UNIVERSITY OF GENOVA

PHD PROGRAM IN BIOENGINEERING AND ROBOTICS

From Constraints to Opportunities: Efficient Object Detection Learning for Humanoid Robots

by

Elisa Maiettini

Thesis submitted for the degree of *Doctor of Philosophy* (32° cycle)

Lorenzo Natale	Supervisor
Lorenzo Rosasco	Supervisor
Giorgio Cannata	Head of the PhD program

Thesis Jury:

Elisa Ricci, *University of Trento, Italy*	External reviewer
Javier Civera, *University of Zaragoza, Spain*	External reviewer
Antonios Gasteratos, *Democritus University of Thrace, Greece*	External reviewer
Ryad Benosman, *University of Pittsburgh, United States*	External reviewer

Dibris

Department of Informatics, Bioengineering, Robotics and Systems Engineering

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Elisa Maiettini
March 2020

Abstract

Reliable perception and efficient adaptation to novel conditions are priority skills for robots that function in ever-changing environments. Indeed, autonomously operating in real world scenarios raises the need of identifying different context's states and act accordingly. Moreover, the requested tasks might not be known a-priori, requiring the system to update on-line. Robotic platforms allow to gather various types of perceptual information due to the multiple sensory modalities they are provided with. Nonetheless, latest results in computer vision motivate a particular interest in visual perception. Specifically, in this thesis, I mainly focused on the object detection task since it can be at the basis of more sophisticated capabilities.

The vast advancements in latest computer vision research, brought by deep learning methods, are appealing in a robotic setting. However, their adoption in applied domains is not straightforward since adapting them to new tasks is strongly demanding in terms of annotated data, optimization time and computational resources. These requirements do not generally meet current robotics constraints. Nevertheless, robotic platforms and especially humanoids present opportunities that can be exploited. The sensors they are provided with represent precious sources of additional information. Moreover, their embodiment in the workspace and their motion capabilities allow for a natural interaction with the environment.

Motivated by these considerations, in this Ph.D project, I mainly aimed at devising and developing solutions able to integrate the worlds of computer vision and robotics, by focusing on the task of object detection. Specifically, I dedicated a large amount of effort in alleviating state-of-the-art methods requirements in terms of annotated data and training time, preserving their accuracy by exploiting robotics opportunity.

Table of contents

List of figures

List of tables

Part I

Introduction

Chapter 1

Motivation and scientific objectives

Fictional movies and books have created a very high expectation on robots capabilities. However, for a long time, the only reliable robotic platforms have mostly been the robotic arms, used in controlled industrial environments. Indeed, functioning in an uncontrolled and continuously changing scenario with eventual human presence, is not trivial for platforms that need to be programmed. Nevertheless, in the last decade, research has made great progress and robotic devices are becoming more popular with a significantly growing impact in everyday life.

One of the crucial skills of robots that need to interact within a dynamic and unconstrained environment is perception. In fact, a device that works in fixed conditions can operate almost blindly following a pre-scripted set of movements to accomplish the required tasks, while in a continuously changing scenario, the robot needs to perceive the environment in order to understand its state. Moreover, reliable accuracy and fast adaptation of the perception system represent two very important requirements in this setting (refer to Fig. 1.1 for examples of future real world use-case scenarios).

While robotic platforms, especially humanoids Metta et al. (2010); Parmiggiani et al. (2017), are provided with multiple sensory modalities, latest results in computer vision, pushed by the advancements of deep learning He et al. (2017, 2015); Redmon et al. (2016); Ren et al. (2015); Russakovsky et al. (2015); Shelhamer et al. (2017), motivate a particular interest in visual perception systems. However, even though state-of-the-art solutions for computer vision tasks perform stunningly well, their applicability to robotics comes with some difficulties Sunderhauf et al. (2018). Indeed, deep learning methods, especially the ones lying in the supervised learning framework, are strongly demanding in terms of the amount of annotated data, training time and computation required, while robotics has a very strict "*budget*" for all of these resources.

Figure 1.1: Pictures exemplifying future real world use-case scenarios of robotic platforms.

