and replacing the remaining part of the architecture with a new classification layer, trained on the new dataset.

The effectiveness of this approach has been empirically demonstrated in many works. [Donahue et al., 2014] are among the first to investigate whether features extracted from modern deep CNNs (they consider AlexNet) trained on a large object database (specifically, ImageNet) have sufficient representational power to perform other visual recognition tasks by using simple linear classifiers. They report results showing that this strategy reliably outperforms other approaches based on traditional features, analyzing also how the choice of the extraction layer impacts generalization performance. Their remarkable results also motivated the release of the CAFFE framework, namely "Convolutional Architecture for Fast Feature Embedding", published with the aim to provide researches with an easy tool to "re-use" pre-trained CNNs [Jia et al., 2014].

To further investigate these ideas, [Sharif Razavian et al., 2014] conducted a series of experiments using another CNN model, namely, the *OverFeat* network [Sermanet et al., 2014], trained on the ILSVRC 2013. They employed the output of internal layers as image representation to tackle diverse recognition tasks, gradually moving further away from the original

task/data the model was trained on, including scene recognition, fine-grained recognition, attribute detection and image retrieval, and confirmed superior results compared to state-of-the-art systems on each of them. A similar empirical assessment was conducted in [Azizpour et al., 2015], where the authors investigate the transferability of such representations w.r.t. different parameters (related to the network training and the feature extraction), reporting interesting results which indicate a correlation between the performance on the tasks at hand and their "distance" from the original ILSVRC.

The potential of this approach is even more relevant in that CNN models trained for object recognition can also be exploited as feature extractors to tackle object localization and detection tasks [Sermanet et al., 2014, Girshick et al., 2014]. In this setting, "salient" regions are detected in the image and encoded into CNN representations, which are finally fed to classifiers in order to obtain a prediction for each region. As anticipated in Sec. 1.2.1 and described in more detail in Chapter 6 (Appendix 6.A.1), we will adopt this approach to localize objects in the robot's visual field and recognize them by exploiting CNNs pre-trained on ImageNet.

Over the years, the generalization capabilities of CNN features have been investigated for the different architectures that have been subsequently proposed in the literature, like *VGG* and *ResNets* (see, e.g., supplementary material in [Simonyan and Zisserman, 2015] and [Mahmood et al., 2016]). To this regard, in Chapter 6 we will consistently compare the performance of features from the different models presented in Sec. 3.3 on several visual recognition tasks on the iCub and R1 humanoids.

4.3.2 Fine-tuning

Another approach that proved to be effective is to use a CNN trained on the large-scale dataset to "warm-start" the learning process on other (potentially smaller) tasks and datasets. This strategy, known as *fine-tuning*, consists in performing back-propagation on the new training set, initializing the parameters of the network to those previously learned (see Fig. 4.1 (Red pipeline)).

The only required modification to the network structure is in the output layer, in order to address the new task (e.g., by changing the number of units to account for the new number of classes to be discriminated, or changing the loss if the new task is not multiclass classification, and so forth). A crucial difference between training a network "from scratch" and fine-tuning it is that, in the latter case, the parameters of most layers are initialized with the result of the first optimization, and may not need to be changed dramatically during the second training phase. Their learning rates (that is, the step size used by the stochastic gradient descent during backpropagation –see Sec. 4.1) can therefore be set to smaller values, or even to zero (no update is done). The learning rate of the output layer, instead (or, of all layers which are

learned from scratch) is usually maintained to higher values. In practice, the warm-restart allows to train the CNN on much smaller datasets, in significantly shorter training times.

[Agrawal et al., 2014] investigated the applicability of fine-tuning to different datasets and tasks, finding that, while large CNNs can also be trained from scratch using a relatively modest amount of data, pre-training on ImageNet significantly improves performance. [Chatfield et al., 2014] provide an extensive in-depth evaluation of both fine-tuning and feature extraction adaptation techniques, comparing them to classical pipelines (based on, e.g., dictionary learning with Improved Fisher Vectors [Perronnin et al., 2010]), on the Pascal VOC [Everingham et al., 2010] and Caltech [Griffin et al., 2007] reference benchmarks. The authors rigorously test many different method variations, considering also subtle technical details, empirically demonstrating that deep learning approaches definitely outperform shallow methods by a large margin.

Interestingly, [Huh et al., 2016] invert the question, and investigate which aspects of ImageNet are more critical for learning CNN models that convey general image representations. They train AlexNet varying the amount of examples, number of classes, ratio between examples per class and classes, etc., and evaluate performance when the resulting model is fine-tuned on other tasks, finding that some changes in the pre-training data, thought to be critical (one above all, the need for extremely large training sets), in fact do not significantly affect transfer performance.

Fine-tuning Issues. A potential advantage of fine-tuning is that it can adapt the parameters of all layers to the new problem, rather than only those in the final layers. However, this flexibility comes at the price of a more involved training process. Indeed, the performance of a fine-tuned network depends on the setting of the (many) hyper-parameters typically involved in training deep CNNs. In particular, fine-tuning can be prone to overfitting, eventually disrupting the visual representation provided by the "off-the-shelf" CNN (where we indicate with this term the model trained on the web database and publicly available). Indeed, we have seen that a key factor in the representational power of features from deep CNNs lies in the variability of the web-collected data corpora on which they are learned. In this sense, a relevant issue is whether it can be in general more or less favorable to adapt the intermediate layers of the network with respect to the new training data (i.e., with higher learning rates) or keep a more conservative approach (i.e., with zero or very low learning rates). A certain caution is required therefore when setting "the degree of adaptation" with respect to the size and richness of the new dataset.

[Soekhoe et al., 2016] specifically address these issues, by comparing the training from scratch with fine-tuning, progressively "freezing" more layers, on larger or smaller target tasks, and showing that, in the latter case, freezing lower layers is critical to avoid overfitting. [Oquab et al., 2014] investigate different solutions that, starting from AlexNet, learn from scratch

fully-connected layers, while fixing convolution ones, and report state of the art results on Pascal VOC. This fine-tuning protocol in fact can be seen as a feature extraction stage, followed by a classification stage when the predictor is a multi-layer Neural Network.

[Yosinski et al., 2014a] take a different perspective and adopt fine-tuning as a tool for evaluating the "degree of generality" of intermediate representations, which they define as the extent to which such representations can be used for other tasks. They progressively learn from scratch more layers, starting from the output one and freezing, or adapting, lower ones. Their study confirms that the specialization of higher layers to the original task comes at the expense of a drop in generalization performance, negatively affecting transferability. Nevertheless, they find that employing features transferred at any layer is better than using random weights, even on rather distant tasks. Also, they interestingly find that representations which are pre-trained and then fine-tuned provide even higher generalization performance on further tasks.

In Chapters 6 and 7, we will consider fine-tuning of off-the-shelf CNN models in robotic settings, addressing these issue and proposing a solution based on a combination of fine-tuning and feature extraction.

Chapter 5

The iCubWorld Project

In this Chapter we present iCubWorld, a data acquisition project started in 2013 and consistently developed during this Thesis. Within this project and as an integral part of the Thesis, two new image datasets for visual object recognition have been acquired and publicly released. In the following, we will introduce the project and describe these datasets.

This Chapter is important for understanding the motivation of the studies reported in Chapters 6, 7 and 8, that will be conducted on the datasets here introduced.

5.1 Introduction

The main goal of this project is to benchmark and progressively develop the visual recognition capabilities of a humanoid robot like the iCub [Metta et al., 2008, Metta et al., 2010]. To this end, iCubWorld datasets are collections of images recording the iCub's view while it observes objects in its typical environment (a laboratory or an office). Specifically, we designed an acquisition setting that reflects the prototypical visual experience of a humanoid robot during a natural interaction with a human, and collected corresponding image datasets. This project is the result of a collaboration between the iCub Facility and the Laboratory for Computational and Statistical Learning at IIT, and the SlipGURU research group at the University of Genoa, started in 2013 with the aim to investigate computer vision and machine learning approaches to the problem of visual recognition in robotics.

The first two releases, with the first benchmarks, were published in 2013 [Fanello et al., 2013b, Ciliberto et al., 2013]. These datasets were relatively small, and served more as proof of concept for the feasibility of the acquisition approach. On the other hand, they highlighted how typical computer vision methods (based, e.g., on dictionary learning), which, as we saw in Sec. 1.3.2, were not commonly employed in robotics, could instead be effective for solving visual recognition problems in these settings, opening the way for further investigation.

As part of this Thesis, the original acquisition setup has been substantially improved, in order to acquire two new releases of increasing size, namely ICUBWORLD28 and ICUBWORLD TRANSFORMATIONS, aimed at investigating complementary aspects of robotic visual recognition. These two releases are large and allow to extensively test the behaviour of latest deep CNNs when trained in robotic settings, offering a faithful and reproducible benchmark for the performance that we can expect from the real system. While ICUBWORLD TRANSFORMATIONS will be the reference benchmark for the studies in Chapters 6 and 7, ICUBWORLD28 will be employed in Chapter 8.

The new acquisition setup is part of a complete "collect-train-deploy" framework (namely, the ICUBWORLD FRAMEWORK) that will be described in Chapter 9. The framework will be soon made available to allow users to (i) acquire visual data by interacting with the robot and afterwards (ii) train/benchmark deep networks on the collected images, in order to finally (iii) deploy them on the robot. We plan to further expand ICUBWORLD by relying on this infrastructure, progressively including more object categories in order to reproduce a typical domestic or office environment, that is, the iCub's world as seen by its own eyes. All ICUBWORLD releases are publicly available on the ICUBWORLD website [1].

In the following, we first provide a general overview of existing datasets for visual object recognition, as well as the motivation and spirit of the project (Sec. 5.2). We then present the acquisition setting (Sec. 5.3) and finally focus on the description of the ICUBWORLD28 (Sec. 5.4) and ICUBWORLD TRANSFORMATIONS (Sec. 5.5) datasets.

5.2 Related Work and Project Goal

In the last two decades, several datasets for visual object recognition have been proposed in computer vision and robotics. In the following, we roughly divide them into three main groups. We will then proceed by pointing out how some critical limitations of existing datasets motivated us to start the ICUBWORLD project.

5.2.1 Datasets for Visual Object Recognition

Computer Vision Datasets. Typical computer vision benchmarks focus on the task of object categorization, known to be a more challenging problem than object identification. The size and difficulty of the released benchmarks progressively increased over the years, starting with Caltech-101 [Fei-Fei et al., 2007] to Caltech-256 [Griffin et al., 2007], LabelMe [Russell et al., 2008], the Pascal VOC [Everingham et al., 2010, Everingham et al., 2015], up to more recent ImageNet Large Scale Visual Recognition Challenges (ILSVRC) [Russakovsky et al.,

[1] https://robotology.github.io/iCubWorld/

2015] and Microsoft COCO [Lin et al., 2014]. The progress in public datasets availability has played a fundamental role in the advancement of the state-of-the-art of recognition methods, culminating in the great success of deep learning recently marked by the ILSVRC 2012-15. A major limitation of these datasets is that each object instance appears only in a single image, in a random context, i.e., they contain only category-level and not instance-level variability.

Turntable Datasets. On the other hand, robotic datasets for visual object recognition historically focused more on the problem of object identification, and most of them are instead specifically meant to provide multiple images of objects while undergoing viewpoint transformations. The COIL-100 [Nene et al., 1996], SOIL-47 [Koubaroulis et al., 2002], ALOI [Geusebroek et al., 2005], Washington RGB-D [Lai et al., 2011], KIT [Kasper et al., 2012], 3DNet [Wohlkinger et al., 2012], BigBIRD [Singh et al., 2014] and SHORT- 100 [Rivera-Rubio et al., 2014] dataset are only some of the most famous available benchmarks. This trend is clearly due to the different application domain, robotics being characterized by the need of identifying specific instances, e.g., for manipulation.

However, these datasets typically approach the recognition problem in relatively simple scenarios, affected by low levels of noise/clutter. The images are usually acquired in "turntable" settings, with uniform background, scanning the scene at fixed camera positions. They mainly contain strictly controlled changes in the 3D point of view, and light in some cases. Other natural visual transformations like scaling, background and setting changes, occlusions, and so forth, that occur in practice, are underrepresented. In fact, as we already anticipated, most of them are not directly targeted to visual recognition, but rather on building on it to investigate grasping/manipulation. Hence, in order to provide accurate annotations of, e.g., the object pose in each image, necessary for reconstruction of point clouds or other 3D models, they constrain the acquisition setting. See [Firman, 2016] for a comprehensive review of RGBD-D datasets.

Hence, although they are acquired in real world scenarios, there is still a gap to be bridged when training recognition systems on them and then deploying to robot agents. Note that this aspect is extremely relevant, since visual biases make it typically difficult to generalize prediction performances across different datasets and applications [Pinto et al., 2008, Torralba and Efros, 2011, Model and Shamir, 2015].

Robot-acquired Datasets. In order to start bridging such a gap, one straightforward approach is to acquire benchmarks on the robot platform itself. The resulting dataset would offer a natural testbed for the visual recognition system, as close as possible to the real application, ensuring that, in particular, the performance measured *off-line* on the benchmark can be expected to hold on the actual robot. Using a robot to acquire a dataset provides other useful advantages, first

among others, on the annotation procedure, that can be automatized by relying on contextual information and interaction with the robot.

Unfortunately, datasets acquired by recording directly from robot cameras are extremely few in the literature. In recent years, we are aware only of [Ramisa et al., 2011, Meger and Little, 2013]. The Rutgers APC RGB-D Dataset [Rennie et al., 2016] is acquired by mounting a depth camera on the end joint of a robot arm. However, it is specifically meant for the Amazon warehouse Picking Challenge (APC), involving the grasping of objects placed inside the bins of shelving unit. Hence, it presents the same limitations of typical RGB-D turntable datasets.

One major limitation of this acquisition approach may be scalability, which is particularly critical in order to collect the necessary data to train modern deep networks. Indeed, making robots physically interact with an object requires much more time than browsing images from the Internet. However, as we mentioned, it is also true that in the robotic setting self-supervision techniques can be exploited in order to automatize, not only the annotation, but also the acquisition itself. Indeed, this is the approach pursued by more recent works as [Oberlin et al., 2015, Pinto and Gupta, 2016, Levine et al., 2016b]. In particular, Oberlin et al. [Oberlin et al., 2015] and Levine et al. [Levine et al., 2016b] "parallelize" the acquisition by employing multiple robots: while in [Levine et al., 2016b] the parallelization is implemented directly in the laboratory, the Million Object Challenge [Oberlin et al., 2015] proposes an infrastructure to involve multiple research groups using Baxter robots [2] to contribute to the acquisition of a large, shared, database of robots' experiences.

One drawback of a totally self-supervised acquisition is that it can prevent from controlling (or also introducing) visual nuisances. Indeed, these works mainly focus on collecting large, "wide" data in order to train deep learning systems for, e.g., detection of grasp points: they do not address the study of the visual recognition problem and are not interested in the analysis of the effects of different noise factors on object identification/categorization performance.

5.2.2 iCubWorld

As it can be observed, none of the above datasets is really suited to extensively study the behaviour of recognition systems in what can be considered the "prototypical" visual experience of a humanoid robotic agent. To this regard, in Sec. 1.4 we considered a hypothetic scenario where a robot is deployed to a user and is required to learn to recognize new objects "on the fly", interacting with humans. We listed some important aspects, characterizing such a setting, which we try to address within this Thesis, starting from the collection of suitable benchmarks. iCubWorld datasets therefore separate from most previous work, in that they are meant to

[2] http://www.rethinkrobotics.com/

model the growing visual experience of a robot, providing a benchmark reproducing typical uncertainties faced by the robot's recognition system during a real world task. Images are directly recorded from the iCub's cameras during a natural interaction with a human teacher (detailed in Sec. 5.3.1) in a semi-controlled setting where objects are captured while undergoing "natural" transformations.

Interestingly, a novel large-scale RGB-D dataset has been recently released by Choi et al. [Choi et al., 2016], which, even if not directly meant as a benchmark for visual recognition (but rather for 3D object reconstruction), is very close to the spirit of ICUBWORLD. The benchmark aims to simulate the broad deployment of 3D scanning systems to consumers and it is composed of over $\sim 10k$ RGB-D sequences representing objects acquired by "unexperienced" operators in their apartment. The operators are just instructed to acquire clean and complete sequences. It is reasonable, in fact, to expect that potential users will be willing to collaborate, showing clearly the object to the system (no matter if this is for 3D reconstruction or visual recognition). The operator holds the object in the hand and rotates it in front of the acquisition setup. The result is somehow similar to ICUBWORLD, considering that the setup consists of a fixed camera and not an interacting robot and, as anticipated, the collection is not structured for visual recognition.

As it will be described in the following, we took advantage of the robotic platform to reduce the human effort in the acquisition-annotation process, by semi-automatizing the procedure and allowing therefore to scale the collection to many objects/sequences. Moreover, other laboratories using the iCub could rely on the same infrastructure and contribute to future releases, similarly to the Million Object Challenge [Oberlin et al., 2015]. Ultimate goal is therefore the construction of a shared database for the study of visual recognition in robotics under different perspectives. Such database would contain multiple releases, "tagged" according to multiple, specific aspects to be investigated. Within this Thesis, for instance, we addressed the ones pointed out in Sec. 1.4, namely the study of representation invariance (ICUBWORLD TRANSFORMATIONS) and incremental and lifelong learning (ICUBWORLD28). Further ICUBWORLD releases will be (hopefully) collected, to address other critical aspects emerged from these studies (pointed out in Chapter 10).

We finally care to note that, since, as also motivated in Sec. 1.3.2, a primary goal of this project was to evaluate the performance that can be achieved by using deep learning methods purely on the visual input, at present we did not make extensive use of the robot's physical capabilities (e.g., object manipulation, exploration) to acquire ICUBWORLD releases. However, the acquisition setting is easy to expand to include more different perceptual scenarios. For instance, a new version of the acquisition application, where images are recorded while the robot manipulates the objects, is already available.

5.3 Acquisition Setup

In this Section, we introduce the acquisition scenario of ICUBWORLD (Sec. 5.3.1) and describe the acquisition setup (Sec. 5.3.2) implemented to collect the ICUBWORLD28 and ICUBWORLD TRANSFORMATIONS datasets (presented, respectively, in Sec. 5.4 and 5.5).

5.3.1 Acquisition Scenario and First Releases

Images in ICUBWORLD datasets are frames recorded from the cameras of the robot while it observes an object. Image annotation is provided in terms of the label of the object, and a bounding box around it. In the following, we describe the Human-Robot-Interaction (HRI) protocol, first proposed in [Fanello et al., 2013a], that we use to automatically acquire annotated images, by exploiting the interaction with the robot and the spatio-temporal context.

The only external supervision is in the form of a human teacher verbally providing the label of the object that is going to be acquired. The teacher approaches the robot and shows the object in the hand; the acquisition starts with a verbal command containing the object's label, e.g. "This is a *sprayer*" (see Fig. 5.1). During the acquisition, the robot focuses on the object of interest, that is localized in the visual field by using bottom-up visual cues and prior assumptions about the scene configuration. We exploit bottom-up cues because relying on model-based tracking methods is neither feasible nor needed in this application. Indeed, we want to focus and maintain the robot's attention on an unknown object, without the need of performing pre-scanning operations that would add overheads to the acquisition procedure. On the other hand, we have prior information, about the expected object's position/motion in the scene, which, as we will see in the following, can be usefully exploited as self-supervisory signal to automatically localize the object and produce the bounding box annotations.

Two acquisition modalities are possible:

Human Mode: the teacher holds the object in the hand, freely moving it, and the robot focuses and tracks the object by exploiting bottom-up cues like motion or disparity (detailed in the following).

Robot Mode: the robot takes the object in the hand and explores it by moving the hand in a random o predefined way, focusing on it by using knowledge of its own kinematics.

We record incoming frames from the robot cameras, together with the information on the bounding box in each frame, for a fixed time period (Human Mode) or for the duration of the exploration period (Robot Mode). The procedure is repeated for all the objects to be acquired.

Object Tracking by Independent Motion Segmentation. Since color-based methods work under strict assumptions on the light condition, the background structure (preferably uniform,

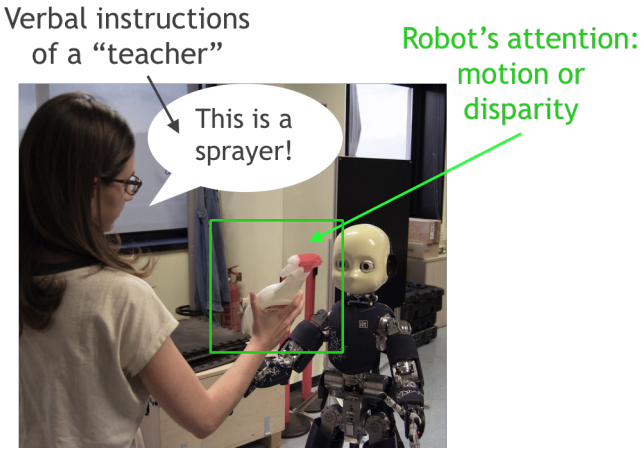

Figure 5.1: **Acquisition Scenario.** A teacher standing in front of the robot pronounces the label of the object of interest and shows it to the robot, by holding it in the hand. The iCub can either track the object while the teacher moves it (Human Mode, depicted), or take it in its hand (Robot Mode).

like a table or a wall) and generally fail in cluttered settings, in the first implementation of the acquisition protocol, a motion-based segmentation was adopted instead. Specifically, the independent motion detection method proposed in [Ciliberto et al., 2011, Ciliberto et al., 2012] was used, which compensates the ego-motion of the camera and segments blobs of pixels consistently moving in the scene. This strategy works well based on the fact that, of course, during the acquisition, the human teacher moves the object. Indeed, a certain degree of motion is naturally applied to the object in order to show it from different faces and poses: an independent motion detection routine as the one in [Ciliberto et al., 2012] can segment the object if this motion is continuous and with a sufficient speed. With this technique, the coordinates of the centroid of the segmented blob on the image plane are computed, and a square bounding box (of arbitrary radius), around the object, is provided. Usually, the radius is chosen (empirically, depending on the acquisition condition) and kept fixed for all images in the dataset.

First releases

In the following, we briefly outline for completeness the first two ICUBWORLD releases acquired in the original scenario described above.

Hello iCubWorld. This is the first release, where the acquisition setup is validated and an initial benchmarking of recognition methods is published [Fanello et al., 2013a]. In Table 5.1

and Fig. 5.2 we report an overview of the dataset and some example images. It is composed of 7 objects: each object was acquired in Human and Robot Modes, repeating the procedure twice to get separate train and test sequences. During the acquisition, a square bounding box around the object was automatically cropped from 320×240 images streaming from the left camera, and saved.

Hello iCubWorld	
#categories	7
#objects/category	1
acquisitions/object	Human & Robot Mode Train & Test
#frames/acquisition	500
tracking cue	Independent Motion
acquired cameras	Left
full images available	No

Table 5.1: Dataset overview.

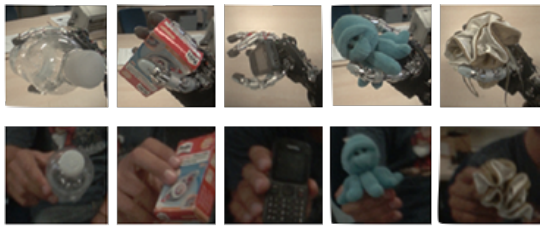

Figure 5.2: Example images showing objects acquired in Human and Robot modes.

Groceries. This is the first release to study object categorization. It is composed of 40 objects divided into 10 categories. For training, 3 objects per category were acquired in Human Mode, by automatically cropping and saving a square bounding box from 640×480 images streaming from the left camera. For testing, 1 new object per category was acquired in the same condition, and also in other 2 different conditions, in order to study generalization across settings (specifically, by changing the human operator and by using the Robot Mode). A small test set where the object is manually segmented and the background is removed is also provided. This dataset was benchmarked in [Fanello et al., 2013b, Ciliberto et al., 2013]. An overview and some example images are provided in Table 5.2 and Fig. 5.3.

5.3.2 New Acquisition Setup

In this Section we briefly mention the two main improvements that were introduced into a new implementation of the described acquisition setup. These changes were necessary for the collection of the latest ICUBWORLD TRANSFORMATIONS release, presented in Sec 5.5, and the development of the project as described in Sec. 5.2. While here we just highlight the innovative aspects of the novel acquisition setup, a more detailed description is left for Chapter 9, where the applications developed along with the Thesis investigation will be described.

Groceries	
#categories	10
#objects/category	3
acquisitions/object	Train: Human Mode 4 Test Sets
#frames/acquisition	200
tracking cue	Independent Motion
acquired cameras	Left
full images available	No

Table 5.2: Dataset overview.

Figure 5.3: Example images from the dataset showing object instances belonging to the 10 categories.

Object Tracking by Depth Segmentation. Relying on motion detection to localize the object forces the human to continuously move the object during the acquisition. As a result, the quality of the images may be degraded by blur, and the acquisition of sequences containing arbitrary visual transformations, possibly requiring the object to be still, is prevented. On the other hand, appearance-based approaches are typically not robust enough, and we observed that model-based approaches require a-priori knowledge about the object to be acquired. Indeed, the most distinguishing feature in the considered acquisition setting is simply the fact that the object of interest is usually always closer to the robot than any other element in the scene.

Starting from this observation, a new object tracking procedure, based on the disparity cue, has been developed and employed for the acquisition of the latest ICUBWORLD TRANSFORMATION release. Specifically, the closest blob in the scene is segmented in the disparity map (as the lightest region) and its centroid is used to make the robot focus on it. The centroid, together with the blob's tightest enclosing rectangular region, are recorded also as localization annotations. We refer to [Pasquale et al., 2016b] for a complete description of the resulting attention system.

Scalability. Another important aspect that was addressed in the implementation of the new acquisition setup was increasing the number of acquirable sequences. Indeed, the size of previous ICUBWORLD releases was inadequate to deal with modern deep learning methods. To this end, we developed an application to scale the described acquisition procedure to hundreds of objects, to be collected also during multiple sessions. This required the automation of many operations that before were performed manually during the acquisition phase and, more importantly, of *all* the post-acquisition steps required to format the raw streams of frames into an ordered dataset of images with standard organization. More details on this software component will be provided in Sec. 9.3.

5.4 iCubWorld28

In this Section we present the ICUBWORLD28 release (ICW28 for short), the first of the two datasets published within the Thesis.

The initial motivation of the dataset was to acquire a first benchmark for assessing the impact of employing pre-trained deep Convolutional Neural Networks (CNNs) for visual object recognition in robotics and specifically in the context of ICUBWORLD. In particular, we aimed to evaluate the robustness of deep CNNs to typical robotic issues, as for instance the difference between training and testing condition, when increasing the task size and difficulty. In fact, our preliminary experiments suggested us that, although remarkably outperforming "shallow" representations, deep CNNs were not providing the close to perfect performance that one would wish for, even on relatively small tasks. We ascribed the reason for this gap to the presence of challenging nuisances, typically affecting objects' appearance in our scenario, like viewpoint, light, setting changes and so forth. As also explained in Sec. 1.4, these findings motivated us to plan our research accordingly.

Specifically, results initially pointed out a critical role of light and setting conditions, preventing classifiers trained on top of representations extracted from deep CNNs in certain scenarios, to generalize to others, even similar. Therefore, in order to evaluate and improve the generalization performance of deep CNNs also across days, we acquired and released ICW28, comprising 28 objects acquired in multiple (4) slightly different setting conditions. The dataset was published in [Pasquale et al., 2015b] (see [Pasquale et al., 2015a] for an extended version), where we provide also a complete empirical assessment of these observations.

In particular, in [Pasquale et al., 2015b] we observe that by incrementally including images of objects acquired in different days, we could remarkably improve the generalization capabilities of predictors. This dataset release has become therefore a reference benchmark to evaluate incremental learning approaches and different recent works employ the ICW28 dataset for this purpose [Keller and Lohan, 2016, Part and Lemon, 2016]. We also used it in our subsequent work (that will be presented in Chapter 8) to evaluate a modified version of the recursive Regularized Least Squares for Classification (RLSC) algorithm.

In the following, after a brief overview of related datasets in the literature, we present ICW28.

5.4.1 Datasets for Incremental Learning

As anticipated, a special focus of this dataset is to study incremental learning approaches to visual object recognition in robotics. While we refer to Chapter 8 for an overview of related work, here we briefly introduce the basic concepts needed to motivate and understand the

acquisition of ICW28.

Usually, incremental learning in the literature is referred to as the problem of updating a previously learned model of some entity (in this case, we can think of an object), when new training examples are provided, in contrast to *batch* procedures that, instead, re-train every time the model from scratch in order to include the newly added examples. In the context of classification, the new examples may belong to one of the learned classes, or to an unknown class. In the first case, the system is expected to "improve" the model of the class for which new examples are available, while in the latter case, the system must create a new class and learn it from the new examples. Referring to Sec. 2.1, if we are in the supervised learning setting, the model can be though as the parameters of a RLSC (as we will see in Chapter 8).

If we consider a humanoid robot that is taught several objects one after the other, like in the presented scenario, the examples are in fact the acquired frames from the robot's cameras. In this setting, learning incrementally means that the robot updates the model of the object while observing it, as new frames arrive, instead of waiting to see the full sequence before start training the model. Accordingly, if a new object is encountered, a new class should be added to the knowledge of the robot without retraining all known classes from scratch. Note that, the identical reasoning can be made when considering the task of object categorization, except that, in this case, the class is a category of objects. Within this perspective, it is evident that any dataset can be used to study incremental learning methods, simply by considering progressively increasing subsets of the overall number of examples and classes available. Indeed, there are works in the literature that use the Washington RGB-D benchmark [Lai et al., 2011] to evaluate incremental learning methods (see, e.g., Chapter 8 and references therein).

However, when considering lifelong learning applications, it is useful to have at disposal multiple different acquisitions of the same object, to simulate the visual experience of a humanoid agent that is repeatedly taught the same object in possibly different conditions, in order to improve its recognition capabilities. The available benchmarks become very few if we restrict to this scenario.

The BigBrother dataset [Franco et al., 2009] is a benchmark for face recognition, created from DVDs of an edition of the "Big Brother" reality show for TV documenting 99 days of permanence of 20 participants in a domesitc environment. The faces of participants are characterized by large variability over the full period, and can be exploited for incremental learning of their identity.

The ADL dataset [Pirsiavash and Ramanan, 2012] is a benchmark for the detection of Activities of Daily Living in first-person camera views and comprises $\sim 1m$ frames of 20 of people performing everyday activities while recording their view with a wearable device. The dataset is annotated with activities, object tracks, hand positions, and interaction events.

iCubWorld28	
#categories	7
#objects/category	4
acquisitions/object	4 Days Train & Test
#frames/acquisition	150
tracking cue	Independent Motion
acquired cameras	Left
full images available	Yes

Table 5.3: Dataset overview.

The IDOL2 (Image Database for rObot Localization 2) dataset [Luo et al., 2007] is particularly relevant since it is a benchmark for place recognition, specifically acquired to investigate incremental learning. Moreover, like ICUBWORLD dataset, it is robot-centric since it was acquired by a camera mounted on two mobile robot platforms while they were driven through an indoor laboratory environment. It contains 24 image sequences of $\sim$ 1k frames each, automatically labeled with one of five different classes (printer area, corridor, kitchen, and so forth). The acquisition procedure was designed to capture the changes in illumination, varying weather conditions, people's activities, change of location for objects and for furniture, also over a quite long time span (6 months).

Finally, partially inspired by ICW28, two new datasets for incremental object recognition on a humanoid platform have been recently released: the TUM-CS [Barkmeyer, 2016] and the IGLU [Azagra et al., 2016] datasets. The first one is an extension of our dataset, comprising images of 11 objects, acquired by recording from the iCub's cameras in three different settings (objects on a table, on a white background and a combination of the two). The second one is a multimodal dataset comprising synchronized speech, depth and RGB data recorded from multiple sources positioned on a robot while interacting with people showing objects to it, repeatedly and in different ways.

As can be noticed, to our knowledge, ICW28 is the first robot-centric dataset addressing the problem of visual object recognition in a lifelong learning setting, inspiring also recent works in a similar direction.

Figure 5.4: **iCubWorld28.** Example images from the dataset showing the 4 object instances belonging to each of the 7 categories.

5.4.2 Dataset Overview

The ICUBWORLD28 dataset comprises images of 28 objects evenly organized into 7 categories. As can be seen in Fig. 5.4, the object categories have been chosen among household products. For each object instance, we acquired a sequence of 220 images, during $\sim$ 20s, in Human Mode. Specifically, the human operator was rotating the object randomly in front of the robot, affecting the object's appearance mainly with 3D rotations. We acquired full 320×640 images streaming from the left camera, plus the localization annotations provided by the independent motion detector described in Sec. 5.3.1.

We repeated the acquisition protocol of the 28 objects twice, in order to obtain two sequences per object that we used as train and test. Moreover, as anticipated, since we aimed to assess incremental learning capabilities over multiple days, we repeated the same acquisition procedure in 4 different days, ending up with 4 datasets (Day 1, to 4) of more than $12k$ images each. From one day to another there is a slight natural change in the setting condition (which we did not constrain on purpose), that is, light, background, object's scale, can be slightly different in sequences of each day.

On the ICUBWORLD website we made available the full images with the annotation of the object's centroid, and also a dataset version where square bounding boxes of size 128×128 have been cropped around the centroid. We also provide a manually segmented version of the dataset acquired on Day 4. The segmentation in this case was performed by clicking on the image and dragging a rectangular shape to enclose the object represented. We aimed to use

this version to measure the recognition performance gap due to the automatic segmentation procedure based on motion with respect to an "ideal" segmentation.

In Table 5.3 an overview of the dataset is reported.

5.5 iCubWorld Transformations

The ICUBWORLD TRANSFORMATIONS dataset (in the following ICWT) is the latest release, acquired with the aim of providing a large-scale benchmark for extensive evaluation of the application of deep CNNs to visual recognition problems in robotics, with a specific focus on invariance to viewpoint transformations. In the following, we briefly review existing benchmarks for the study of invariance and provide an overview of this dataset.

5.5.1 Datasets for Invariance

We have seen in Sec. 4.2.2 that to study the invariance properties of recognition systems to visual nuisances, particular datasets are needed. Considering for instance viewpoint transformations, for a rigorous analysis it is useful to consider each transformation in isolation from others, and hence multiple image sequences, each representing an object whose appearance undergoes only one transformation, are needed in this case. Datasets with similar characteristics are not common in the literature and we have seen that, often, the problem is addressed by rendering 3D object models [Pinto et al., 2011, Peng et al., 2015]. This approach provides flexibility and control over arbitrary transformations, as 3D viewpoint, scale, background, light and so forth. The NORB dataset [LeCun et al., 2004] is one of the most famous benchmarks for invariance. It was acquired in a turntable setting, then the images were artificially perturbed to introduce the desired variability factors. It is not obvious, however, how systems trained on synthetic images would generalize to real world scenarios. Therefore, while, on the one hand, this strategy provides a practical way of generating the necessary data to perform extensive and rigorous analyses, on the other hand there is still a (potentially large) gap to be bridged before achieving results that can be expected to hold also in real world settings.

Indeed, these considerations are shared with recent work: [Borji et al., 2016] released the iLab-20M dataset, with which they aim to create the first large-scale benchmark that allows to study the behavior of visual recognition methods under multiple perspectives, including invariance to real world visual transformations. Beyond providing a high number of object examples per category (25 to 160), similarly to classical computer vision datasets, iLab-20M provides also a high number of images per object ($\sim 20k$), which are also tagged by transformation. While images are recorded in a turntable setting (plus an additional part acquired by a robotic arm), this benchmark overcomes many of the limitations of previous

datasets recorded in similar conditions (that we mentioned in Sec. 5.2), in terms of size and nuisance factors, providing an extremely useful benchmark for the study of invariance.

On the other hand, this dataset is still far from modeling the prototypical visual experience of a humanoid robot, characterized by wider, noisier, transformations, and the continuous addition of new training evidence, which typically, as we have seen, cannot be acquired in such a constrained setting. To this end, the goal of ICWT is to provide a middle-way benchmark between the rigour of datasets for invariance and the realism of robot-centric images. Indeed, as anticipated, the peculiarity of ICWT, motivating its name, is that each object is represented in multiple image sequences while it undergoes isolated visual transformations (such as rotations, scaling, background changes). Being acquired from a moving robotic head, the considered transformation may not be perfectly isolated, but rather dominant, with respect to other variations, which are kept minimal.

To our knowledge, ICWT is the first dataset to address invariance-related questions in robotics and accounts a much wider range of visual transformations with respect to previous datasets. We believe that ICWT could be a significant contribution also in the direction of invariance analysis on real, rather than synthetic, visual transformations.

5.5.2 Dataset Overview

The ICWT dataset includes objects from 20 categories that can be typically found in a domestic environment. As can be seen in Fig. 5.6, reporting a sample image for each category, 11 categories (in Red) are also in the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) 2012 [Russakovsky et al., 2015], that is, we found semantically and visually similar classes among the 1000 of the classification challenge. The remaining 9 categories do not appear ILSVRC but belong (or are similar) to a synset in the larger ImageNet dataset [Deng et al., 2009]. The WordNet sub-tree related to the 20 categories comprising the dataset was reported in Fig. 1.6.

For each category, we collected 10 object examples. To provide a qualitative intuition of the semantic variability within a given category, namely the different visual appearance of objects in the same category, Fig. 5.7 shows a sample image from the 10 instances in the *mug* category. Finally, for completeness, Fig. 5.8 reports one example image for each of the 200 object instances in the dataset, indicating also the reference number of the associated synset in ImageNet. Like in Fig. 5.6, Red categories are part of the ILSVRC, Black categories belong, or at least are similar, to synsets in ImageNet. As it can be noticed, the semantic variability of ICWT allows to test both object categorization and identification capabilities.

Notably, the dataset is also segmented in different sets of sequences. Specifically, for each object instance, 5 different image sequences were acquired, while the human supervisor was

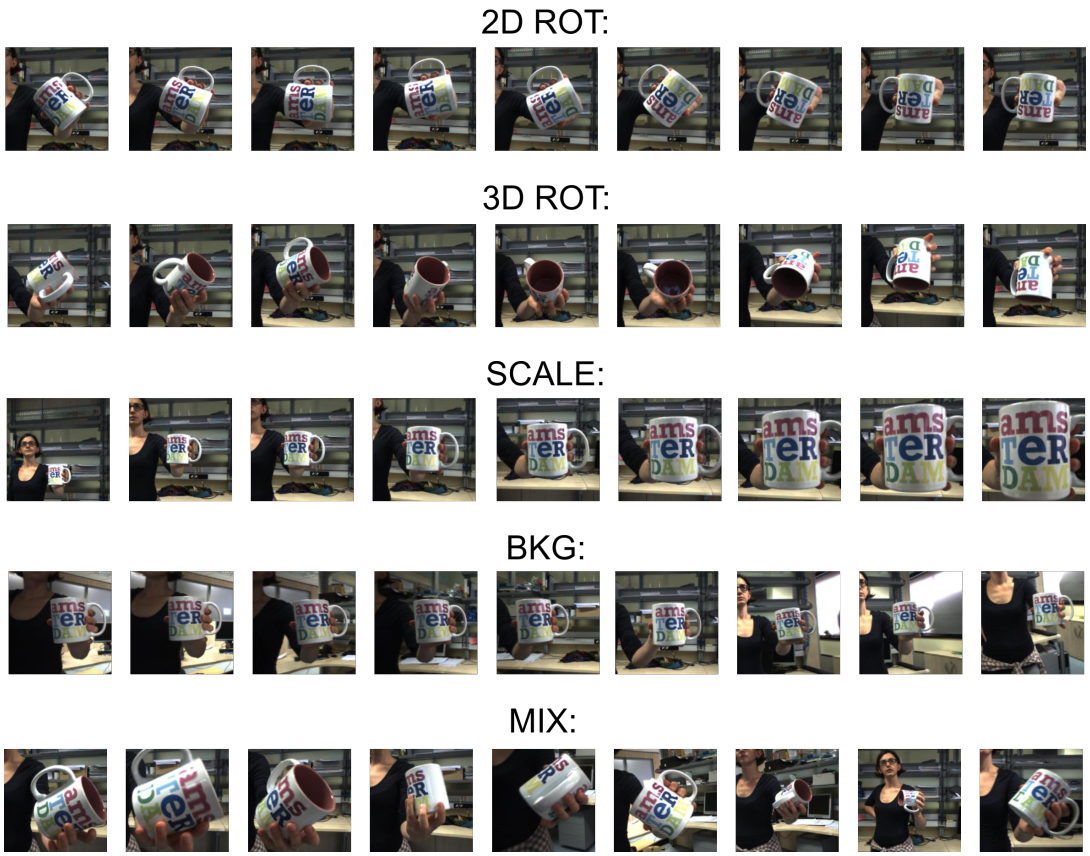

Figure 5.5: **iCWT visual transformations.** Excerpts from the sequences acquired for one *mug*, representing the object while it undergoes specific visual transformations.

applying a different visual transformation to the object. Fig. 5.5 reports excerpts of these sequences, which contain, respectively:

$2D$ **Rotation** The human rotated the object parallel to the camera image plane. The same scale and position of the object were mantained (see Fig. 5.5, first row).

$3D$ **Rotation** Similarly to $2D$ rotations, the object was kept at same position and scale. However, this time the human applied a generic rotation to the object (not parallel to the image plane). As a consequence different "faces" of the object where shown to the camera (Fig. 5.5, second row).

Scale The human moved the object towards the cameras and back, thus changing the object's scale in the image. No change in the object orientation (no $2D$ or $3D$ rotation) was applied (Fig. 5.5, third row).

Background The human moved the object around the robot, keeping approximately the same

iCubWorld Transformations	
#categories	20
#objects/category	10
acquisitions/object	4 Transf. + Mix 2 Days
#frames/acquisition	$\sim$ 150 (4 Transf.) $\sim$ 300 (Mix)
tracking cue	Depth
acquired cameras	Left & Right
full images available	Yes

Table 5.4: Dataset overview.