Therefore, the main objectives of this Ph.D. project have been in the direction of aiming to integrate these two worlds, by finding solutions that allow to have state-of-the art performance within the constrains imposed by robotics, considering the specific task of object detection.

1.1 Supervised learning framework for vision tasks

The *supervised learning* framework has proved to be effective in a wide variety of tasks and specifically brought great results in the field of computer vision. The main concept behind this framework is that of finding (*learning*) a mathematical model which is able to describe a certain data space, represented by a given subset of its elements, called *training set*. More specifically, the given points drawn from this data space represent associations between some samples and corresponding *labels*. The aim of the supervised learning is that of approximating the underlying mapping function such that, if new samples are drawn from the same data space (the *test set*), their labels can be *predicted* with a certain confidence. Therefore, the main assumption behind the supervised learning framework is the existence of an available training set, composed by annotated samples that represent the original data space.

The great majority of the state-of-the-art approaches which address computer vision tasks are based on a machine learning field that is called *deep learning*. This framework refers to all the methods where the model to be learned is given by multiple levels of composition of functions Goodfellow et al. (2016), obtaining so called *architectures*. The performance achieved by these methods are impressive He et al. (2017, 2015); Redmon et al. (2016); Ren et al. (2015); Russakovsky et al. (2015); Shelhamer et al. (2017), however, they are typically

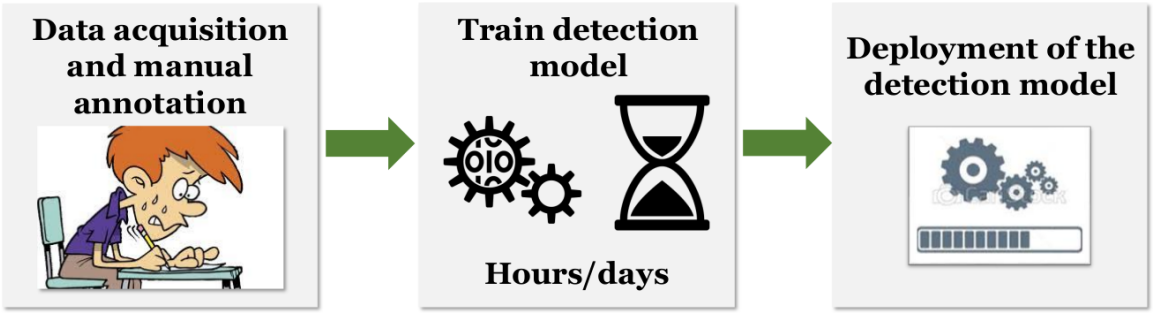


Figure 1.2: Representation of the standard pipeline used to adapt to a practical use case a state-of-the-art algorithm in computer vision.

characterized by a huge amount of parameters that need to be tuned at training time. This trait produces two main drawbacks. First of all, considering that deep learning methods are usually trained within the fully supervised learning framework, the enormous amount of parameters to tune dramatically increases the quantity of labeled data required. Secondly, the learning process used to identify the best parameters typically requires long optimization time and great computation capabilities. The extraordinary performance achieved are appealing for applied fields like robotics, however, the aforementioned requirements are strong and make their adoption difficult.

In the computer vision community, collections of annotated images for different tasks Deng et al. (2009); Everingham et al. (2010); Lin et al. (2014) are available and can be used as benchmarks for comparison with the state-of-the-art. However, these are general purpose datasets that cannot be used in practical settings where specific tasks are required. A common solution to this problem, that has proved to work in many practical cases Pasquale et al. (2016a); Schwarz et al. (2015) is that of splitting the learning of the model in two steps. During the first step, the parameters of the model are initialized with random values and then optimized on one of the available datasets on the web (a standard practice is to use the Imagenet dataset Deng et al. (2009)). Subsequently, during the second step, the obtained parameters are *fine-tuned* on the (usually relatively small) amount of available data for the task at hand. However, some annotated data is required and the amount can be still significant, depending on the task and on the architecture chosen.

Therefore, the resulting steps that are required to adapt a state-of-the-art method to function properly on a specific task are reported in Fig. 1.2. A first step of data acquisition and manual annotation is required to collect a sufficient training set. Then, the model needs to be optimized on the new collected data (either by training it from scratch or by fine-tuning it), with long training time. Finally, the obtained model can be deployed in the application. The aforementioned pipeline does not scale well in practical settings, where the required

task cannot be fixed a-priori but might change during functioning, since it requires both an impressive human effort and long adaptation time.

Nevertheless, the idea of splitting the optimization process, jointly with recent advancements in the field of machine learning Rudi et al. (2017) can lead to impressive speed-up in the learning pipeline. Moreover, computer vision literature proposes interesting solutions to address the data requirement, based on the so called *Weakly supervised learning* framework. These techniques allow to alleviate the assumptions required for the training set, being of interest in many practical fields.

1.2 The robotics setting

Robots might be asked to accomplish very different tasks and each of them could require a specific level of cognition of the environment. For instance, for navigation purposes, a robot might need to recognize the presence of specific landmarks for self-localization or detecting obstacles for avoiding collisions, while for object manipulation, a more precise detection, on the pixel level, might be required. However, ensuring accurate perception and fast adaptation capabilities represents a crucial aspect for all scenarios.

While high reliability is a very important requirement, functioning conditions are not ideal. Indeed, needing to work in real world scenarios, the workspace might be articulated and cluttered, bringing very specific challenges. For instance, in the case of navigation, the landmarks are required to be detected reliably from varying distances, view points and light conditions, while, for grasping an object, a precise identification and segmentation is required even in the case where the object is occluded by other distractors. Moreover, the setting considered can change dynamically both in terms of conditions, since the working space can change dramatically, and in terms of required tasks. For instance, considering an object detection task, there might be the necessity of detecting novel objects or to detect the already known ones within unseen conditions. In both cases, a learning model needs to be updated and adapted to the new situation in order to preserve the required accuracy. Moreover, this adaptation needs to happen quickly when the task changes, in order to ensure the continuity of the system, possibly avoiding considerable human effort and long waiting time.

Nevertheless, while the robotic setting entails specific challenges and imposes certain constraints for the accomplishment of a task, having a robotic platform provides also numerous opportunities. First of all, the embodiment of the robot in the working environment can be exploited in various ways. For example, it can autonomously explore the scene, also modifying it, in order to either improve the confidence of the perception model or to update it by acquiring new data for retraining. Moreover, the physical presence of the robot

can be exploited for allowing, for instance, a more natural interaction to ask for help or further information in case of need. Finally, the robotic setting presents multiple sources of additional information. For example, the spatio-temporal coherence in the stream of images coming from the robot and the data coming from the other sensors of the platform (e.g. depth sensor, sensitive skin, etc.).

1.3 Scientific objectives of the project

The main effort of this Ph.D. project has been in the integration of the two worlds of computer vision and humanoid robotics, considering the specific task of object detection (i.e. the task of localizing and recognizing multiple objects), as this capability is at the basis of more sophisticated tasks. The common thread of all the carried out activities has been looking for solutions which allow to maintain performance of state-of-the-art methods in computer vision within the constraints imposed by the robotic setting, by exploiting its opportunities.

More specifically, the main aim of the thesis has been to improve the pipeline represented in Fig. 1.2, by making the learning process for object detection as natural and effortless as possible. For the data acquisition process, the main objective has been twofold. First of all, the aim has been to exploit human robot interaction within constrained scenarios, to acquire automatically annotated data. Secondly, the target has been to exploit the continuous stream of (unlabeled) images coming from the cameras of the robot at inference time and its exploration capabilities to improve the detection model, by leveraging on an extension of the supervised learning framework, i.e. the *weakly supervised learning*. Furthermore, regarding the learning step (see Fig. 1.2), the objective has been to device a strategy to reduce the time required to optimize an object detection model, allowing for faster adaptation to environment's changes while maintaining state-of-the-art accuracy.

Since the purpose of this project was to improve robotic systems requiring object detection, a further objective of the thesis has been to systematically test the obtained results both with off-line experiments considering different datasets and by deploying them in real robotic applications.