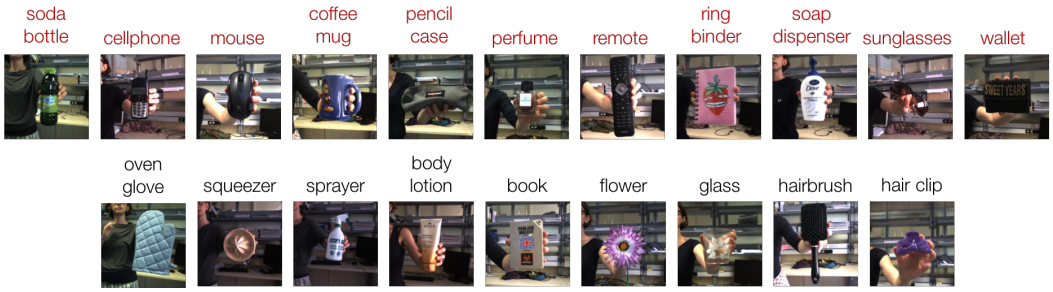

Figure 5.6: **iCWT categories.** For each category in iCWT, we report one example image for one object instance. (Red) categories appear also in the ILSVRC 2012 dataset. (Black) categories appear in ImageNet but not in ILSVRC.

distance (scale) and pose of the object with respect to the camera plane. During this acquisition process only the background changes while the object appearance remains approximately the same (Fig. 5.5, fourth row).

Mix The human moved the object freely in front of the robot, as a person would naturally do when showing a new item to a child. In this sequence all nuisances in all combinations can appear (Fig. 5.5, fifth row).

Each sequence is composed by approximately 150 images acquired at 8 frames per second in the time interval of $20s$, except for the *Mix* one that lasted $40s$ and comprises $\sim$ 300 images.

The acquisition of the 200 objects was split in multiple sessions performed in different days. The acquisition location was always the same (with little changes in the background across days). The light condition was not artificially controlled, since we wanted to investigate

Figure 5.7: **iCWT semantic variability.** Sample images for the different object instances in the *mug* category to provide a qualitative intuition of the semantic variability in the dataset. See Fig. 5.8 for more example objects.

its role as a further nuisance: to this end, we acquired objects at different times of the day and in different days, so that light conditions are slightly changing across the 200 objects (but not within the five sequences of an object, which were all acquired in the time span of few minutes). Moreover, we repeated the acquisition of each object in two different days, so that we ended up with 10 sequences per object, containing 5 visual transformations in 2 different light conditions.

The camera resolution is 640×480: we recorded the centroid and bounding box provided by the tracker at each frame, as explained in Sec. 5.3.2. Both left and right images were acquired, to allow for offline computation of the disparity map and potentially further improvement of the object's localization and segmentation mask. Table 5.4 summarizes the main characteristics of iCWT.

Figure 5.8: **iCWT object instances.** Example images for the 200 objects comprising the dataset.

Chapter 6

Are we Done with Object Recognition? The iCub's Perspective.

In this and in the next Chapter we present our investigation on the benefits and limits of using deep Convolutional Neural Networks (CNNs) for object recognition in real world robotic settings. In this Chapter, we conduct a general analysis of the problem, focalizing some critical aspects. To this regard, in Chapter 7 we will specifically consider the invariance properties of deep CNNs to strong visual nuisances typically affecting the real world scenario.

The study reported in this Chapter has been motivated by the observations we made in [Pasquale et al., 2015a, Pasquale et al., 2015b], not included here. Instead, we report here the more recent work [Pasquale et al., 2017], which provides a complete account of the results observed so far.

6.1 Introduction

As we have seen in Chapter 4, deep learning methods are leading a revolution in Artificial Intelligence, a key role being played by the outstanding performances provided by latest deep Convolutional Neural Networks (CNNs) on a variety of vision-related tasks. Starting from the ground breaking results of AlexNet in 2012 [Krizhevsky et al., 2012], deep CNNs progressively improved the state-of-the-art on the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [Russakovsky et al., 2015], and their remarkable effectiveness proved to be not limited to this benchmark. Indeed, as anticipated in Chapter 1, the progress in robotics could be incredibly boosted by such a breakthrough in visual recognition, ultimately removing one of the main bottlenecks that today prevent the deployment of autonomous agents in unconstrained environments.

To this end, motivated by the observations exposed in Sec. 1.4, we started an effort to isolate and quantify the benefits and limitations, if any, of using deep learning approaches for visual object recognition problems in robotic settings. We designed the acquisition setting of iCubWorld, reflecting a prototypical visual experience of a humanoid robot interacting with people to learn new objects. We first released the iCubWorld28 (iCW28) dataset (Sec. 5.4), on which we performed a "preliminary" study [Pasquale et al., 2015b] (see [Pasquale et al., 2015a] for an extended version). With this analysis, we assessed the clear superiority of deep CNNs over recognition pipelines based, e.g., on dictionary learning, adopted at that time on the iCub. On the other hand, we also assessed that the problem was far from being solved and that, even on a relatively simple task (compared to the ILSVRC), like the one of identifying between the 28 objects in the dataset, there was still a large margin of improvement. Indeed, with iCW28 we started identifying some key factors impacting performance in robotic settings, which we decided to further investigate.

In order to expand the horizon or our observations, we proceeded with the collection of the iCubWorld-Transformations (iCWT) dataset (Sec. 5.5), comprising much more objects and challenging visual transformations. Provided with the large and varied iCWT benchmark, we performed extensive tests, that we will present in the following, including categorization and identification tasks, using various state of the art CNN architectures and learning methods.

We began checking to which extent architectures already trained (on web data corpora) could be directly tested on iCWT. While obtaining results much better than chance, these systems did not perform accurately enough in this case –as perhaps one could expect. We hence followed the recent trend based on the general idea of transfer learning (see Sec. 4.3), and confirmed the effectiveness of such strategy, obtaining substantial improvements. While impressive, however, these methods did not quite provide the close to perfect accuracy one would wish for. We hence proceeded taking a closer look at our results, starting from the question of whether the missing gap could be imputed to lack of data. Indeed, CNNs are known to need massive amount of data to work, such that data-augmentation is often used to improve results. As we discuss in next Sections, investigating this latter question identified a clear gap between the object recognition task in robotics and in typical applications considered in learning and vision, where deep learning approaches excel, such as image retrieval and large-scale image classification. Along the way, our analysis allowed to test invariance properties of the considered networks, quantifying their merits not only for identification, but also for categorization. We will report this part of the work separately in the next Chapter.

In a nutshell, our results confirm that deep learning can indeed provide stunning performances, but a gap still needs to be bridged for seamless deployment in robotics. Indeed, the

discussion of our empirical findings will be complemented with a critical review, reported at conclusion of this Thesis (Chapter. 10), of some of the main avenues of improvements, from a pure machine learning perspective but also taking extensive advantage of the robotic platform.

In the next Section, we discuss some related work, while the rest of the Chapter is organized as follows: in Sec. 6.2 we specify the details of the learning methods considered for our empirical analysis, which is reported in Sec. 6.3 for the categorization task and in Sec. 6.4 for object identification. For both settings, we complement our results by considering in Sec. 6.5 also the common situation where the system is trained and tested on different days. Sec. 6.6 finally concludes the study. We supply many additional tests and technical details on the adopted methods in the Appendices. In particular, in Appendix 6.D we validate our learning methods on the Washington RGB-D dataset [Lai et al., 2011], by obtaining results comparable or better than the state of the art on this benchmark.

6.1.1 Related Work

In Sec. 4.3 we reported recent studies assessing that transferring visual representations previously learned by deep CNNs on large, web-collected data corpora, can provide much better performance, on a variety of benchmarks and tasks, than other methods entirely learning (shallow or deep) representations on the specific considered domains. We reviewed some works in the computer vision literature that, starting from deep CNNs trained on large-scale visual recognition problems, investigated both the direct use of representations extracted from their layers, and their fine-tuning, to address smaller recognition tasks. Today, deep CNNs are receiving growing attention also in robotics and, as we outlined in Sec. 1.3.2, are being progressively adopted for a variety of problems. In this Section, we limit the discussion to the work on object recognition, which is more relevant to the study described in this Chapter.

Representations extracted from CNN models pre-trained on ImageNet have been proved to advance the state-of-the-art on the Washington RGB-D dataset [Schwarz et al., 2015], and also on different place recognition datasets [Sünderhauf et al., 2015, Sünderhauf et al., 2016]. While [Sünderhauf et al., 2015] directly match representations with Nearest Neighbor (NN) to identify places, [Schwarz et al., 2015, Sünderhauf et al., 2016] feed them to SVMs, similarly to our work on ICW28 [Pasquale et al., 2015b]. Other works considering the Washington RGB-D dataset [Eitel et al., 2015] investigated fine-tuning of CNN models pre-trained on ImageNet, slightly outperforming previous approaches. Both [Schwarz et al., 2015] and [Eitel et al., 2015] devise solutions to exploit also the depth information, by encoding the depth map through a colorization scheme and feeding it to the same pre-trained models employed for RGB images, both obtaining an improvement of $\sim 6\%$ with respect to using only RGB information, in line with previous works. In Appendix 6.D we will also report results on

RGB-D Washington, directly comparing with [Schwarz et al., 2015] and [Eitel et al., 2015]. The benefit of adapting representations learned on ImageNet, rather than entirely learning them in the target (robotic) domain has been assessed also for conventional "non-deep" models in the recent work of [Goehring et al., 2014], where the authors consider the Office [Saenko et al., 2010] dataset.

In our work, we adopt a more challenging setting than [Schwarz et al., 2015, Eitel et al., 2015, Goehring et al., 2014], in that CNNs are trained with visual data directly acquired by the robot during a natural interaction. The fact that recognizing objects in a real world scenario poses challenges which are not suitably modeled by classical computer vision benchmarks has been highlighted in the past [Pinto et al., 2008, Rodner et al., 2013]. Recently, the limitations of currently available turntable datasets like the Washington RGB-D, Office, BigBird datasets, have also been pointed out in the literature [Model and Shamir, 2015, Borji et al., 2016]. By directly benchmarking on the robot, we definitely reduce the gap between the reported and the expected performances of the system in its real usage, without losing reproducibility.

The striking performances reported by deep representations learned on web data corpora in a wide range of other tasks and datasets, recently raised the question of whether the problem of dataset bias [Torralba and Efros, 2011] could be finally overcome by taking such an approach. Extensive analyses reported in [Hoffman et al., 2013, Tommasi et al., 2015] show that this problem is indeed alleviated, but not solved, by deep learning systems trained on large-scale datasets. In this respect, it becomes crucial to evaluate the applicability of strategies to adapt deep CNNs to real world, robotic settings. For this reason, we investigate feature extraction from pre-trained CNNs or fine-tuning, spanning over a range of architectures, experimental settings and training approaches. We are aware that the two considered approaches may not be the only ones, and that the literature of transfer learning offers a large variety of solutions to improve performance over the considered methods (see, e.g., [Hoffman et al., 2013, Tommasi et al., 2016]). However, a major focus of this study is to assess the effectiveness of these commonly adopted techniques in a typical robot vision scenario, as a first step towards future improvements.

To our knowledge, there is no work in the robotics literature that addresses an extensive evaluation of these aspects. Works considering the use of deep CNNs in robotics either refer to datasets acquired in turntable setups, like the Washington RGB-D, yet improving performance on these benchmarks, but optimizing a single recognition task rather than considering a real application to open settings. Works relying on robot-acquired datasets, on the other hand, so far focused more on the aspect of scaling up the collection of many points in order to demonstrate that general representations could be learned even with non-conventional forms of supervision. However, it is not clear yet how to best adapt deep CNNs for visual

Table 6.1: Feature extraction layers for the four architectures considered in this work. We used the notation adopted in the literature, in which the number identifies the layer number and the label specifies its type (i.e. fully connected or pooling layer).

Model	Output Layer
CaffeNet	fc6 or fc7
GoogLeNet	pool5/7x7_s1
VGG-16	fc6 or fc7
ResNet-50	pool5

recognition to situations where the robot has, for instance, to continuously learn new objects, in challenging unpredictable scenarios. Indeed, deep CNNs are known for requiring large amounts of examples, and solutions to collect and annotate this data on robotic platforms must be devised. To this end, it is important to understand which kind of examples are most critical to effectively train deep networks for the different visual recognition tasks, and with which adaptation strategy. In this Chapter we take an operative approach and perform a battery of experiments addressing such problems.

6.2 Methods

In this Section we provide details on the network architectures and the transfer learning techniques that we investigate. We will build on the concepts introduced in Chapters 3 and 4, about the structure of deep CNNs (Sec. 3.2) and their training through Stochastic Gradient Descent (SGD) with backpropagation (Sec. 4.1). We refer in particular to Sec. 3.3 and 4.3, respectively for the presentation of the adopted models and transfer learning methods.

As anticipated, we will consistently make use of the ICWT dataset introduced in Sec. 5.5.

6.2.1 CNN Architectures

For our analysis we selected the four architectures that achieved best performance on the ILSVRC between 2012 and 2015: *CaffeNet*, *VGG-16*, *GoogLeNet* and *ResNet-50*. These models, trained on the ILSVRC 2012 dataset for the object categorization task (that is, we recall, predicting the category of an object depicted in an image) are publicly available. We used their implementation in the CAFFE [Jia et al., 2014] framework, which is the software we used for all our experiments. For a description of the architectures and links to their public implementations, see Sec. 3.3.

6.2.2 Feature Extraction

The first strategy that we investigated to transfer knowledge from CNNs trained on ILSVRC to iCWT is to use models as they are (i.e., "off-the-shelf"), feeding them with images and extracting corresponding vector representations as the output of one of their latest layers. Such representations can be then fed to classifiers in order to perform the desired recognition task. In Sec. 4.3.1 we provided a short overview and related work on this approach, often referred to as *feature extraction*. The specific layers whose activations have been extracted as image representations for our experiments are reported in Tab. 6.1 for the four considered architectures. We refer to Appendix 6.A.1 for details about the image preprocessing applied to the images before feeding them to the networks. Indeed, we recall that in this setting images in iCWT are frames recorded by the robot cameras, from which we extracted a bounding box around the object of interest using the depth-driven attention system presented in Sec. 5.3.2.

We then used a Regularized Least Squares Classifier (RLSC) with a Gaussian Kernel on top of the CNN-extracted features. In particular we implemented the Nystrom subsampling approach considered in [Rudi et al., 2015,Rudi et al., 2016], which is computationally appealing for mid and large-scale settings.

In Appendix 6.B.1, we provide details about the choice of the network layers used for extracting representations and model selection for RLSC.

6.2.3 Fine-tuning

The second transfer learning approach is referred to as *fine-tuning* and we presented it in Sec. 4.3.2. It consists in taking a CNN model trained on ILSVRC, replace the final output layer in order to consider the new, desired task and "continue" its training with SGD on the new (usually called *target*) domain (in our case iCWT), by using the off-the-shelf model as a "warm-start" for all the other layers.

In our experiments we performed fine-tuning only for *CaffeNet* and *GoogleNet*, which are representative of most recent architectures and were significantly faster to fine-tune than *VGG-16* and *ResNet-50*. We considered two main regimes for fine-tuning a network: one updating only one or more fully-connected (FC) layers, while keeping the others fixed, and another one more aggressively adapting to the target training dataset all layers, comprising the convolution (C) layers. This was done by selecting different learning rates for the neurons in each layer, i.e., the size of gradient steps at each iteration of backpropagation. We refer to these two protocols as *conservative* and *adaptive* and report the corresponding parameters in Table 6.2 as a reference. In the experiments discussed in this Chapter we consider only these strategies since we did not observe significant performance differences following other

Table 6.2: Fine-tuning protocols for *CaffeNet* and *GoogLeNet*. Base LR is the starting learning rate of all layers that are initialized with the original model. The FC layers that are learned from scratch are indicated using their names in CAFFE models (2^{nd} row), specifying the starting learning rate used for each of them. For the other parameters, we refer the reader to CAFFE documentation.

	CaffeNet		**GoogLeNet**	
	adaptive	*conservative*	*adaptive*	*conservative*
Base LR	1e-3	0	1e-4	0
Learned FC Layers	fc8: 1e-2	fc8: 1e-4 fc7: 1e-4 fc6: 1e-4	loss3/classifier: 1e-2 loss1(2)/classifier: 1e-3 loss1(2)/fc: 1e-3	
Dropout (%)	fc7: 50 fc6: 50		pool5/drop_7x7_s1: 60 loss1(2)/drop_fc: 80	
Solver **LR Decay Policy**	SGD Polynomial (exp 0.5)		Adam No decay	
# Epochs	6	36	6	
Batch Size	256		32	

choices of parameters. We refer to Appendix 6.B.2 for an analysis of using other fine-tuning parameters and a discussion on the model selection protocols used in our experiments.

6.3 Results (Object Categorization)

In this Section we present our empirical investigation of the described deep learning methods in the robotic setting of ICUBWORLD. While here we consider the task of object categorization, in Sec. 6.4 we will consider the problem of object identification.

6.3.1 Deep Learning and (the Need for) Knowledge Transfer

Modern datasets for visual recognition comprise an extremely large number of images (e.g. 1 Million for the ILSVRC challenge) depicting objects in a wide range of natural scenes. This extreme variability opens the question of whether datasets such as ICUBWORLD, which represent a much smaller "reality", could be interpreted simply as sub-domains of larger ones. If this were the case, powerful models as deep CNNs trained on ImageNet would achieve good recognition performance on ICUBWORLD as well, without any re-training or adaptation. While this result would lead to the appealing scenario where a robot can simply download and use a visual classifier "off-the-shelf" (i.e. trained a web collected data corpora), previous analysis has shown that most computer vision datasets are essentially biased, ultimately preventing

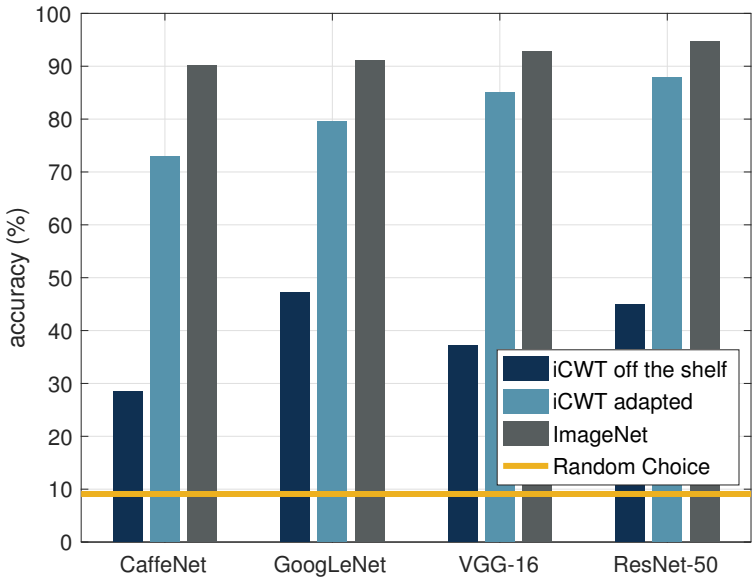

Figure 6.1: Average classification accuracy of off-the-shelf networks (trained on ILSVRC) tested on ICWT (Dark Blue) or on ImageNet itself (Gray). The test sets for the two datasets are restricted to the 11 shared categories (see Sec. 5.5). (Light Blue) reports the classification accuracy when the same networks are "transferred" to ICWT (see Sec. 6.3.1 for details). The (Orange line) shows the recognition chance of a random classifier.

a learning algorithm from generalizing from one domain to the other [Torralba and Efros, 2011, Khosla et al., 2012, Stamos et al., 2015, Model and Shamir, 2015].

To address this question, we evaluated four off-the-shelf CNNs trained on ILSVRC (specified in Sec. 6.2.1) for the task of object categorization on ICWT. For these experiments we restricted the test set to the 11 categories of ICWT that appear also in the ILSVRC challenge (see Sec. 5.5 and Fig. 5.6). As a reference, we compared these results with the average accuracy reported by the same networks on the corresponding 11 categories of the ImageNet dataset. The test set in ICWT was composed, for each category, by the images of all 10 object instances, comprising all 5 transformations for one day and the left camera (unless differently specified we always used this camera for the experiments), for a total of ~ 9000 images per category. For ImageNet, we downloaded the images for the corresponding 11 categories (refer to Fig. 5.8 for the synset numbers), comprising on average ~ 1300 images per category.

Fig. 6.1 reports the average classification accuracy on ICWT (Dark Blue) and ImageNet (Gray). It can be immediately observed that there is a substantial drop in performance of $\sim 50-60\%$ when testing on ICWT rather than ImageNet, suggesting that differences between

the two datasets exist, and in particular that iCubWorld is not a sub-domain of ImageNet.

Specifically, a possible reason explaining this performance degradation may lie in the presence in iCWT of critical visual nuisances (e.g., firstly, viewpoint transformations) typically affecting real world settings, which are not represented in large collections of images "in the wild" like ImageNet (see, e.g., Sec. 4.2.1 and the work of [Pinto et al., 2008]). While we refer to [Hoffman et al., 2013, Tommasi et al., 2015] and also [Rodner et al., 2013, Herranz et al., 2016] for an in-depth analysis of ImageNet biases, outside the scope of this work, in Appendix 6.A.3 we provide some further empirical evidence of the gap between iCWT and ImageNet. We care to point out that for a fair comparison we have restricted the output of the off-the-shelf networks to the 11 classes considered in the experiment with iCWT (from the original 1000-dimensional output vector provided by the off-the-shelf networks); however, for completeness, in Appendix 6.A.2 we report also the same performance when considering the entire 1000-dimensional prediction. We refer to Appendix 6.A for technical details on the experiment, including image preprocessing steps (6.A.1).

Knowledge Transfer

The drop in performance observed in Fig. 6.1 implies that it is necessary to train the recognition system on the target domain. However, the experiment also shows that all the networks performed substantially better than chance (Orange line). This suggests that the models did retain some knowledge about categories' appearance. Therefore, rather than training a novel architecture "from scratch" (which typically requires large amounts of training data, time and computational resources), we see that the alternative approach of re-using and transferring such knowledge, by "adapting" the models trained on ImageNet to our new setting, could, indeed, provide some advantage. We have seen in Sec. 4.3 how, recently, topics related to the idea of transferring visual representations have received considerable attention from the deep learning community, because this provides an effective approach to apply deep learning methods to smaller datasets.

To this end, in Fig. 6.1 we also report the classification accuracy achieved by models where knowledge transfer has been applied (Light Blue). In particular, for this experiment we followed the protocol described in Sec. 6.2.2, where RLSC predictors are trained on feature vectors extracted from the layers of the CNNs. We created a training set by choosing 9 instances from each category, while keeping the 10^{th} instance for testing. We repeated the experiment for 10 trials in order to allow each instance of a category to be used in the test set. Fig. 6.1 reports the average accuracy over these trials. We observe a sharp improvement for all models, which achieve a remarkable classification accuracy in the range of $\sim 70 - 90\%$. However, we also notice that the corresponding performance on ImageNet is still significantly

higher (although, interestingly, such gap seems to be reduced for more recent architectures).

What are the reasons for this gap? In the following, we empirically address this question, showing that the observed behavior is due to fundamental differences between robot vision and image retrieval settings.

6.3.2 Do we need more data?

Deep Learning methods require large amounts of data in order to be trained effectively. This is particularly true when training a network from "scratch", but valid also (albeit on a smaller scale) to knowledge transfer techniques. Indeed, as mentioned in Sec. 4.1.2, a common practice when training a CNN is to perform *data augmentation*, namely, to artificially increase the dataset size by applying synthetic transformations to the original images (e.g. rotations, reflections, crops, illumination changes, etc.). From this perspective, the robotic setting seems particularly favorable. Indeed, data augmentation can be performed by simply acquiring more *frames* depicting an object while viewpoint or illumination change naturally.

However, if, on the one hand, acquiring more frames of an object can be particularly convenient, on the other hand, in robotics it is typically expensive to gather images depicting different object *instances* (since the robot needs to actually observe them), as instead are in typical benchmarks like ImageNet. In this Section, we investigate the impact of these aspects in robot vision, taking iCubWorld as a testbed. Note that, in the following we use the term *instance* (or simply object) to refer to a specific object belonging to a given category, while *frame* denotes a single image depicting a scene or, in this case, an object.

What do we gain by adding more frames?

We considered a 15-class categorization task on iCWT and compared the performance of CNNs trained on an increasing number of example frames. We created training sets of increasing size by sampling $N = 10, 50, 150$ and 300 frames from each transformation sequence of an object (we recall that each object in iCWT is represented by 10 sequences containing 5 isolated visual transformations acquired in 2 days). For each category (see Appendix 6.C for the list of categories) we used 7 objects for training, 2 for validation and 1 for test. Validation and test sets contained all images available for the corresponding instances. In order to account for statistical variability, we repeated the experiment for 10 trials, each time leaving out a different object instance for testing. For this experiment, we considered only one of the two available days in iCWT. Also, we considered only the left camera, apart from when sampling 300 frames, where we drew also from the right sequence. The number of frames per category was therefore ranged from 350 to 10500.

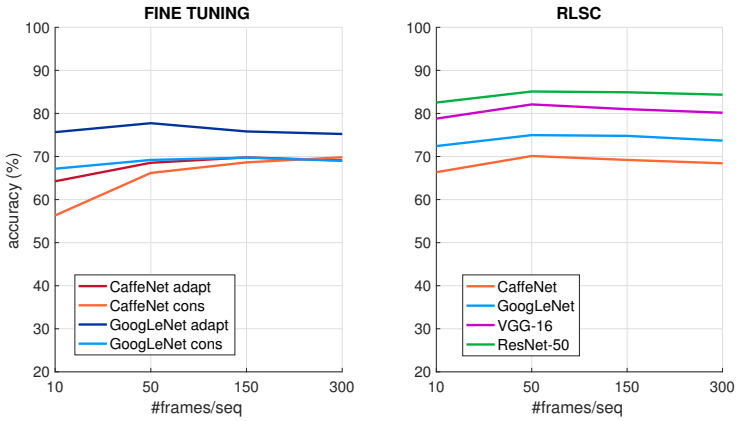

Figure 6.2: **Recognition accuracy vs # frames**. (Left) Accuracy of *CaffeNet* and *GoogleNet* models fine-tuned according to the *conservative* and *adaptive* strategies (see Sec. 6.2.3). (Right) Accuracy of RLSC classifiers trained over representations extracted from the 4 architectures considered in this work (see Sec. 6.2.2).

Fig. 6.2 reports the average classification accuracy of different learning models as more training examples are provided. Surprisingly, most methods achieve their highest accuracy already when trained on the smallest training set and do not show any improvement when new data is available. This finding is in contrast with our expectations, since it implies that increasing the dataset size is not needed in this setting.

Some "secondary" observations can also be made from these results:

- Fine-tuning does not lead to particularly improved performance with respect to using RLSC (both for *CaffeNet* and *GoogLeNet* and for both fine-tuning strategies).
- We confirm the ILSVRC "trend", with more recent networks outperforming "older" ones, with *VGG-16* features being better than those of *GoogLeNet* when using feature extraction with RLSC.
- Note that *CaffeNet* performs worse when training data is scarce because of the high number of parameters to be learned in the 3 FC layers (see Sec. 6.2.3 and Appendix 6.B).
- To further support these findings, in Appendix 6.C we report results for the same experiment performed using less example instances per category.

What do we gain by adding more instances?

We evaluated the impact of adding instead new object instances when training a recognition system. We consider this process as increasing the *semantic* variability of the training set, in contrast to increasing the *geometric* variability by showing more frames of the same object instance. We considered the same 15-class categorization task as previous experiment and

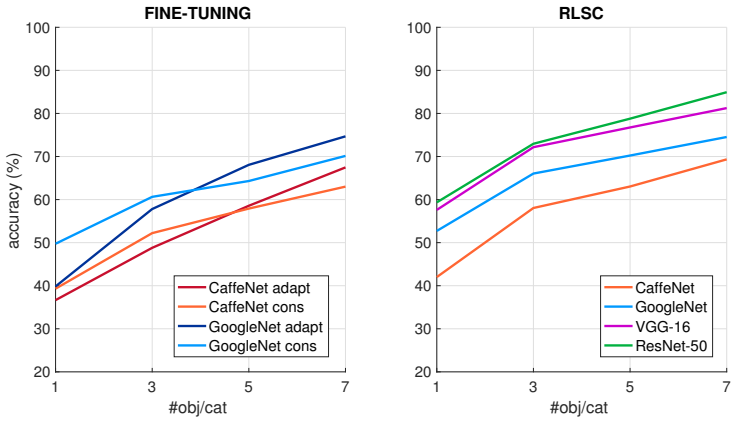

Figure 6.3: **Recognition accuracy vs # instances** (number of object instances available during training). (Left) Accuracy of *CaffeNet* and *GoogleNet* models fine-tuned according to the *conservative* and *adaptive* strategies (see Sec. 6.2.3). (Right) Accuracy of RLSC classifiers trained over representations extracted from the 4 architectures considered in this work (see Sec. 6.2.2).

created four training sets containing an equal number of 900 frames per category, but sampled from an increasing number of instances per category, namely $1, 3, 5$ and 7. Frames were randomly sub-sampled from the 5 available transformation sequences per object, i.e., we took all 900 frames from one instance per category, then we took 300 frames from each of the 3 instances per category, and so forth. Validation and test sets are maintained the same as previous experiment and contained all frames for the remaining 2 and 1 instance (per category) respectively. We repeated again the experiment for 10 trials.

Fig. 6.3 reports the accuracy of the different models trained on this task. Results show that, increasing the semantic variability, dramatically improves the accuracy of all models by a similar margin (around $\sim 5 - 10\%$ for each added instance). Moreover, even if the number of categories is relatively small, from the observed trends we see that all models do not saturate accuracies with 7 instances per category.

Also here some secondary observations are in order:

- Fine-tuning both *CaffeNet* and *GoogLeNet* seems to be more critically affected than RLSC by the lack of training instances. Indeed, the two methods achieve similar performance when 7 objects per category are available (also in line with previous experiment reported in Fig. 6.2), however fine-tuning drops more when fewer instances are available.
- We confirm ILSVRC results also in this setting, with more recent networks outperforming previous ones and with *VGG-16* features outperforming *GoogleNet*.

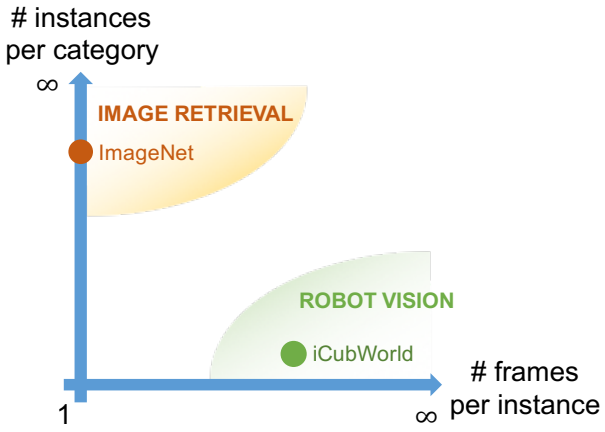

Figure 6.4: Different training regimes from robot vision and image retrieval settings.

- To further support our findings, in Appendix 6.C we report results for an equal experiment but discriminating only between 10 or 5 categories.

Robot Vision and Image Retrieval

In this Section we considered some of the main challenges associated to visual recognition in robot vision. While both robot vision and image retrieval address the same visual recognition problem, they are cast within two different training regimes. We have pointed out here the two main causes of such difference (depicted in Fig. 6.4): 1) *semantic variability*, namely the amount of different object *instances* available for each category and 2) the amount of *frames* per instance.

Typically, in image retrieval settings a large number of images depicting different object instances for each category is available, but for each such instance very few images (actually one, if the dataset is web collected) are provided. As an example, ILSVRC training set comprises ~ 1000 instances for each of the 1000 object categories and 1 image per instance is provided. On the opposite end of the spectrum, in robot vision settings it is easy to gather more example images for a single instance (the robot can move around and observe an object from multiple points of view). However, there is a limitation in the total number of categories and instances that can be experienced.

Our analysis has shown that the limited amount of semantic variability typically occurring in robotics can dramatically reduce recognition accuracy of deep learning systems. Moreover, contrarily to our expectations, we observed that this limitation cannot be alleviated by simply feeding more object views to the network. Indeed, classifiers trained on datasets containing variable number of images per object (but same semantic variability) achieved identical

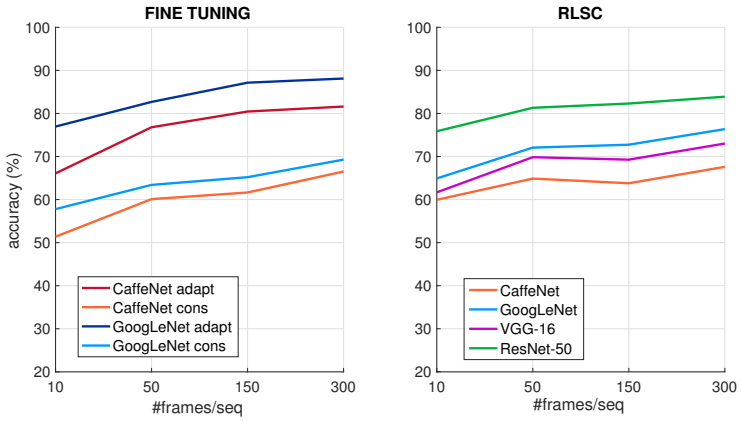

Figure 6.5: **Recognition accuracy vs # frames (Identification)**. (Left) Accuracy of *CaffeNet* and *GoogleNet* models fine-tuned according to the *conservative* and *adaptive* strategies (see Sec. 6.2.3). (Right) Accuracy of RLSC classifiers trained over representations extracted from the 4 architectures considered in this work (see Sec. 6.2.2).

performance. This can be problematic since in robotics there is a non negligible overhead in collecting and acquiring examples of a novel instance, and often it is simply not possible, while collecting more views of an object comes at a low cost. Unfortunately, adopting "data augmentation" strategies to artificially increase the semantic variability is not as easy as creating geometric or light variabilities. To this end, in Chapter 10 we will propose some possible solutions to this problem.

6.4 Results (Object Identification)

We focused so far on visual categorization problems, namely to assign an unknown object instance to its corresponding category, previously learned by observing other instances belonging to it. To complement our observations, in this Section we move to the task of object *identification* (or *instance* recognition), which consists in labeling a given image according to the object instance appearing in it (previously learned by observing other images of the instance). As we mentioned in Sec. 1.3.2, this problem is indeed very relevant to robotic settings, since reliable instance recognition skills would provide the system with finer control over tasks such as manipulation, grasping, etc. Also, considering a domestic scenario, the robot could be asked to use a *specific* object rather than *some* object in a category (e.g. "iCub, bring me *my mug*" instead of "*a mug*"). Therefore, in parallel to the experiments on categorization, we assessed the performance of pre-trained deep CNNs on the task of object identification in robotics.

6.4.1 Knowledge Transfer: from Categorization to Identification

The identification problem has been approached historically with methods based on keypoints extraction and template matching [Philbin et al., 2008, Collet et al., 2011, Muja et al., 2011]. However, it has been observed that approaches relying on more holistic visual representations perform typically better in low/mid-resolution settings such as ICUBWORLD [Ciliberto et al., 2013]. Moreover, recently, representations provided by deep CNNs proved to be effective for the task of instance retrieval [Sharif Razavian et al., 2014, Babenko et al., 2014]. Following these recent results, in this Section we will focus on the same transfer learning methods considered in Sec. 6.2.

We have shown how knowledge acquired on ImageNet transfers to the ICUBWORLD domain to tackle categorization tasks (Sec. 6.3.1). A natural question is whether the same strategy would be similarly favorable for object identification. In particular, in Sec. 6.3 we have observed that semantic variability is critical to learning the classes, whereas adding more images of a given instance is almost redundant. In object identification, intraclass semantic variability does not enter the game since each class is a single instance, represented in many different images. Does, in this case, geometric variability play a crucial role in learning?

We addressed this question by considering an object identification task on ICWT, where we compared CNN models trained with an increasing number of example images. The setting is similar to the one used for categorization but for a 50-class object identification problem: we chose 50 object instances by taking all instances from the *book*, *flower*, *glass*, *hairbrush* and *hairclip* categories, which do not appear in the ILSVRC dataset. We created four training sets containing respectively $10, 50, 150$ and 300 images per object, sampled randomly from the 4 transformation sequences $2D$ *Rot,* $3D$ *Rot, Scale* and *Bkg* (see Sec. 5.5). From each training set, 20% random images were retained for model selection. The images from the *Mix* sequence were instead used to test the classification accuracy of the methods considered. As for the categorization experiment, only the images from a single day were used for this experiment, and only the left camera was used – apart from when sampling 300 frames per sequence.

Fig. 6.5 reports the average accuracy of models trained on a growing number of example images. These results are in stark contrast with their counterpart on categorization (Sec. 6.3): we can clearly notice that adding more training data definitely improves the recognition capabilities of all networks. This suggests that having access to more views of an instance allows the learning systems to create a more nuanced model of the object. Apparently, this richer model does not provide any advantage in categorization settings, but is extremely useful to recognize the same object in new images.

In line with this observation, we notice also that in this setting the *adaptive* fine-tuning strategy significantly outperforms the competitors, suggesting that the representation learned

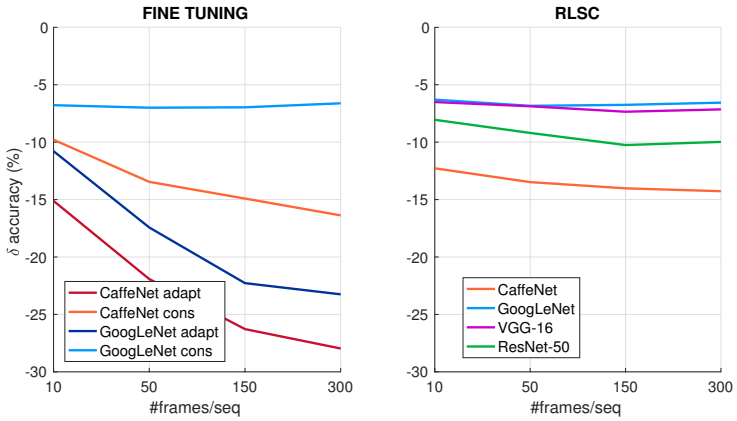

Figure 6.6: **Regognition accuracy vs # frames - Generalization across days**. Drop in performance observed when testing the models trained for the categorization task of Fig. 6.2 on a day different from the one where they were trained on. It can be noted that when fine-tuning on more frames from one day, then the performance drop on another day remarkably increases, particularly with the *adaptive* strategy.

for categorization on ImageNet, albeit beneficial to the task, can be improved to adapt to the object identification setting. These findings confirm and extend recent similar results from the instance retrieval literature [Babenko et al., 2014, Gordo et al., 2016].

6.5 Generalization Across Days

It is reasonable to imagine that, once trained, a robotic recognition system would be required to work for a certain period, during which the objects can appear not only under different viewpoints, but also in different conditions. In this Section we test therefore the robustness of the considered deep CNNs with respect to variations such as changes of illumination, background, etc., which are neither semantic nor geometric.

In order to account for this variability, we recall that, in the current release of ICWT, for each object instance we acquired image sequences in two different days, so that we naturally have images with light and small background setting changes. For the sake of clarity, in previous Sections we did not report on tests related to this aspect, which is however relevant. Indeed, we always considered only one among the two available days in the dataset. However, when training for visual recognition on a given day and testing on a second one, the possibly small contextual variation in the scene can cause the system to experience a degradation of its recognition accuracy. In the experiments reported below, we report this loss for the two categorization tasks in Sec. 6.3 and the identification task in Sec. 6.4, as the difference between

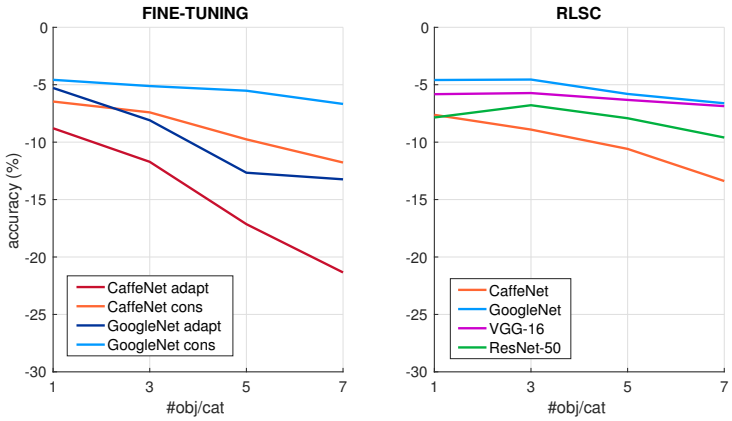

Figure 6.7: **Recognition accuracy vs # instances - Generalization across days**. Drop in performance observed when testing the models trained for the categorization task of Fig. 6.3 on a day different from the one where they were trained on.

the average classification accuracy observed when testing a model on the same day it was trained on and on the other available day in ICWT.

Object Categorization

We start considering the models trained for the categorization experiment in Fig. 6.2 and Fig. 6.3. We simply test these models on a day different from the one used for training (that is, we took the images of the object instances selected for testing in the original experiment, but from the second available day). Fig. 6.6 and Fig. 6.7 report the loss in performance observed respectively for the case where we tested on the relevance of the training set size (see Fig. 6.2) and of semantic variability (see Fig. 6.3).

In both settings all models exhibit a substantial drop in performance, starting from a minimum loss of 5% accuracy that achieves even around $20 - 30\%$ for the more *adaptive* fine-tuning strategy. This is remarkable since this latter strategy is also the most commonly adopted when fine-tuning CNNs. Indeed, while it appeared to be the most effective when tested on the same day, actually it was also the one most overfitting the variability in that day, being therefore less capable of generalizing to other days. Interestingly, however, when using less aggressive strategies such as the *conservative* fine-tuning or RLSC on top of off-the-shelf features, modern networks such as *GoogleNet* and *ResNet*-50 seem more robust.

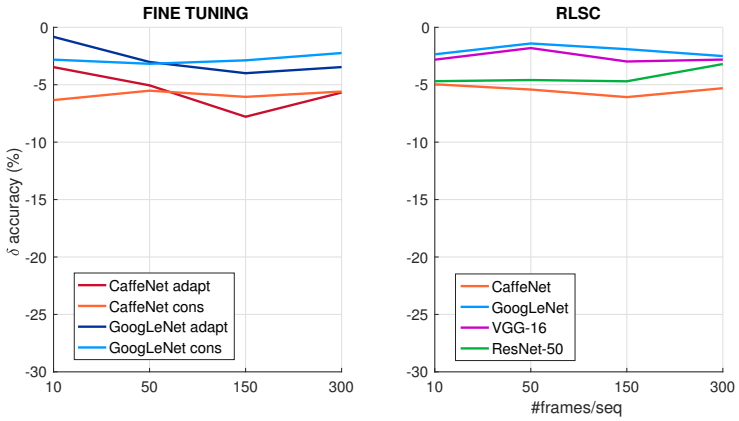

Figure 6.8: **Identification accuracy vs # frames - Generalization across days**. Drop in performance (difference between test accuracy) observed when testing the models trained for the identification task reported in Sec. 6.4.1 on the same day of training and on a different one.

Object Identification

Given the dramatic performance loss observed for the *adaptive* fine-tuning in categorization, we would expect an even worse accuracy drop affecting the *identification* task considered in Sec. 6.4, where, we recall, such strategy was definitely outperforming the other two approaches on the same day of training (see Fig. 6.5). Surprisingly, however, in Fig. 6.8 we notice that, albeit experiencing a drop in performance when generalizing to another day, for the *adaptive* fine-tuning such loss is small and on par with the other strategies considered in this work. Indeed, it can be noticed that all methods exhibit a performance drop around $\sim 5\%$ in accuracy and generally smaller than for categorization.

The Need to Learn Incrementally

From these findings, it is evident that, when only sequences acquired in a single day/condition are available for training a recognition system, a non negligible drop must be expected when testing it in a different situation, even when the underlying visual representations are provided by powerful deep CNNs trained on million examples and the differences amount to small light and setting changes. Indeed, these results confirm similar observations that we made in [Pasquale et al., 2015b, Pasquale et al., 2015a] on ICW28.

It becomes crucial therefore being able to exploit the possibility, intrinsic in the robotic setting, of using training evidence collected by the robot in more different situations. In this regard, it would be ideally desirable to update incrementally the learned object models as new examples arrive, rather than re-training "from scratch" at every time. These observations

motivated our work on incremental learning presented in Chapter 8.

6.6 Conclusion

In this Chapter we have described the first part of our study on the application of latest deep learning methods, namely, Convolutional Neural Networks (CNNs), to visual recognition in robot vision. We challenged these models on tasks specifically designed to represent the visual experience of a humanoid robot. Our experiments confirm deep learning is a remarkable step forward, leading to high performances for object categorization and identification, with a positive trend that corresponds to the new architectures proposed in the literature.

However, we identified a substantial gap still exists between the results that can be obtained in this and the web-scale image classification domain. Our analysis shows that, for categorization problems, this is firstly due to the scarce semantic variability, characterizing the robot vision context, because of the intrinsic cost of acquiring visual experience on different object instances. Yet, adoption of robotic systems in real applications requires to push even further the performance requirement to approach $\sim 100\%$. Therefore, there is still a lot that needs to be done to reach the required levels of robustness and usability.

To this end, we identified also other potential issues that need to be addressed to bridge this gap, namely, the presence of challenging viewpoint transformations and "long-term" context variations like, e.g, those occurring between one day and another. In the following two chapters, we will consider them, with the ultimate goal of building a visual recognition system that is robust to viewpoint transformations (Chapter 7) and can be updated incrementally (Chapter 8) to gain also robustness to other light and context changes.

In Chapter 10 we will complement some of the critical observations pointed out by this study with a review of possible avenues of improvement and related directions of research.

Appendices

6.A Off-the-shelf models and Dataset Bias

In Sec. 6.3.1 we observed that a direct application of off-the-shelf CNNs to ICWT leads to poor performance. In this Section we provide some additional information on the experiment reported in Fig. 6.1. Notice that the image processing protocol described in the following is the one that we adopted in all our experiments, unless differently specified.

6.A.1 Image Preprocessing

When applying deep CNNs to our setting, we first evaluated the impact of considering coarser or finer regions around the object of interest in the images. Indeed, we recall that each image in ICWT is annotated with the coordinates of the object's centroid and with a bounding box provided by segmentation of the depth map (that is the smallest rectangle enclosing the object's blob, see Sec. 5.3.2 for details). We took advantage of this localization in order to avoid considering the full images for classification and we compared different strategies to extract a crop around the object from the images. Specifically, we tried either to extract a square crop of fixed size centered on the object's centroid, or to use the bounding box provided by the depth map segmentation. In both cases we tried two sizes: using a radius of 256 or 384 for the square crop, and leaving a margin of 30 or 60 pixels around the depth's bounding box.

Then, since deep CNNs require a fixed sized input (227×227 for *CaffeNet* and 224×224 for the other models), different preprocessing strategies can be applied in order to feed an image of arbitrary size to a network. We reproduced the operations that are specified on the reference page of each particular CAFFE model. For convenience we report them in Tab. 6.A.1. These, in general, consist in the subtraction of the mean image (or the mean pixel value) of the dataset on which the model has been trained on, and in the extraction of a grid of crops (eventually at multiple scales as for *VGG-16*) of the required size. Usually, the final prediction is computed by aggregating the predictions of more crops to gain robustness. Since in our case the image is in fact already a crop around the object, we compared the advantage of this grid approach versus just considering the central crop (and a single scale in the case of *VGG-16*). Fig. 6.A.1 reports therefore the same experiment of Fig. 6.1, but testing off-the-shelf CNNs on ICWT with the different described cropping strategies. It can be noticed that performing a finer localization of the object in the image (Blue and Green) provides better performance for all architectures. Also, considering more than the central crop provides a little advantage (for *VGG-16* this is even detrimental). This is reasonable, because we already localized the object which is assumed to be centered in the considered image region.

Table 6.A.1: Preprocessing operations executed on images before feeding the networks.

Model	Mean Subtraction	Scaling	Crop Extraction
CaffeNet ResNet-50	image	mean image size (256×256)	2×2 grid + center mirrored
GoogLeNet	pixel	256×256	2×2 grid + center mirrored
VGG-16	pixel	shorter side to 256, 384, 512	5×5 grid at each scale mirrored

We therefore decided to apply CNNs by extracting a square region of 256×256 from iCWT images, and considering only the central crop (Light Green in Fig. 6.A.1). In this way we avoided scaling the image (for *VGG-16* we considered the smallest scale). The performance reported in Fig. 6.1 (Dark Blue) corresponds to this preprocessing, which is also used in all experiments. We care to note that we decided to extract fixed-size regions instead of relying on the bounding box of the depth map, since we aimed at evaluating, in the following of our analysis, also invariance properties to geometric transformations (as, e.g, scaling). We finally note that, when applying the networks to ImageNet, we used instead the multi-crop strategy suggested for each of the architectures (see Tab. 6.A.1).

6.A.2 1000-class Categorization Results

In Fig. 6.1 we reported the accuracy of off-the-shelf CNNs on iCWT and ImageNet. Since the task was reduced to a 11-class categorization setting, we considered the prediction of the networks limited to the 11 corresponding labels rather than the full vector of 1000 scores normally produced by a CNN trained on the ILSVRC. For completeness, in Fig. 6.A.2 we report the resulting accuracy of the same CNNs but taking as prediction the class achieving maximum score over all available 1000 – even if 889 classes were actually not represented neither in the iCWT nor in the reduced ImageNet. As can be noticed, performance drops even below chance on iCWT (the "real" chance is now $1/1000$ rather than $1/11$).

6.A.3 Viewpoint Biases

In this Section we follow-up the analysis reported in Sec. 6.3.1, discussing potential biases in ImageNet, which could prevent off-the-shelf models from generalizing well, when tested on iCubWorld without applying knowledge transfer techniques. To this end, we report a series of excerpts showing the frame-by-frame predictions of some of the considered CNNs when

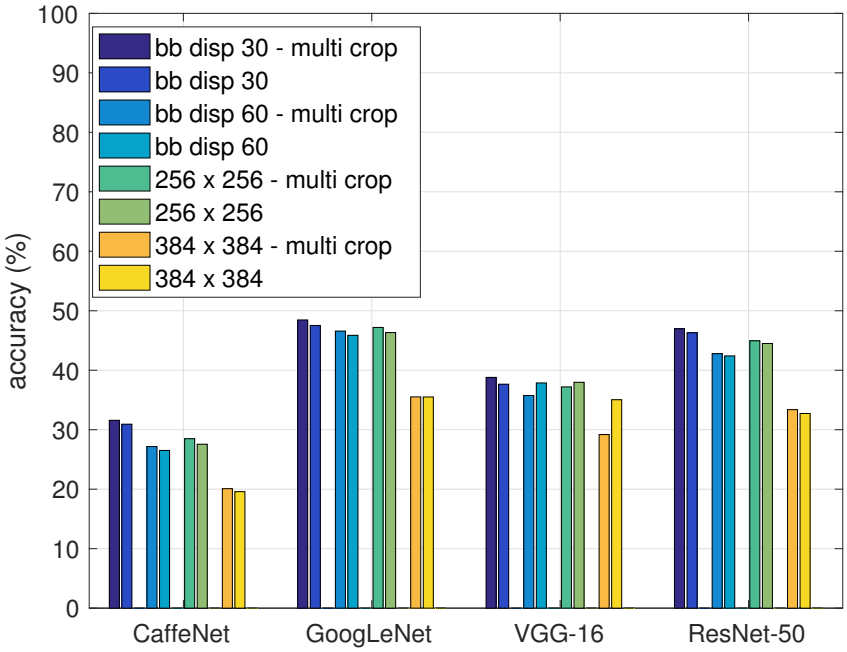

Figure 6.A.1: Average classification accuracy of off-the-shelf networks tested on ICWT segmenting the object according to different strategies.

tested on ICWT sequences. This analysis is qualitative, but allows to better understand what frames of ICWT are actually "harder" to recognize for the models and compare them with prototypical examples in ImageNet.

In particular, we consider *GoogLeNet* (similar observations were made for the other architectures) and in Fig. 6.A.3- 6.A.4 we report its frame-by-frame predictions on a subset of the test set considered in Fig. 6.1 (specifically, the 5 best recognized categories and two specific visual transformations: *Scale* and *2D Rotation*). We show predictions separately for each category and, in each plot, we represent the sequences of the 10 object instances of that category as rows of a matrix (the temporal dimension, or frame index, is reported in the horizontal axis). In each row, the frame-by-frame predictions over the sequence are represented as vertical bars: $White$ if the prediction is correct, Red if it is wrong (and $Black$ if the sequence is terminated). As in Fig. 6.1, predictions are computed as the maximum score among those of the 11 classes present in test set.

Fig. 6.A.3 represents sequences containing the *Scale* transformation. It can be noticed that, since during the acquisition of this sequence the operator was moving the object back and forth in front of the robot, starting close and slowly moving backward (eventually re-approaching the

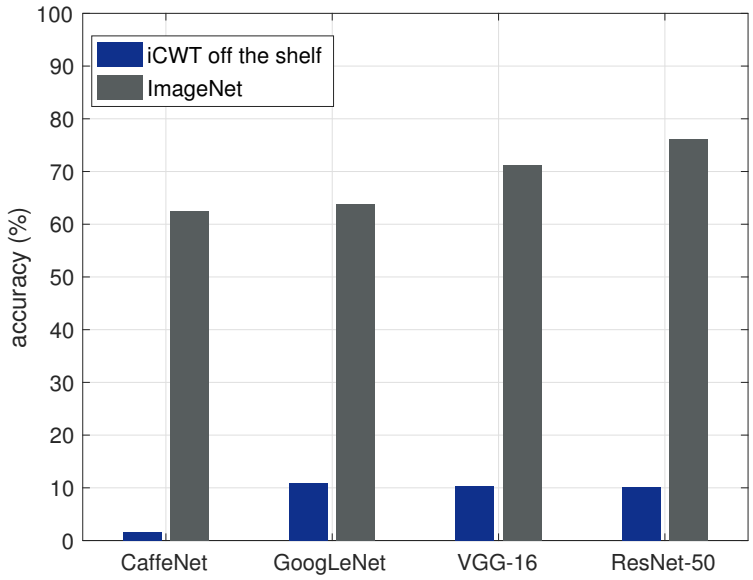

Figure 6.A.2: Average classification accuracy of off-the-shelf networks (trained on ILSVRC) tested on iCWT (Dark Blue) or on ImageNet itself (Gray). The test sets for the two datasets are restricted to the 11 shared categories (see Sec. 5.5, Fig. 5.8).

robot in some sequences), the CNN manages to recognize the object only when this appears at a relatively large scale (white bars mostly concentrated in the first half of rows). This qualitative observation confirms recent studies [Rodner et al., 2013, Herranz et al., 2016], highlighting the problem of the scale bias in ImageNet, and generally in all "object-centric" image datasets. Indeed, often, as in this case, the scale bias is a main barrier towards the direct application of off-the-shelf systems in real-world problems, where objects can appear at a smaller scale (like in the second part of our sequences). Note that, in any case, the image is always pre-cropped and we do not feed the full image to the classification network. In Fig. 5.5 it is possible to observe an example of images in iCWT taken at different scales.

In Fig. 6.A.4 we report the same for sequences containing *2D Rotation*. In this case, the operator was rotating the object (always maintaining the same face visible to the robot) at a constant speed. Indeed, it can be noticed that on many sequences there are periodic time intervals when the CNN does (or does not) recognize the object, corresponding to configurations in which that category appears more or less frequently in the ImageNet dataset. For example, we observed that soap dispensers and mugs in ImageNet are mostly placed on tables, and consequently are never recognized when they are, e.g. upside-down.

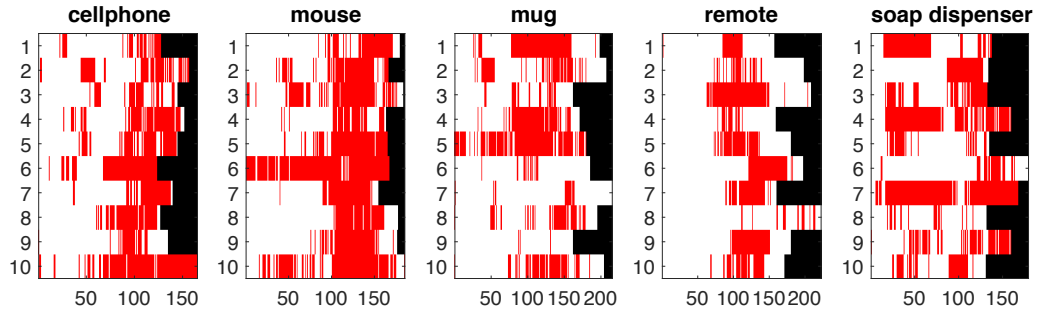

Figure 6.A.3: Frame-by-frame predictions of *GoogLeNet* on image sequences containing the *Scale* transformation, reported for 5 categories in different plots. The sequences of the 10 object instances belonging to each category are represented as matrix rows (the temporal dimension, or frame index, is reported in the horizontal axis)). In each row, the frame-by-frame predictions over the sequence are represented as vertical bars: $White$ if the prediction is correct, Red if it is wrong (and $Black$ if the sequence is terminated).

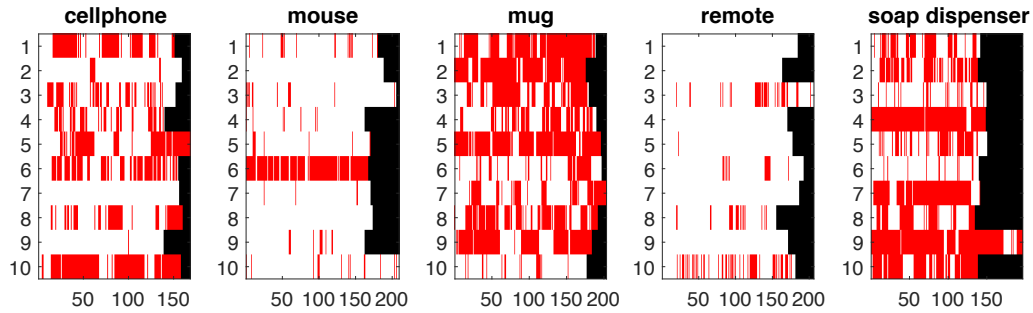

Figure 6.A.4: Same as Fig. 6.A.3, but on image sequences containing 2*D Rotation*.

6.B Model (Pre) Selection

In this Section we provide details regarding the parameter selection of the learning methods adopted in this work, which were described in Sec. 6.2.

6.B.1 Feature Extraction and RLSC

Here we specify the design choices that we made to implement each of the steps described in Sec. 6.2.2.

Image Preprocessing. When feeding images to the networks in order to extract their representation, we applied the preprocessing pipeline that was explained and justified in Sec. 6.A.1.

Feature Extraction. For each architecture, the layers considered as output to extract image representations were specified in Tab. 6.1. For *CaffeNet* and *VGG-16* we indicated either *fc6* or *fc7* layers, since these networks have more than one FC layer that can provide vector representations of the full image, and we used one or the other depending on the setting. Indeed, it has been observed (see Sec. 4.3.1) that the best output layer choice depends on the distance between the original and the target domain, with, generally, higher layers being preferable for more similar domains. Indeed, we will see in the following that our results confirm this empirical observation also on ICWT.

Classifier. We resorted to [Rudi et al., 2015] for the Nystrom subsampling approach that we used to implement the Gaussian Kernel RLSC. This algorithm involves determining some hyper-parameters, the most critical being the number m of training examples to subsample to approximate the kernel matrix.

In order to evaluate the best output layer for *CaffeNet* and *VGG-16*, and also to assign a reasonable value to m, we considered one small and one large categorization tasks on iCWT, representative of the smallest and largest task that we expected to run in our analyses, and compared the RLCS accuracy on them when varying these parameters. We fixed a 15-class categorization problem, training on 7 object instances per category and validating on 2. For each instance, we considered the sequences of all 5 visual transformations and one camera. We considered only one day, among the two available in iCWT, for training, and tested on the instance left out from the training set, in both days. For the large task, we considered all frames per sequence, leading to ~ 6300 training examples per class, whereas for the small task we subsampled ~ 4 frames per sequence, leading to ~ 170 examples per class (similarly to what we did in the first experiment of Sec. 6.3.

Fig. 6.B.2 and 6.B.1 reports the average accuracy respectively for the small and large experiment. We performed the two experiments using the representations from the four considered architectures (reported separately), using either *fc6* layer (Gray) or *fc7* (Pink) for *CaffeNet* and *VGG-16*. In each case, we increased logarithmically the value of m (horizontal axis) starting from $\sqrt{N}$, N being the size of the training set. For the small experiment we stopped at N, whereas for the large experiment we stopped at values where we observed very little performance improvements with respect to much longer training times. The performance on the same day on which we trained is reported in Light (Gray/Pink), whereas the performance on the other day is reported in Dark colors.

We observed that *fc6* features consistently performed better than *fc7*, confirming the observations above regarding the fact that lower layers can be better when there is a certain

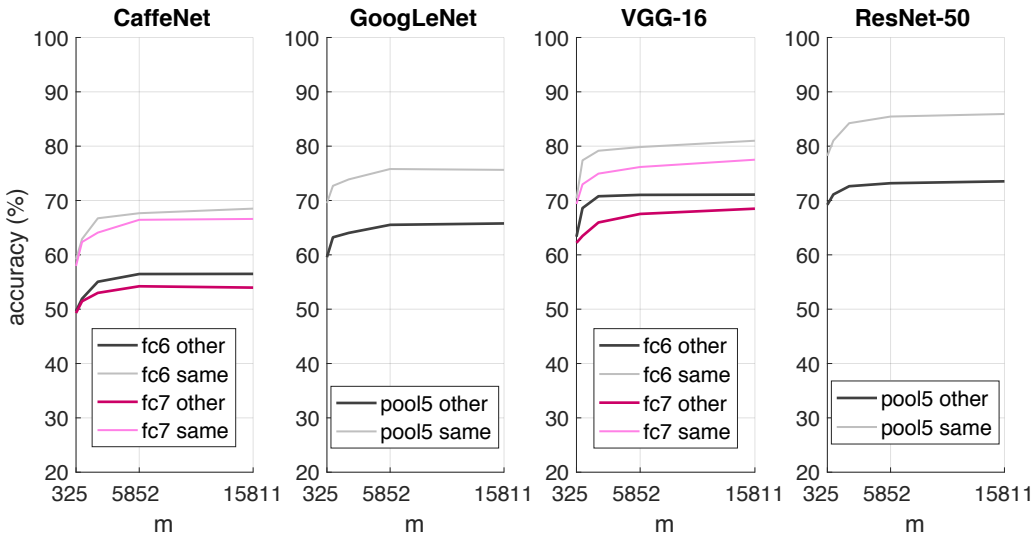

Figure 6.B.1: **Classification accuracy reported by training RLSC over representations extracted by the four considered architectures**. A "large" categorization experiment is performed (with $\sim N = 95000$ training examples) and m is varied between $\sqrt{N}$ and 15000 (horizontal axis). Performance on the same training day (Light colors) and on another one (Dark colors) is reported.

distance between the original training set (ILSVRC) and the target one (ICWT). Indeed, in Chapter 7 we will extract features from models tuned on ICWT finding that in that case instead *fc7* performs better than *fc6*. We therefore decided to use the *fc6* layer when extracting representations from off-the-shelf *CaffeNet*/*VGG-16* on images of iCWT. We also observed that relatively small values of m could provide good accuracies and therefore we fixed $m = \min(15000, N)$ in all experiments.

6.B.2 Fine-tuning

In this Section we provide details on how we performed the fine-tuning procedure and on the choice of the two strategies, *adaptive* and *conservative*, reported in Tab. 6.2.

General Protocol

The image preprocessing pipeline executed in this case was similar to the one reported in Sec. 6.A.1: once a square 256×256 region around the object's centroid was extracted, we either subtracted the mean image (*CaffeNet*) or pixel (*GoogLeNet*) of the training set. When training, the rest of processing was performed within CAFFE, specifically extracting a random

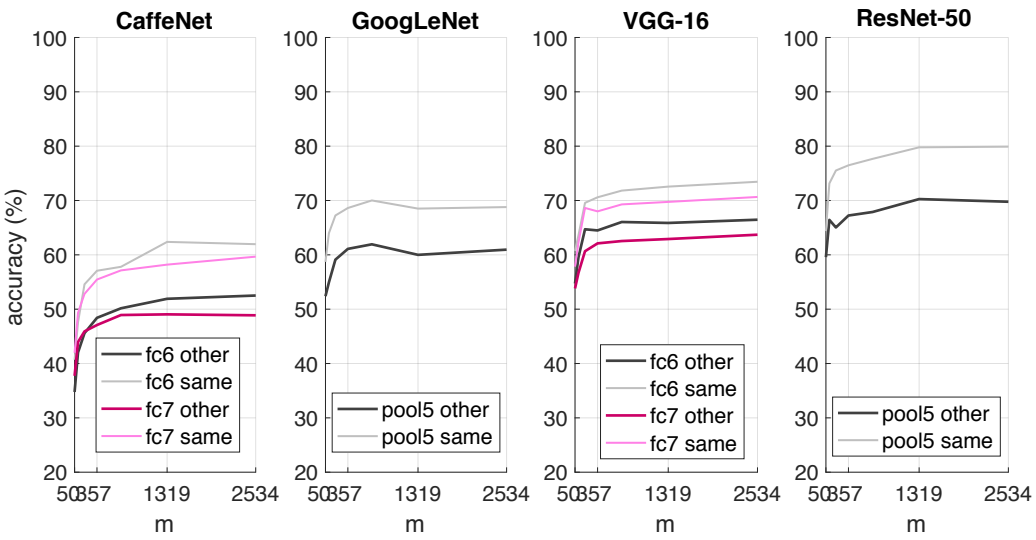

Figure 6.B.2: **Classification accuracy reported by training RLSC over representations extracted by the four considered architectures**. Similar to Fig. 6.B.1 but on a "small" categorization experiment ($\sim N = 2500$ training examples), varying m between $\sqrt{N}$ and N.

227×227 crop (randomly mirrored horizontally) before passing it to the network. At test phase, the prediction over one image was taken by extracting the central crop instead of a random one (see Sec. 6.A.1).

The training set was shuffled before starting the fine-tuning process since we observed that similarity of images within a batch negatively affected convergence of the backpropagation algorithm (see also Sec. 4.1.2). When fine-tuning a network, we evaluated the model's performance on a validation set every epoch and we finally chose the epoch achieving highest validation accuracy.

We left the batch size to the values specified in CAFFE (and reported in Tab. 6.2).

Since we needed to perform a high number of fine-tuning experiments, in general we tried to keep the number of iterations – or, equivalently, of epochs – low, compromising between the increase in the training time and the performance improvement provided. The number of epochs was hence fixed observing that, in general, the performance was increasing slowly after $\sim 6 - 8$ epochs for all models but for the *conservative* fine-tuning of *CaffeNet*, that involves the learning of all parameters of the 3 FC layers and takes more time to converge. In this case, we fixed the number of epochs to ~ 36.

It is possible that tuning more accurately the number of epochs for the different experiments, and in particular letting the learning to proceed for longer times, slight improvements may be

obtained. However, a similar reasoning also applies to the Nystrom RLSC, where we set m to a relatively small value, compromising with the training times. In general, one criterium we followed was to achieve comparable training times between fine-tuning and feature extraction with RLSC. Indeed, we were interested in comparing the application of the two methods to a robotic domain, producing an overall performance assessment of the trends rather than focusing on optimizing specific tasks.

Hyper parameters Choice

In this Section we report on the empirical analysis that we performed in order to understand the effect and the relative importance of the hyper parameters involved in fine-tuning deep architectures as *CaffeNet* or *GoogLeNet* in our scenario.

While model selection is per se an open problem when dealing with deep networks, in our setting this issue is even more complicated by the fact that we do not have a fixed reference task on which to optimize the training (as e.g., can be the ILSVRC), but we instead plan to span over wide range of tasks, comprising small or large training sets. Indeed, the study presented in this work aims at comparing models trained on different categories, objects, transformations, and so forth.

We considered therefore the same two experiments that we used to perform parameter selection for the RLSC approach (the "small" and "large" categorization tasks described in Sec. 6.B.1) and we fine-tuned *CaffeNet* and *GoogLeNet* on them by varying the values of multiple hyper parameters. In the following, we report this analysis separately for the two architectures.

CaffeNet. We considered the parameters appearing in Tab. 6.2 and tried different combination of values for them, according to the considerations exposed in the following.

When fine-tuning CNNs on ICWT from off-the-shelf models, we followed the criterium that either layers are initialized with off-the-shelf weights, and their learning rates are all equal to the so called "base" learning rate of the network, or we learn them from scratch, setting therefore their learning rates to higher values. For *CaffeNet*, we experimented learning from scratch either only the last fully-connected (FC) layer $fc8$, or also $fc7$ and $fc6$ (we tried to learn from scratch also convolution (C) layers but with no success). We started setting the learning rate of $fc8$ to 10^{-2}; then, each time we learned from scratch one FC layer more, as an empirical rule we decreased the learning rates of the FC layers by a factor of 10: therefore we tried learning from scratch $fc8$ and $fc7$ with learning rate 10^{-3}, or $fc8$,$fc7$ and $fc6$ with learning rate 10^{-4}. Independently, for each of the three choices we swept the base learning rate from 0 to a maximum of 10^{-3} – we observed that higher base learning rates generally

prevented the training from converging. Note that all these values are in fact the *initial* learning rates, since then their value is usually decreased during SGD, according to some decay policy (see Sec. 2.2)

In the following we list, for each fine-tuning parameter, the values that we tried. We considered all possible combinations of values across parameters. For the parameters that are missing here, we kept their value as specified in *Caffe* reference models.

Base LR: starting learning rate of the layers that are initialized with the parameters of the off-the-shelf model. We tried $10^{-3}, 5 * 10^{-4}, 10^{-4}, 5 * 10^{-5}, 10^{-5}, 10^{-6}, 0$.

Learned FC Layers: which fully-connected (FC) layers are learned from scratch and their specific starting LR. As anticipated, we tried to learn (i) only $fc8$ with starting LR set to 10^{-2}, or (ii) including also $fc7$ and (iii) finally also $fc6$. As an empirical rule, every time we included one more layer to learn from scratch, we decreased the starting LR of these layers of a factor of 10 (hence 10^{-3} in (ii) and 10^{-4} in (iii)).

Dropout %: in FC layers. We tried 50% (default CAFFE value) or 65%.

Solver: algorithm used for the SGD. We used the standard *SGD* solver [Bottou, 2012] in CAFFE.

LR Decay Policy: decay of the learning rates. We tried either polynomial decay, where the learning rate at iteration t is given by $\eta_t = base\_lr(1 - \frac{t}{T})^p$ with $p = 0.5$ or $p = -3$, T being the total number of iterations, or step decay, decreasing the LR of a factor of 10 every $\frac{T}{3}$ iterations.

We observed that the dropout percentage had a small influence and left it to the default value. We also observed that the polynomial decay with smaller slope was consistently better (of $\sim 5 - 10\%$ accuracy). The most critical parameters were instead the base LR and the numbers of FC layers learned from scratch. In Fig. 6.B.3 we report as an example the accuracy obtained respectively when learning only $fc8$ (Left) or $fc6$, $fc7$ and $fc8$ (Right). In each case, we decreased the base LR of all other layers from 10^{-3} to 0 (horizontal axis). Performance is reported, as in Fig. 6.B.2 and 6.B.1, for both the same day of training (Light Gray) and a different one (Dark Gray). While we tried all values of base learning rate on the "small" experiment (Continuous Line), for the "large" experiment we chose one small, medium and large value (Dots).

For interpreting the results, it can be useful to recall that the strategy that learns from scratch the three FC layers (Right) has many more free parameters than the one learning only the last FC layer (Left). Indeed, we notice that this latter strategy is more robust to the small-scale scenario, for any base LR (Continuos Line in the range 40-60% to the Left and 20-40% to the Right).

We also observe that, when we add frames to the training set, both strategies benefit from

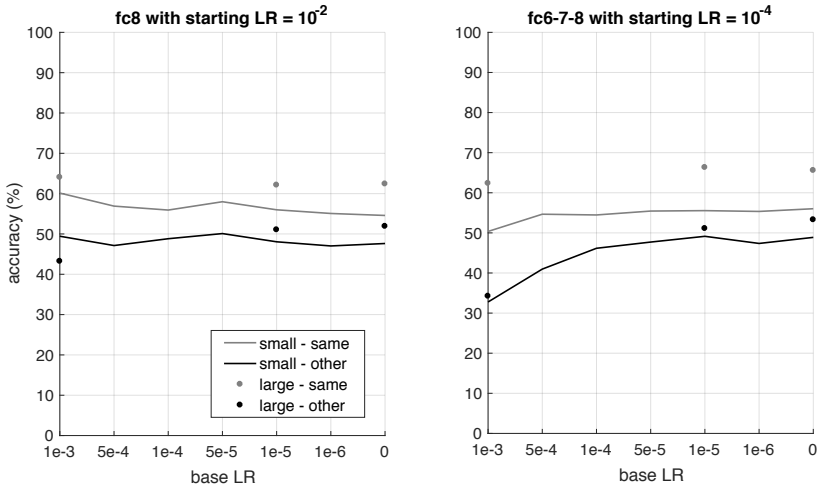

Figure 6.B.3: **Classification accuracy provided by fine-tuning *CaffeNet* according to different strategies**: either learning from scratch only $fc8$ (Left) or $fc6$, $fc7$ and $fc8$ (Right). Performance is reported for both the same day of training (Light Gray) and a different one (Dark Gray). We tried multiple values of base LR on a "small" training set (Continuous Line), and only one small, medium and large value (Dots) on a "large" one.

learning slowly (i.e., when we pass from the Continuos Line to the corresponding Dot we gain more with smaller base LRs). Indeed, in the large-scale experiment learning $fc6$-7-8 with small base LR achieves the best performance. On the contrary, learning only $fc8$ with large base LR almost does not benefit from seeing more frames and even overfits the day of training.

Therefore, since in our analysis we aimed at varying the size of the training set across a wide range, we chose two representative strategies providing best performance respectively in the small and large-scale experiment: learning $fc6$-7-8 with small (actually 0) base LR, that we call the *conservative* strategy, to account for our larger-scale settings, and learning only $fc8$ with large base LR, that we call the *adaptive* strategy, to account for smaller-scale settings.

GoogLeNet. We performed for *GoogLeNet* a similar analysis to the one reported for *CaffeNet*.

We recall that the considered *GoogLeNet* architecture is composed of "main" branch, ending with one FC layer (called *loss3/classifier*), and two identical "auxiliary" branches, ending with two FC layers (called *loss1(2)/fc* and *loss1(2)/classifier*). We refer to [Szegedy et al., 2015] for a detailed description of the model. Considering this structure, we then explored the following parameters:

Base LR: varied as for *CaffeNet*.

Learned FC Layers: we always learned *loss3/classifier* from scratch with starting LR

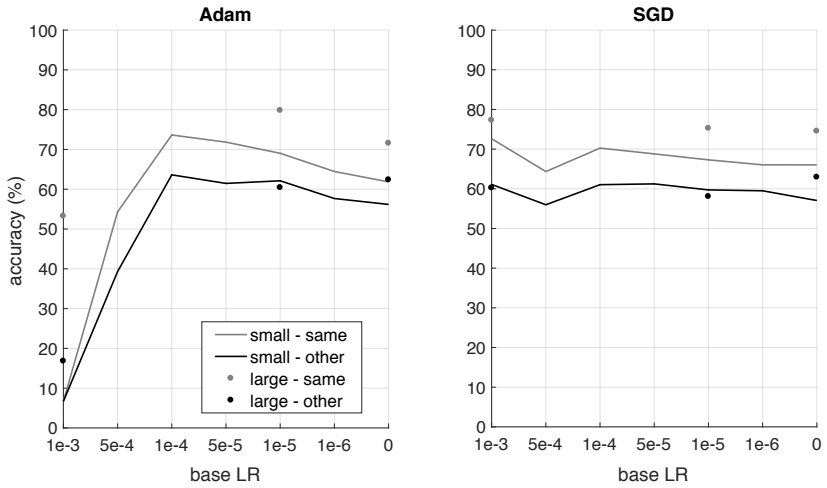

Figure 6.B.4: **Classification accuracy provided by fine-tuning** *GoogLeNet* **according to different strategies**: either using the *Adam* solver (Left) or *SGD* (Right). The rest of the figure is similar to Fig. 6.B.3

equal to 10^{-2}; regarding the auxiliary branches, we tried (i) to cut them out, (ii) to learn also *loss1(2)/classifier* from scratch with starting LR equal to 10^{-2}, or, finally, (iii) to learn from scratch also *loss1(2)/fc*, setting the starting LR of these layers and of *loss1(2)/classifier* to 10^{-3} (following the same heuristic as for *CaffeNet* of decreasing the LR of a factor of 10 when adding one more FC layer to be learned from scratch).

Dropout %: we tried either default CAFFE values (40% for *loss3/classifier* and 70% for *loss1(2)/fc*) or a higher percentage: 60% for *loss3/classifier* and 80% for *loss1(2)/fc*.

Solver: we tried either *SGD* or *Adam* [Kingma and Ba, 2015] solvers in CAFFE.

LR Decay Policy: when using *SGD*, we used polynomial decay with $p = 0.5$ or $p = -3$; when using *Adam*, we maintained the learning rate constant.

As for *CaffeNet*, we tried all combinations and left the parameters not mentioned to default values in CAFFE.

We first observed that, differently from *CaffeNet*, overall for this architecture the impact of learning the FC layers from scratch or not learning them was very small, with the three tested strategies behaving very similarly. We chose the last one (iii), that was slightly better than the others. We also observed a small benefit from using higher dropout percentages.

A slightly more critical aspect was instead the choice of the solver. To this end, in Fig. 6.B.4 we report the accuracy obtained respectively when using the *Adam* solver (Left) or *SGD* (Right). In the latter case the polynomial decay with smaller slope is reported, since we observed again that it was consistently better. Performance is reported, as in Fig. 6.B.3, for both the same day

of training (Light Gray) and a different one (Dark Gray). We varied again the base LR from 10^{-3} to 0 (horizontal axis), trying all values on the "small" experiment (Continuous Line) and one small, medium and large value for the "large" experiment (Dots).

It can be observed that, while the *SGD* solver is more robust to different choices of base LR, *Adam* provides slightly better accuracies for mid-range values of base LR, both for the small and the large experiment. We therefore opted for this latter solver. Note that we tried the *Adam* solver also for *CaffeNet*, but we observed that it was not converging for that architecture and the tuning protocols considered in the analysis.

Similar observations as for the *CaffeNet* architecture led us to conclude that learning slowly (with a smaller base LR) can be preferable when we add more frames to the training set, since the model is less prone to overfit the training condition (i.e. with base LR equal to 0 we gain performance on both days when passing from the Continuos Line to the corresponding Dot). Larger base LR can be preferable instead when the training set is smaller. As reported in Tab. 6.2 therefore, also for *GoogLeNet* we identified an *adaptive* strategy with base LR equal to 0 and a *conservative* strategy with base LR equal to 10^{-4}.

6.C More Experiments on Categorization

In this Section we complement the experiments discussed in Sec. 6.3. These results are reported to show that the behavior observed for the specific choices considered in the paper (e.g. number of object classes considered for the tasks) holds also in other settings.

For the sake of completeness, we specify that in Sec. 6.3 we considered the 15-class categorization task comprising the following categories: $cellphone$, $mouse$, *coffee mug*, *pencil case*, $perfume$, $remote$, *ring binder*, *soap dispenser*, $sunglasses$, $flower$, $wallet$, $glass$, $hairbrush$, *hair clip*, $book$.

What do we gain by adding more frames?

Fig. 6.C.1 reports the results of the same experiment shown in Fig. 6.2, but performed including only 3 object instances per category in the training sets.

These results further confirm that, in this setting, adding frames without increasing semantic variability cannot be used as a viable strategy to improve categorization accuracy, which remains definitely lower than the performance achieved using 7 example instances (Fig. 6.2). Moreover, we observe that, when having too few instances with respect to the number of images per instance (as it happens in this experiment when using, e.g., 300 frames per sequence), fine-tuning deep CNNs even worsens their performance on categorization with respect to keeping the original representation learned on ImageNet. This confirms our insights exposed

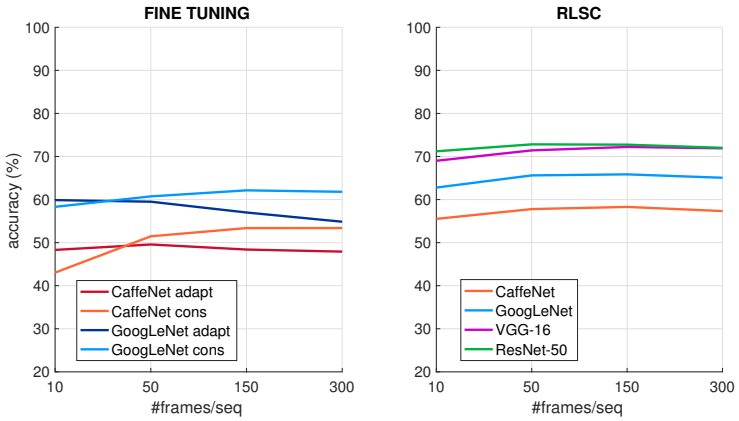

Figure 6.C.1: **Recognition accuracy vs # frames** (number of object views available during training). Same experiment as Fig. 6.2 but executed by training only on 3 example instances per category.

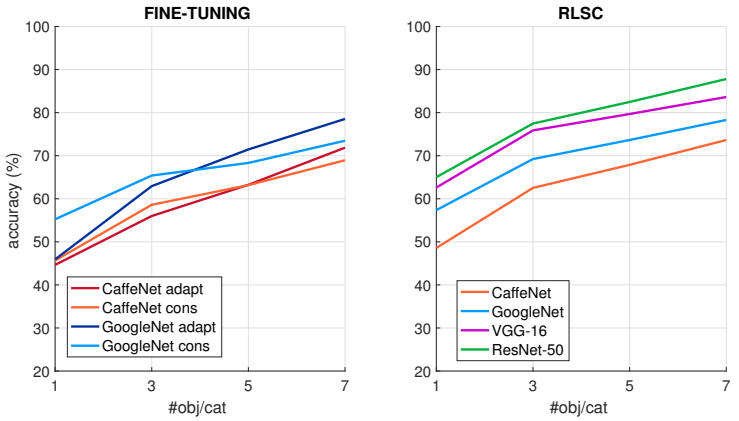

Figure 6.C.2: **Recognition accuracy vs # instances** (number of object instances available during training). Same experiment as Fig. 6.3 executed for a 10-class categorization problem.

in Sec. 6.3 regarding the fact that the most suitable dataset format for categorization is indeed one maximizing semantic variability, as in image retrieval, and that the visual data typically collectable in robot vision settings limits performance of deep CNNs.

What do we gain by adding more instances?

Fig. 6.C.2 and 6.C.3 report the results of an experiment analogous to the one shown in Fig. 6.3, but for respectively 10 and 5 object categories rather than 15. As can be noticed, the observations reported in Sec. 6.3 apply also here: the accuracy increases remarkably as more example instances per category are made available (without saturating), supporting the claim

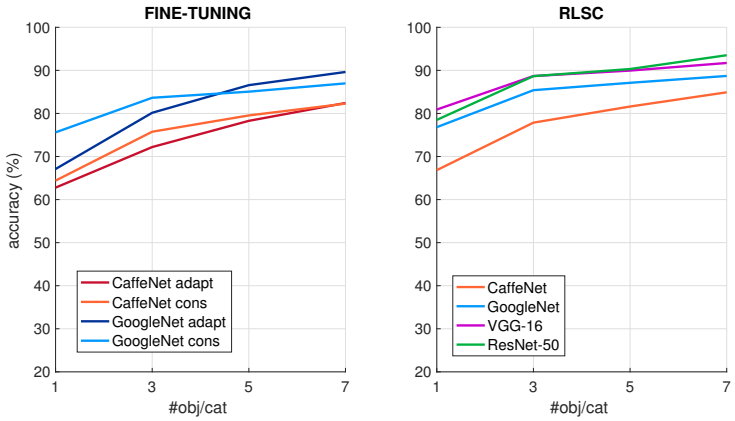

Figure 6.C.3: **Recognition accuracy vs # instances** (number of object instances available during training). Same experiment as Fig. 6.3 executed for a 5-class categorization problem.

that semantic variability is indeed critical for categorization even in settings that involve very few categories.

6.D Comparison with Washington RGB-D

In this Section we apply the methods that we investigated in this work on ICWT to the Washington RGB-D benchmark [Lai et al., 2011]. Since ICWT is a new data collection, the goal of this Section is to provide a performance reference for our approaches, with respect to other works in the literature that recently improved the state-of-the-art on this benchmark.

Washington RGB-D. Washington RGB-D (in the following WRGB-D for brevity) is a turntable dataset comprising 300 objects divided into 51 categories. It contains a variable number of object instances per category (from 3 to 14), ~ 6 on average.

For each object instance, three sequences were acquired by an RGB-D camera mounted at ~ 1 meter from the object and different elevation angles relative to the horizontal table $(30, 45, 60^\circ)$. In each sequence, the camera did one revolution around the table at a constant speed, recording ~ 250 frames at 20Hz, for a total of ~ 250k RGB and depth frames (640×480 resolution) in the dataset.

The dataset provides an object segmentation mask based on depth and color, that the authors used to release a version with cropped RGB-D frames that tightly include the object. In our experiments we used this version, as is done in the papers we will compare our results to. Also, usually for both categorization and identification tasks only every fifth frame of the sequences is considered, and we followed this practice, using the same 41877-sized dataset

Table 6.D.1: **Categorization results on Washington RGB-D, comparison with state-of-the-art:** RLSC on top of extracted features from the 4 considered architectures is compared with results in [Schwarz et al., 2015]. The horizontal line separates the use of *CaffeNet* from other architectures.

Method	Accuracy (%)
Schwarz et al. 2015	83.1 ± 2.0
CaffeNet RLSC	83.1 ± 2.8
GoogLeNet RLSC	84.6 ± 2.8
VGG-16 RLSC	87.5 ± 2.3
ResNet-50 RLSC	$\mathbf{89.4 \pm 3.1}$

version employed in the literature.

6.D.1 Object Categorization Benchmark

The categorization task usually considered on this dataset was proposed in [Lai et al., 2011] and consists in discriminating between the 51 classes in the dataset, by leaving out one instance per category for testing and using the remaining ones for training. The authors performed 10 trials, each time randomly sampling the test instances and, since they released the lists of the 10 splits, usually in the literature the same splits are used and performance is averaged over them. We followed this practice. Each split consists of $\sim 35k$ training and $7k$ test images.

We evaluated on this task the 4 CNN architectures considered in this work, with the related transfer learning methods. We used the settings described respectively in Sec. 6.2.2 and 6.2.3 for feature extraction followed by RLSC and fine-tuning. We used a random 20% of the training set for validation. Table 6.D.1 compares our results obtained with RLSC with the performance reported in [Schwarz et al., 2015], while Table 6.D.2 compares our results obtained with fine-tuning with the performance reported in [Eitel et al., 2015]. All results refer to the use of only RGB frames, without the depth information.

Table 6.D.2: **Categorization results on Washington RGB-D, comparison with state-of-the-art:** *adaptive* and *conservative* fine-tuning of *CaffeNet* and *GoogLeNet* is compared with results in [Eitel et al., 2015]. The horizontal line separates the use of *CaffeNet* from other architectures.

Method	Accuracy (%)
Eitel et al. 2015	$\mathbf{84.1 \pm 2.7}$
CaffeNet FT adapt	81.9 ± 2.7
CaffeNet FT cons	83.6 ± 2.4
GoogLeNet FT adapt	82.4 ± 2.8
GoogLeNet FT cons	83.9 ± 2.3

In both papers the authors use only *CaffeNet*. Specifically, in [Schwarz et al., 2015] they preprocess the images by using the depth segmentation mask to remove the background around the object; then they extract features from $fc7$ and $fc8$ and give their concatenation to linear SVMs. With respect to this work, we avoid complex preprocessing steps (see Appendix 6.A.1), we extract earlier features, and employ the Nystrom subsampling approach at the classifier level, which allows to save computation and use a Gaussian Kernel instead of a linear one. Overall, as can be seen in Table 6.D.1, with *CaffeNet* we achieve similar classification accuracy, 83.1% and, in line with the literature and our results on ICWT, more recent networks gradually improve performance on this benchmark.

In [Eitel et al., 2015] the authors fine-tune *CaffeNet*: they apply a standard image preprocessing, similar to ours, but isotropically rescaling the images and tiling the borders with artificial context. Their fine-tuning protocol is rather "aggressive", with a remarkably high initial learning rate of 0.01 for all layers, reduced by a factor of 10 every $\frac{T}{3}$ epochs and running the training for $T \sim 100$ epochs (with a batch size of 128). Such a long training would take ~ 8 hours on an NVIDIA Tesla k40 GPU. To this regard, we note that, since we observed that fine-tuning for longer times in our experiments generally provided smaller improvements, in our analyses we considered relatively fewer epochs, in order to be able to perform more experiments in reasonable time. Indeed, even if we do not achieve 84.1%, considering also the variance associated to such measurements, reporting 83.6% in just ~ 30 minutes (the time required to complete our training on a Tesla k40 GPU) can be considered a good result. It can be noticed that *GoogLeNet* fine-tuned with our strategies on this benchmark does not report particularly good results. This may be due to the fact that the specified strategies are the result of a model selection (see Sec. 6.B.2) that led us to choose them as the ones reporting best results on representative recognition tasks on ICWT. Therefore, such strategies may not be optimal for WRGB-D. To this end, we tried varying some hyper parameter (e.g., scaling learning rates to higher/lower values, or changing the CAFFE solver, and managed to improve performance on WRGB-D by some percentage point (we do not report these numbers since are out of the scope of the work).

6.D.2 Object Identification Benchmark

Like categorization, also the identification task usually considered on this dataset is the one proposed in [Lai et al., 2011] and consists in discriminating between the 300 objects in the dataset. For each object, the two sequences recorded at 30° and 60° elevation degrees are used for training, and the one recorded at 45° for testing. Globally, therefore, the training set comprises 28009 frames and the test set 13868 frames.

We evaluated again the 4 CNN architectures, with the transfer learning methods described

Table 6.D.3: **Identification results on Washington RGB-D, comparison with state-of-the-art:** RLSC on top of extracted features from the 4 considered architectures is compared with results in [Schwarz et al., 2015]. The horizontal line separates the use of *CaffeNet* from other architectures.

Method	Accuracy (%)
Schwarz et al. 2015	92.0
CaffeNet RLSC	94.0
GoogLeNet RLSC	94.3
VGG-16 RLSC	94.5
ResNet-50 RLSC	**96.0**

Table 6.D.4: **Identification results on Washington RGB-D, comparison with state-of-the-art:** *adaptive* and *conservative* fine-tuning of *CaffeNet* and *GoogLeNet* is compared with results in [Held et al., 2016]. The horizontal line separates the use of *CaffeNet* from other architectures.

Method	Accuracy (%)
Held et al. 2016	93.3
CaffeNet FT adapt	**94.0**
CaffeNet FT cons	92.7
GoogLeNet FT adapt	93.9
GoogLeNet FT cons	92.5

in Sec. 6.2.2 and 6.2.3. We used again a random 20% of the training set for validation. Table 6.D.3 compares our RLSC results with the performance reported in [Schwarz et al., 2015], while Table 6.D.4 compares our fine-tuning results with the performance reported in [Held et al., 2016]. Again, all methods use only RGB frames, without the depth information, and the two referred papers use *CaffeNet*.

From Table 6.D.3 it can be noticed that our feature extraction pipeline provides better results than [Schwarz et al., 2015] for this task (94% with respect to 92%). Apart from the differences in the image preprocessing and in the layers at which they extract the network output, we note that, in [Schwarz et al., 2015], the authors adopt, for the identification task, a hierarchical approach, where, first, the classifiers discriminate between categories, and then between object instances within categories.

In [Held et al., 2016] the authors address the problem of one-shot object identification on WRGB-D by fine-tuning deep CNNs in order to improve their viewpoint invariance. We will enter in the merit of their work in Chapter 7, when we will compare their approach with our work on invariance. However, here we compare our methods with the results that they report on the standard identification task.

They fine-tune *CaffeNet* by fixing convolution layers to the original values of the off-the-shelf model; then, they set the initial learning rate to 10^{-4} for $fc6$ and $fc7$, which they initialize also to off-the-shelf values; finally, the learn from scratch, with an initial learning rate of 10^{-3}, the last $fc8$ layer. Compared to our strategies, their protocol is similar to our *conservative* approach, in that they fix convolution layers; however, when fine-tuning, we followed the criterium that either layers are initialized with off-the-shelf weights, and their learning rates are all equal to the base learning rate of the network, or we learn them from scratch, setting their learning rates to higher values. Therefore, in our *conservative* approach we learned from scratch all three convolution layers with initial learning rate 10^{-4} (see Appendix 6.B.2 for more detailed explanation of our strategies). Another important difference is that they fine-tune for 100k iterations (with a batch size set to 256 and 28k training points, these are 900 epochs), that is considerably higher than our fine-tuning times (of the order of ~ 36 epochs at maximum). Overall, we achieve higher performance with the *adaptive* fine-tuning, confirming our results observed in Sec. 6.4 that, when changing task, it is beneficial to change more deeply the network layers.

Chapter 7

Object Identification from Few Examples by Improving the Invariance of Deep CNNs

In this Chapter we will deepen the analysis presented in Chapter 6, to better quantify to which extent the recognition performance of the considered deep Convolutional Neural Networks (CNNs) is impacted by the different visual nuisances contained in the setting of ICUBWORLD TRANSFORMATIONS (ICWT). We will exploit the peculiar structure of ICWT that, as we described in Chapter 5, contains image sequences representing objects while undergoing isolated visual transformations, to evaluate – and improve – the invariance properties of deep CNNs with respect to specific viewpoint and light changes typically affecting objects' appearance in real world situations.

The content of this Chapter builds on the observations previously published in [Pasquale et al., 2016a] and further extends the results reported therein.

7.1 Introduction

Assessing the invariance of recognition systems to visual nuisances is particularly relevant in robotics, because the real world is an infinite source of variabilities and a robot is required to learn the appearance of objects just from a limited set of example images, while being able to reliably recognize them in many other different configurations. In such setting, it becomes critical to rely on good visual representations that map images of objects into discriminative and robust representations.

While in Sec. 4.2.2 we overviewed recent work aiming to get insights on the invariance

properties of deep CNNs pre-trained on ImageNet, in our analysis in previous Chapter we verified that off-the-shelf CNNs usually must be "adapted" to the target domain in order to provide satisfactory performance. Representations extracted from off-the-shelf models can be fed to classifiers, or such models can be fine-tuned on the new domain. In this latter case, we saw that, on the one hand, it is possible to perform a light tuning, by updating only the last few layers of the network; on the other hand, a more aggressive tuning can be performed to change the layers more "deeply" based on the new data. We considered all these strategies and observed that it was not evident when one should be preferable over others. To this end, a first contribution of this work is to investigate how the different approaches to fine-tuning can affect the invariance properties of off-the-shelf models, specifically related to the kind of transformations observed during the tuning.

By building on these observations, we will identify which strategies disrupt and which others instead can improve the invariance of off-the-shelf deep networks, and devise an effective pipeline achieving remarkable performance on object identification tasks with just few example images. We assess our results on our ICWT benchmark and also on the standard Washington RGB-D dataset [Lai et al., 2011]. As a by product of this work, such pipeline has also been deployed on the iCub and R1 robots, resulting, as we will see in Chapter 9, in an application to teach robots to recognize new objects on the fly, from just few example images, in challenging real world scenarios.

We started investigating ideas related to invariance of deep CNNs in [Pasquale et al., 2016a], where we considered a subset of the categories and transformations present in ICWT. We then expanded the study in [Pasquale et al., 2017] and, in this Chapter, we provide a unified version of the work.

The rest of the Chapter is organized as follows: Sec. 7.1.1 reviews the approach adopted in our study in comparison to recent related work; Sec. 7.2 reports on a series of observations that led us to identify the most effective strategies to improve the invariance of off-the-shelf networks; finally, in Sec. 7.3 we build on these observations to devise an object identification pipeline that we evaluate on the ICWT dataset, and that we also validate on the Washington RGB-D benchmark (Sec. 7.4). Sec. 7.5 concludes the work, discussing potentials and limits of the proposed approach.

7.1.1 Methods and Related Work

In this Section we describe the approach that we take for the presented study, by comparing with recent related work.

The first problem that is encountered when aiming to investigate invariance properties of recognition methods to specific visual nuisances is getting suitable data. Indeed, we discussed

in Sec. 4.2.2 that, due to the lack of benchmarks representing objects while undergoing multiple, isolated, variability factors, most works (see, e.g., recently, [Aubry and Russell, 2015, Peng et al., 2015]) approach the problem by relying on synthetic imagery. This provides perfect control and flexibility over the kind of transformations applied to the images. However, as we explained previously, we are also interested in obtaining reproducible results that we can expect to hold in real world conditions. Indeed, these and other motivations (see Sec. 5.5) led us to the acquisition of ICWT. In this Chapter, we specifically exploit the tagged sequences provided by ICWT for our investigation of invariance. To our knowledge, among the (few) studies using real sequences (see, e.g., [LeCun et al., 2004, Goodfellow et al., 2009, Bakry et al., 2015, Borji et al., 2016]), our analysis is carried on over a notable wide range of transformations (scaling, in- and out-of-plane rotations, background and light changes).

We consider, like in Chapter 6, object categorization and identification tasks on ICWT. Specifically, we refer to the same experimental settings (see Sec. 6.3 and 6.4) and investigate the effect of specific visual transformations on the observed performance of the considered models. Following previous work (see Sec. 4.2.2), we measure the representation invariance of the considered CNNs by systematically training and testing models on different transformations contained in ICWT. Indeed, since our ultimate goal is devising effective recognition systems for real robotic platforms, our primary concern is evaluating and improving the performance of such systems in different training/testing conditions. Specifically, we will measure the performance of Regularized Least Square Classifiers (RLSC) trained on the representations extracted from the intermediate layers of CNNs. We refer to Sec. 6.2.2 for details, since in this Chapter we apply the same procedure.

After an initial assessment of the invariance properties of models trained on ImageNet (Sec. 7.2), we investigate how fine-tuning can affect such properties (Sec. 7.3). To this end, we apply the *conservative* and *adaptive* fine-tuning protocols described in Sec. 6.2.3, and specifically focus on the effect of this latter one (that modifies all layers, comprising convolution ones) to the internal network representation. These experiments will allow us to understand the most effective strategies to fine-tune off-the-shelf networks in order to improve the invariance of their internal representations to the viewpoint transformations present in ICWT. Indeed, by exploiting improved representations, we will finally be able to devise an effective recognition pipeline achieving remarkable performance on object identification tasks with just few example images.

Our results are in agreement with recent work in the instance retrieval literature [Babenko et al., 2014, Gordo et al., 2016] that observed particular benefits in fine-tuning off-the-shelf CNNs on datasets containing more images of the same object in different configurations. In particular, while it is true that off-the-shelf representations provide an “astounding” baseline

for a wide range of visual recognition tasks, including instance retrieval [Sharif Razavian et al., 2014], on the other hand it is also true that, when switching from categorization to identification, as we observed in Sec. 6.4, intra-class semantic variability leaves room to viewpoint (and light, context and so forth) variabilities potentially playing a decisive role for the performance of recognition methods.

In Sec. 7.4, we will validate the proposed approach on the problem setting that is considered in [Held et al., 2016], where the authors address one-shot object identification tasks on Washington RGB-D (WRGB-D in the following) by means of an analogous strategy. With respect to [Held et al., 2016], we extend the analysis of the problem to different CNN architectures and apply a more effective pipeline. Moreover, we consider different visual transformations, beyond the 3D viewpoint included in WRGB-D, and the more challenging robot-centric setting of ICWT. Indeed, with our pipeline, we achieve superior performance on one-shot object identification on WRGB-D. Finally, by including in the analysis also the object categorization task, we highlight some potential limitations of the proposed solution.

7.2 Evaluating Viewpoint Invariance of Off-the-shelf CNNs

In this Section, we start evaluating the invariance properties of the representations extracted by CNN models off-the-shelf (that is, trained on the ImageNet Large Scale Visual Recognition Challenge or ILSVRC), with respect to the different visual transformations present in ICWT. We will also observe how fine-tuning the networks on different transformations can "reshape" their performance on the experienced and not experienced transformations. As anticipated, we will conduct the analysis with respect to object categorization (Sec. 7.2.1) and identification (Sec.7.2.2) tasks.

7.2.1 Object Categorization

In Sec. 6.3 we saw that models trained on few frames achieve identical accuracy to those trained on many more. These results suggest that the corresponding networks are embedded with a robust data representation. In the following, we quantify to what extent this representation is invariant to the specific visual transformations represented in ICWT.

To this end, we repeat the experiment described in Sec. 6.3, by individually considering the different transformations instead of mixing them all. We consider the same 15-class categorization problem, where we use 7 object instances per category for training, 2 for validation and 1 for testing (repeating for 10 trials). However, here we do not mix examples from all the five available sequences (*2D Rotation, 3D Rotation, Scale, Background* and *Mix*). Instead we perform training (and validation) using only an individual transformation, and then

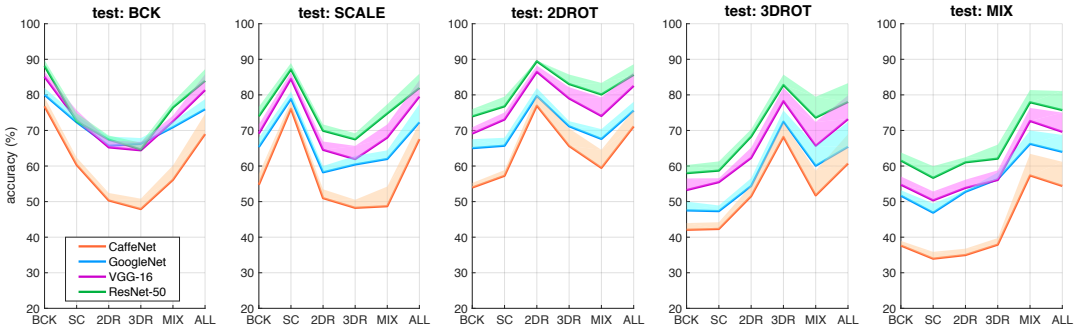

Figure 7.1: **Generalization performance across visual transformations:** RLSC models are trained on one of the 5 transformations present in ICWT (specified in the horizontal axis), and then tested on all transformations. Accuracy is reported separately in each plot for each tested transformation. Shaded areas represent the improvement achieved by including all frames of a sequence instead of subsampling ~ 20 of them.

test the model on the others. As in Sec. 6.3, we limit our analysis to only one of the two available days in ICWT. Indeed, in this setting we aim to study the effect of isolated viewpoint transformations, without considering other nuisances due, for example, to changes in the light and setting conditions.

We consider a "large-scale" and a "small-scale" training regimes, including, respectively, all images from a sequence, or subsampling images of a factor of 7 (as we did in Fig. 6.3). The size of the training set is therefore equal to $\sim 150 * 7 = 1050$ images per category in the first regime and 150 images per category in the subsampled regime. The aim is to verify what we observed in Sec. 6.3, that, while including more instances in the training set was critical, including more frames seemed to be not.

Fig. 7.1 and Fig. 7.2 report the generalization capabilities of models trained on a single transformation and tested on a different one. As a reference, we considered also a 6^{th} training set (*All*), analogous to those used in the experiments in Sec. 6.3 and obtained by randomly sampling points from all other 5 training sets, keeping similar size. We report the accuracy of the subsampled training sets with a bold line, and the improvement achieved by considering all frames from the sequences with a shaded area of the same color.

Specifically, in Fig. 7.1 we report the accuracy achieved by applying of RLSC on top of the representations extracted from off-the-shelf CNNs. This is done to evaluate the invariance of the representations learned on ImageNet. Instead, in Fig. 7.2 we report the accuracy achieved by fine-tuning *CaffeNet* and *GoogLeNet* according to the two *adaptive* and *conservative* strategies (compared with the performance achieved by training RLSC on top of the off-the-

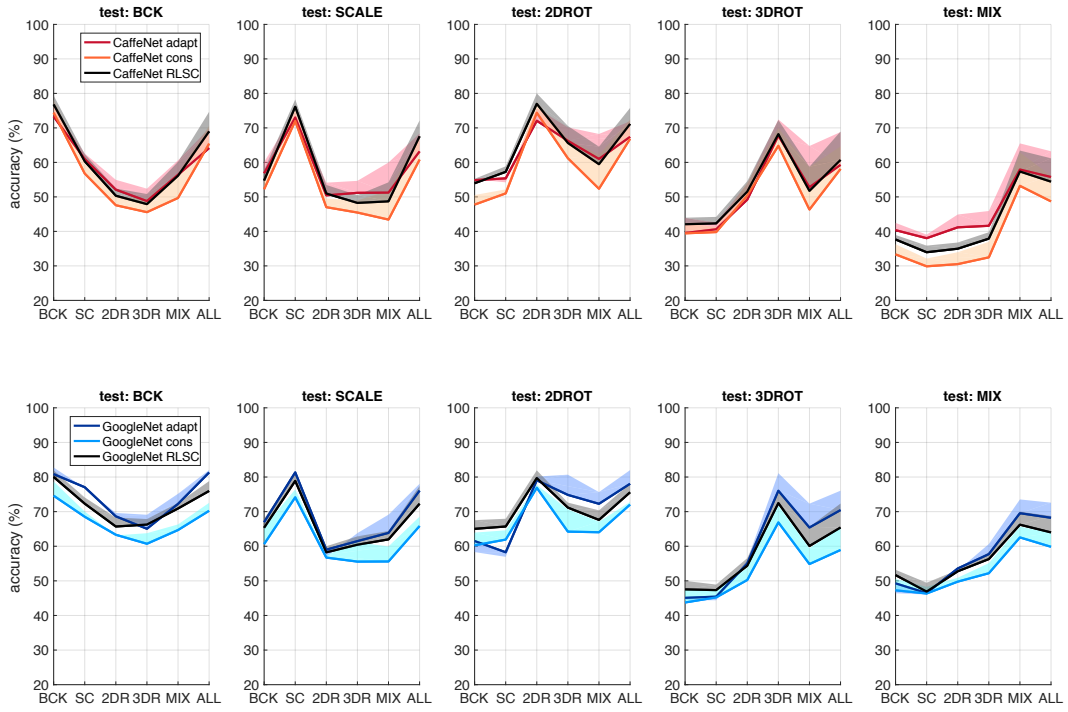

Figure 7.2: **Generalization performance across visual transformations:** *CaffeNet* and *GoogLeNet* are fine-tuned according to the *adaptive* and *conservative* strategies on one of the 5 transformations present in ICWT (specified in the horizontal axis), and then tested on all transformations. Accuracy is reported separately in each plot for each tested transformation. Shaded areas represent the improvement achieved by including all frames of a sequence instead of subsampling ~ 20 of them. The accuracy of applying RLSC on top of off-the-shelf representations, shown in Fig. 7.1, is also reported here (Black) for comparison with fine-tuned models.

shelf representation). This is done to evaluate how the across-transformation generalization performance of CNNs can be affected by fine-tuning on individual transformations in ICWT.

We first confirm that, for all networks and methods, increasing example frames for specific transformations does not improve generalization performance, neither on the experienced transformation itself, nor on others. Exceptions are the cases when training on the *Mix* or *All* sets, where, in fact, frames contain random and varied transformations and adding examples helps generalizing to the individual transformations. We hence confirm our previous observations from Sec. 6.3 and deduce that subsampling frames, to include more transformations, does not degrade, but rather helps, performance.

Overall, the models exhibit invariance only to transformations that have been included

– even with a small number of examples – in the training set. Indeed, in all cases the best performance is obtained when training and testing are performed on the same transformation and when all transformations are included in the training set (*All*). Remarkable performance drops are observed when trying to generalize to transformations not observed during training. This effect is expected and more remarkable for those cases where the model is trained, e.g., only on a single face of the object (that is, *Bkg*, *2D Rot* or *Scale* sequences) and, of course, cannot generalize to other viewpoints (*3D Rot* or *Mix* sequences). However, it is worth noting that quite big gaps are observed also between in-plane transformations.

Regarding Fig. 7.2 and fine-tuning, we notice that the *adaptive* strategy provides better performance than the *conservative* one. This is interesting and confirms, in this setting, what we observed in Sec. 6.3, suggesting that adapting the convolution layers to the target domain can generally be better (with the only exception of cases when very few example objects per category are available – see Fig. 6.3 – where it may be preferable to rely more on off-the-shelf models). In particular, fine-tuning adaptively on a sufficient number of frames of varied transformations (i.e., on *Mix* and *All* sequences, shaded areas) is the strategy that provides best generalization performance across transformations. Indeed, this experiment confirms an analogous test that we performed in [Pasquale et al., 2016a] (reported there in Fig. 3) and expands the observations that we made there (using *CaffeNet* and in-plane transformations), to 3D viewpoint changes and also the architecture of *GoogLeNet*. In Sec. 7.2.2 we will have a further confirmation of the benefit of adaptive fine-tuning when looking at the results for the object identification task.

We conclude with some "secondary observations" on the presented results:

- It is useful to notice that training on *Mix* achieves substantially good performance (not as good as *All*, but remarkable) when tested on every specific transformation. This suggests that showing an object to the robot in a natural way, with all transformations appearing in random combinations, instead of systematically collecting sequences comprising individual transformations, is a feasible approach to obtain predictors invariant to these transformations.
- Rather different models learned on ImageNet exhibit the same invariance pattern. This further confirms what we observed from this experiment, i.e., that, while these architectures do not have specific hard-coded invariances (apart from translation invariance because of convolutions), they learn to be invariant to the experienced visual nuisances.
- Regarding Fig. 7.1, we can see that the ILSVRC trend (*CaffeNet* $<$ *GoogLeNet* $<$ *ResNet-50*) is confirmed in all specific combinations of train/test transformations, and moreover, over the years the models exhibit increased invariance across transformations (i.e., the improvement from one architecture to another is more enhanced where testing

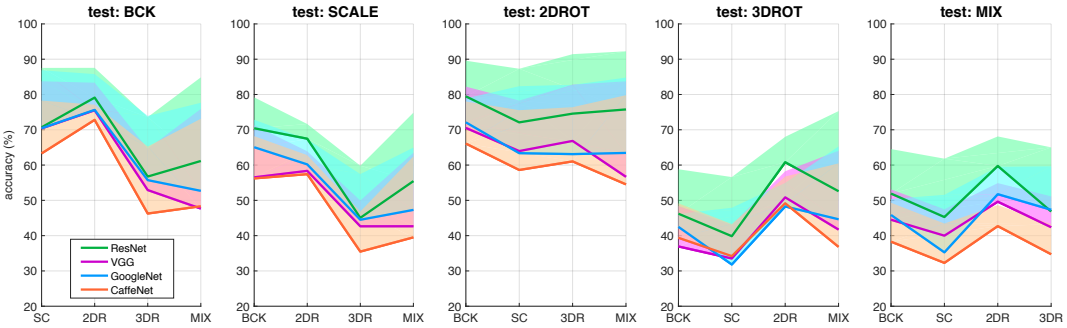

Figure 7.3: **Generalization performance across visual transformations (Identification):** RLSC models are trained on one of the 5 transformations present in ICWT (specified in the horizontal axis), and then tested on the other transformations. Accuracy is reported separately in each plot for each tested transformation. Shaded areas represent the improvement achieved by including all frames of a sequence instead of subsampling ~ 10 of them.

on an unseen transformation). This suggests that more recent architectures learned from ImageNet to be gradually more invariant; on the other hand, there is still a gap to be filled in some way.

- In Fig. 7.2 we observe that it is preferable to reuse the fully-connected layers of the models learned on ImageNet (as it is done by RLSC), rather then learning them from scratch (as it is done by the *conservative* fine-tuning). This result will be further confirmed on object identification tasks, suggesting that, indeed, these layers retain a certain degree of invariance that is useful to reuse.

7.2.2 Object Identification

In Sec. 6.4 we observed that, differently from object categorization, the accuracy of the models for object identification was greatly improved by providing more example images of the objects. To better understand this fact, analogously to the previous Section, here we repeat the same experiment of Sec. 6.4, but splitting the training and test sets "per transformation". Hence, we repeat the 50-class object identification task of Sec. 6.4, restricting the training set to contain images of objects undergoing a single transformation. The models are then tested on the remaining transformations, to assess the ability of the learned system to generalize to unseen viewpoints. Accordingly, we consider only one day among the two available in the dataset, to isolate viewpoint changes from other nuisances.

In Fig. 7.3 and Fig. 7.4, we report the accuracy of models trained on a single transformation and tested on all others. As for categorization, we simulate a "small-scale" and a "large-scale"

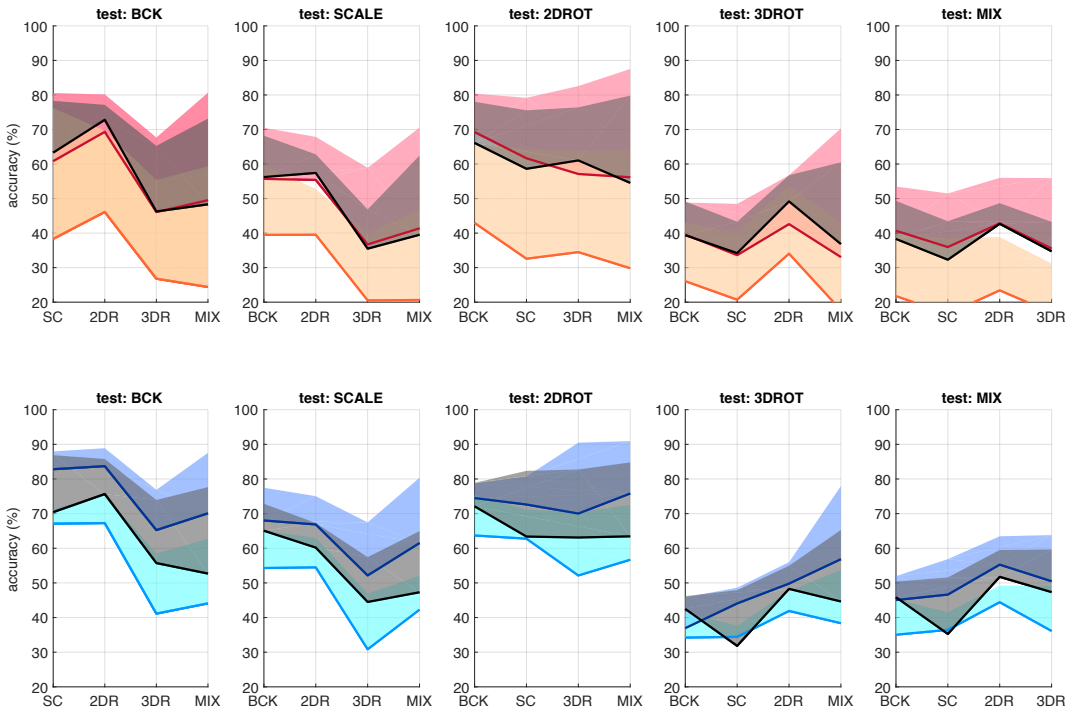

Figure 7.4: **Generalization performance across different visual transformations (Identification):** *CaffeNet* and *GoogLeNet* are fine-tuned according to the *adaptive* and *conservative* strategies on one of the 5 transformations present in ICWT (specified in the horizontal axis), and then tested on the other transformations. Accuracy is reported separately in each plot for each tested transformation. Shaded areas represent the improvement achieved by including all frames of a sequence instead of subsampling ~ 10 of them. The accuracy of applying RLSC on top of off-the-shelf representations, shown in Fig. 7.1, is also reported here (Black) for comparison with fine-tuned models.

regime, by training respectively on 10 (continuous line) or 150 (shaded area) frames per object. Note that, differently from categorization, here we do not test on the transformation used for training, since in this case it coincides with the training set. Accordingly, the *All* training set is not present in this case.

As in the previous Section, we first report, in Fig. 7.3, the results achieved by training RLSC on features extracted from the four models trained on ImageNet, in order to evaluate the viewpoint invariance of off-the-shelf representations. Then, in Fig. 7.4, we restrict to *CaffeNet* and *GoogLeNet* and compare the RLSC accuracy with the performance achieved by fine-tuning the architectures according to the *adaptive* and *conservative* strategies, to evaluate how fine-tuning on a transformation affects the across-transformation generalization performance.

It can be first noticed that the remarkable improvement observed in Sec. 6.4 when adding more frames is confirmed and clarified in this setting: not only, as expected, adding more frames containing mixed and varied transformations (that is, from the *Mix* sequence) helps to generalize to the individual transformations, but also adding frames from a single visual transformation always improves performance on the others. Indeed, if we restrict to using few frames and consider RLSC on top of off-the-shelf representations (Fig. 7.3, bold line), quite a low performance is reported across visual transformations, with large gaps from one tested transformation to another. This suggests that the off-the-shelf representations are definitely not invariant to the transformations observed in ICWT.

However, if we proceed to observe Fig. 7.4, we can see that, by adaptively fine-tuning the networks on enough frames, we can remarkably improve the across-transformation generalization performance of both *CaffeNet* and *GoogLeNet* (the shaded areas achieve highest accuracy in all cases). This suggests that adapting the internal representation of the CNNs is very helpful, not only because we are in fact changing the task, but possibly also because, provided with enough example frames, these networks learn viewpoint invariances from the sequences in ICWT. We recall that we already observed this effect (that nevertheless was much less evident) in the categorization experiment of previous Section.

Unfortunately, in real world situations, where a robot typically has to constantly learn to identify new objects on the fly, from a few images, fine-tuning networks on high numbers of example images, as are available in the large-scale regime, is not very suitable. To this end, in the following we will investigate whether it is possible to exploit this capability of deep networks of learning viewpoint invariances, to boost the object identification performance, while keeping the number of example frames needed low.

Before proceeding, we list again some additional observations:

- From Fig. 7.4 it can be noticed that overall the *conservative* fine-tuning provides remarkably worse performance, especially for *CaffeNet*. In the small-scale regime, this is simply due to the fact that that learning from scratch the fully-connected layers on the subsampled sequences (that, in this case, comprise only $\sim 10 * 50 = 500$ training examples), provides poor results. However, it is interesting to note how, also when the full sequences are provided, it is still preferable to start from, or even just rely on, the weights learned on ImageNet (as it is done, respectively by the *adaptive* fine-tuning and RLSC). This confirms what we observed in the previous Section, that these layers probably retain useful invariance properties also for identification tasks. Indeed, this is confirmed also by the literature, as we saw in Sec. 4.2.2.
- In line with previous ideas, from Fig. 7.3 we notice how some observations made for categorization in fact hold also for object identification: we can still observe that, even if

it is less evident, the different CNN models learned on ImageNet exhibit a very similar invariance pattern, with modern networks outperforming previous ones and ResNet-50 achieving the highest accuracy by a large margin.

7.3 Improving Viewpoint Invariance of Off-the-shelf CNNs

The results reported in Fig. 7.1 and 7.3 seem to indicate that the internal representations extracted by off-the-shelf CNNs are not invariant to the transformations affecting objects' apperarance in ICWT. At the same time, in Fig. 7.2 and in particular in Fig. 7.4 we observed how, by fine-tuning the models on sequences containing many examples of multiple, varied transformations, it seems possible to remarkably improve their robustness to such transformations. In this Section, we investigate whether the observed increase in the across-transformation generalization performance of tuned models is actually caused by the fact that their internal representation has become more invariant to such transformations.

To this end, we refer again to the same object categorization and identification experiments of Sec. 7.2.1 and 7.2.2, and compare the performance achieved by RLSC trained on representations extracted from (i) off-the-shelf networks and (ii) networks that have been previously fine-tuned on image sequences of ICWT. Of course, the training set used for fine-tuning in this latter case is a separate subset of ICWT, not overlapping with the categories involved in the categorization/identification tasks. Since we want to evaluate the invariance of the underlying representations, we train the RLSC in the "small-scale" regime, that is, on just 10 frames per object in the identification task (and 20 in categorization). Indeed, investigating this setting is particularly appealing for robotics because, if we could rely on invariant representations, we could perform object recognition from very few example images.

For this study, we restricted the analysis to *CaffeNet* and *GoogLeNet*, the two architectures for which we considered fine-tuning throughout this work. Specifically, we fine-tuned them with the "adaptive" strategy on a subset of ICWT: namely, the remaining part of the dataset comprising the 5 categories used neither in the object identification nor categorization experiment (*oven glove*. *squeezer*, *sprayer*, *body lotion* and *soda bottle*). In order to better assess the usefulness of this fine-tuning stage, we tried to fine-tune the networks according to four different possible tasks (described in the following). Afterwards, we proceeded in considering the object categorization and identification experiments of Sec. 7.2.1 and 7.2.2 (small-scale regimes), and compared the performance achieved by training RLSC on top of the representations extracted from (i) off-the-shelf models with (ii) each of the fine-tuned models. For RLSC, we adopted the same protocol used in previous Sections.

The four tasks that we considered to fine-tune the networks are:

- **iCWT – Identification Task (iCWT id)**. We considered a 50-class identification task involving all objects in the considered subset of iCWT. We used all 5 sequences per object, one day and one camera, keeping a random 20% for validation.
- **iCWT – Categorization Task (iCWT cat)**. The same dataset as (iCWT id) but fine-tuning was performed on a 5-class object categorization task where training and validation sets were obtained following the same protocol as in Sec. 6.3.
- **iCWT + ImageNet (iCWT + ImNet)**. This dataset is the union of the iCWT subset used above with the corresponding 5 synsets in ImageNet. This dataset is conceived to test the possibility to have the CNN jointly learning the semantic variability contained in ImageNet examples and the viewpoint variability contained in iCWT examples of categories. Note that part of the 5 categories in the subset do not appear in ILSVRC but are in the larger ImageNet dataset (see Fig. 5.8 for the synset codes). All images in the synsets were used. Fine-tuning was performed on a 5-class categorization task (keeping the same protocol of Sec. 6.3).
- **ImageNet (refresh ImNet)**. This dataset contains the ImageNet synsets corresponding to the object categories on which the CNN will be, later, trained and tested on: namely, the 5 categories for the identification task of Sec. 7.2.2 (*book*, *flower*, *glass*, *hairbrush* and *hairclip*) and the 15 categories (listed in Appendix 6.C) for the categorization task of Sec. 7.2.1. Fine-tuning was performed on categorization tasks of course (keeping a random 20% for validation). This dataset is conceived to test whether it is beneficial to "refresh" the network on the categories about the task at hand (that may be possibly available on-line), even if they are different from the images/tasks that the robot will observe.

Fig. 7.5 compares the accuracies achieved by the four fine-tuned representations on the categorization task, and we report as a reference also the off-the-shelf representation (that is the same as in Fig. 7.1). Fig. 7.6 does the same but for the identification task, comparing the four fine-tuned representations with the off-the-shelf one (reported from in Fig. 7.3).

Before proceeding, we note that for this experiment we performed a similar analysis to the one reported in Appendix 6.B.1, in order to assess the best output layer to be used to extract visual representations from the architectures, observing that, for *CaffeNet* models fine-tuned on iCWT, *fc7* features were providing better performance than *fc6* in this case. This can be explained considering that "more specialized" features can be extracted from models that have been adapted to our setting, while "more general" features of off-the-shelf models provide better generalization performance on iCWT. Therefore, in order to compare the different approaches in their "best" conditions, we report *fc7* features for models tuned on iCWT, and *fc6* for the others.

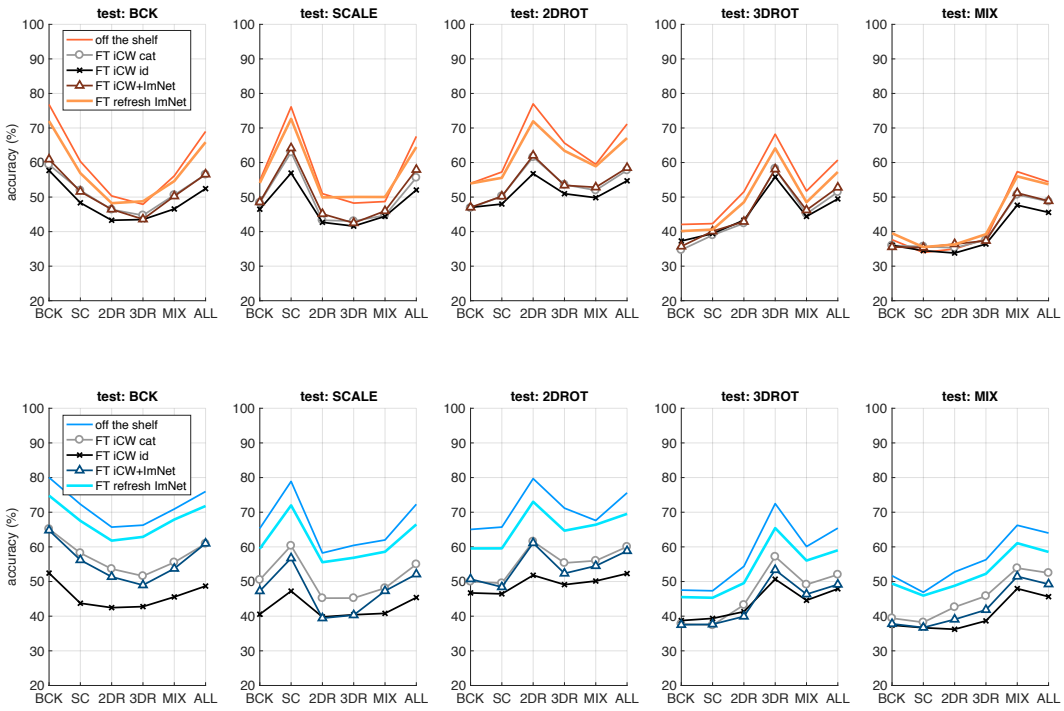

Figure 7.5: **Generalization performance across different visual transformations:** Same experiment as in Fig. 7.1, but using different visual representations, provided by fine-tuning *CaffeNet* (top, Orange) or *GoogLeNet* (bottom, Blue) on different datasets including ICWT and/or ImageNet images (see Sec. 7.3).

We first observe in Fig. 7.5 that fine-tuning on any of the datasets described above worsen the performance on the categorization task. We hypothesize this can be due to two possible reasons: 1) networks trained on the ILSVRC are already highly optimized for categorization, and there is no gain in adapting their representation, neither on the target ICWT domain, nor by "refreshing" on a subset of categories – as, e.g., in the (ImNet) case; 2) when fine-tuning on ICWT, the negative effect of the limited semantic variability in the dataset, with respect to ImageNet, may "overcome" the potential benefits of increasing the representation invariance to viewpoint transformations. Indeed, this result motivates some of the proposed directions of future investigation that we discuss in Chapter 10.

On the contrary, Figure 7.6 reports a remarkable improvement of the object identification performance, provided by representations extracted from models fine-tuned on ICWT. Fine-tuning on the identification task (iCWT id) is particularly advantageous, by far surpassing other models. Comparing these results with those in Fig. 7.3, we see that (iCWT id) trained on

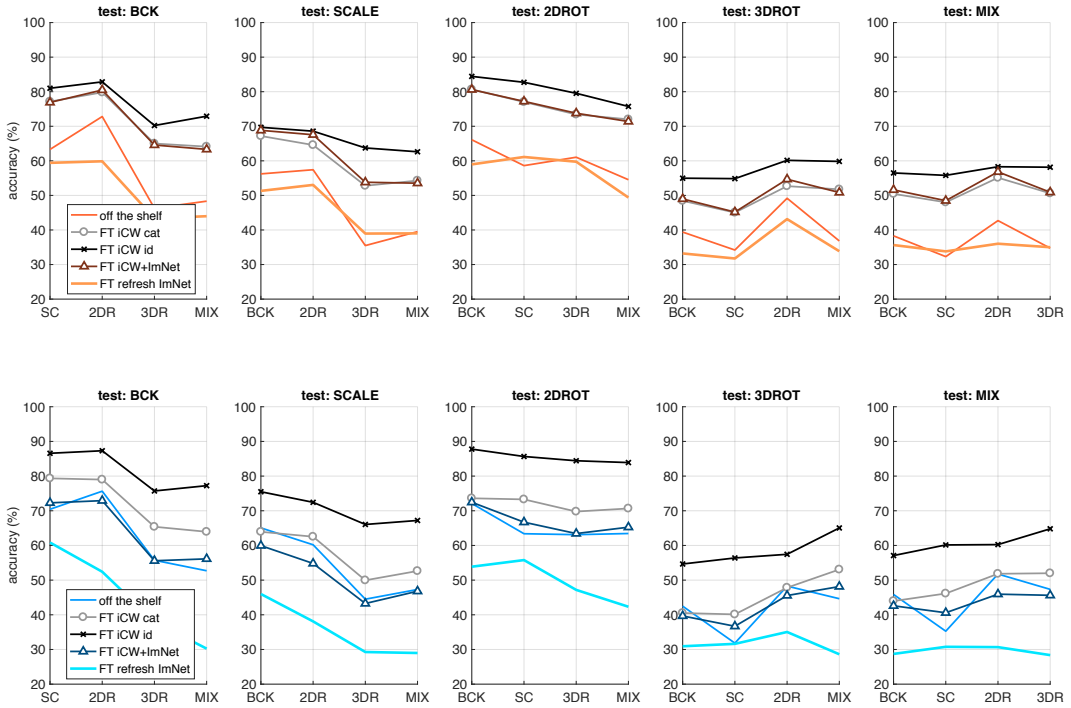

Figure 7.6: **Generalization performance across different visual transformations (Identification):** Same experiment as in Fig. 7.3, but using different visual representations, provided by fine-tuning *CaffeNet* (top, Orange) or *GoogLeNet* (bottom, Blue) on different datasets including ICWT and/or ImageNet images (see Sec. 7.3).

10 example images per object performs even better than off-the-shelf representations trained on 150 images per object. Moreover, the performance of (iCWT id) are in general more stable across different transformations, suggesting that the preliminary fine-tuning could have indeed allowed the CNNs to become more invariant to transformations in ICWT.

Interestingly, the models tuned on ICWT for categorization (iCWT cat) also perform better than the off-the-shelf baseline, further suggesting that the observed improvement of (iCWT id) may not be due only to the adaptation of the task, but indeed to better viewpoint invariance. Finally, note that we compared the off-the-shelf *fc6* features of *CaffeNet* also with the same *fc6* features of models fine-tuned on ICWT, confirming also in this case the higher performance of the latter ones.

From these results, we can conclude that fine-tuning network models on sequences of ICWT can indeed provide robust features to viewpoint changes, that can be used to solve object identification tasks from very few example images. Note that this experiment expands the results reported in [Pasquale et al., 2016a] to a different architecture besides *CaffeNet*, namely

GoogLeNet, considering also 3D viewpoint transformations in addition to in-plane rotations and scaling. Moreover, by considering also the problem of categorization, we highlighted the limitations of the presented approach to improve performance on this latter task. Indeed, as we observed in Chapter 6, the intra-class semantic variability involved in this case heavily affects the recognition capabilities of models. Building on these observations, in Chapter 10 we will propose possible directions of research to address these critical challenges in the robotic setting. In the following Section, we compare the proposed approach with a recent work [Held et al., 2016], addressing a similar problem on the Washington RGB-D benchmark.

7.4 One-shot Object Identification on Washington RGB-D

In previous Sections, to investigate the invariance of CNNs, we devised object recognition tasks where the models were trained on few example images of objects (typically undergoing only a single visual transformation), and were then required to generalize to unseen viewpoints. We highlighted how this setting is interesting for robotics, since it well represents the typical recognition tasks that a robot is asked to perform in real world applications. Throughout our analysis, we found that fine-tuning models on sequences of ICWT, representing objects undergoing different viewpoint transformations, remarkably improves the robustness of their internal representations to such viewpoint changes, with respect to off-the-shelf models. Building on these observations, we were able to encode images into robust features, and perform challenging object identification tasks by training statistical tools like RLSC on very few examples.

While in the previous Section we tested the proposed pipeline on the ICWT dataset, in this Section we validate it on the setting considered in a recent work [Held et al., 2016], where the authors address the problem of one-shot object identification, defining a reference task on the Washington RGB-D (WRGB-D) benchmark [Lai et al., 2011] and addressing it with a similar approach. In the following, we briefly review for convenience their setup, which we reproduced, in order to compare with our method.

Robust Single-View Instance Recognition [Held et al., 2016]

The one-shot object identification task on WRGB-D that is considered by Held et al. is similar to the standard 300-class identification task usually adopted in the literature (that we described in Sec. 6.D.2), except that, in this case, the training set is reduced to be only a single image per object, randomly sampled from the sequence recorded at 30°.

The authors address this task by fine-tuning deep CNNs. Specifically, they start from the same *CaffeNet* off-the-shelf model trained on ILSVRC. Then, similarly to our analysis, they

compare two solutions: the first solution (the baseline) fine-tunes *CaffeNet* on the one-shot identification task; the second solution, first, fine-tunes *CaffeNet* on a so-called "multi-view" dataset, comprising a certain number of object instances represented under different viewpoints; then, they start from the tuned model, and further fine-tune it on the one-shot identification task. Their idea is similar to our approach, in that they observe that the network model that has been previously fine-tuned on a set of sequences representing objects under multiple viewpoints achieves improved performance on a one-shot identification task requiring recognition systems to be viewpoint invariant.

In order to demonstrate that the observed improvement is caused by better invariance and not, e.g., by other adaptation processes, similarly to what we did, as multi-view dataset they use a different set of objects from the ones contained in WRGB-D. Specifically, they use the BigBird dataset [Singh et al., 2014], a benchmark acquired in a similar turntable setting, whose images are not too different from the ones in WRGB-D. They care to remove one object instance that is represented also in WRGB-D, and subsample 100 frames per object, evenly including all viewpoints available in the dataset, ending up with a dataset of 124 objects in 12400 frames.

The fine-tuning protocol that they adopt in their "multi-view pre-training" on BigBird is the same that they use afterwards on the one-shot identification task, except that in this latter training all learning rates are reduced by a factor of 10. Specifically, they (i) fix convolution layers, (ii) fine-tune $fc6$ and $fc7$ with initial learning rate of 10^{-3} (10^{-4} in the second fine-tuning), by initializing the weights with the off-the-shelf model, and (iii) learn $fc8$ from scratch with initial learning rate of 10^{-2} (10^{-3}). This protocol is similar to our *conservative* strategy, but they use the off-the-shelf initialization for $fc6$ and $fc7$, in line with our observations made in the previous Section, that it is beneficial to exploit an ImageNet warm restart also for fully-connected layers. In addition, they also try fine-tuning convolution layers, similarly to our *adaptive* strategy.

Before addressing the one-shot task, in [Held et al., 2016] they first report the performance that they achieve by fine-tuning *CaffeNet* with the proposed protocol on the standard WRGB-D identification task (without any multi-view preliminary fine-tuning). Indeed, in Sec. 6.D.2, we already evaluated our methods on this task, surpassing their results (Tab. 6.D.4). In the following, we will compare the approach presented in previous Section with their pipeline for one-shot identification on WRGB-D described above.

7.4.1 Comparison

To compare with [Held et al., 2016], we first evaluate differences between the proposed methods, by simply replacing our pipeline in their setting, using the same BigBird subset to

Table 7.1: **One-shot object identification results on Washington RGB-D, comparison with state-of-the-art:** our approach, based on training RLSC on features extracted by a *CaffeNet* model, is compared with results in [Held et al., 2016], where the network is fine-tuned on the task. See in the text for a detailed description of the different methods.

Method	**Pre Fine-tuning (FT)**	**One-shot Identification**	**Accuracy** (%)	**Gain** (%) **of Pre FT**
[Bo et al., 2013]	-	HMP	42.3	
[Held et al., 2016]	-	FT	59.2	
	BB aug	FT	63.9	+4.7
	BB aug	FT (all layers)	65.1	+5.9
Ours	-	RLSC	59.5	
	BB	RLSC	66.2	+6.7
	iCWT	RLSC	66.8	+7.3

preliminary fine-tune the networks. Then, we evaluate also replacing BigBird with the iCWT subset that we used in the previous Section.

In both cases, we apply the pipeline that reported best results in the object identification tasks on iCWT: we first fine-tune *CaffeNet* according to the "adaptive" strategy on the dataset representing the objects under multiple viewpoints; then, we address the one-shot identification task by training RLSC on the representations extracted from the fine-tuned models. As a baseline, we train RLSC on off-the-shelf representations.

When training RLSC on 1 image per object, we set the model hyper-parameters according to preliminary tests. To account for statistical oscillations, we repeated the experiment for 20 trials, every time randomly sampling training images, observing for all approaches standard deviations under 1% of recognition accuracy.

Table 7.1 reports results. The first row reports, as a general reference of "shallow" methods (i.e., not using pre-trained deep CNNs), the performance achieved by [Held et al., 2016] applying the method proposed in [Bo et al., 2013]. Then, the following three rows report results achieved by the authors, respectively with their baseline – second row – and with the preliminary fine-tuning on BigBird (BB aug) – third and fourth rows. As we anticipated, they try either keeping the convolution layers fixed (FT) or fine-tuning them (FT all layers, fourth row), finding that this latter strategy provides better improvement over the baseline (+5.9% instead of +4.7%). It is worth noting that they augment the BigBird subset by synthetically placing the objects over random backgrounds – for this reason we indicated it as (BB aug), whereas we do not.

The following rows report our results: RLSC trained on off-the-shelf features (the baseline), or on features extracted from models fine-tuned on BigBird (BB) or the iCWT subset used

in Sec. 7.3 (precisely, the (ICWT id)). It can be first noticed that we achieve comparable or superior performance both with the baseline and with the preliminary fine-tuning. It is particularly interesting that our model fine-tuned on the ICWT subset is indeed able to provide the same performance gain than the model fine-tuned on BigBird. Indeed, while in all the other cases the dataset used for the preliminary fine-tuning either was the same (as in the previous Section), or was very similar (as in this experiment when fine-tuning on BigBird) to the final dataset used for the object identification task, in this case the two are completely different. Differences between the two domains can be clearly appreciated just looking at the example images of ICWT provided in Sec. 5.5 and comparing with images in WRGB-D. Therefore, while in the other cases one could have still argued that the improvement in performance on the considered identification tasks could have been due to, at leat partially, the fact that the model had been already adapted to the task domain, in this case this is not possible. Hence, the improvement must be due to better robustness of the features to viewpoint changes.

7.4.2 Discussion

To conclude, we confirm a remarkable performance improvement brought by our approach also on the one-shot identification task on WRGB-D. With this test, we have extended the results of [Held et al., 2016] to more challenging settings, where the dataset for preliminary fine-tuning is remarkably different from the final domain of the object identification task.

Moreover, we studied the efficacy of this solution with respect to different specific transformations represented both in the dataset used for fine-tuning and in the final recognition task, extending the evaluation to more, wider transformations beyond 3D viewpoint changes. Indeed, in [Pasquale et al., 2016a] (precisely, in Tab. II) we observe how by including more transformations (namely, light changes between one day and the other) in the dataset used for fine-tuning, provides improved invariance to the experienced transformations. Building on those results, here we evaluated how features fine-tuned on multiple transformations (all available ones in ICWT) can generalize to the individual ones.

On the choice of combining deep CNNs and Kernel methods. In [Pasquale et al., 2016a] we evaluate also the alternative technique, here not reported, of using the fine-tuned model as warm restart to further fine-tune the network on the final object identification task (like in [Held et al., 2016]), rather then extracting its representation to train RLSC on the final task. We show there (Fig. 5) that starting from a previously fine-tuned model provides as well better performance than directly fine-tuning the off-the-shelf model, confirming therefore [Held et al., 2016]. Nevertheless, we opted for combining fine-tuning with standard classifiers like RLSC for at least three reasons, explained in the following.

First, it is not trivial to properly fine-tune deep CNNs when learning from few examples. To this regard, in [Pasquale et al., 2016a] we show how fine-tuning achieves the RLSC performance only when more examples are provided. In fact, [Held et al., 2016] achieve remarkable performance just by using a training set of 300 examples, by setting the fine-tuning hyper-parameters (which layers, learning rates, decay policies, number of iterations) to specific values (and we managed to reproduce their results). However, we also showed (see, e.g. Sec. 6.B.2) that these hyper-parameters generally depend on the size and kind of recognition task, and performing model selection when training deep CNNs is still an open issue, usually addressed by very computational intensive approaches like grid or random search (see Sec. 4.1.2). On the other hand, classical statistical tools like RLSC or SVMs depend on few hyper-parameters that can be set more easily.

Second, fine-tuning is not yet a practical method to be used in situations where the robot has to learn on the fly, and these one-shot tasks, in our intention, are precisely conceived to simulate such setting. In these scenarios the training has to be very fast, possibly allowing the incremental inclusion of new objects. As we will see in Chapter 8, while this problem has been investigated for classical statistical tools, this is still a largely unexplored field regarding deep networks.

Third, and actually our first motivation in Sec. 7.3, we opted for using statistical tools on top of features extracted from deep networks because we were interested in evaluating invariance properties of such features, and RLSC is a stable predictor to measure such properties through recognition accuracy. Indeed, considering the approach of [Held et al., 2016], since their second fine-tuning actually changes the features learned in the preliminary multi-view fine-tuning, it is not evident that the improvement can actually be attributed to the previous stage, rather than to the final fine-tuning step.

7.5 Conclusion

In this Chapter, we deepened the analysis of Chapter 6 and studied the viewpoint invariance properties of the internal representations of modern deep Convolutional Neural Networks (CNNs). We exploited the structure of the novel iCubWorld Transformations dataset and evaluated the performance achieved by networks trained and tested on specific visual transformations contained in the image sequences comprising the dataset. This allowed us to assess the robustness of models in real world conditions, differentiating from previous work that mainly adopted synthetic imagery to perform similar analyses.

We first assessed the invariance of off-the-shelf models and then how this is affected by fine-tuning the models on sequences representing specific or multiple transformations. For

our evaluation, we extracted deep representations from the internal layers of CNN models and trained Regularized Leas Squares Classifiers (RLSC) on top. Specifically, we considered the scenario where the human supervisor provides only few glimpses of the objects of interest to the robot, and tested the generalization performance of classifiers on multiple sequences containing different visual transformations.

Our analysis shows that fine-tuning the internal layers of a network adapts the invariance of the representation to the nuisances contained in the new dataset. By leveraging this effect, we fine-tuned two CNN models (*CaffeNet* and *GoogLeNet*) over multiple visual transformations to obtain a robust representation for object identification. We observed that using the fine-tuned models as feature extractors and feeding such adapted representations to RLSC provides an effective pipeline to learn to identify a wide range of objects from minimal amounts of data.

We conclude that adopting conventional classifiers based on Kernel methods to leverage a deep visual representation, previously trained on large data collections to be robust to specific nuisances, is an effective approach to achieve reliable object identification from few example frames in online robotic scenarios. In line with theses findings, we believe that the off-the-shelf representations trained on ImageNet are already highly optimized for object categorization and indeed we did not manage to improve their properties by fine-tuning in our setting.

Chapter 8

Incremental Learning of New Objects with Fixed Update Time

While relying on good representations is fundamental in order to allow robots to reliably learn objects on the fly, it is not the only critical component of a robotic recognition system. Indeed, in the considered scenario where a robot encounters new objects as it explores the environment, possibly guided by a teacher, another important part is allowing the knowledge about objects to be built and refined incrementally. The recognition system should update the internal model of known objects, as new example images are gathered. Moreover, since it is not possible to hard-code into the system beforehand all possible classes that the robot will be required to recognize, the system should also allow the inclusion of new classes without the need of re-training the known ones from scratch.

In this Chapter, we move therefore to consider object recognition in the context of lifelong learning. We propose an incremental variant of the Regularized Least Square for Classification (RLSC) algorithm used in previous Chapters, and exploit its structure to seamlessly add new classes to the learned model. The content of this Chapter has been previously published in [Camoriano et al., 2017].

8.1 Introduction

Most of the deep learning methods that recently achieved stunning performances have been developed for off-line (or "batch") settings, where the entire training set is available beforehand. The problem of updating a learned model on-line has been addressed in the literature [French, 1999, Sayed, 2008, Duchi et al., 2011, Goodfellow et al., 2013b], but most algorithms proposed in this context do not take into account challenges that are characteristic of realistic lifelong learning applications. Specifically, in on-line classification settings, a major challenge is to

cope with the situation in which a novel class is added to the model. Indeed, 1) most learning algorithms require the number of classes to be known beforehand and not grow indefinitely, and 2) the imbalance between the few examples of the new class (potentially just one) and the many examples of previously learned classes can lead to unexpected and undesired behaviors [Elkan, 2001]. More precisely, in this work we observe both theoretically and empirically that the new and under-represented class is likely to be ignored by the learning model, in favor of classes for which more training examples have already been observed, until a sufficient number of examples are provided also for such class.

Several methods have been proposed in the literature to deal with class imbalance in the batch setting by "rebalancing" the misclassification errors accordingly [Elkan, 2001, Steinwart and Christmann, 2008, He and Garcia, 2009]. However, as we point out in this work, rebalancing cannot be applied to the on-line setting without re-training the entire model from scratch every time a novel example is acquired. This would incur in computational learning times that increase at least linearly in the number of examples, which is clearly not feasible in scenarios where training data grows indefinitely.

In this work, we propose a novel method that learns incrementally both with respect to the number of examples and classes, and accounts for potential class unbalance. Our algorithm builds on a recursive version of Regularized Least Squares algorithm for Classification (RLSC) [Rifkin, 2002, Rifkin et al., 2003] to achieve fixed incremental learning times when adding new examples to the model, while efficiently dealing with imbalance between classes

We evaluate the approach on the MNIST [LeCun et al., 1998] machine learning benchmark and on two visual object recognition datasets for robotics, namely, the Washington RGB-D dataset [Lai et al., 2011] and the iCubWorld28 dataset. Empirical evidence shows that our approach achieves comparable or higher classification performance than its batch counterpart when classes are unbalanced, while being significantly faster.

The Chapter is organized as follows: Sec. 8.1.1 overviews related work on incremental learning and class imbalance. Sec. 8.2 discusses the impact of class imbalance in the considered learning setting of RLSC, presenting two approaches that have been adopted in the literature to deal with this problem. Sec. 8.3 reviews the recursive version of the RLSC algorithm. In Sec. 8.4 we build on previous Sec. 8.2 and 8.3 to derive the proposed method, which extends recursive RLSC to allow for the addition of new classes with fixed update time, while dealing with class imbalance. In Sec. 8.5 we report on the empirical evaluation of our method, concluding the paper in Sec. 8.6.

8.1.1 Related Work

Incremental Learning. The problem of learning from a continuous stream of data has been addressed in the literature from multiple perspectives. The simplest strategy is to re-train the system on the updated training set, whenever a new example is received [Xiao et al., 2014, Jain et al., 2014]. The model from the previous iteration can be used as an initialization to learn the new predictor, reducing training times. However, these approaches require to store all the training data and to retrain over all the points at each iteration. Hence, their computational complexity increases, at least linearly, with the number of examples.

Incremental approaches that do not require to keep previous data in memory can be divided in *stochastic* and *recursive* methods. Stochastic techniques assume training data to be randomly sampled from an unknown distribution and offer asymptotic convergence guarantees to the ideal predictor [Duchi et al., 2011]. However, it has been empirically observed that these methods do not perform well when seeing each training point only once, hence requiring to perform "multiple passes" over the data [Hardt et al., 2016, Lin et al., 2016]. This problem has been referred to as the "catastrophic effect of forgetting" [French, 1999], which occurs when training a stochastic model only on new examples while ignoring previous ones, and has recently attracted the attention of the deep learning literature [Srivastava et al., 2013, Goodfellow et al., 2013a].

Recursive techniques are based, as the name suggests, on a recursive formulation of batch learning algorithms. Such formulation typically allows to compute the current model in closed form (or with few operations independent from the number of examples) as a combination of the previous model and the new observed example [Sayed, 2008, Laskov et al., 2006]. As we discuss in more detail in Sec. 8.3, the algorithm proposed in this work is based on a recursive method.

Learning with an Increasing Number of Classes. Most classification algorithms have been developed for batch settings and therefore require the number of classes to be known a priori. This assumption is often broken in incremental settings, since new examples could belong to previously unknown classes. The problem of dealing with an increasing number of classes has been addressed in the contexts of transfer learning or *learning to learn* [Thrun, 1996]. These settings consider a scenario where T linear predictors have been learned to model T classes. Then, when a new class is observed, the associated predictor is learned with the requirement of being "close" to a linear combination of the previous ones [Tommasi et al., 2010, Tommasi et al., 2012, Kuzborskij et al., 2013]. Other approaches have been recently proposed where a class hierarchy is built incrementally as new classes are observed, allowing to create a taxonomy and exploit possible similarities among different classes [Xiao et al.,

2014, Sun and Fox, 2016]. However, all these methods are not incremental in the number of examples, and require to retrain the system every time a new point is received.

Class Imbalance. Problems related to class imbalance were previously studied in the literature [Elkan, 2001, He and Garcia, 2009, Steinwart and Christmann, 2008] and are addressed in Sec. 8.2. Methods to tackle this issue have been proposed, typically re-weighting the misclassification loss [Tommasi et al., 2010] to account for class imbalance. However, as we discuss in Sec. 8.4.2 for the case of the least squares loss, these methods cannot be implemented incrementally. This is problematic, since imbalance among multiple classes often arises in on-line settings, even if temporarily, for instance when examples of a new class are observed for the first time.

8.2 Dealing with Class Imbalance

We refer to Sec. 2.1 for the introduction and definition of the supervised learning problem considered in this Chapter. Here we maintain the same notation, therefore all quantities involved are also defined therein.

The Effect of Class Imbalance on the Optimal Bayes Classifier

For the sake of simplicity, we start by considering a binary classification problem, postponing the extension to multiclass classification to the end of the Section. In particular, we address the problem of finding the optimal Bayes classifier b^*, such that (see Eq. (2.6)-(2.7)):

$$b^* = \underset{f:\mathcal{X}\to\{-1,1\}}{\arg\min} \int_{\mathcal{X}\times\{-1,1\}} \mathbf{1}(y - f(x)) d\rho(x, y), \tag{8.1}$$

with

$$b^*(x) = \begin{cases} 1 & \text{if } \rho(1|x) > \rho(-1|x) \\ -1 & \text{otherwise} \end{cases} \tag{8.2}$$

for all $x \in \mathcal{X}$. This classification rule associates every $x \in \mathcal{X}$ to the class y with highest likelihood $\rho(y|x)$. This approach can lead to unexpected and undesired behaviors in situations where the two classes are not balanced. To see this, let us denote $\gamma = \rho(y = 1) = \int_{\mathcal{X}} d\rho(y = 1, x)$ and notice that, by applying the Bayes' theorem to Eq. (8.2), an example x is labeled $y = 1$ whenever

$$\rho(x|1) > \rho(x| - 1)\frac{(1 - \gamma)}{\gamma}. \tag{8.3}$$

Hence, when γ is close to one of its extremal values 0 or 1 (i.e. $\rho(y = 1) >> \rho(y = -1)$ or vice-versa), one class becomes clearly preferred with respect to the other and is almost always selected.

In Fig. 8.1 we report an example of the effect of unbalanced data by showing how the decision boundary (white dashed curve) of the optimal Bayes classifier from Eq. (8.2) varies as γ takes values from 0.5 (balanced case) to 0.9 (very unbalanced case). As it can be noticed, while the classes maintain the same shape, the decision boundary is remarkably affected by the value of γ.

In a robotic setting, this effect could be critically suboptimal for two reasons: 1) we would like the robot to recognize with high accuracy even objects that are less common to be seen; 2) in on-line incremental settings, whenever a novel object is observed for the first time, only few training examples are available (in the extreme case, just one) and we need a loss weighting fairly also underrepresented classes.

Rebalancing the Loss

Here we consider a general approach to "rebalance" the classification loss of the learning problem of Eq. (8.1), similar to the ones adopted in [Elkan, 2001, Steinwart and Christmann, 2008]. We begin by noticing that in the balanced setting, namely for $\gamma = 0.5$, the classification rule at Eq. (8.3) is equivalent to assign class 1 whenever $\rho(x|1) > \rho(x|-1)$ and vice-versa. Here, we want to slightly modify the misclassification loss in Eq. (8.1) to recover this same rule also in unbalanced settings. To do so, we propose to apply a weight $w(y) \in \mathbb{R}$ to the loss $\mathbf{1}(y - f(x))$, obtaining the problem

$$b^*_w = \underset{f:\mathcal{X}\to\{-1,1\}}{\arg\min} \int_{\mathcal{X}\times\{-1,1\}} w(y)\mathbf{1}(y - f(x))\, d\rho(x, y).$$

Analogously to the non-weighted case, the solution to this problem is

$$b^*_w(x) = \begin{cases} 1 & \text{if } \rho(1|x)w(1) > \rho(-1|x)w(-1) \\ -1 & \text{otherwise.} \end{cases} \tag{8.4}$$

In this work, we take the weights w to be $w(1) = \frac{1}{\gamma}$ and $w(-1) = \frac{1}{1-\gamma}$. Indeed, by applying again the Bayes' theorem to Eq. (8.4) and replacing the chosen values for the weights, we recover the original rule:

$$b^*_w(x) = \begin{cases} 1 & \text{if } \rho(x|1) > \rho(x|-1) \\ -1 & \text{otherwise} \end{cases} \tag{8.5}$$

which corresponds to the (unbalanced) optimal Bayes classifier in the case $\gamma = 0.5$, as desired.

Fig. 8.1 compares the unbalanced and rebalanced optimal Bayes classifiers for different values of γ. Notice that rebalancing leads to solutions that are invariant to the value of γ (compare the black decision boundary with the white one).

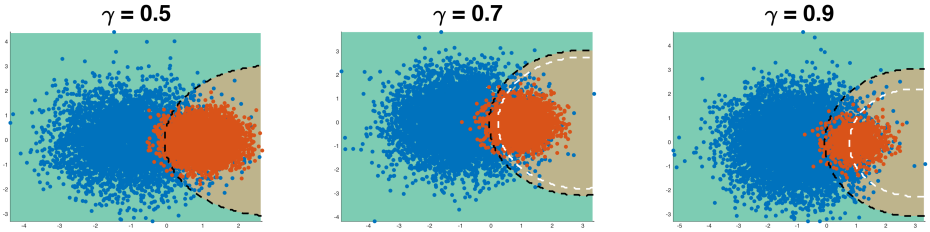

Figure 8.1: **Bayes decision boundaries for standard** (dashed white line) **and rebalanced** (dashed black line) **binary classification losses** for multiple values of $\gamma = \rho(y = 1)$ from 0.5 to 0.9. Data is sampled according to a Gaussian $\rho(x|y) \sim \mathcal{N}(\mu_y, \sigma_y)$ with $\mu_1 = (-1, 0)^\top$, $\mu_{-1} = (1, 0)^\top$, $\sigma_1 = 1$ and $\sigma_{-1} = 0.3$. The boundaries coincide when $\gamma = 0.5$ (balanced data), while they separate as γ increases.

Rebalancing the Least Squares Loss

So far we analysed the effect of class imbalance on the optimal Bayes classifier. However, in Sec. 2.1.2 we explained how, since in practice only a finite set $\{x_i, y_i\}_{i=1}^n$ of training examples is provided, the following *surrogate* problem is usually considered instead:

$$f^* = \arg\min_{f:\mathcal{X}\to\mathbb{R}} \int_{\mathcal{X}\times\{-1,1\}} (y - f(x))^2 d\rho(x, y), \tag{8.6}$$

where the 0-1 loss is replaced by the square loss. Indeed, we showed how the solution of this problem takes the form

$$f^*(x) = 2\rho(1|x) - 1 = \rho(1|x) - \rho(-1|x), \tag{8.7}$$

such that $sign(f^*(x))$ recovers the optimal classifier $b^*(x)$ and, at the same time, the minimization of the associated *regularized empirical risk*:

$$\hat{f} = \arg\min_{f:\mathcal{X}\to\mathbb{R}} \quad \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2 + R(f) \tag{8.8}$$

provides a convex optimization problem very well studied in the literature.

In the following, we show how the strategy of changing the weight of the classification loss presented in the previous Section in order to prevent the disrupting effects of class imbalance on the optimal Bayes classifier, can be naturally extended to its least squares surrogate.

It is in fact possible to consider the weighted least squares problem:

$$f_w^* = \arg\min_{f:\mathcal{X}\to\mathbb{R}} \quad \int_{\mathcal{X}\times\{-1,1\}} w(y)(y - f(x))^2 d\rho(x, y), \tag{8.9}$$

and recover again the (weighted) rule $b^*_w(x) = sign(f^*_w(x))$ like in the non-weighted setting. Indeed, by direct calculation it follows that Eq. (8.9) has solution

$$f^*_w(x) = \frac{\rho(1|x)w(1) - \rho(-1|x)w(-1)}{\rho(1|x)w(1) + \rho(-1|x)w(-1)}. \tag{8.10}$$

Now, if we assume $w(1) > 0$ and $w(-1) > 0$ (as we set them in this work), the denominator of Eq. (8.10) is always positive and therefore $sign(f^*_w(x)) > 0$ if and only if $\rho(1|x)w(1) > \rho(-1|x)w(-1)$, as desired.

Recoding the Least Squares Loss

Another approach to recover the rebalanced optimal Bayes classifier via least squares surrogate is to apply a suitable *coding* function to the class labels $y = \{-1, 1\}$, namely:

$$f^*_c = \underset{f:\mathcal{X}\to\mathbb{R}}{\arg\min} \int_{\mathcal{X}\times\{-1,1\}} (c(y) - f(x))^2 d\rho(x, y), \tag{8.11}$$

where $c : \{-1, 1\} \to \mathbb{R}$ maps the labels y into scalar codes $c(y) \in \mathbb{R}$. Analogously to the unbalanced (and uncoded) case, the solution to Eq. (8.11) is

$$f^*_c(x) = c(1)\rho(1|x) - c(-1)\rho(-1|x) \tag{8.12}$$

which, for $c(y) = w(y)$, corresponds to the numerator of Eq. (8.10). Therefore, the optimal (rebalanced) Bayes classifier is recovered again by $b^*_w(x) = sign(f^*_c(x))$.

Multiclass Rebalancing and Recoding

In the multiclass setting, the optimal Bayes decision rule corresponds to the function $b^* : \mathcal{X} \to \{1, \dots, T\}$, that assigns a label $t \in \{1, \dots, T\}$ to $x \in \mathcal{X}$ when $\rho(t|x) > \rho(s|x)\ \forall s \neq t$, with $t, s \in \{1, \dots, T\}$. Consequently, the rebalanced decision rule would assign class t, whenever $\rho(y = t|x)w(t) > \rho(y = s|x)w(s)\ \forall s \neq t$, where the function $w : \{1, \dots, T\} \to \mathbb{R}$ assigns a weight to each class. Generalizing the binary case, in this work we set $w(t) = \frac{1}{\rho(y=t)}$, where we denote $\rho(y = t) = \int_{\mathcal{X}} d\rho(t, x)$, for each $t \in \{1, \dots, T\}$.

As we saw in Sec. 2.1, the surrogate least squares multiclass approach is recovered by adopting a *one-vs-all* strategy, formulated as the vector-valued problem:

$$f^* = \underset{f:\mathcal{X}\to\mathbb{R}^T}{\arg\min} \int_{\mathcal{X}\times\{1,\dots,T\}} \|e_y - f(x)\|^2 d\rho(x, y), \tag{8.13}$$

where $e_t \in \{0, 1\}^T \subseteq \mathbb{R}^T$ is a vector of the canonical basis $\{e_1, \dots, e_T\}$ of $\mathbb{R}^T$, i.e., the one-hot encoding of the label t. Analogously to the derivation of Eq. (8.7), it can be shown

that the solution to this problem corresponds to $f^*(x) = (\rho(1|x), \dots, \rho(T|x))^\top \in \mathbb{R}^T$ for all $x \in \mathcal{X}$. Consequently, we recover the optimal Bayes classifier by

$$b^*(x) = \arg\max_{t=1,\dots,T} f^*(x)^t, \tag{8.14}$$

where $f^*(x)^t$ denotes the t-th entry of the vector $f^*(x) \in \mathbb{R}^T$.

The extensions of recoding and rebalancing sapproaches to this setting follow analogously to the binary setting discussed in previous Section. In particular, the coding function $c : \{e_1, \dots, e_T\} \to \mathbb{R}$ consists in mapping the one-hot encoding e_t into $c(e_t) = \frac{e_t}{\rho(y=t)}$.

8.3 Incremental RLSC

In this Section, we briefly review the recursive formulation of the solution to the Regularized Least Squares for Classification (RLSC) algorithm presented in Sec. 2.1.5, which can be used to apply incremental updates to the model.

We recall from Sec. 2.1.5 that the problem consists in finding

$$\widehat{W} = \arg\min_{W \in \mathbb{R}^{d \times T}} \|Y - XW\|_F^2 + \lambda \|W\|_F^2. \tag{8.15}$$

Here we show that it is possible to derive a recursive formulation of the closed form solution of this problem:

$$\widehat{W} = (X^\top X + \lambda I_d)^{-1} X^\top Y \in \mathbb{R}^{d \times T} \tag{8.16}$$

which allows to incrementally update $\widehat{W}$ in fixed time as new training examples are observed [Sayed, 2008].

Consider a learning process where training data are provided one at a time to the system. At iteration k we would need to compute $W_k = (X_k^\top X_k + \lambda I_d)^{-1} X_k^\top Y_k$, where $X_k \in \mathbb{R}^{k \times d}$ and $Y_k \in \mathbb{R}^{k \times T}$ are the matrices whose rows correspond to the first k training examples. The computational cost for evaluating W_k is $O(kd^2)$ for the matrix products and $O(d^3)$ for the inversion. This is undesirable in the on-line setting, since k can grow indefinitely.

Here we show that we can compute W_k incrementally from W_{k-1} in $O(Td^2)$. To see this, first notice that by construction

$$X_k = [X_{k-1}^\top, x_k]^\top \qquad Y_k = [Y_{k-1}^\top, e_{y_k}]^\top,$$

and therefore, if we denote $A_k = X_k^\top X_k + \lambda I_d \in \mathbb{R}^{d \times d}$ and $b_k = X_k^\top Y_k \in \mathbb{R}^{d \times T}$, we obtain the recursive formulations:

$$A_k = X_k^\top X_k + \lambda I_d = X_{k-1}^\top X_{k-1} + x_k^\top x_k + \lambda I_d = A_{k-1} + x_k^\top x_k + \lambda I_d \tag{8.17}$$

$$b_k = X_k^\top Y_k = X_{k-1}^\top Y_{k-1} + x_k e_{y_k}^\top = b_{k-1} + x_k e_{y_k}^\top \tag{8.18}$$

Computing b_k from b_{k-1} requires $O(d)$ operations (since e_{y_k} has all zero entries but one). Computing A_k from A_{k-1} requires $O(d^2)$, while the inversion A_k^{-1} requires $O(d^3)$.

To reduce the cost of the (incremental) inversion, we recall that, for a positive definite matrix A_k, for which its Cholesky decomposition $A_k = R_k^\top R_k$ is known (with $R^k \in \mathbb{R}^{d\times d}$ upper triangular), the inversion A_k^{-1} can be computed in $O(d^2)$ [Golub and Loan, 1996]. In principle, computing the Cholesky decomposition of A_k still requires $O(d^3)$, but we can apply a rank-one update to the Cholesky decomposition at the previous step, namely $A_k = R_k^\top R_k = R_{k-1}^\top R_{k-1} + x_k x_k^\top = A_{k-1} + x_k x_k^\top$, which is known to require $O(d^2)$ [Björck, 1996]. Several implementations are available for the Cholesky rank-one updates; in our experiments we used the MATLAB routine CHOLUPDATE.

Therefore, the update W_k from W_{k-1} can be computed in $O(Td^2)$, since the most expensive operation is the multiplication $A_k^{-1} b_k$. In particular, this computation is independent of the current number k of training examples seen so far, making this algorithm suited for the on-line setting.

8.4 Incremental RLSC with Class Extension and Recoding

In this Section, we present our approach to incremental multiclass classification, where we account for the possibility to extend the number of classes incrementally and apply the recoding strategy introduced in Sec. 8.2. The full algorithm is reported in Alg. 2.

8.4.1 Class Extension

We introduce a modification of recursive RLSC which allows to extend the number of classes in constant time with respect to the number of examples seen so far. Let T_k denote the number of classes seen up to iteration k. We have two possibilities:

1. The new sample (x_k, y_k) belongs to a known class, this meaning that $e_{y_k} \in \mathbb{R}^{T_{k-1}}$ and $T_k = T_{k-1}$.
2. (x_k, y_k) belongs to a new class, implying that $y_k = T_k = T_{k-1} + 1$.

In the first case, the update rules for A_k, b_k and W_k explained in Sec. 8.3 can be directly applied. In the second case, the update rule for A_k remains unchanged, while the update of b_k needs to account for the increase in size (since $b_k \in \mathbb{R}^{d\times(T_{k-1}+1)}$). However, we can modify the update rule for b_k without increasing its computational cost by first adding a new column of zeros $\mathbf{0} \in \mathbb{R}^d$ to b_{k-1}, namely

$$b_k = [b_{k-1}, \mathbf{0}] + x_k^\top e_{y_k} \tag{8.19}$$

which requires $O(d)$ operations. Therefore, with the strategy described above it is indeed possible to extend the classification capabilities of the incremental learner during on-line

Algorithm 2 Incremental RLSC with Class Recoding

Inputs: Hyperparameters $\lambda > 0$, $\alpha \in [0, 1]$
Output: Learned weights W_k at each iteration
Initialize: $R_0 \leftarrow \sqrt{\lambda} I_d$, $b_0 \leftarrow \mathbf{0}$, $\gamma_0 \leftarrow \mathbf{0}$, $T \leftarrow 0$
Increment: Observed example (x_k, y_k):
  if $(y_k = T + 1)$ **then**
    $T \leftarrow T + 1$
    $\gamma_{k-1} \leftarrow [\gamma_{k-1}^\top, 0]^\top$
    $b_{k-1} \leftarrow [b_{k-1}, \mathbf{0}]$
  end if
  $\gamma_k \leftarrow \gamma_{k-1} + e_{y_k}$
  $\Gamma_k \leftarrow k \cdot \text{diag}(\gamma_k)^{-1}$
  $b_k \leftarrow b_{k-1} + x_k^\top e_{y_k}$
  $R_k \leftarrow \text{CHOLESKYUPDATE}(R_{k-1}, x_k)$
  $W_k \leftarrow R_k^{-1}(R_k^\top)^{-1} b_k (\Gamma_k)^\alpha$
return W_k

operation, without re-training it from scratch. In the following, we address the problem of dealing with class imbalance during incremental updates by performing incremental recoding.

8.4.2 Incremental Recoding

The main algorithmic difference between standard RLSC and the variant with recoding is in the matrix Y containing output training examples. Indeed, according to the recoding strategy, the vector e_t associated to an output label t is coded into $c(e_t) = \frac{e_t}{\rho(y=t)}$. In the batch setting, this can be formulated in matrix notation as

$$W = (X^\top X + \lambda I_d)^{-1} X^\top Y \Gamma$$

where the original output matrix is substituted by its encoded version $Y\Gamma \in \mathbf{R}^{n \times T}$, with Γ the $T \times T$ diagonal matrix whose t-th diagonal element is $\Gamma^{(t,t)} = \frac{1}{\rho(y=t)}$. Clearly, in practice $\rho(y = t)$ is estimated empirically (e.g. by $\hat{\rho}(y = t) = n_t/n$, the ratio between the number n_t of training examples belonging to class t and the total number n of examples).

The above formulation is favorable for the on-line setting. Indeed, we have

$$X_k^\top Y_k \Gamma_k = b_k \Gamma_k = (b_{k-1} + x_k^\top y_k)\Gamma_k \tag{8.20}$$

where Γ_k is the diagonal matrix of the (inverse) class distribution estimators $\hat{\rho}$ up to iteration k. Γ_k can be computed incrementally in $O(T)$ by keeping track of the number k_t of examples belonging to t and then computing $\hat{\rho}_k(y = t) = k_t/k$ (see Alg. (2) for how this update was implemented in our experiments).

Note that the above step requires $O(dT)$, since updating the (uncoded) b_k from b_{k-1} requires $O(d)$ and multiplying b_k by a diagonal matrix requires $O(dT)$. All the above computations are dominated by the product $A_k^{-1}b_k$, which requires $O(Td^2)$. Therefore, our algorithm is computationally equivalent to the standard incremental RLSC approach.

Coding as a Regularization Parameter. Depending on the amount of training examples seen so far, the estimator k_t/k could happen to not approximate $\rho(y = t)$ well. In order to mitigate this issue, we propose to introduce a parameter $\alpha \in [0, 1]$ and raise Γ_k element-wise to the power of α (indicated by $(\Gamma_k)^\alpha$). Indeed, it can be noticed that for $\alpha = 0$ we recover the (uncoded) standard RLSC, since $(\Gamma_k)^0 = I_T$, while $\alpha = 1$ applies full recoding. In Sec. 8.5.2 we discuss an efficient heuristic to find α in practice.

On Incremental Rebalancing. Note that, differently from recoding, the loss-rebalancing strategy (Eq. (8.9)) cannot be implemented incrementally. Indeed, the solution of the rebalanced empirical RLS is

$$W_k = (X_k^\top \Sigma_k X_k + \lambda I_d)^{-1} X_k^\top \Sigma_K Y_k \tag{8.21}$$

with Σ_k a $k \times k$ diagonal matrix whose i-th entry is equal to $\Sigma_k^{(i,i)} = \frac{1}{\hat{\rho}(y=y_i)}$. Since Σ_k changes at every iteration with the class distributors estimators, it is not possible to derive a rank-one update rule for $(X_k^\top \Sigma_k X_k + \lambda I_d)^{-1}$ as for standard RLSC.

8.5 Experiments

We empirically assessed the performance of Alg.2 on the MNIST machine learning benchmark and on the robotics visual recognition datasets Washington RGB-D (WRGB-D) and iCubWorld28 (iCW28).

To evaluate the improvement provided by the incremental recoding when classes are imbalanced, we compared the accuracy of the proposed method with the standard recursive RLSC presented in Sec. 8.3. As a competitor in terms of accuracy, we also considered the rebalanced approach presented in Eq. (8.21) (which, we recall, cannot be implemented incrementally).

8.5.1 Experimental Protocol

We adopted the following experimental protocol[1]:

1. Given a dataset with T classes, we simulated a scenario where a new class is observed by selecting $T - 1$ of them to be "balanced" and the remaining one to be under-represented.

[1]Code available at https://github.com/LCSL/incremental_multiclass_RLSC

2. We trained a classifier on the balanced classes, using a randomly sampled dataset containing n_{bal} examples per class (specified below for each dataset). We sampled a validation set with $n_{bal}/5$ examples per class.
3. We incrementally trained the classifier from the previous step by sampling online n_{imb} examples for the T-th class. Model selection was performed *using exclusively the validation set of the balanced classes*, following the strategy described in Sec. 8.5.2.
4. To measure performance, we sampled a separate test set containing n_{test} examples per class (both balanced and under-represented) and we measured the accuracy of the algorithms on the test set while they were trained incrementally.

For each dataset, we averaged results over multiple independent trials randomly sampling the validation set. In Table 8.1 we report the test accuracy on the imbalanced class and on the entire test set.

Datasets

MNIST [LeCun et al., 1998] is composed of 60K 28×28 centered greyscale pictures of digits from 0 to 9. We addressed the 10-class digit recognition problem and in our experiment we considered a smaller subset as training set, with $n_{bal} = 1$K examples per class. The test set was obtained by sampling $n_{test} = 200$ examples per class. We used the raw pixels of the images as inputs for the linear classifiers.

iCubWorld28 [Pasquale et al., 2015b] was introduced in Sec. 5.4 and specifically collected with the purpose of providing a benchmark for incremental object recognition in robotics. We addressed the 28-class object identification task, using all available acquisition sessions per object and randomly sampling $n_{bal} = 700$ and $n_{test} = 700$ examples per class

In [Pasquale et al., 2015b], we started investigating incremental learning on ICW28, by evaluating the benefit of incrementally refining the model of known objects as new example frames were observed. Such benefit was particularly evident when example images from a new day of acquisition were added to the training set. In that work, we adopted the standard recursive extension of RLSC (Sec. 8.3), experimenting only the addition of examples belonging to known classes. In this work, we evaluate the proposed incremental RLSC with addition of new classes (and recoding). We follow the same protocol adopted there, by feeding the linear classifiers with image representations extracted as the output of the *fc*7 activation layer of the *CaffeNet* Convolutional Neural Network trained on ImageNet (see Sec. 3.3). We refer to [Pasquale et al., 2015b] for more details about the procedure for feature extraction.

Washington RGB-D [Lai et al., 2011] was introduced in Sec. 6.D. We addressed the 51-class object categorization task, averaging results over the ten splits specified in [Lai et al., 2011] (where, we recall, for each category a random instance is left out for testing). As in

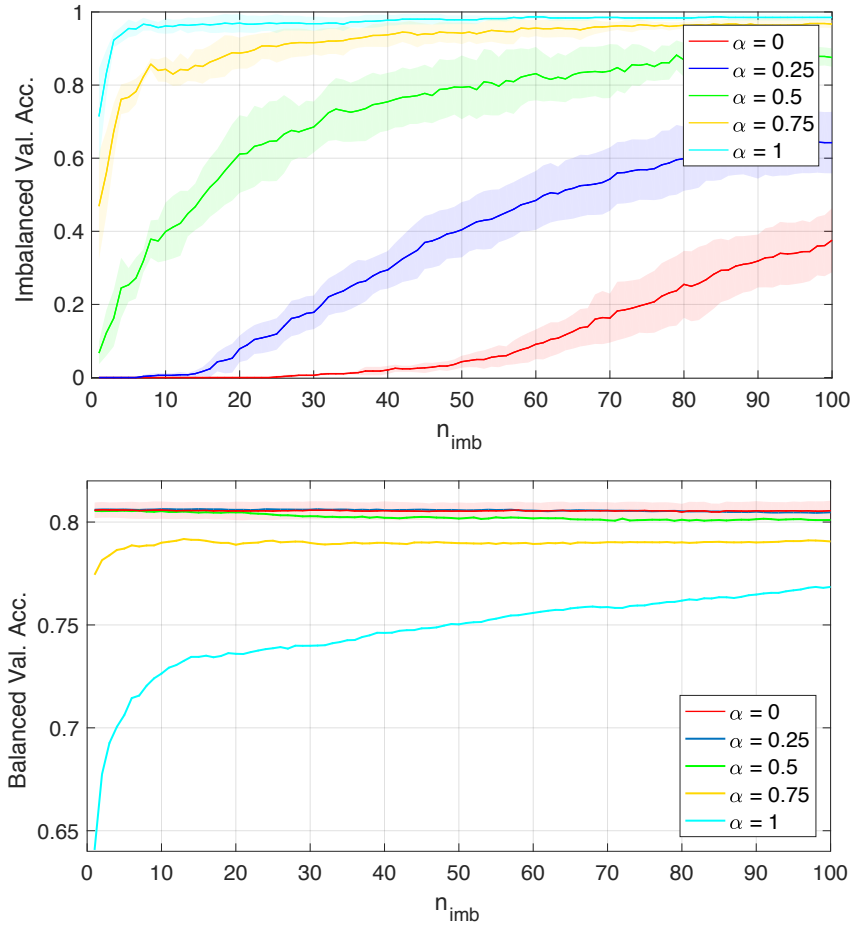

Figure 8.2: **Classification accuracy on** iCW28 **imbalanced** (Top) **and balanced** (Bottom) **classes for models trained according to Alg.** (2) with varying α and best λ within a predefined range (chosen at each iteration and for each α). Growing α from 0 to 1 allows to find a model that maintains same performance on known classes while improving on the under-represented one.

Sec. 6.D.1, we subsampled one cropped RGB frame every five from the full dataset, following the standard procedure. We sampled $n_{bal} = 500$ and $n_{test} = 400$ images per class (an average of ~ 900 images are available for each category) and performed feature extraction analogously to *iCubWorld28*, using the output of CAFFENET's $fc6$ layer.

8.5.2 Model Selection

In traditional batch learning settings for RLSC, model selection for the hyper-parameter λ is typically performed via hold-out, k-fold or similar cross-validation techniques. In the

incremental setting these strategies cannot be directly applied since examples are observed on-line, but a simple approach to create a validation set is to hold out every i-th example without using it for training (in our experiments we set $i = 6$). At each iteration, multiple candidate models are trained incrementally, each for a different value of λ, and the one with highest validation accuracy is selected for prediction.

However, following the same argument of Sec. 8.2, in presence of class imbalance this strategy would often select classifiers that ignore the under-represented class. Rebalancing the validation loss (see Sec. 8.2) does not necessarily solve the issue, but could rather lead to overfit the under-represented class, degrading the accuracy on other classes since errors count less on them. Motivated by empirical evidence discussed below, in this work we have adopted a model selections heuristic for λ and α in Alg. 2, which guarantees to not degrade accuracy on well-represented classes, while at the same time achieving higher or equal accuracy on the under-represented one.

Our strategy evaluates the accuracy of the candidate models on the incremental validation set, but *only for classes that have a sufficient number of examples* (e.g. classes with less examples than a pre-defined threshold are not used for validation). Then, we choose the model with largest $\alpha \in [0, 1]$ for which such accuracy is higher or equal to the one measured for $\alpha = 0$, namely without coding. Indeed, as can be seen in Fig. 8.2 for validation experiments on ICW28, as α grows from 0 to 1, the classification accuracy on the under-represented class increases (Top), while it decreases on the remaining ones, Fig. 8.2 (Bottom). Our heuristic chooses the best trade-off for α so that performance does not degrade on well-known classes but at the same time it will often improve on the under-represented one.

8.5.3 Results

In Table 8.1 we report the results of the three methods on *MNIST*, *iCubWorld28* and *RGB-D* for a single under-represented class (digit "8", class 28 and *tomato*, respectively). We observed a similar behaviour for other classes. We show both the accuracy on all classes (Total Acc., Left) and on the under-represented one (Imbalanced Acc., Right). We note that, on the under-represented class, Alg. 2 (*RC*) consistently outperforms the RLSC baseline (*N*), which does not account for class imbalance and learns models that ignore the class. Also the total accuracy of *RC* results higher. Interestingly, on the two robotics tasks, *RC* outperforms the loss rebalancing approach (*RB*), particularly when very few examples of the under-represented class are available. This is favorable since, as we said, the rebalancing approach cannot be implemented incrementally (Sec. 8.4.2).

To offer a clear intuition of the improvement provided by our method, in Fig. 8.3 we show the accuracy of Naïve RLSC (Red) and Alg. 2 (Blue), separately on the under-represented

Table 8.1: Incremental classification accuracy for Naïve (N) RLSC, Rebalanced (RB) and Recoding (RC) (see Alg. 2). Following the procedure described in Sec. 8.5.2, we set $\alpha = 0.7$ for *MNIST*, $\alpha = 0.6$ for *iCubWorld28* and $\alpha = 0.7$ for *RGB-D*.

Dataset	n_{imb}	Total Acc. (%)			Imbalanced Acc. (%)		
		N	RB	RC	N	RB	RC
MNIST $n_{bal} = 1000$	1	79.2 ± 0.3	$\mathbf{79.7 \pm 0.4}$	$\mathbf{79.7 \pm 0.6}$	0.0 ± 0.0	7.4 ± 7.7	$\mathbf{9.5 \pm 4.9}$
	5	79.1 ± 0.3	$\mathbf{82.5 \pm 0.7}$	80.3 ± 0.6	0.0 ± 0.0	$\mathbf{39.6 \pm 6.2}$	17.5 ± 6.6
	10	79.2 ± 0.3	$\mathbf{83.6 \pm 0.7}$	81.0 ± 0.6	0.0 ± 0.0	$\mathbf{49.5 \pm 5.7}$	25.1 ± 5.3
	50	79.2 ± 0.3	$\mathbf{85.5 \pm 0.3}$	83.9 ± 0.5	0.0 ± 0.0	$\mathbf{73.5 \pm 3.3}$	49.1 ± 3.5
	100	79.2 ± 0.4	$\mathbf{85.9 \pm 0.4}$	85.1 ± 0.5	2.0 ± 0.9	$\mathbf{75.5 \pm 2.7}$	62.7 ± 2.9
	500	85.5 ± 0.3	$\mathbf{86.2 \pm 0.3}$	$\mathbf{86.1 \pm 0.3}$	66.9 ± 1.1	$\mathbf{78.5 \pm 0.9}$	$\mathbf{77.8 \pm 1.1}$
iCub $n_{bal} = 700$	1	77.6 ± 0.3	76.8 ± 0.1	$\mathbf{77.7 \pm 0.3}$	0.0 ± 0.0	0.4 ± 0.6	$\mathbf{8.0 \pm 11.4}$
	5	77.6 ± 0.3	77.9 ± 0.1	$\mathbf{78.6 \pm 0.3}$	0.0 ± 0.0	8.1 ± 3.9	$\mathbf{38.5 \pm 9.7}$
	10	77.6 ± 0.3	78.3 ± 0.4	$\mathbf{78.9 \pm 0.2}$	0.0 ± 0.0	23.7 ± 10.8	$\mathbf{49.6 \pm 5.6}$
	50	77.7 ± 0.2	$\mathbf{80.0 \pm 0.2}$	$\mathbf{80.0 \pm 0.1}$	5.4 ± 4.1	73.9 ± 7.3	$\mathbf{75.0 \pm 5.5}$
	100	78.6 ± 0.1	$\mathbf{80.2 \pm 0.1}$	80.1 ± 0.2	39.1 ± 3.6	85.9 ± 4.0	$\mathbf{86.5 \pm 3.0}$
	500	80.2 ± 0.2	$\mathbf{80.1 \pm 0.1}$	$\mathbf{80.1 \pm 0.2}$	89.3 ± 2.5	93.8 ± 2.0	$\mathbf{94.8 \pm 1.9}$
RGB-D $n_{bal} = 500$	1	80.4 ± 2.2	78.6 ± 3.2	$\mathbf{83.3 \pm 3.2}$	0.0 ± 0.0	62.0 ± 42.1	$\mathbf{72.2 \pm 26.3}$
	5	80.4 ± 2.2	83.0 ± 2.1	$\mathbf{83.9 \pm 2.6}$	0.0 ± 0.0	91.7 ± 12.8	$\mathbf{99.9 \pm 0.3}$
	10	80.4 ± 2.2	$\mathbf{83.8 \pm 1.8}$	$\mathbf{83.6 \pm 2.6}$	2.8 ± 2.4	94.7 ± 8.4	$\mathbf{100.0 \pm 0.0}$
	50	82.3 ± 2.2	$\mathbf{84.3 \pm 1.9}$	83.5 ± 2.9	96.6 ± 3.7	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$
	100	82.4 ± 2.1	$\mathbf{84.4 \pm 2.0}$	83.5 ± 2.8	100.0 ± 0.0	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$
	500	82.3 ± 2.1	$\mathbf{84.1 \pm 2.0}$	$\mathbf{84.1 \pm 2.8}$	100.0 ± 0.0	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$

class (Top), the balanced classes (Middle), and all classes (Bottom), as they are trained on new examples. For this experiment we let each class be under-represented and averaged the results. It can be noticed that Alg. 2 is much better on the imbalanced class, while being comparable on the balanced ones, resulting in overall improved performance.

We point out that the total accuracy on all datasets for $n_{imb} = 500$ is comparable with the state of the art. Indeed, on *MNIST* we achieve $\sim 86\%$ accuracy, which is slightly lower than the one reported in [LeCun et al., 1998] for a linear classifier on top of raw pixels (this is reasonable, since we are using much fewer training examples). The total accuracy of Alg. 2 on *RGB-D* is approximately 84%, which is comparable with the state of the art on this dataset [Schwarz et al., 2015]. On the *iCubWorld28* dataset we achieve $\sim 80\%$ accuracy, which is in line with the results reported in Fig. 8 of [Pasquale et al., 2015a] (extended version of [Pasquale et al., 2015b]).

8.6 Conclusion

In this Chapter we addressed the problem of learning on-line with an increasing number of classes. With the ultimate goal of building a visual object recognition application for a lifelong

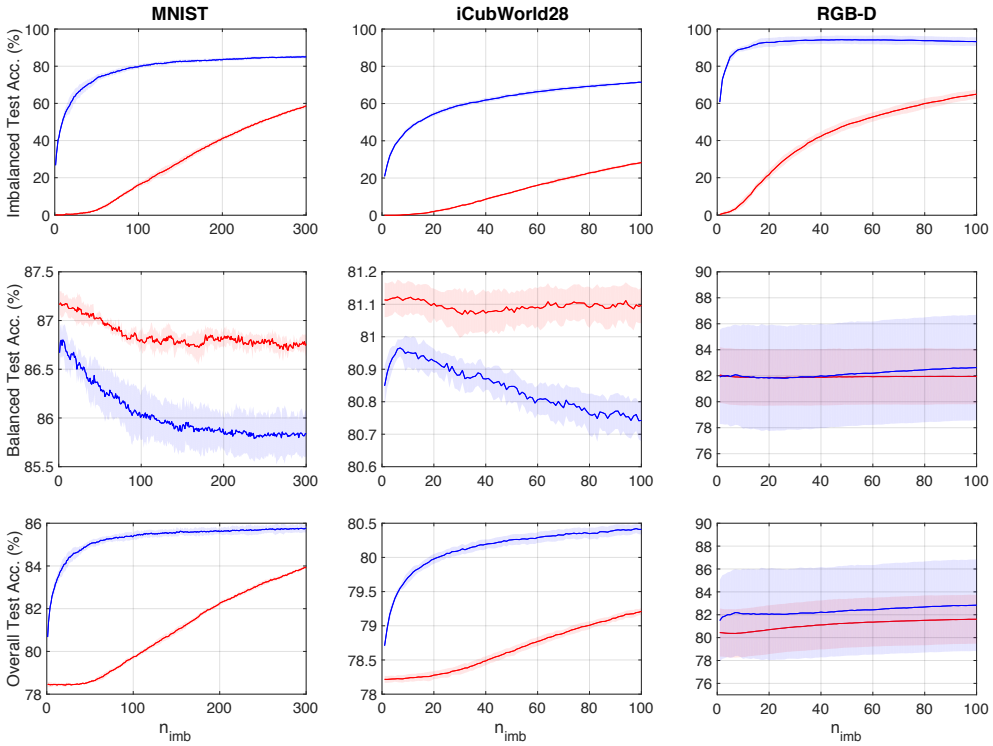

Figure 8.3: Average classification accuracy of the standard incremental RLSC (Red) and the variant proposed in this work (Blue) over the imbalanced (Top), balanced (Middle) and all (Bottom) classes. The models are incrementally trained as n_{imb} grows, as described in Sec. 8.5.1.

learning robotic setting, we focused on problems related to class imbalance, which naturally arises when a new category/object is observed for the first time. To address these issues, we proposed a variant of the recursive RLSC (Regularized Least Squares for Classification) algorithm, which (i) incorporates new classes incrementally and (ii) dynamically applies class recoding when new examples are observed. Updates are performed in constant time with respect to the growing number of training examples. We evaluated the proposed algorithm on a standard machine learning benchmark and on two robotics datasets, showing that our approach is indeed favourable in online settings when classes are imbalanced.

We note that, in principle, for the experiments where we used features extracted from a Convolutional Neural Network, we could have also directly trained the network online, by Stochastic Gradient Descent (backpropagation). While works empirically investigating this end-to-end approach in settings where new classes are to be progressively included into the model exist [Käding et al., 2016], this is still a largely unexplored field, the study of which is

not in the scope of this work. The method we propose allows to update a predictor without using training data from previous classes in a fast and stable way, and, by relying on rich deep representations learned offline, is proven to be competitive with the state of the art, while being more suitable for online applications.

Future research will focus on strategies to exploit knowledge of known classes to improve classification accuracy on new ones, following recent work [Tommasi et al., 2010, Tommasi et al., 2012, Kuzborskij et al., 2013, Sun and Fox, 2016].

Chapter 9

Robotic Applications: Putting it All Together

Providing autonomous agents with robust and adaptive recognition capabilities for real world scenarios poses scientific but also technological challenges. In this Thesis we addressed both: in previous Chapters, we reported on our investigation of the use of modern deep networks for visual recognition in robotics, focusing on key aspects like learning on the fly and learning incrementally; in this Chapter, we show how the deployment of the proposed solutions on real platforms resulted into a fast, adaptive, visual recognition system remarkably improving the capabilities of the robots.

After introducing the two platforms used as testbed for our experiments, the iCub and R1 humanoid robots (Sec. 9.1), we provide an overview of the developed applications (Sec. 9.2). The first one is the iCubWorld Framework (Sec. 9.3), a set of software tools that have been implemented to collect the iCubWorld datasets presented in Chapter 5. The second one is the On the fly Recognition application (Sec. 9.4), an interactive demo where the robot learns to recognize new objects taught by a human, relying on the pipeline based on Convolutional Neural Networks and the Regularized Least Squares for Classification algorithm described in previous Chapters.

9.1 Robot Platforms

We provide a very brief introdution to the two platforms consistently used during this Thesis, the iCub (Sec. 9.1.1) and R1 (Sec. 9.1.2) humanoids. Both have been designed with the goal of realizing general-purpose assistants for people in possibly multiple scenarios.

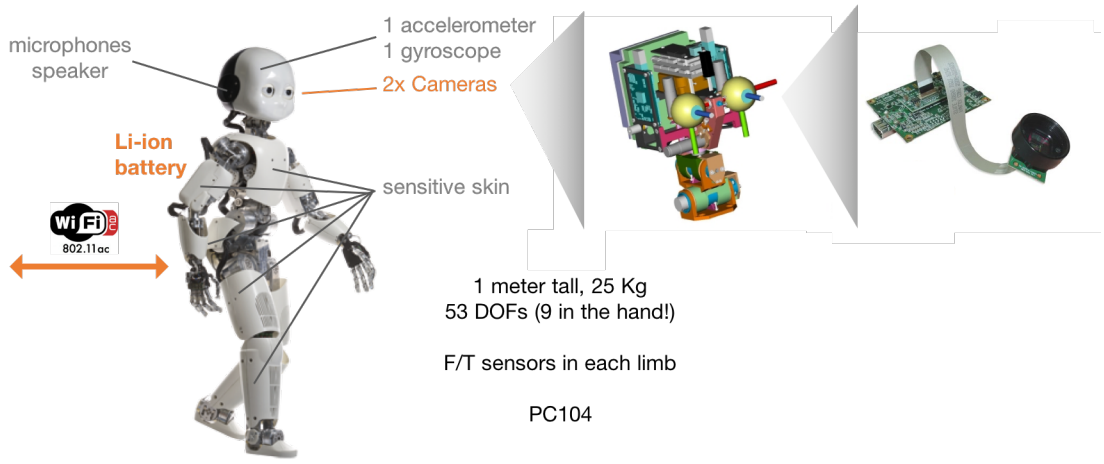

Figure 9.1: **iCub**: Full body (Left); Head Kinematics (Middle); Dragonfly Camera (Right).

9.1.1 iCub

An Open Source Platform for Research. The iCub [Metta et al., 2010][1] (Fig. 9.1) is a fully humanoid robot specifically conceived as a testbed for research in human cognition and Artificial Intelligence (AI). The motivation behind the accurate humanoid design is the *embodied cognition* hypothesis [Metta et al., 2008], arguing that we learn cognitive skills by interacting with the environment and other humans, using the limbs and senses, and, consequently, our internal model of the world is determined by the form of the human body. Therefore, the iCub's design is aimed to allow the robot to interact with the world in the same way a child would do, through an accurate reproduction of the human perceptual system and articulation [Metta et al., 2008].

The robot was initially developed within the FP-6 European Project RobotCub and built by the Istituto Italiano di Tecnologia (IIT) in 2004. At present, more than 30 laboratories in Europe, US, Korea and Japan are using this platform for their research and more than 70 people work on this project at the iCub Facility department, IIT, in Genoa. The name is a partial acronym, *cub* standing for *Cognitive Universal Body*. The hardware design, software and documentation are all open source, released under the GPL/LGPL license. Since the first iCub robots were built in 2004, the design has undergone several revisions and improvements. With respect to the very first versions, the latest iCub has for example smaller and more dexterous hands, lighter and more robust legs with greater joint angles, to permit walking [Tsagarakis et al., 2009], a progressively more extended skin covering, and so forth.

[1] http://www.iCub.org

Short Description. While we report in Appendix 9.A.1 a detailed description of the robot, here we just highlight the features of its "visual system", which is the part most involved in the developed applications. This consists in a stereo pair in a swivel mounting inside the head, with 2 dragonfly cameras positioned where the eyes would be located on a human (see Fig. 9.1). Each camera streams RGB images of 640×480 pixels at frame rate of 33Hz. Since the robot was not initially designed for autonomous operation, it was consequently not equipped with on board processors suitable for this. Instead, an umbilical cable provides power and network connection. Therefore, all computational-intensive applications, including visual processing, are executed on external machines. However, the most recent iCub version is equipped with a WiFi connection and a battery mounted as a backpack such that the robot can move freely in the environment.

Software Infrastructure. The iCub software relies on the YARP (Yet Another Robot Platform) [Metta et al., 2006a] middleware. YARP is a set of C++ libraries, protocols, and tools to keep software modules cleanly independent of specific devices and operative systems. In this way, the applications for the robot can be "composed" by connecting multiple software components or YARP *modules*, each providing a specific functionality and running independently on a machine, communicating peer-to-peer data and instructions with other modules. Like iCub, also the YARP middleware is open source, released under the LGPL. Currently, iCub and YARP software are hosted on Github within the `robotology` organization [2]. Accordingly, all modules and applications result of this work are publicly available under the same organization (the specific repositories will be described in Chapter 9).

9.1.2 R1

"Your Personal Humanoid". R1 (Fig. 9.2) is a recent platform designed at the iCub Facility, with the aim of commercializing the robot as a general-purpose assistant for public or professional environments (as hotels, hospitals, shopping centres and, eventually, private houses).

The design and materials of R1 are such to definitely reduce its cost (from ~ 250.000euro, current price of one iCub robot, to an estimate of ~ 20.000euro for R1's prototypes and ~ 3.000euro in production), while maintaining most of the capabilities of the iCub. It is made of plastic and fiber of carbon and metal (50%), ~ 1.25m high and ~ 50Kg weigh.

[2] https://www.github.com/robotology

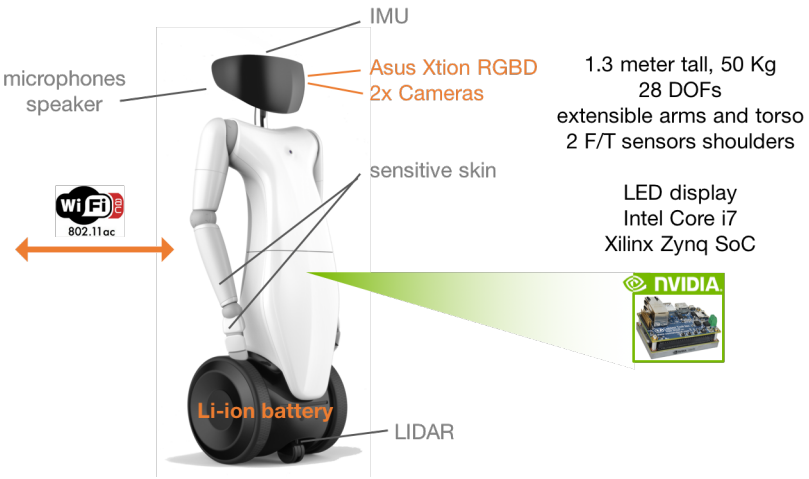

Figure 9.2: **R1**

Short Description. With respect to the iCub, it is equipped with a fixed stereo pair, complemented with an additional RGB-D sensor (see Fig. 9.2). The RGB output image of the depth sensor is 320×240 pixels, while the left/right cameras are synchronized and stream full HD images (1920×1080), both at at frame rate of 33Hz. Overall, with respect to the iCub, R1 is provided with a more robust and detailed perception of the scene structure, which is particularly critical for navigation. Like the iCub, it is also equipped with microphones and speakers enabling verbal communication.

Differently from iCub, R1 is a fully autonomous platform provided with WiFi connectivity, a battery with $\sim$ 3h of autonomy, and three computational units: an FPGA (Field Programmable Gate Array), an Intel i7 CPU, and an NVIDIA Tegra (Jetson TX1). This latter device is a quite powerful embedded system that supports a Linux distribution and that is endowed with a quad-core ARM processor, 4GB RAM and an integrated 256-core CUDA GPU. The system provides 1TFLOPS of FP16 compute performance in 10 watts of power and has been specifically released for deploying computer vision and deep learning algorithms to mobile platforms like robots, cars, personal devices etc. Indeed, we will see in Sec. 9.4.1 that the full visual recognition system developed in this work runs on it in real time.

Software Infrastructure. This robot shares the same YARP middleware with iCub and therefore most of the software written for this latter one runs, with minor adaptation, also on R1. In particular, the recognition application produced for this work runs on both platforms, replacing only the parts that rely on the physical structure of the robots. For instance, on the iCub, the depth map must be computed from the stereo pair, while, on R1, it is provided

by the RGB-D sensor. Similarly, the implementation of the visual tracking of a target will be platform-specific. However, modules providing high-level cognitive functionalities and behaviors can be interchanged seamlessly between the two platforms. This includes the visual recognition pipeline developed in this work. Such flexibility is one of the advantages provided by the modularity of YARP middleware.

As for the iCub, in Appendix 9.A.2 we expand this brief description providing some more details on the robot.

9.2 Overview

In this Section we provide an overview of the developed applications. The overall aim of the provided software tools is to make the set of operations that we performed in previous Chapters, throughout our investigation, reproducible for other users of the robots. In particular, in Chapter 7 we observed that fine-tuning off-the-shelf CNN models, like AlexNet, on image sequences representative of the variability factors to which the system should be robust, does improve the invariance properties of the internal representation of these models with respect to such factors. In particular, we considered viewpoint variations and collected the ICUBWORLD TRANSFORMATIONS dataset. We then fine-tuned models on such dataset and, by encoding images into fine-tuned representations, we were able to learn objects' appearance from few examples, by training Regularized Least Square Classifiers (RLSC) on top.

The set of applications developed makes this "cycle" reproducible for other users with minimum coding effort. The ICUBWORLD FRAMEWORK provides software tools to collect annotated visual data from the iCub or R1 platforms and fine-tune (or train) deep CNNs on the acquired data, in order to obtain robust representations. The ON THE FLY RECOGNITION application finally allows to use the trained representations within a recognition pipeline, to teach new objects to the robot "on the fly".

In the following, we briefly describe how the different software components allow to perform the main three phases involved in such process:

Data Collection: A first component is a data acquisition application to collect annotated visual sequences. This application is the one that has been developed to acquire the latest ICUBWORLD TRANSFORMATION release (see Sec. 5.3.2) and will be described in Sec. 9.3.1.

Training: The second component allows to evaluate the fine-tuning (or training) of different CNN architectures, with possibly different protocols, on the collected dataset (or subsets of it), in order to produce a model conveying a deep representation with good properties with respect to the desired recognition task. This consists in a set of MATLAB routines

(interfacing with CAFFE) that has been developed to perform the experiments reported in Chapters 6 and 7, and it will be described in Sec. 9.3.2. Together with the acquisition application, it constitutes the ICUBWORLD FRAMEWORK.

Deployment: Once the acquisition and offline training phases are done, the fine-tuned CAFFE model can be deployed into the visual recognition pipeline of the robot, where it can be used to encode in real time images of objects into corresponding representations. To this end, a representation extraction module has been implemented, which uses NVIDIA software tools to optimize CAFFE models and provide real time predictions (also on memory-constrained embedded devices like the GPU onboard the R1 robot). The representation extraction step is followed by a classification stage, where RLSC are trained on the fly to learn new objects. The performance of the resulting recognition pipeline composed of representation extraction and RLSC can be evaluated within the ON THE FLY RECOGNITION demo, described in Sec. 9.4.

All components are independent and can be employed either separately or in sequence to (i) adapt a visual representation to potentially new settings and (ii) insert it into a fast learning pipeline to teach the robot new objects on the fly. In the following, we describe more in detail each part.

9.3 iCubWorld Framework

In this Section we briefly present the main functionalities of the ICUBWORLD acquisition application (Sec. 9.3.1) and of the MATLAB routines to train CAFFE models on the acquired sequences (Sec. 9.3.2), comprising together the ICUBWORLD FRAMEWORK.

9.3.1 Acquisition

The ICUBWORLD acquisition setup has been introduced in Sec. 5.3.1. Here we specifically consider the "Human Mode", the modality employed for the acquisition of the latest release: a human teacher standing in front of the robot holds an object in the hand and pronounces its label; the robot focuses on the object and tracks it, by exploiting bottom up visual cues, for a time period during which frames (annotated with the object's label and bounding box) are recorded. A schema of the application is reported in Fig. 9.3. Each box is a YARP module, which runs independently on a node of the iCub's cluster and communicates with other nodes through a YARP network (see Sec. 9.1).

As it can be noticed, the application is composed of two main parts: a first one (`Disparity Segmentation` and `Motor Control` modules) is dedicated to continuously focus the robot's gaze on the object of interest (in this case, the depth cue is used, as it will be explained

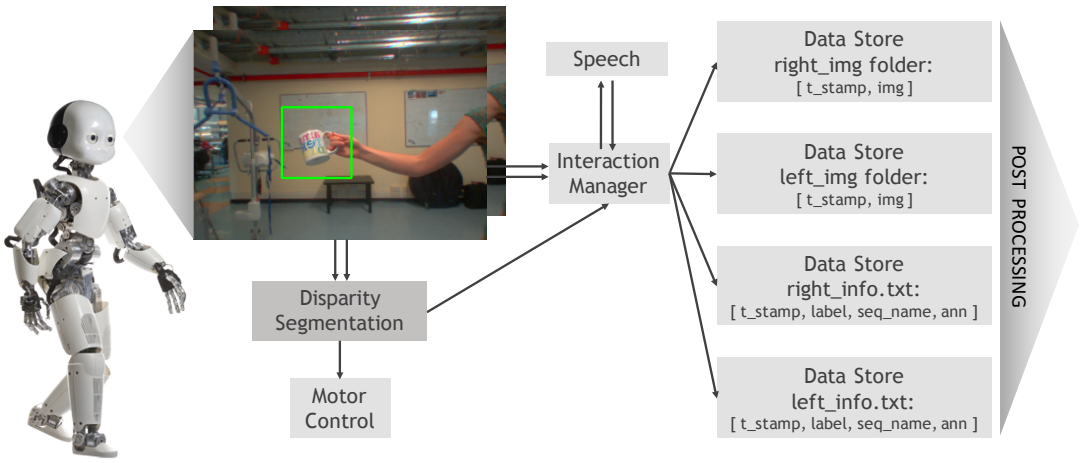

Figure 9.3: **iCubWorld**: schema of the acquisition application (Human Mode).

in the following); a second one (`Speech`, `Interaction Manager` and `Data Store` modules) instead manages the verbal interaction with the human supervisor and starts/stops accordingly the recordings.

In the following, we shortly illustrate the object localization method, based on depth segmentation, and the software infrastructure to automatically check, clean and organize the acquired raw data.

Object Tracking by Depth Segmentation

As explained in Sec. 5.3.2, the depth cue is a robust distinguishing feature to localize the object of interest in the considered setting, because this is assumed to be the closest entity to the robot. To this end, we devised a depth-based attention system, whose operation is showed in Fig. 9.4, reporting three excerpts from the acquisitions of three objects (one per column). In the top row, an RGB frame and its annotation, superimposed in terms of object's label and bounding box, are represented. In the middle row, the corresponding disparity map of the scene is reported. Finally, in the bottom row, the segmentation of the lightest (that is, the closest) region in the disparity map is shown to produce a rough mask on the object of interest, which can be used to compute (i) the object's 3D centroid and (ii) its enclosing bounding box. The centroid is sent to a motor control module, which focuses the robot's gaze on the object. The bounding box is instead used to annotate the frame (and is reported also in the top row). The result is that the robot continuously focuses on the closest object in its visual field.

A complete description of the system is published in [Pasquale et al., 2016b]. To realize it, we first improved the stereo perception of the iCub robot in order to provide dense, robust and

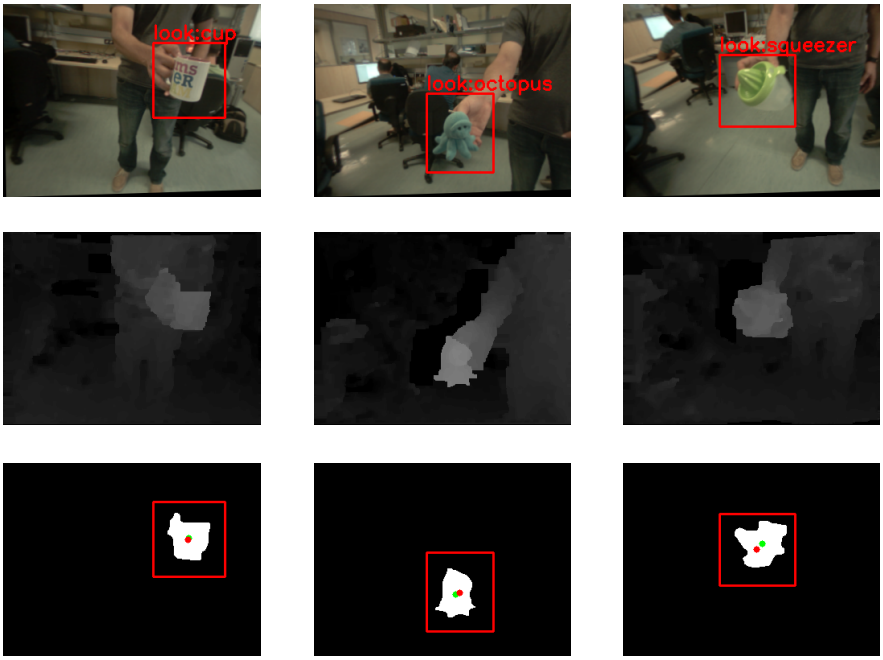

Figure 9.4: **Depth-based Localization Pipeline.** Three frames (extracted from the video attached to [Pasquale et al., 2016b] as supplemental material) showing the effectiveness of the proposed segmentation system. Top: resulting crop in the left frame, labeled by the operator using speech. Middle: disparity map. Bottom: segmented disparity map, with the closest blob, its centroid and enclosing bounding box. Red and green centroids correspond to two different temporal smoothing windows. See [Pasquale et al., 2016b] for details.

real time 3D information usable for tracking. Then, we devised a simple, lightweight disparity segmentation algorithm. Finally, we closed the loop by streaming the centroid coordinates to the robot's gaze controller.

An adapted version of this system has also been implemented on the R1 robot, with the following differences:

Depth Map: as anticipated in Sec. 9.1, R1 is endowed with an RGB-D camera directly outputting the depth map.

Depth Segmentation: this part is common to both iCub and R1 and the module's code is hosted under the `robotology` GitHub organization[3].

Gaze Control: of course the gaze control is platform-specific and we relied on available modules.

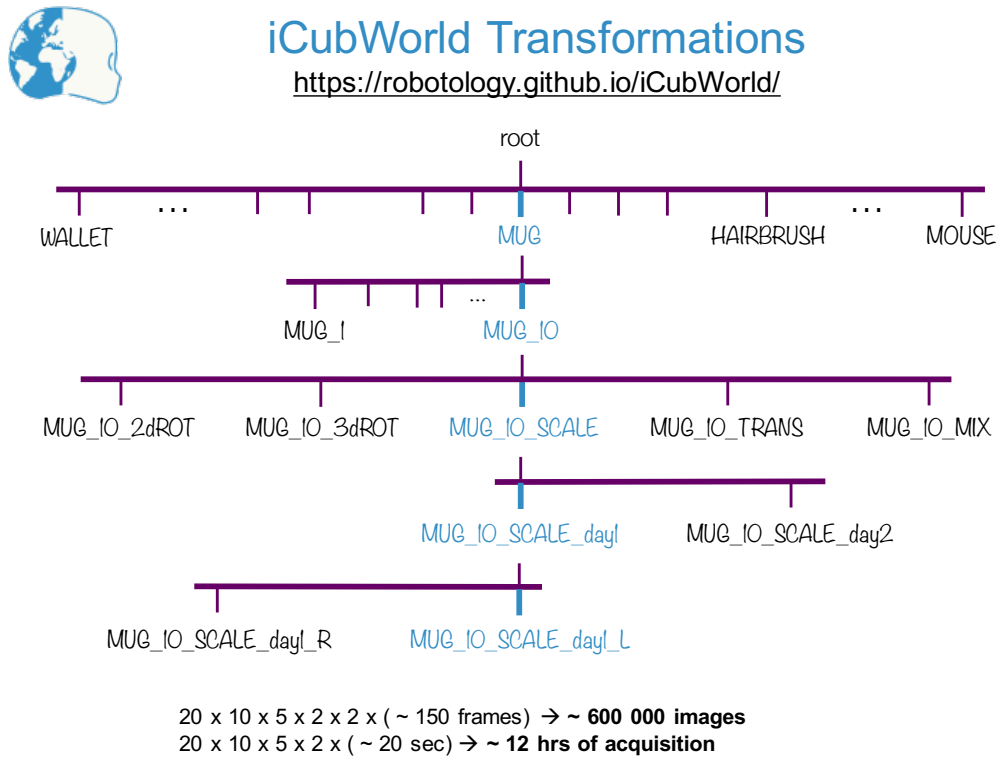

Figure 9.5: **iCubWorld**: directory tree.

Scalable Application

Another fundamental innovation introduced in the acquisition setup allows to scale the number of collected sequences, with the automation of several operations during the acquisition and in the post-processing of the recorded data. This was necessary, since the ICUBWORLD TRANSFORMATION overall comprises 4k image sequences. Specifically, all post-acquisition steps to format the raw streams of frames/annotations into an ordered set of folders were automated. As it can be observed in Fig. 9.3, images are first recorded into a unique folder, where they can be associated only to their timestamp. Image-related information is also written in a unique file, where each row contains the timestamp (to be associated with that of the corresponding image), the label and the name of the specific sequence that is being acquired (e.g., 2*D Rot*, *Scale*, etc...), and finally the localization annotation (see Fig. 9.3 for an example). All this information must be checked, cleaned and finally formatted into a folder tree like the one depicted in Fig. 9.5, representing an example of the structure of ICUBWORLD

[3] https://www.github.com/robotology/segmentation/dispBlobber

TRANSFORMATION.

A MATLAB set of functions was developed to perform these steps semi-automatically. With this setup, users can acquire a new dataset by (i) running the acquisition application, interacting with the robot and showing the desired objects, and, afterwards, (ii) running the post-processing application, which takes the raw sequences and produces the dataset in a standard format. We plan to make this code soon available on the ICUBWORLD website[4] in order to involve multiple laboratories to use this application to collect a shared database for visual recognition on the iCub.

9.3.2 Training

Once the dataset has been collected and prepared, it can be used to train or fine-tune CNN models. To this end, a set of MATLAB functions has been developed, providing support for calling CAFFE APIs.

To train a CAFFE model three definition files are needed, providing, respectively: the list of training/test images, the network structure (i.e., the sequence of layers with specified parameters) and the training protocol (i.e., SGD hyper-parameters). The aim of the provided routines is to help the user producing these files in an programmatic way. In particular, regarding the creation of training/test sets, if the dataset is formatted as a folder tree (as for example the one reported in Fig. 9.5), it is possible to create arbitrary sets including, e.g. only certain categories, or certain objects per category, or sequences per object, etc., by simply listing the directories to be included at each level. Regarding the creation of the architecture and SGD configuration files, once a "base" model and protocol are chosen (e.g., *CaffeNet* with the default protocol), it is possible to create many variations of them, by listing the different values to be tried for each parameter that has to be changed. A plotting tool is included to help inspecting the performance of the experimented models.

This framework has been developed to perform the experiments reported in Chapters 6, 7 and the code will be publicly released on the same ICUBWORLD website.

9.4 On the Fly Object Recognition

In this Section we describe the visual recognition application that has been developed to teach new objects to the robot on the fly, in an analogous setting to the one adopted for the acquisition of ICUBWORLD. The application is called ON THE FLY OBJECT RECOGNITION and allows to directly test the robustness of visual representations trained, e.g., on ICUBWORLD datasets, in a real world setting. Notably, this tool is also used within other applications, relying on

[4] https://robotology.github.io/iCubWorld/

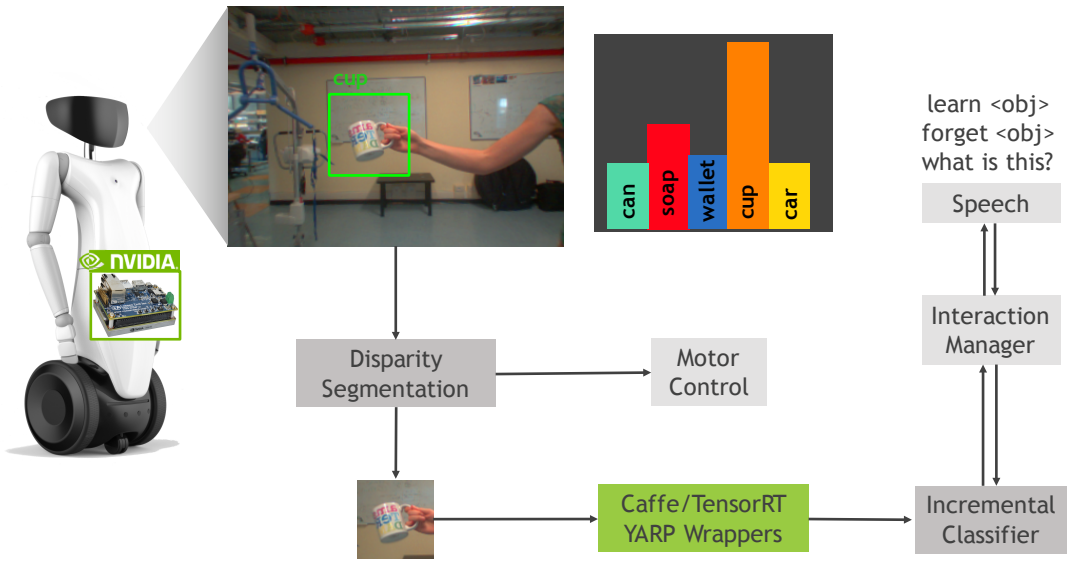

Figure 9.6: **On the fly Object Recognition**: application schema.

the outcome of visual recognition to perform further actions (see, e.g., [Fischer and Demiris, 2016]).

Framework	**Model**	**Precision**	Time (batch=1)	Power cons.
CAFFE	*CaffeNet*	FP32	33Hz	7.2W
TENSORRT	*CaffeNet*	FP16	43Hz	6.6W
CAFFE	*GoogLeNet*	FP32	24Hz	9.6W
TENSORRT	*GoogLeNet*	FP16	40Hz	8W

Table 9.1: **caffeCoder and GIECoder**: comparison of computational time and power consumption between respectively CAFFE and TENSORRT implementations of *CaffeNet* or *GoogLeNet*, running on the Jetson TX1 card onboard the R1 robot.

A sketch of the application is depicted in Fig. 9.6 and some snapshots of the demo are also represented in Fig. 1.2. As in the ICUBWORLD acquisition scenario, a human teacher standing in front of the robot shows an object holding it in the hand and pronounces its label; a routine exploiting a bottom-up visual cue allows the robot to focus on the new entity of interest and track it for a certain time period, during which cropped frames around the object are encoded into vector representations and fed to a classifier module. Once done, the object is learned, and the human can hence show it again to the robot, this time asking to recognize it. The procedure can be repeated for all the objects that the robot has to learn. Also, specific objects can be

forgotten with an analogous verbal command.

The application is implemented in YARP and is composed of different modules, the most important ones reported in Fig. 9.6. A localization module segments a salient region in the image and extracts a rectangular crop around it, that is fed to the recognition pipeline. This latter is composed by (i) a representation extraction module that encodes images into vector representations and (ii) a classifier module that is trained and tested on the extracted representations. A speech recognition/production module tackles the verbal interaction and a motor controller moves the robot's gaze towards the localized region. The robot continuously localizes, tracks and recognizes salient regions in incoming frames, and the visual output, exemplified in Fig. 9.6, is an histogram of class probabilities (that is, the real time output of the classifier), whose class with maximum score is chosen as the predicted object's label. The histogram can also be averaged over a time window to increase the stability of predictions, accordingly to the results that we observed in [Pasquale et al., 2015b]. A manager module triggers the training of the classifier depending on the processed verbal commands to train or forget the objects.

At present, the object localization and tracking is dealt with the same attention system described in Sec. 9.3.1, exploiting the depth as discriminant cue for the object. Nevertheless, alternative solutions could be plugged at this level. Similarly to the acquisition of ICUBWORLD, also this demo can work in a different modality, where the robot takes the object in its hand and explores it, instead of tracking it in the scene (we called it the "Robot Mode" in Sec. 5.3.1). To demonstrate the robustness of the recognition pipeline however, the "Human Mode" is more suited since it allows to teach objects in shorter times and to "stress" the system with more challenging conditions.

The code is hosted in the GitHub `robotology` organization[5] and is implemented both for the iCub and R1 platforms. In the following, we briefly describe the two components of the recognition pipeline (while we skip the speech and motor controller modules since these are not part of this work).

9.4.1 Representation Extraction

This step is performed by a YARP module that wraps a CNN model implemented in a deep learning framework, CAFFE in this case. Specifically, the module receives a batch of images as input, feeds them to a CAFFE model, runs a forward pass and provides their corresponding representations as the output of one of the intermediate layers of the architecture. Every CAFFE model can be used, since the module internally calls CAFFE APIs. For instance, either an off-the-shelf model, or one fine-tuned within the ICUBWORLD FRAMEWORK, can be employed.

[5] https://www.github.com/robotology/onthefly-recognition

Also the layer at which to extract representations can be changed (e.g., when choosing the output layer, the predictions of the network can also be directly obtained). The module, called `caffeCoder`, is hosted in the `himrep` repository[6] (that stands for `hierarchical image representations`), together with the YARP module previously employed for visual recognition on the iCub robot, which was based on a Sparse Coding dictionary learning strategy proposed in [Fanello et al., 2013c].

Note that `caffeCoder` requires a GPU (Graphics Processing Unit) in order to process images in real time. When using the iCub, the module runs (like all other modules) on external nodes, which can be easily provided with a GPU. However, we care to point out that the module runs in real time also on the embedded GPU onboard the R1 robot.

Optimizing CAFFE models for R1. To this regard, a second module has also been implemented, with the same functionality of `caffeCoder` but exploiting the recently released NVIDIA software tool TENSORRT to process CAFFE models and produce further optimized implementations. In particular, the TENSORRT[7] library optimizes the implementation of deep CNN models in various supported frameworks (comprising CAFFE), in order to reduce their memory and computational requirements. This library processes the definition of the architecture of a CNN model (e.g., the order and size of convolution, pooling, non linearity layers, etc.) and produces an equivalent model, with a more efficient implementation for each layer, which runs faster, particularly on embedded GPUs like R1's NVIDIA Jetson TX1. The module is hosted at the same `himrep` repository under the name of `GIECoder`, where GIE stands for GPU INFERENCE ENGINE, the initial name of the TENSORRT framework. The module, while originally developed for R1, it is in fact usable on every supported GPU.

In Table 9.1 the computational time of one feedforward pass of a single image through *CaffeNet* or *GoogLeNet* is reported, comparing `caffeCoder` and `GIECoder` modules running on the NVIDIA Jetson TX1 onboard the R1 robot. Also the power consumption is reported, as the difference between the Jetson's consumption with or without the module running. As can be noticed, both modules achieve (almost) real time performance. Moreover, TENSORRT allows to run complex models like *GoogLeNet* on memory-constrained platforms like the one considered, with even better performance. Note that TENSORRT offers the possibility of optimizing computations with FP16 precision with particular efficiency, with negligible numerical differences in the output representations that do not affect predictions.

The full ON THE FLY RECOGNITION demo including the `GIECoder` can run entirely on the Jetson TX1 in real time, without the need of external additional computational resources.

[6] https://www.github.com/robotology/himrep
[7] https://developer.nvidia.com/tensorrt

9.4.2 Classification

The incremental version of RLSC proposed in Chapter 8 is currently under development and will be released soon. For the time being, on the robots we use the `linearClassifier` module, hosted in the `himrep` repository and employing the `liblinear`[8] library.

9.5 Conclusions

In this Chapter, we have presented the software tools that have been developed and made available as a product of this Thesis. The iCubWorld Framework comprises a data acquisition application, which allows to quickly record large-scale collections of annotated image sequences while interacting with the robot. An additional set of MATLAB functions accelerate the use of the acquired sequences to train or fine-tune deep CNN models by interfacing with Caffe APIs.

The On the fly Recognition application allows to assess the performance of the proposed recognition pipeline in a similar human-robot interaction scenario, where new objects can be taught to the robot on the fly. An optimized module extracts in real time image representations as the output of an intermediate layer of a CNN model, and a downstream classifier is trained and tested online in the representation space. The CNN model can be chosen off-the-shelf or fine-tuned, possibly on sequences previously acquired within the iCubWorld Framework.

The overall system allows users to quickly train a deep CNN on the recognition problem of interest, and assess the efficacy of the resulting visual representation within the same real world scenario where the robot will be expected to operate.

[8] https://www.csie.ntu.edu.tw/~cjlin/liblinear/

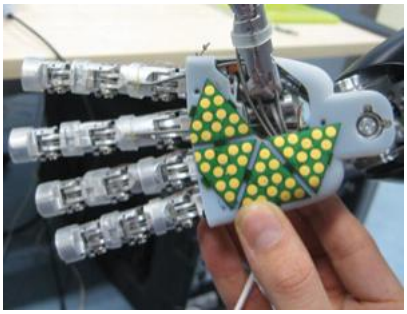

Figure 9.A.1: **iCub**: Skin sensors.

Appendices

9.A Robots Details

We expand the two descriptions provided in Sec. 9.1, adding some more details on the adopted robotic platforms.

9.A.1 iCub

The iCub humanoid is $\sim$ 1m high and $\sim$ 25 kg in weight. In the following, its main features are listed.

Actuators. The robot has 53 actuated Degrees Of Freedom (DOFs), distributed in this way:

arms: 7 each

hands: 9 each (3 for the thumb, 2 for the index, 2 for the middle finger, 1 for the coupled ring and little finger, 1 for the adduction/abduction). As can be noticed, the hands have been designed to support sophisticate manipulation skills.

head: 6 (3 for the neck and 3 for the cameras)

torso: 3

legs: 6 each

It also has lines of red LEDs representing mouth and eyebrows mounted behind the face panel for making facial expressions, and speakers.

Sensors. The iCub is one of the few platforms in the world with a sensitive full-body skin to deal with safe physical interaction with the environment. At present, the skin covers the chest, arms and forearms, hand palms and finger tips (see Fig. 9.A.1), legs and calfs.

We already described the stereo pair enabling 3-dimensional perception of the environment. Some versions of the robot are also equipped with two asynchronous bio-inspired Dynamic Vision Sensors (DVS) [Lichtsteiner et al., 2008], event-based cameras that allow the perception

of fast changes in the scene with super real time temporal resolution.

Two microphones are mounted where the ears are in humans. A gyroscope and an accelerometer are also mounted on the head to act like the human vestibular system. Joint encoders and 6-axis F/T sensors in each limb are also present.

Computational Capabilities. The robot is controlled by an on board PC104 controller which communicates with actuators and sensors using CANBus. It is not equipped with other on board processors and, while in previous versions, an umbilical cable provided power and network connection, the most recent version is endowed with a WiFi connection and a battery mounted as a backpack.

Software. As anticipated, the iCub software relies on the YARP (Yet Another Robot Platform) [Metta et al., 2006a] middleware, a set of C++ libraries, protocols, and tools to keep software modules cleanly independent on specific devices and operative systems. Its functionality is equivalent to the Robot Operative System (ROS)[9] [Quigley et al., 2009], both aiming to ease the complexity of developing robotics applications through software modularity. By means of the YARP middleware, applications can be composed by connecting multiple YARP modules running independently on nodes of a cluster and communicating peer-to-peer data and instructions through a YARP network. Indeed, YARP supports asynchronous or synchronous communication and many protocols like tcp, udp, multicast, local, MPI, mjpeg-over-http, XML/RPC, tcpros. It is written in C++, like most of iCub software, but it includes interfaces to other languages like Python, LUA, Matlab. It supports Windows, Linux/Debian, MacOs, iOs and Android, and flexible interfacing with the hardware devices used on the robot platform.

9.A.2 R1

The R1 humanoid is made of plastic and fiber of carbon and metal (50%), ~ 1.25m high and ~ 50 kg in weight. It is simpler than the iCub, and very robust, accordingly with its commercial purpose. In the following we provide a brief comparison between the two platforms:

Actuators. With respect to the iCub, R1 replaces legs with wheels, achieving a 2Km/h speed (threshold set mainly for safety reasons). It has plier-like hands, with a spherical wrist that can lift up to 1.5Kg. Globally, it has 28 DOFs:

arms: 8 each

hands: 2 each (1 for the thumb, 1 for the fingers).

head: 2 (neck)

torso: 4

wheels: 1 each

[9] http://www.ros.org/

Notably, its arms and torso are extensible (13cm and 15cm respectively), to allow reaching for far objects. Differently from iCub, its head is fully covered with a LED monitor to communicate expressions and sentences (and of course it is provided with speakers).

Sensors. R1's skin does not cover the full body but only the most critical parts including forearms, hand palms and the thumb tip. We already described its visual system, composed of both a fixed stereo pair and one RGB-D sensor. Also, a LIDAR (Laser Imaging Detection And Ranging) is mounted on the basis.

Finally, a microphone, gyroscope and an accelerometer are mounted on the head, and 6-axis F/T sensors on the shoulders.

Computational Capabilities. We already pointed out that R1 is a fully autonomous platform, provided with WiFi connectivity, a battery and three powerful computational units: an FPGA (Field Programmable Gate Array), an Intel i7 CPU, and an NVIDIA Tegra (Jetson Tx1), which we can exploit to run the developed recognition system in real time, as described in Sec. 9.4.1.

Software. As described, R1 shares the same YARP middleware with iCub, facilitating also software reusability.

Chapter 10

Future Directions

In this Chapter, we consider possible directions for future research, in order to address the challenges that emerged from this study and mitigate the impact of the gap with web-scale image classification settings when performing visual recognition in robotics. The goal of this discussion is to consider different research topics together in the same picture to present how, in our opinion, the visual recognition problem in robotics could be addressed from multiple fronts.

Improving invariance

We showed that modern deep Convolutional Neural Networks (CNNs) can learn specific invariances from data, and our results confirm that improving representation invariance is crucial to perform visual recognition in real world scenarios [Pinto et al., 2008, Pinto et al., 2011, Borji et al., 2016, Poggio and Anselmi, 2016]. Within this perspective, the effects of training these models on sequences that represent objects undergoing specific visual nuisances should be definitely further investigated. In particular, it would be interesting to gain more insights on the relevancy of viewpoint invariance in categorization tasks, an aspect which evidenced unexpected results in our setting and which has also been considered in recent work employing the iLab-20M dataset [Zhao et al., 2016, Zhao and Itti, 2016].

Increasing semantic variability

We clearly observed how this point is critical to object categorization. The simplest method is to share data acquired from different robotic platforms. We have seen in Sec. 5.2 how [Levine et al., 2016b] use this strategy to collect the necessary data to learn hand-eye coordination with deep CNNs, or, similarly, the Million Object Challenge [Oberlin et al., 2015] which aims to acquire a dataset of $1M$ object models by sharing acquisitions from different laboratories

owning a Baxter robot [1]. Along a similar direction, we explained how we plan to expand iCubWorld with the help of the community of the iCub robot. Beyond expanding the dataset in the direction of semantic variability, this would also allow to collect multiple acquisitions of the same object in different conditions, extending the analysis presented in this work in multiple directions.

An alternative, or complementary, approach, is *data augmentation*. We have seen in Sec. 4.1.2 how this method is largely employed to increase viewpoint/illumination variability by applying synthetic transformations to the images in the training set. Visual augmentation to cope for semantic variability is a much more challenging problem, although recent work on *inverse graphics* [Mansinghka et al., 2013, Kulkarni et al., 2014, Kulkarni et al., 2015] is a starting point in this direction.

Exploiting 3D Contextual Information

As we saw in Sec. 1.3.2, the exploitation of 3D information in robotics has largely been investigated. Recently, the depth has been integrated in CNNs with encouraging results [Schwarz et al., 2015, Eitel et al., 2015, Redmon and Angelova, 2015, Carlucci et al., 2016]. While [Schwarz et al., 2015, Eitel et al., 2015, Redmon and Angelova, 2015] leverage off-the-shelf representations learned with RGB data and include the depth channel through colorization schemes, [Carlucci et al., 2016] synthetically generate a large-scale RGB-D dataset and train a CNN model entirely on it, reporting further improved performance with respect to previous approaches.

The spatial structure of the scene can also be exploited as an additional self-supervisory signal, or prior information, to make the prediction more robust. Examples are the work of [Song et al., 2015], where, for instance, the 3D reconstruction of the room helps discriminating between known objects and newly introduced ones. Interestingly, [Pillai and Leonard, 2015] implement a SLAM-aware detection system where the 3D reconstruction of the objects in the scene allows to project them back on frames and make their localization on the images more robust. With this approach, object instances could also be autonomously discovered by the robot.

Exploiting Temporal Contextual Information

While it is important to push the limits of visual recognition at the frame level, a robot is typically exposed to a continuous stream of images, in which the visual information is highly correlated. We showed in [Pasquale et al., 2015b] how the prediction accuracy of a system can be remarkably improved by exploiting this correlation, even with trivial solutions as the temporal averaging of predictions. More elaborated solutions have been proposed in the

[1] http://www.rethinkrobotics.com/

literature, including the use of recent Recurrent Convolutional Neural Networks [Donahue et al., 2015].

The temporal correlation among consecutive frames can be exploited as well as an additional self-supervisory signal. For instance, in [Wang and Gupta, 2015] and [Goroshin et al., 2015b, Goroshin et al., 2015a, Jayaraman and Grauman, 2015, Agrawal et al., 2015], visual representations are learned in absence of label annotations at each frame, by enforcing, e.g., the representations of consecutive frames to be similar.

Self-supervised Learning

We opted for the use of a human teacher in the acquisition of iCubWorld Transformation because we needed a relatively fine control on the movement of the objects. However, iCubWorld could be extended by introducing self-supervision in the acquisition process. Indeed, we mentioned in Sec. 1.3.2 that a way to reduce the cost of supervision is to implement explorative strategies in which the robot autonomously interacts with the environment to collect training samples. Strategies specific to the robotic domain could be devised by integrating multiple sensory modalities [Sinapov et al., 2014, Higy et al., 2016] and a repertoire of explorative actions (like grasping or pushing objects [Montesano et al., 2008, Fitzpatrick et al., 2003, Högman et al., 2016, Pinto et al., 2016]) designed to extract useful cues. The work of [Pinto et al., 2016] is interesting because, while there has been significant work in the vision and robotics community to develop vision algorithms for performing robotic tasks, as grasping, they reverse the pipeline and use robotic tasks for learning visual representations.

Incremental and Multi-task Learning

We pointed out in Sec. 6.5 how this is a crucial aspect in the development of an adaptive recognition system that must guarantee reliability in lifelong learning scenarios. To this end, firstly, suitable datasets should be collected (extending, e.g., iCubWorld28 to more days, conditions, and so forth). Then, leveraging the proposed incremental pipeline based on RLSC as a tool that allows to keep computations feasible, further investigation of the behavior of the considered learning models in such settings should be carried on.

Within this perspective, in line with the literature on "learning to learn" and transfer learning [Thrun, 1996], future research should also focus on strategies to exploit knowledge of previous, well-represented classes, to improve classification accuracy on novel and under-represented ones. Indeed, as empirically observed in recent work [Tommasi et al., 2010, Kuzborskij et al., 2013, Sun and Fox, 2016], sharing information and structures among classification tasks can dramatically improve performance.

To this regard, while in this work we adopted a typical approach to categorization, where no similarity or relation is assumed among different classes, object categories do have common visual features (e.g. wheel-like parts, legs, handles and so on) and sharing such information within the learning model could improve the overall accuracy of a system, especially in presence of scarce training data.

The literature on multi-task learning proposes different ways to model the relations across tasks/categories and directly enforcing them on the problem when available a priori [Crammer and Singer, 2000, Micchelli and Pontil, 2004, Evgeniou et al., 2005, Joachims et al., 2009, Fergus et al., 2010, Lozano and Sindhwani, 2011] or learn them when unknown [Argyriou et al., 2008, Jacob et al., 2008, Minh and Sindhwani, 2011, Dinuzzo et al., 2011, Sindhwani et al., 2012, Ciliberto et al., 2015a, Ciliberto et al., 2015b].

Bibliography

[Agrawal et al., 2015] Agrawal, P., Carreira, J., and Malik, J. (2015). Learning to see by moving. In *The IEEE International Conference on Computer Vision (ICCV)*. 16, 191

[Agrawal et al., 2014] Agrawal, P., Girshick, R., and Malik, J. (2014). *Analyzing the Performance of Multilayer Neural Networks for Object Recognition*, pages 329–344. Springer International Publishing, Cham. 72

[Aldoma et al., 2012a] Aldoma, A., Tombari, F., Di Stefano, L., and Vincze, M. (2012a). *A Global Hypotheses Verification Method for 3D Object Recognition*, pages 511–524. Springer Berlin Heidelberg, Berlin, Heidelberg. 14

[Aldoma et al., 2013] Aldoma, A., Tombari, F., Prankl, J., Richtsfeld, A., Stefano, L. D., and Vincze, M. (2013). Multimodal cue integration through hypotheses verification for rgb-d object recognition and 6dof pose estimation. In *2013 IEEE International Conference on Robotics and Automation*, pages 2104–2111. 14

[Aldoma et al., 2012b] Aldoma, A., Tombari, F., Rusu, R. B., and Vincze, M. (2012b). *OUR-CVFH – Oriented, Unique and Repeatable Clustered Viewpoint Feature Histogram for Object Recognition and 6DOF Pose Estimation*, pages 113–122. Springer Berlin Heidelberg, Berlin, Heidelberg. 14

[Aldoma et al., 2011] Aldoma, A., Vincze, M., Blodow, N., Gossow, D., Gedikli, S., Rusu, R. B., and Bradski, G. (2011). Cad-model recognition and 6dof pose estimation using 3d cues. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 585–592. 14

[Aleotti et al., 2012] Aleotti, J., Rizzini, D. L., and Caselli, S. (2012). Object categorization and grasping by parts from range scan data. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 4190–4196. 14

[Alexe et al., 2012] Alexe, B., Deselaers, T., and Ferrari, V. (2012). Measuring the objectness of image windows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2189–2202. 6

[Anselmi et al., 2015] Anselmi, F., Leibo, J. Z., Rosasco, L., Mutch, J., Tacchetti, A., and Poggio, T. (2015). Unsupervised learning of invariant representations. *Theoretical Computer Science*. 39, 65

[Anselmi et al., 2016] Anselmi, F., Rosasco, L., and Poggio, T. (2016). On invariance and selectivity in representation learning. *Information and Inference*, 5(2):134–158. 39, 65

[Argyriou et al., 2008] Argyriou, A., Evgeniou, T., and Pontil, M. (2008). Convex multi-task feature learning. *Machine Learning*, 73. 192

[Assael et al., 2016] Assael, Y. M., Shillingford, B., Whiteson, S., and de Freitas, N. (2016). LipNet: Sentence-level Lipreading. *ArXiv e-prints*. 13

[Aubry and Russell, 2015] Aubry, M. and Russell, B. C. (2015). Understanding Deep Features with Computer-Generated Imagery. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2875–2883. 68, 135

[Azagra et al., 2016] Azagra, P., Mollard, Y., Golemo, F., Murillo, A., Lopes, M., and Civera, J. (2016). A multimodal dataset for interactive and incremental learning of object models. In *submitted to ICRA 2017*. 86

[Azizpour et al., 2015] Azizpour, H., Sharif Razavian, A., Sullivan, J., Maki, A., and Carlsson, S. (2015). From generic to specific deep representations for visual recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 71

[Babenko et al., 2014] Babenko, A., Slesarev, A., Chigorin, A., and Lempitsky, V. (2014). *Neural Codes for Image Retrieval*, pages 584–599. Springer International Publishing, Cham. 109, 110, 135

[Baishya and Bäuml, 2016] Baishya, S. and Bäuml, B. (2016). Robust material classification with a tactile skin using deep learning. In *IEEE International Conference on Intelligent Robots and Systems*. 16

[Bakry et al., 2015] Bakry, A., Elhoseiny, M., El-Gaaly, T., and Elgammal, A. (2015). Digging Deep into the Layers of CNNs: In Search of How CNNs Achieve View Invariance. *arXiv preprint*, pages 1–20. 68, 135

[Barkmeyer, 2016] Barkmeyer, N. (2016). Learning to recognize new objects using deep learning and contextual information. Master's thesis, Technische Universitat Munchen - Institute for Cognitive Systems (TUM - ICS). 86

[Bartlett et al., 2006] Bartlett, P., Jordan, M., and McAuliffe, J. (2006). Convexity, classification, and risk bounds. *Journal of the American Statistical Association*. 26

[Bartolozzi et al., 2016] Bartolozzi, C., Natale, L., Nori, F., and Metta, G. (2016). Robots with a sense of touch. *Nat Mater*, 15(9):921–925. 15

[Bay et al., 2006] Bay, H., Tuytelaars, T., and Van Gool, L. (2006). *SURF: Speeded Up Robust Features*, pages 404–417. Springer Berlin Heidelberg, Berlin, Heidelberg. 14, 39

[Bergstra and Bengio, 2012] Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305. 63

[Bishop, 2006] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA. 9, 10, 28, 29, 42, 51

[Björck, 1996] Björck, Å. (1996). *Numerical Methods for Least Squares Problems*. 161

[Bo et al., 2013] Bo, L., Ren, X., and Fox, D. (2013). *Unsupervised Feature Learning for RGB-D Based Object Recognition*, pages 387–402. Springer International Publishing, Heidelberg. 14, 149

[Borji et al., 2016] Borji, A., Izadi, S., and Itti, L. (2016). ilab-20m: A large-scale controlled object dataset to investigate deep learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 18, 68, 88, 98, 135, 189

[Botterill et al., 2008] Botterill, T., Mills, S., and Green, R. (2008). Speeded-up bag-of-words algorithm for robot localisation through scene recognition. In *2008 23rd International Conference Image and Vision Computing New Zealand*, pages 1–6. 15

[Bottou, 2012] Bottou, L. (2012). *Stochastic Gradient Tricks*, volume 7700, pages 430–445. Springer. 34, 123

[Boureau et al., 2010a] Boureau, Y. L., Bach, F., LeCun, Y., and Ponce, J. (2010a). Learning mid-level features for recognition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2559–2566. 41, 47

[Boureau et al., 2010b] Boureau, Y.-L., Ponce, J., and LeCun, Y. (2010b). A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 111–118. 41, 47

[Boureau et al., 2011] Boureau, Y. L., Roux, N. L., Bach, F., Ponce, J., and LeCun, Y. (2011). Ask the locals: Multi-way local pooling for image recognition. In *2011 International Conference on Computer Vision*, pages 2651–2658. 41, 47

[Boyd and Vandenberghe, 2004] Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press, New York, NY, USA. 32, 33

[Cadieu et al., 2014] Cadieu, C. F., Hong, H., Yamins, D. L. K., Pinto, N., Ardila, D., Solomon, E. A., Majaj, N. J., and DiCarlo, J. J. (2014). Deep neural networks rival the representation of primate it cortex for core visual object recognition. *PLoS computational biology*, 10:e1003963. 50

[Camoriano et al., 2017] Camoriano, R., Pasquale, G., Ciliberto, C., Natale, L., Rosasco, L., and Metta, G. (2017). Incremental robot learning of new objects with fixed update time. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 153

[Carlucci et al., 2016] Carlucci, F. M., Russo, P., and Caputo, B. (2016). A deep representation for depth images from synthetic data. *ArXiv e-prints*. 190

[Chatfield et al., 2011] Chatfield, K., Lempitsky, V., Vedaldi, A., and Zisserman, A. (2011). The devil is in the details: an evaluation of recent feature encoding methods. In *British Machine Vision Conference*. 41

[Chatfield et al., 2014] Chatfield, K., Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference*. 13, 72

[Chitta et al., 2011] Chitta, S., Sturm, J., Piccoli, M., and Burgard, W. (2011). Tactile Sensing for Mobile Manipulation. *IEEE Transactions on Robotics*, 27(3):558–568. 15

[Choi and Christensen, 2012] Choi, C. and Christensen, H. I. (2012). 3d textureless object detection and tracking: An edge-based approach. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3877–3884. 14

[Choi et al., 2016] Choi, S., Zhou, Q.-Y., Miller, S., and Koltun, V. (2016). A large dataset of object scans. *arXiv:1602.02481*. 79

[Ciliberto et al., 2012] Ciliberto, C., Fanello, S. R., Natale, L., and Metta, G. (2012). A heteroscedastic approach to independent motion detection for actuated visual sensors. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3907–3913. IEEE. 6, 81

[Ciliberto et al., 2013] Ciliberto, C., Fanello, S. R., Santoro, M., Natale, L., Metta, G., and Rosasco, L. (2013). On the impact of learning hierarchical representations for visual recognition in robotics. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3759–3764. 15, 65, 75, 82, 109

[Ciliberto et al., 2015a] Ciliberto, C., Mroueh, Y., Poggio, T., and Rosasco, L. (2015a). Convex learning of multiple tasks and their structure. In *International Conference on Machine Learning*. 192

[Ciliberto et al., 2011] Ciliberto, C., Pattacini, U., Natale, L., Nori, F., and Metta, G. (2011). Reexamining lucas-kanade method for real-time independent motion detection: Application to the icub humanoid robot. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4154–4160. IEEE. 6, 81

[Ciliberto et al., 2015b] Ciliberto, C., Rosasco, L., and Villa, S. (2015b). Learning multiple visual tasks while discovering their structure. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 131–139. 192

[Cloutman, 2013] Cloutman, L. L. (2013). Interaction between dorsal and ventral processing streams: Where, when and how? *Brain and Language*, 127(2):251 – 263. 4

[Collet et al., 2009] Collet, A., Berenson, D., Srinivasa, S. S., and Ferguson, D. (2009). Object recognition and full pose registration from a single image for robotic manipulation. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 48–55. 14

[Collet et al., 2011] Collet, A., Martinez, M., and Srinivasa, S. S. (2011). The moped framework: Object recognition and pose estimation for manipulation. *The International Journal of Robotics Research*, 30(10):1284–1306. 14, 109

[Collet and Srinivasa, 2010] Collet, A. and Srinivasa, S. S. (2010). Efficient multi-view object recognition and full pose estimation. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2050–2055. 14

[Corke, 2011] Corke, P. (2011). *Robotics, Vision and Control*. Springer Tracts in Advanced Robotics. 2

[Crammer and Singer, 2000] Crammer, K. and Singer, Y. (2000). On the learnability and design of output codes for multiclass problems. In *In Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, pages 35–46. 192

[Csurka et al., 2004] Csurka, G., Dance, C. R., Fan, L., Willamowski, J., and Bray, C. (2004). Visual categorization with bags of keypoints. In *In Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22. 11

[Dahiya et al., 2010] Dahiya, R. S., Metta, G., Valle, M., and Sandini, G. (2010). Tactile Sensing - From Humans to Humanoids. *IEEE Transactions on Robotics*, 26(1):1–20. 15

[Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1. 11, 39

[Dansereau et al., 2016] Dansereau, D. G., Singh, S. P., and Leitner, J. (2016). Interactive computational imaging for deformable object analysis. In *2016 International Conference on Robotics and Automation (ICRA)*. IEEE. 15

[Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*. 89

[Desimone and Ungerleider, 1989] Desimone, R. and Ungerleider, L. G. (1989). Neural mechanisms of visual processing in monkeys. *Handbook of neuropsychology*, 2:267–299. 3

[DiCarlo and Cox, 2007] DiCarlo, J. J. and Cox, D. D. (2007). Untangling invariant object recognition. *Trends in Cognitive Sciences*, 11(8):333 – 341. 64, 68

[DiCarlo et al., 2012] DiCarlo, J. J., Zoccolan, D., and Rust, N. C. (2012). How does the brain solve visual object recognition? *Neuron*, 73:415–34. 64, 65

[Dimashova et al., 2013] Dimashova, M., Lysenkov, I., Rabaud, V., , and Eruhimov, V. (2013). Tabletop object scanning with an rgb-d sensor. In *3rd Workshop on Semantic Perception, Mapping and Exploration (SPME)*. 14

[Dinuzzo et al., 2011] Dinuzzo, F., Ong, C. S., Gehler, P., and Pillonetto, G. (2011). Learning output kernels with block coordinate descent. *International Conference on Machine Learning*. 192

[Doersch et al., 2015] Doersch, C., Gupta, A., and Efros, A. A. (2015). Unsupervised visual representation learning by context prediction. In *The IEEE International Conference on Computer Vision (ICCV)*. 15

[Donahue et al., 2015] Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., and Darrell, T. (2015). Long-term recurrent convolutional networks for visual recognition and description. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 191

[Donahue et al., 2014] Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2014). Decaf: A deep convolutional activation feature for generic visual recognition. In Jebara, T. and Xing, E. P., editors, *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 647–655. JMLR Workshop and Conference Proceedings. 70

[Dosovitskiy and Brox, 2016] Dosovitskiy, A. and Brox, T. (2016). Inverting visual representations with convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 66

[Duchi et al., 2011] Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159. 36, 153, 155

[Edelman and Poggio, 1991] Edelman, S. and Poggio, T. (1991). Models of object recognition. *Current Opinion in Neurobiology*, 1(2):270 – 273. 10

[Eitel et al., 2015] Eitel, A., Springenberg, J. T., Spinello, L., Riedmiller, M., and Burgard, W. (2015). Multimodal deep learning for robust rgb-d object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 681–687. 16, 97, 98, 129, 130, 190

[Ekvall et al., 2005] Ekvall, S., Kragic, D., and Hoffmann, F. (2005). Object recognition and pose estimation using color cooccurrence histograms and geometric modeling. *Image and Vision Computing*, 23(11):943 – 955. 14

[Elkan, 2001] Elkan, C. (2001). The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*. 154, 156, 157

[Endres and Hoiem, 2010] Endres, I. and Hoiem, D. (2010). *Category Independent Object Proposals*, pages 575–588. Springer Berlin Heidelberg, Berlin, Heidelberg. 6

[Everingham et al., 2015] Everingham, M., Eslami, S. M. A., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2015). The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136. 10, 76

[Everingham et al., 2010] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338. 72, 76

[Evgeniou et al., 2005] Evgeniou, T., Micchelli, C. A., and Pontil, M. (2005). Learning multiple tasks with kernel methods. In *Journal of Machine Learning Research*, pages 615–637. 192

[Fanello et al., 2013a] Fanello, S. R., Ciliberto, C., Natale, L., and Metta, G. (2013a). Weakly supervised strategies for natural object recognition in robotics. *IEEE International Conference on Robotics and Automation*, pages 4223–4229. 80, 81

[Fanello et al., 2013b] Fanello, S. R., Ciliberto, C., Santoro, M., Natale, L., Metta, G., Rosasco, L., and Odone, F. (2013b). iCub World: Friendly Robots Help Building Good Vision Data-Sets. In *2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 700–705. 75, 82

[Fanello et al., 2014] Fanello, S. R., Noceti, N., Ciliberto, C., Metta, G., and Odone, F. (2014). Ask the image: supervised pooling to preserve feature locality. In *IEEE Computer Vision and Pattern Recognition*. 41

[Fanello et al., 2013c] Fanello, S. R., Noceti, N., Metta, G., and Odone, F. (2013c). Multi-class image classification - sparsity does it better. In *Computer Vision Theory and Applications (VISAPP), 2013 International Conference on*. 183

[Fei-Fei et al., 2007] Fei-Fei, L., Fergus, R., and Perona, P. (2007). Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 106(1):59–70. 65, 76

[Fei-Fei and Perona, 2005] Fei-Fei, L. and Perona, P. (2005). A bayesian hierarchical model for learning natural scene categories. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 524–531 vol. 2. 40

[Fergus et al., 2010] Fergus, R., Bernal, H., Weiss, Y., and Torralba, A. (2010). Semantic label sharing for learning with many categories. *European Conference on Computer Vision*. 192

[Filliat, 2007] Filliat, D. (2007). A visual bag of words method for interactive qualitative localization and mapping. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3921–3926. 15

[Finn and Levine, 2016] Finn, C. and Levine, S. (2016). Deep Visual Foresight for Planning Robot Motion. *ArXiv e-prints*. 16

[Firman, 2016] Firman, M. (2016). RGBD Datasets: Past, Present and Future. In *CVPR Workshop on Large Scale 3D Data: Acquisition, Modelling and Analysis*. 77

[Fischer and Demiris, 2016] Fischer, T. and Demiris, Y. (2016). Markerless perspective taking for humanoid robots in unconstrained environments. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3309–3316. 181

[Fischler and Bolles, 1981] Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395. 14

[Fishman, 1997] Fishman, R. S. (1997). Gordon holmes, the cortical retina, and the wounds of war. *Documenta Ophthalmologica*, 93(1):9–28. 46

[Fitzpatrick et al., 2003] Fitzpatrick, P., Metta, G., Natale, L., Rao, S., and Sandini, G. (2003). Learning About Objects Through Action – Initial Steps Towards Artificial Cognition. volume 3, pages 3140–3145. 191

[Fitzpatrick et al., 2008] Fitzpatrick, P., Needham, A., Natale, L., and Metta, G. (2008). Shared challenges in object perception for robots and infants. *Infant and Child Development*, 17(1):7–24. 15

[Franco et al., 2009] Franco, A., Maio, D., and Maltoni, D. (2009). *The Big Brother Database: Evaluating Face Recognition in Smart Home Environments*, pages 142–150. Springer Berlin Heidelberg, Berlin, Heidelberg. 85

[French, 1999] French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*. 153, 155

[Fukushima, 1980] Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202. 12, 45

[Geusebroek et al., 2005] Geusebroek, J.-M., Burghouts, G. J., and Smeulders, A. W. (2005). The amsterdam library of object images. *International Journal of Computer Vision*, 61(1):103–112. 77

[Gibson, 1979] Gibson, J. J. (1979). *The Ecological Approach to Visual Perception*. Houghton Mifflin. 15

[Girshick et al., 2014] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 6, 71

[Goehring et al., 2014] Goehring, D., Hoffman, J., Rodner, E., Saenko, K., and Darrell, T. (2014). Interactive adaptation of real-time object detectors. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1282–1289. IEEE. 98

[Golub and Loan, 1996] Golub, G. H. and Loan, C. F. V. (1996). *Matrix Computations*. 161

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep learning. Book in preparation for MIT Press. 42, 58

[Goodfellow et al., 2009] Goodfellow, I., Lee, H., Le, Q. V., Saxe, A., and Ng, A. Y. (2009). Measuring invariances in deep networks. In Bengio, Y., Schuurmans, D., Lafferty, J. D., Williams, C. K. I., and Culotta, A., editors, *Advances in Neural Information Processing Systems 22*, pages 646–654. Curran Associates, Inc. 68, 135

[Goodfellow et al., 2013a] Goodfellow, I., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. (2013a). Maxout networks. In *Proceedings of The 30th International Conference on Machine Learning*, pages 1319–1327. 155

[Goodfellow et al., 2013b] Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A., and Bengio, Y. (2013b). An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*. 153

[Gordo et al., 2016] Gordo, A., Almazán, J., Revaud, J., and Larlus, D. (2016). *Deep Image Retrieval: Learning Global Representations for Image Search*, pages 241–257. Springer International Publishing, Cham. 110, 135

[Gordon and Lowe, 2006] Gordon, I. and Lowe, D. G. (2006). *What and Where: 3D Object Recognition with Accurate Pose*, pages 67–82. Springer Berlin Heidelberg, Berlin, Heidelberg. 14

[Gorges and Navarro, 2010] Gorges, N. and Navarro, S. E. (2010). Haptic Object Recognition using Passive Joints and Haptic Key Features. pages 2349–2355. 15

[Goroshin et al., 2015a] Goroshin, R., Bruna, J., Tompson, J., Eigen, D., and LeCun, Y. (2015a). Unsupervised learning of spatiotemporally coherent metrics. In *The IEEE International Conference on Computer Vision (ICCV)*. 191

[Goroshin et al., 2015b] Goroshin, R., Mathieu, M. F., and LeCun, Y. (2015b). Learning to linearize under uncertainty. In *Advances in Neural Information Processing Systems*, pages 1234–1242. 191

[Griffin et al., 2007] Griffin, G., Holub, A., and Perona, P. (2007). Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology. 10, 72, 76

[Hardt et al., 2016] Hardt, M., Recht, B., and Singer, Y. (2016). Train faster, generalize better: Stability of stochastic gradient descent. *International Conference on Machine Learning*. 155

[He and Garcia, 2009] He, H. and Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*. 154, 156

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *The IEEE International Conference on Computer Vision (ICCV)*. 13, 44, 50, 51

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 12, 54, 62

[Held et al., 2016] Held, D., Thrun, S., and Savarese, S. (2016). Robust single-view instance recognition. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2152–2159. 16, 131, 136, 147, 148, 149, 150, 151

[Herranz et al., 2016] Herranz, L., Jiang, S., and Li, X. (2016). Scene Recognition With CNNs: Objects, Scales and Dataset Bias. In *Conference on Computer Vision and Pattern Recognition*, pages 571–579. 103, 117

[Higy et al., 2016] Higy, B., Ciliberto, C., Rosasco, L., and Natale, L. (2016). Combining sensory modalities and exploratory procedures to improve haptic object recognition in robotics. *IEEE-RAS International Conference on Humanoid Robots*. 15, 191

[Hinterstoisser et al., 2011] Hinterstoisser, S., Holzer, S., Cagniart, C., Ilic, S., Konolige, K., Navab, N., and Lepetit, V. (2011). Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *2011 International Conference on Computer Vision*, pages 858–865. 14

[Hinterstoisser et al., 2010] Hinterstoisser, S., Lepetit, V., Ilic, S., Fua, P., and Navab, N. (2010). Dominant orientation templates for real-time detection of texture-less objects.

In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2257–2264. 14

[Hinton et al., 2012] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. 62

[Hoffman et al., 2013] Hoffman, J., Tzeng, E., Donahue, J., Jia, Y., Saenko, K., and Darrell, T. (2013). One-Shot Adaptation of Supervised Deep Convolutional Models. 98, 103

[Högman et al., 2016] Högman, V., Björkman, M., Maki, A., and Kragic, D. (2016). A Sensorimotor Learning Framework for Object Categorization. *IEEE Transactions on Cognitive and Developmental Systems*, 8(1):15–25. 191

[Hosoda and Iwase, 2010] Hosoda, K. and Iwase, T. (2010). Robust haptic recognition by anthropomorphic bionic hand through dynamic interaction. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1236–1241. IEEE. 15

[Hsiao and Hebert, 2014] Hsiao, E. and Hebert, M. (2014). Occlusion reasoning for object detection under arbitrary viewpoint. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(9):1803–1815. 14

[Hu, 1962] Hu, M.-K. (1962). Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, 8(2):179–187. 10

[Hubel and Wiesel, 1959] Hubel, D. H. and Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, 148(3):574–591. 46

[Hubel and Wiesel, 1962] Hubel, D. H. and Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160(1):106–154. 45, 46, 49

[Huber and Hebert, 2003] Huber, D. F. and Hebert, M. (2003). Fully automatic registration of multiple 3d data sets. *Image and Vision Computing*, 21(7):637 – 650. Computer Vision beyond the visible spectrum. 14

[Huh et al., 2016] Huh, M., Agrawal, P., and Efros, A. A. (2016). What makes ImageNet good for transfer learning? 72

[Hung et al., 2005] Hung, C. P., Kreiman, G., Poggio, T., and DiCarlo, J. J. (2005). Fast readout of object identity from macaque inferior temporal cortex. *Science*, 310(5749):863–866. 49

[Huttenlocher et al., 1993] Huttenlocher, D. P., Klanderman, G. A., and Rucklidge, W. J. (1993). Comparing images using the hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–863. 14

[Ioffe and Szegedy, 2015] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. volume 37, pages 448—456. 62

[Itti, 2000] Itti, L. (2000). *Models of bottom-up and top-down visual attention*. PhD thesis, California Institute of Technology. 5

[Jacob et al., 2008] Jacob, L., Bach, F., and Vert, J.-P. (2008). Clustered multi-task learning: a convex formulation. *Advances in Neural Information Processing Systems*. 192

[Jain et al., 2014] Jain, L. C., Seera, M., Lim, C. P., and Balasubramaniam, P. (2014). A review of online learning in supervised neural networks. *Neural Computing and Applications*. 155

[Jamali et al., 2016] Jamali, N., Ciliberto, C., Rosasco, L., and Natale, L. (2016). Active perception: Building objects' models using tactile exploration. *IEEE-RAS International Conference on Humanoid Robots*. 15

[Jarrett et al., 2009] Jarrett, K., Kavukcuoglu, K., Ranzato, M., and LeCun, Y. (2009). What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th International Conference on Computer Vision*, pages 2146–2153. 41, 51

[Jayaraman and Grauman, 2015] Jayaraman, D. and Grauman, K. (2015). Learning image representations tied to ego-motion. In *The IEEE International Conference on Computer Vision (ICCV)*. 16, 191

[Jeannerod, 1997] Jeannerod, M. (1997). *The cognitive neuroscience of action.* Blackwell Publishing. 15

[Jegou et al., 2010] Jegou, H., Douze, M., Schmid, C., and Perez, P. (2010). Aggregating local descriptors into a compact image representation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3304–3311. 40

[Jia et al., 2012] Jia, Y., Huang, C., and Darrell, T. (2012). Beyond spatial pyramids: Receptive field learning for pooled image features. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3370–3377. 41

[Jia et al., 2014] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature

embedding. In *Proceedings of the ACM International Conference on Multimedia - MM '14*, pages 675–678. ACM Press. 52, 61, 70, 99

[Joachims et al., 2009] Joachims, T., Hofmann, T., Yue, Y., and Yu, C.-N. (2009). Predicting structured objects with support vector machines. *Commun. ACM*, 52(11):97–104. 192

[Käding et al., 2016] Käding, C., Rodner, E., Freytag, A., and Denzler, J. (2016). Fine-tuning deep neural networks in continuous learning scenarios. In *ACCV Workshop on Interpretation and Visualization of Deep Neural Nets (ACCV-WS)*. 168

[Kaltenbach et al., 1992] Kaltenbach, J. A., Czaja, J. M., and Kaplan, C. R. (1992). Changes in the tonotopic map of the dorsal cochlear nucleus following induction of cochlear lesions by exposure to intense sound. *Hearing research*, 59(2):213–223. 46

[Kasper et al., 2012] Kasper, A., Xue, Z., and Dillmann, R. (2012). The kit object models database: An object model database for object recognition, localization and manipulation in service robotics. *The International Journal of Robotics Research*, 31(8):927–934. 77

[Ke and Sukthankar, 2004] Ke, Y. and Sukthankar, R. (2004). Pca-sift: a more distinctive representation for local image descriptors. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–506–II–513 Vol.2. 39

[Keller and Lohan, 2016] Keller, I. and Lohan, K. S. (2016). Analysis of illumination robustness in long-term object learning. In *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 240–245. 84

[Kemp et al., 2007] Kemp, C. C., Edsinger, A., and Torres-Jara, E. (2007). Challenges for robot manipulation in human environments [grand challenges of robotics]. *IEEE Robotics Automation Magazine*, 14(1):20–29. 2

[Khosla et al., 2012] Khosla, A., Zhou, T., Malisiewicz, T., Efros, A. A., and Torralba, A. (2012). *Undoing the Damage of Dataset Bias*, pages 158–171. Springer Berlin Heidelberg, Berlin, Heidelberg. 102

[Killackey et al., 1995] Killackey, H. P., Rhoades, R. W., and Bennett-Clarke, C. A. (1995). The formation of a cortical somatotopic map. *Trends in neurosciences*, 18(9):402–407. 46

[Kingma and Ba, 2015] Kingma, D. and Ba, J. (2015). Adam: A Method for Stochastic Optimization. In *3rd International Conference for Learning Representations (ICLR)*. 36, 125

[Koubaroulis et al., 2002] Koubaroulis, D., Matas, J., and Kittler, J. (2002). Evaluating colour-based object recognition algorithms using the soil-47 database. In *in Asian Conference on Computer Vision*, pages 840–845. 77

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc. 12, 44, 50, 51, 52, 53, 62, 95

[Kulkarni et al., 2014] Kulkarni, T. D., Mansinghka, V. K., Kohli, P., and Tenenbaum, J. B. (2014). Inverse graphics with probabilistic cad models. *arXiv preprint arXiv:1407.1339*. 190

[Kulkarni et al., 2015] Kulkarni, T. D., Whitney, W. F., Kohli, P., and Tenenbaum, J. (2015). Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems*, pages 2539–2547. 190

[Kumar et al., 2015] Kumar, S., Odone, F., Noceti, N., and Natale, L. (2015). Object segmentation using independent motion detection. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 94–100. 6

[Kuzborskij et al., 2013] Kuzborskij, I., Orabona, F., and Caputo, B. (2013). From n to n+ 1: Multiclass transfer incremental learning. In *Computer Vision and Pattern Recognition (CVPR)*. 155, 169, 191

[Lai et al., 2011] Lai, K., Bo, L., Ren, X., and Fox, D. (2011). A large-scale hierarchical multi-view rgb-d object dataset. In *2011 IEEE International Conference on Robotics and Automation*, pages 1817–1824. 9, 14, 19, 68, 77, 85, 97, 128, 129, 130, 134, 147, 154, 164

[Laskov et al., 2006] Laskov, P., Gehl, C., Krüger, S., and Müller, K.-R. (2006). Incremental support vector learning: Analysis, implementation and applications. *Journal of machine learning research*. 155

[Lazebnik et al., 2006] Lazebnik, S., Schmid, C., and Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2169–2178. 41

[LeCun et al., 1989] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551. 12, 41, 50, 58

[LeCun et al., 1998] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324. 10, 21, 41, 50, 52, 58, 61, 154, 164, 167

[LeCun et al., 2004] LeCun, Y., Huang, F. J., and Bottou, L. (2004). Learning methods for generic object recognition with invariance to pose and lighting. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–97–104 Vol.2. 88, 135

[Lee et al., 2006] Lee, H., Battle, A., Raina, R., and Ng, A. Y. (2006). Efficient sparse coding algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, pages 801–808. 41

[Leitner et al., 2014] Leitner, J., Förster, A., and Schmidhuber, J. (2014). Improving robot vision models for object detection through interaction. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pages 3355–3362. IEEE. 15

[Lenc and Vedaldi, 2015] Lenc, K. and Vedaldi, A. (2015). Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 67, 68

[Lenz et al., 2015] Lenz, I., Lee, H., and Saxena, A. (2015). Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5):705–724. 16

[Levine et al., 2016a] Levine, S., Finn, C., Darrell, T., and Abbeel, P. (2016a). End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40. 16

[Levine et al., 2016b] Levine, S., Pastor, P., Krizhevsky, A., and Quillen, D. (2016b). Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection. *arXiv:1603.02199 [cs]*. arXiv: 1603.02199. 16, 78, 189

[Lichtsteiner et al., 2008] Lichtsteiner, P., Posch, C., and Delbruck, T. (2008). A 128x128 120db 15us latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid-State Circuits*, 43(2):566–576. 185

[Lin et al., 2016] Lin, J., Camoriano, R., and Rosasco, L. (2016). Generalization properties and implicit regularization for multiple passes sgm. *International Conference on Machine Learning*. 155

[Lin et al., 2013] Lin, M., Chen, Q., and Yan, S. (2013). Network In Network. *ArXiv e-prints*. 54

[Lin et al., 2014] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). *Microsoft COCO: Common Objects in Context*, pages 740–755. Springer International Publishing, Cham. 77

[Logothetis et al., 1995] Logothetis, N. K., Pauls, J., and Poggio, T. (1995). Shape representation in the inferior temporal cortex of monkeys. *Current Biology*, 5(5):552–563. 49

[Lowe, 2004] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110. 11, 14, 39

[Lozano and Sindhwani, 2011] Lozano, A. and Sindhwani, V. (2011). Block variable selection in multivariate regression and high-dimensional causal inference. *Advances in Neural Information Processing Systems*. 192

[Luo et al., 2007] Luo, J., Pronobis, A., Caputo, B., and Jensfelt, P. (2007). Incremental learning for place recognition in dynamic environments. In *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'07)*, pages 721–728, San Diego, CA, USA. 86

[Mahendran and Vedaldi, 2015] Mahendran, A. and Vedaldi, A. (2015). Understanding deep image representations by inverting them. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 39, 66

[Mahmood et al., 2016] Mahmood, A., Bennamoun, M., An, S., and Sohel, F. (2016). ResFeats: Residual Network Based Features for Image Classification. *ArXiv e-prints*. 71

[Malach et al., 2002] Malach, R., Levy, I., and Hasson, U. (2002). The topography of high-order human object areas. *Trends in Cognitive Sciences*, 6(4):176 – 184. 49

[Mansinghka et al., 2013] Mansinghka, V., Kulkarni, T. D., Perov, Y. N., and Tenenbaum, J. (2013). Approximate bayesian image interpretation using generative probabilistic graphics programs. In *Advances in Neural Information Processing Systems*, pages 1520–1528. 190

[Marĉelja, 1980] Marĉelja, S. (1980). Mathematical description of the responses of simple cortical cells∗. *J. Opt. Soc. Am.*, 70(11):1297–1300. 11

[Marr, 1982] Marr, D. (1982). *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. Henry Holt and Co., Inc., New York, NY, USA. 10

[Marr and Hildreth, 1980] Marr, D. and Hildreth, E. (1980). Theory of edge detection. *Proceedings of the Royal Society of London B: Biological Sciences*, 207(1167):187–217. 11

[McCulloch and Pitts, 1943] McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133. 43

[Meger and Little, 2013] Meger, D. and Little, J. J. (2013). *The UBC Visual Robot Survey: A Benchmark for Robot Category Recognition*, pages 979–991. Springer International Publishing, Heidelberg. 78

[Metta et al., 2006a] Metta, G., Fitzpatrick, P., and Natale, L. (2006a). Yarp: yet another robot platform. *International Journal on Advanced Robotics Systems*, 3(1):43–48. 173, 186

[Metta et al., 2010] Metta, G., Natale, L., Nori, F., Sandini, G., Vernon, D., Fadiga, L., von Hofsten, C., Rosander, K., Lopes, M., Santos-Victor, J., Bernardino, A., and Montesano, L. (2010). The icub humanoid robot: an open-systems platform for research in cognitive development. *Neural networks : the official journal of the International Neural Network Society*, 23(8-9):1125–34. 75, 172

[Metta et al., 2006b] Metta, G., Sandini, G., Natale, L., Craighero, L., and Fadiga, L. (2006b). Understanding mirror neurons: a bio-robotic approach. *Interaction studies*, 7(2):197–232. 15

[Metta et al., 2008] Metta, G., Sandini, G., Vernon, D., Natale, L., and Nori, F. (2008). The icub humanoid robot: An open platform for research in embodied cognition. In *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*, PerMIS '08, pages 50–56, New York, NY, USA. ACM. 75, 172

[Micchelli and Pontil, 2004] Micchelli, C. A. and Pontil, M. (2004). Kernels for multi-task learning. *Advances in Neural Information Processing Systems*. 192

[Mikolajczyk and Schmid, 2004] Mikolajczyk, K. and Schmid, C. (2004). Scale & affine invariant interest point detectors. *International journal of computer vision*, 60(1):63–86. 39

[Mikolajczyk and Schmid, 2005] Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630. 11

[Miller, 1995] Miller, G. A. (1995). Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41. 8

[Milner and Goodale, 2006] Milner, D. and Goodale, M. (2006). *The visual brain in action*. Oxford University Press. 4

[Minh and Sindhwani, 2011] Minh, H. Q. and Sindhwani, V. (2011). Vector-valued manifold regularization. *International Conference on Machine Learning*. 192

[Mishkin et al., 1983] Mishkin, M., Ungerleider, L. G., and Macko, K. A. (1983). Object vision and spatial vision: two cortical pathways. *Trends in Neurosciences*, 6:414–417. 3

[Mnih et al., 2013] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing Atari with Deep Reinforcement Learning. *ArXiv e-prints*. 13

[Mnih et al., 2015] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533. 13

[Model and Shamir, 2015] Model, I. and Shamir, L. (2015). Comparison of Data Set Bias in Object Recognition Benchmarks. *IEEE Access*, 3:1953–1962. 77, 98, 102

[Montesano et al., 2008] Montesano, L., Lopes, M., Bernardino, A., and Santos-Victor, J. (2008). Learning object affordances: From sensory–motor coordination to imitation. *IEEE Transactions on Robotics*, 24(1):15–26. 191

[Moreels and Perona, 2007] Moreels, P. and Perona, P. (2007). Evaluation of features detectors and descriptors based on 3d objects. *International Journal of Computer Vision*, 73(3):263–284. 14

[Muja et al., 2011] Muja, M., Rusu, R. B., Bradski, G., and Lowe, D. G. (2011). Rein - a fast, robust, scalable recognition infrastructure. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2939–2946. 14, 109

[Murphy, 2012] Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. 30

[Mutch and Lowe, 2008] Mutch, J. and Lowe, D. G. (2008). Object class recognition and localization using sparse features with limited receptive fields. *International Journal of Computer Vision*, 80(1):45–57. 4, 49, 65

[Natale et al., 2004] Natale, L., Metta, G., and Sandini, G. (2004). Learning haptic representation of objects. 15

[Nemirovskii et al., 1983] Nemirovskii, A., Yudin, D. B., and Dawson, E. R. (1983). Problem complexity and method efficiency in optimization. 34

[Nene et al., 1996] Nene, S. A., Nayar, S. K., and Murase, H. (1996). Columbia object image library (coil-100. Technical report. 77

[Netzer et al., 2011] Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*. 13

[Newcombe et al., 2011] Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohi, P., Shotton, J., Hodges, S., and Fitzgibbon, A. (2011). Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136. 14

[Nguyen et al., 2016] Nguyen, A., Kanoulas, D., Caldwell, D. G., and Tsagarakis, N. G. (2016). Detecting object affordances with convolutional neural networks. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2765–2770. 16

[Nguyen et al., 2015] Nguyen, A., Yosinski, J., and Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 427–436. 66

[Oberlin et al., 2015] Oberlin, J., Meier, M., Kraska, T., and Tellex, S. (2015). Acquiring Object Experiences at Scale. In *AAAI-RSS Special Workshop on the 50th Anniversary of Shakey: The Role of AI to Harmonize Robots and Humans*. Blue Sky Award. 78, 79, 189

[Oquab et al., 2014] Oquab, M., Bottou, L., Laptev, I., and Sivic, J. (2014). Learning and transferring mid-level image representations using convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 72

[Papazov and Burschka, 2011] Papazov, C. and Burschka, D. (2011). *An Efficient RANSAC for 3D Object Recognition in Noisy and Occluded Scenes*, pages 135–148. Springer Berlin Heidelberg, Berlin, Heidelberg. 14

[Part and Lemon, 2016] Part, J. L. and Lemon, O. (2016). Incremental on-line learning of object classes using a combination of self-organizing incremental neural networks and deep convolutional neural networks. In *Workshops of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 84

[Pasquale et al., 2015a] Pasquale, G., Ciliberto, C., Odone, F., Rosasco, L., and Natale, L. (2015a). Real-world Object Recognition with Off-the-shelf Deep Conv Nets: How Many Objects can iCub Learn? *ArXiv*, abs/1504.03154. 84, 95, 96, 112, 167

[Pasquale et al., 2015b] Pasquale, G., Ciliberto, C., Odone, F., Rosasco, L., and Natale, L. (2015b). Teaching iCub to recognize objects using deep Convolutional Neural Networks. volume 43, pages 21–25. 84, 95, 96, 97, 112, 164, 167, 182, 190

[Pasquale et al., 2016a] Pasquale, G., Ciliberto, C., Rosasco, L., and Natale, L. (2016a). Object identification from few examples by improving the invariance of a deep convolutional neural network. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4904–4911. 133, 134, 139, 146, 150, 151

[Pasquale et al., 2017] Pasquale, G., Ciliberto, C., Rosasco, L., and Natale, L. (2017). Are we Done with Object Recognition? The iCub-robot Perspective. *Submitted to The International Journal of Robotics Research*. 95, 134

[Pasquale et al., 2016b] Pasquale, G., Mar, T., Ciliberto, C., Rosasco, L. A., and Natale, L. (2016b). Enabling depth-driven visual attention on the icub humanoid robot: Instructions for use and new perspectives. *Frontiers in Robotics and AI*, 3(35). 6, 83, 177, 178

[Peng et al., 2015] Peng, X., Sun, B., Ali, K., and Saenko, K. (2015). Exploring invariances in deep convolutional neural networks using synthetic images. In *International Conference on Learning Representations (Workshop track)*. 68, 88, 135

[Perronnin et al., 2010] Perronnin, F., Sánchez, J., and Mensink, T. (2010). *Improving the Fisher Kernel for Large-Scale Image Classification*, pages 143–156. Springer Berlin Heidelberg, Berlin, Heidelberg. 72

[Philbin et al., 2008] Philbin, J., Chum, O., Isard, M., Sivic, J., and Zisserman, A. (2008). Lost in quantization: Improving particular object retrieval in large scale image databases. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE. 41, 109

[Pillai and Leonard, 2015] Pillai, S. and Leonard, J. (2015). Monocular slam supported object recognition. In *Proceedings of Robotics: Science and Systems (RSS)*, Rome, Italy. 190

[Pinto et al., 2016] Pinto, L., Gandhi, D., Han, Y., Park, Y.-L., and Gupta, A. (2016). The Curious Robot: Learning Visual Representations via Physical Interactions. *arXiv:1604.01360 [cs]*. arXiv: 1604.01360. 16, 191

[Pinto and Gupta, 2016] Pinto, L. and Gupta, A. (2016). Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3406–3413. 16, 78

[Pinto et al., 2011] Pinto, N., Barhomi, Y., Cox, D. D., and DiCarlo, J. J. (2011). Comparing state-of-the-art visual features on invariant object recognition tasks. In *2011 IEEE Workshop on Applications of Computer Vision (WACV)*, pages 463–470. 65, 68, 88, 189

[Pinto et al., 2008] Pinto, N., Cox, D. D., and DiCarlo, J. J. (2008). Why is real-world visual object recognition hard? *PLoS computational biology*, 4(1):e27. i, 2, 51, 65, 77, 98, 103, 189

[Pirsiavash and Ramanan, 2012] Pirsiavash, H. and Ramanan, D. (2012). Detecting activities of daily living in first-person camera views. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2847–2854. IEEE. 85

[Poggio and Anselmi, 2016] Poggio, T. and Anselmi, F. (2016). *Visual Cortex and Deep Networks: Learning Invariant Representations*. The MIT Press, Cambridge, MA, USA. 39, 49, 65, 189

[Poggio et al., 1985] Poggio, T., Torre, V., and Koch, C. (1985). Computational vision and regularization theory. *Nature*, 317(6035):314–319. 10

[Polyak and Juditsky, 1992] Polyak, B. T. and Juditsky, A. B. (1992). Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855. 63

[Prankl et al., 2015] Prankl, J., Aldoma, A., Svejda, A., and Vincze, M. (2015). Rgb-d object modelling for object recognition and tracking. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 96–103. 14

[Quadros et al., 2012] Quadros, A., Underwood, J. P., and Douillard, B. (2012). An occlusion-aware feature for range images. In *2012 IEEE International Conference on Robotics and Automation*, pages 4428–4435. 14

[Quigley et al., 2009] Quigley, M., Faust, J., Foote, T., and Leibs, J. (2009). Ros: an open-source robot operating system. 186

[Ramisa et al., 2011] Ramisa, A., Aldavert, D., Vasudevan, S., Toledo, R., and Lopez de Mantaras, R. (2011). The iiia30 mobile robot object recognition dataset. In *Robotica 2011*. 78

[Redmon and Angelova, 2015] Redmon, J. and Angelova, A. (2015). Real-time grasp detection using convolutional neural networks. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1316–1322. 16, 190

[Rennie et al., 2016] Rennie, C., Shome, R., Bekris, K. E., and Ferreira De Souza, A. (2016). A dataset for improved rgbd-based object detection and pose estimation for warehouse pick-and-place. *IEEE Robotics and Automation Letters (RA-L) [Also accepted to appear at the 2016 IEEE International Conference on Robotics and Automation (ICRA)]*, 1:1179 – 1185. 78

[Riesenhuber and Poggio, 1999] Riesenhuber, M. and Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nature neuroscience*, 2(11):1019–25. 4, 10, 12

[Rifkin and Klautau, 2004] Rifkin, R. and Klautau, A. (2004). In defense of one-vs-all classification. *Journal of Maching Learning Research*, 5:101–141. 27

[Rifkin et al., 2003] Rifkin, R., Yeo, G., and Poggio, T. (2003). Regularized least-squares classification. *Nato Science Series Sub Series III Computer and Systems Sciences*. 154

[Rifkin, 2002] Rifkin, R. M. (2002). *Everything old is new again: a fresh look at historical approaches in machine learning*. PhD thesis, Massachusetts Institute of Technology. 154

[Rivera-Rubio et al., 2014] Rivera-Rubio, J., Idrees, S., Alexiou, I., Hadjilucas, L., and Bharath, A. A. (2014). Small hand-held object recognition test (SHORT). In *2014 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 524–531. 77

[Roberts, 1963] Roberts, L. G. (1963). *Machine perception of three-dimensional soups*. PhD thesis, Massachusetts Institute of Technology. 10

[Rodner et al., 2013] Rodner, E., Hoffman, J., Donahue, J., Darrell, T., and Saenko, K. (2013). Towards Adapting ImageNet to Reality: Scalable Domain Adaptation with Implicit Low-rank Transformations. 98, 103, 117

[Rosasco, 2016] Rosasco, L. (2016). Introductory machine learning notes. 24, 29

[Rosenblatt, 1958] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386. 42, 44

[Rudi et al., 2015] Rudi, A., Camoriano, R., and Rosasco, L. (2015). Less is more: Nystrom computational regularization. In *Advances in Neural Information Processing Systems*, pages 1648–1656. 100, 119

[Rudi et al., 2016] Rudi, A., Camoriano, R., and Rosasco, L. (2016). Generalization properties of learning with random features. *arXiv preprint arXiv:1602.04474*. 100

[Rumelhart et al., 1988] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1. 35, 58

[Russakovsky et al., 2015] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252. 9, 12, 51, 76, 89, 95

[Russakovsky et al., 2012] Russakovsky, O., Lin, Y., Yu, K., and Fei-Fei, L. (2012). *Object-Centric Spatial Pooling for Image Classification*, pages 1–15. Springer Berlin Heidelberg, Berlin, Heidelberg. 41

[Russell et al., 2008] Russell, B. C., Torralba, A., Murphy, K. P., and Freeman, W. T. (2008). Labelme: A database and web-based tool for image annotation. *International Journal of Computer Vision*, 77(1):157–173. 76

[Rust and DiCarlo, 2010] Rust, N. C. and DiCarlo, J. J. (2010). Selectivity and tolerance ”invariance” both increase as visual information propagates from cortical area v4 to it. *Journal of Neuroscience*, 30(39):12978–12995. 49, 65

[Rusu et al., 2010] Rusu, R. B., Bradski, G., Thibaux, R., and Hsu, J. (2010). Fast 3d recognition and pose using the viewpoint feature histogram. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2155–2162. 14

[Saenko et al., 2010] Saenko, K., Kulis, B., Fritz, M., and Darrell, T. (2010). *Adapting Visual Category Models to New Domains*, pages 213–226. Springer Berlin Heidelberg, Berlin, Heidelberg. 98

[Sánchez et al., 2013] Sánchez, J., Perronnin, F., Mensink, T., and Verbeek, J. (2013). Image classification with the fisher vector: Theory and practice. *International Journal of Computer Vision*, 105(3):222–245. 40

[Sayed, 2008] Sayed, A. H. (2008). *Adaptive Filters*. Wiley-IEEE Press. 153, 155, 160

[Schneider et al., 2009] Schneider, A., Sturm, J., Stachniss, C., Reisert, M., Burkhardt, H., and Burgard, W. (2009). Object identification with tactile sensors using bag-of-features. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 243–248. 15

[Schneider, 1969] Schneider, G. E. (1969). Two Visual Systems. *Science*, 163(3870):895–902. 3

[Schwarz et al., 2015] Schwarz, M., Schulz, H., and Behnke, S. (2015). Rgb-d object recognition and pose estimation based on pre-trained convolutional neural network features. In *2015*

IEEE International Conference on Robotics and Automation (ICRA), pages 1329–1335. 16, 97, 98, 129, 130, 131, 167, 190

[Sermanet et al., 2014] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and Lecun, Y. (2014). *Overfeat: Integrated recognition, localization and detection using convolutional networks*. 6, 12, 70, 71

[Serre et al., 2007] Serre, T., Wolf, L., Bileschi, S., Riesenhuber, M., and Poggio, T. (2007). Robust object recognition with cortex-like mechanisms. *IEEE transactions on pattern analysis and machine intelligence*, 29(3):411–426. 4, 12, 15, 49

[Sharif Razavian et al., 2014] Sharif Razavian, A., Azizpour, H., Sullivan, J., and Carlsson, S. (2014). Cnn features off-the-shelf: An astounding baseline for recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 13, 70, 109, 136

[Shawe-Taylor and Cristianini, 2004a] Shawe-Taylor, J. and Cristianini, N. (2004a). *Kernel methods for pattern analysis*. Cambridge university press. 25

[Shawe-Taylor and Cristianini, 2004b] Shawe-Taylor, J. and Cristianini, N. (2004b). *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA. 30

[Shrivastava et al., 2016] Shrivastava, A., Sukthankar, R., Malik, J., and Gupta, A. (2016). Beyond Skip Connections: Top-Down Modulation for Object Detection. *ArXiv e-prints*. 6

[Silver et al., 2016] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489. 13

[Simonyan et al., 2014] Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR Workshop*. 12, 50, 53, 66

[Simonyan and Zisserman, 2015] Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*. 71

[Sinapov et al., 2014] Sinapov, J., Schenck, C., Staley, K., Sukhoy, V., and Stoytchev, A. (2014). Grounding semantic categories in behavioral interactions: Experiments with 100 objects. *Robotics and Autonomous Systems*, 62(5):632–645. 15, 191

[Sindhwani et al., 2012] Sindhwani, V., Lozano, A. C., and Minh, H. Q. (2012). Scalable matrix-valued kernel learning and high-dimensional nonlinear causal inference. *CoRR*, abs/1210.4792. 192

[Singh et al., 2014] Singh, A., Sha, J., Narayan, K. S., Achim, T., and Abbeel, P. (2014). Bigbird: A large-scale 3d database of object instances. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 509–516. 14, 77, 148

[Soekhoe et al., 2016] Soekhoe, D., van der Putten, P., and Plaat, A. (2016). *On the Impact of Data Set Size in Transfer Learning Using Deep Neural Networks*, pages 50–60. Springer International Publishing, Cham. 72

[Song et al., 2015] Song, S., Lichtenberg, S. P., and Xiao, J. (2015). Sun rgb-d: A rgb-d scene understanding benchmark suite. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 190

[Srinivas et al., 2016] Srinivas, S., Sarvadevabhatla, R. K., Mopuri, K. R., Prabhu, N., Kruthiventi, S. S. S., and Babu, R. V. (2016). A taxonomy of deep convolutional neural nets for computer vision. *Frontiers in Robotics and AI*, 2:36. 42

[Srivastava et al., 2014] Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958. 62

[Srivastava et al., 2013] Srivastava, R. K., Masci, J., Kazerounian, S., Gomez, F., and Schmidhuber, J. (2013). Compete to compute. In *Advances in neural information processing systems*. 155

[Stallkamp et al., 2012] Stallkamp, J., Schlipsing, M., Salmen, J., and Igel, C. (2012). Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 32:323 – 332. Selected Papers from {IJCNN} 2011. 13

[Stamos et al., 2015] Stamos, D., Martelli, S., Nabi, M., McDonald, A., Murino, V., and Pontil, M. (2015). Learning with dataset bias in latent subcategory models. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 07-12-June, pages 3650–3658. IEEE. 102

[Steder et al., 2011] Steder, B., Rusu, R. B., Konolige, K., and Burgard, W. (2011). Point feature extraction on 3d range scans taking into account object boundaries. In *2011 IEEE International Conference on Robotics and Automation*, pages 2601–2608. 14

[Steinwart and Christmann, 2008] Steinwart, I. and Christmann, A. (2008). *Support vector machines*. Springer Science & Business Media. 24, 25, 26, 154, 156, 157

[Sturm et al., 2013] Sturm, J., Bylow, E., Kahl, F., and Cremers, D. (2013). Copyme3d: Scanning and printing persons in 3d. In *German Conference on Pattern Recognition (GCPR)*. 14

[Sun and Fox, 2016] Sun, Y. and Fox, D. (2016). NEOL : Toward Never-Ending Object Learning for Robots. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1621–1627. 155, 169, 191

[Sünderhauf et al., 2016] Sünderhauf, N., Dayoub, F., McMahon, S., Talbot, B., Schulz, R., Corke, P., Wyeth, G., Upcroft, B., and Milford, M. (2016). Place categorization and semantic mapping on a mobile robot. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5729–5736. 16, 97

[Sünderhauf et al., 2015] Sünderhauf, N., Shirazi, S., Dayoub, F., Upcroft, B., and Milford, M. (2015). On the performance of convnet features for place recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 4297–4304. 16, 97

[Sung et al., 2016] Sung, J., Jin, S. H., Lenz, I., and Saxena, A. (2016). Robobarista: Learning to manipulate novel objects via deep multimodal embedding. *CoRR*, abs/1601.02705. 15

[Szegedy et al., 2015] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 12, 53, 54, 124

[Szegedy et al., 2014] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2014). Intriguing properties of neural networks. In *International Conference on Learning Representations*. 66

[Tang et al., 2012] Tang, J., Miller, S., Singh, A., and Abbeel, P. (2012). A textured object recognition pipeline for color and depth image data. In *2012 IEEE International Conference on Robotics and Automation*, pages 3467–3474. 14

[Taylor and Kleeman, 2003] Taylor, G. and Kleeman, L. (2003). Fusion of multimodal visual cues for model-based object tracking. In *In Australasian Conference on Robotics and Automation (ACRA2003), Brisbane,Australia*. 14

[Thrun, 1996] Thrun, S. (1996). Is learning the n-th thing any easier than learning the first? *Advances in neural information processing systems*. 155, 191

[Tieleman and Hinton, 2012] Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. Technical report. 36

[Tombari et al., 2010] Tombari, F., Salti, S., and Di Stefano, L. (2010). *Unique Signatures of Histograms for Local Surface Description*, pages 356–369. Springer Berlin Heidelberg, Berlin, Heidelberg. 14

[Tommasi et al., 2016] Tommasi, T., Lanzi, M., Russo, P., and Caputo, B. (2016). *Learning the Roots of Visual Domain Shift*, pages 475–482. Springer International Publishing, Cham. 98

[Tommasi et al., 2010] Tommasi, T., Orabona, F., and Caputo, B. (2010). Safety in numbers: Learning categories from few examples with multi model knowledge transfer. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE. 155, 156, 169, 191

[Tommasi et al., 2012] Tommasi, T., Orabona, F., Kaboli, M., and Caputo, B. (2012). Leveraging over prior knowledge for online learning of visual categories. In *BMVC*. 155, 169

[Tommasi et al., 2015] Tommasi, T., Patricia, N., Caputo, B., and Tuytelaars, T. (2015). A deeper look at dataset bias. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9358, pages 504–516. Springer International Publishing. 98, 103

[Torralba and Efros, 2011] Torralba, A. and Efros, A. A. (2011). Unbiased look at dataset bias. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1521–1528. 77, 98, 102

[Tsagarakis et al., 2009] Tsagarakis, N. G., Vanderborght, B., Laffranchi, M., and Caldwell, D. G. (2009). The mechanical design of the new lower body for the child humanoid robot. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4962–4968. 172

[Uijlings et al., 2013] Uijlings, J. R. R., van de Sande, K. E. A., Gevers, T., and Smeulders, A. W. M. (2013). Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171. 6

[Ungerleider and Haxby, 1994] Ungerleider, L. G. and Haxby, J. V. (1994). 'What' and 'where' in the human brain. 3

[Vaillant et al., 1994] Vaillant, R., Monrocq, C., and Cun, Y. L. (1994). Original approach for the localisation of objects in images. *IEE Proceedings - Vision, Image and Signal Processing*, 141(4):245–250. 6

[van Polanen and Davare, 2015] van Polanen, V. and Davare, M. (2015). Interactions between dorsal and ventral streams for controlling skilled grasp. *Neuropsychologia*, (79):186–191. 4

[Vapnik, 1998] Vapnik, V. (1998). *Statistical learning theory*. Wiley. 9, 10, 24

[Vezzani et al., 2016] Vezzani, G., Jamali, N., Pattacini, U., Battistelli, G., Chisci, L., and Natale, L. (2016). A Novel Bayesian Filtering Approach to Tactile Object Recognition. *IEEE-RAS International Conference on Humanoid Robots*. 15

[Viola and Jones, 2001] Viola, P. and Jones, M. (2001). Robust real-time object detection. In *International Journal of Computer Vision*. 11

[Wager et al., 2013] Wager, S., Wang, S., and Liang, P. S. (2013). Dropout training as adaptive regularization. In *Advances in neural information processing systems*, pages 351–359. 62

[Wan et al., 2013] Wan, L., Zeiler, M., Zhang, S., Cun, Y. L., and Fergus, R. (2013). Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1058–1066. 62

[Wang et al., 2010] Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., and Gong, Y. (2010). Locality-constrained linear coding for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3360–3367. 41

[Wang and Gupta, 2015] Wang, X. and Gupta, A. (2015). Unsupervised learning of visual representations using videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802. 15, 191

[Werbos, 1974] Werbos, P. (1974). Beyond regression: New tools for prediction and analysis in the behavioral sciences. 58

[Wohlkinger et al., 2012] Wohlkinger, W., Aldoma Buchaca, A., Rusu, R., and Vincze, M. (2012). 3DNet: Large-Scale Object Class Recognition from CAD Models. In *IEEE International Conference on Robotics and Automation (ICRA)*. 77

[Xiang et al., 2014] Xiang, Y., Mottaghi, R., and Savarese, S. (2014). Beyond pascal: A benchmark for 3d object detection in the wild. In *IEEE Winter Conference on Applications of Computer Vision*, pages 75–82. 68

[Xiao et al., 2010] Xiao, J., Hays, J., Ehinger, K. A., Oliva, A., and Torralba, A. (2010). Sun database: Large-scale scene recognition from abbey to zoo. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3485–3492. 9

[Xiao et al., 2014] Xiao, T., Zhang, J., Yang, K., Peng, Y., and Zhang, Z. (2014). Error-driven incremental learning in deep convolutional neural network for large-scale image classification. In *Proceedings of the 22nd ACM international conference on Multimedia*. ACM. 155

[Xie et al., 2013] Xie, Z., Singh, A., Uang, J., Narayan, K. S., and Abbeel, P. (2013). Multi-modal blending for high-accuracy instance recognition. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2214–2221. 14

[Yamins and DiCarlo, 2016] Yamins, D. L. and DiCarlo, J. J. (2016). Using goal-driven deep learning models to understand sensory cortex. *Nature neuroscience*, 19(3):356–365. 49, 50

[Yamins et al., 2014] Yamins, D. y. K., Hong, H., Cadieu, C. F., Solomon, E. A., Seibert, D., and DiCarlo, J. J. (2014). Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the National Academy of Sciences of the United States of America*. 50

[Yang et al., 2009] Yang, J., Yu, K., Gong, Y., and Huang, T. (2009). Linear spatial pyramid matching using sparse coding for image classification. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1794–1801. 41

[Yosinski et al., 2014a] Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014a). How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, pages 3320–3328. 73

[Yosinski et al., 2014b] Yosinski, J., Clune, J., Fuchs, T., and Lipson, H. (2014b). Understanding neural networks through deep visualization. In *ICML Workshop on Deep Learning*. 50, 66

[Zeiler and Fergus, 2014] Zeiler, M. D. and Fergus, R. (2014). *Visualizing and Understanding Convolutional Networks*, pages 818–833. Springer International Publishing, Cham. 50, 66, 67, 68

[Zhang et al., 2016] Zhang, F., Leitner, J., Upcroft, B., and Corke, P. (2016). Vision-Based Reaching Using Modular Deep Networks: from Simulation to the Real World. *ArXiv e-prints*. 16

[Zhao et al., 2016] Zhao, J., Chang, C.-k., and Itti, L. (2016). Learning to Recognize Objects by Retaining other Factors of Variation. *ArXiv e-prints*. 189

[Zhao and Itti, 2016] Zhao, J. and Itti, L. (2016). Improved Deep Learning of Object Category using Pose Information. *ArXiv e-prints*. 189

[Zhou et al., 2010] Zhou, X., Yu, K., Zhang, T., and Huang, T. S. (2010). *Image Classification Using Super-Vector Coding of Local Image Descriptors*, pages 141–154. Springer Berlin Heidelberg, Berlin, Heidelberg. 40