Chapter 2

Outline of the thesis

The outline of this thesis is organized as follows. In Part II, I provide an overview of the theoretical background and state-of-the-art approaches regarding the main research topics that have been of interest for this project. Specifically, in Chapter 3, I briefly introduce the supervised learning framework illustrating how it is adapted when it is applied to the domain of images. Then, in Chapter 4, I provide a definition of the object detection task, describing the metrics used for comparison with the state-of-the-art and explaining the main proposed methods to address this problem, starting from some remarkable historical approaches arriving to tackle some more recent solutions. In the same chapter, I also introduce the main issues of acquiring an annotated image dataset for object detection and I analyze the main research trends in this field for robotics. To conclude this part, in Chapter 5, I introduce the weakly supervised learning framework as an extension of the fully supervised one, explaining its importance in applied fields, like robotics, and describing some possible scenarios that are covered by this framework.

In Part III, I provide a description of the main algorithmic contribution of this project. More specifically, in Chapter 6, I explain the automatic data collection procedure proposed to allow for a natural data acquisition, avoiding manual annotation. Subsequently, in Chapter 7, I give details about the proposed on-line learning algorithm which reduces the training time for the task of object detection. Finally, in Chapter 8, I provide a description of how this on-line learning algorithm has been integrated with a weakly supervised learning strategy with the aim of improving generalization capabilities while reducing human effort. For each contribution, I describe the methods proposed and the experimental analysis carried out to demonstrate the obtained results.

In Part IV, I describe how the algorithmic achievements have been conveyed in a robotic application. Specifically, in Chapter 9, I give details of the application obtained by integrating

the automatic data collection procedure and the on-line learning algorithm for object detection. Furthermore, Chapter 10 explains how this robotic pipeline has been made more robust against scale variation within a specific application scenario.

Finally, in Part V, I conclude this thesis. More specifically, in Chapter 11, I summarize the contributions and the achievements obtained with this project and in Chapter 12 I discuss them, drawing directions for future work.

Part II

Background and State of the Art

Chapter 3

Background

Over the years, *Machine learning (ML)* algorithms demonstrated to be able to revolutionize our daily lives. In fact, they are widely used in many applied research fields (e.g. medical research, robotics, computer vision, biology etc.) and they represent fundamental tools for applications in devices used in our everyday life. However, the term *machine learning* (also referred to as *artificial intelligence*) is sometimes overused or misused. The original definition was coined by Arthur Samuel in 1959 Samuel (1959), a pioneer in the field of computer science, who stated that "[machine learning] *gives computers the ability to learn without being explicitly programmed"*. This concept has been formalized, then, by Tom Mitchell in 1997 Mitchell (1997). He said that *"A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E"*. This means that according to this framework, in order to have a ML algorithm, we should explicitly and formally define a task T, a performance measure P and instantiate the concept of experience E.

While there are tasks that are relatively straightforward to be solved by computers, (e.g. all those problems that can be formalized by a list of mathematical rules), the real challenge to ML is to solve the tasks that are difficult to program in a machine. Surprisingly, this is true very often for tasks that are easy for humans. For instance, recognizing faces in images or spoken words, detecting objects, creating captions for pictures etc. are tasks that are solved by human in, at most, a few seconds but proved to require years of research to be addressed properly by machines. Considerable progresses have been made in the last decades in the field of ML. Beyond the tremendous theoretical advances in fundamental research on this field, other important reasons for its great success (which also motivated theoretical studies, in a virtuous circle) is the availability of huge amount of data to train and test models and, the increasing computational power that can be used for the required processing.

In this context, the provisioning of data plays a fundamental role since it represents the experience from which the model should learn, as per the aforementioned definition of Tom Mitchell. Depending on which assumptions are made on the data, different ML problems can be defined. During this project, in the robotic setting, I firstly considered the so called *fully supervised learning* scenario, where the aim is to "learn" a function with the assumption that all the information for the required learning task are provided. I describe this scenario in the following sections (Sec. 3.1), providing details about the notable case of having images as data to learn from (Sec. 3.2). Afterwards, I considered the case where partial information is provided about the data, so called *weakly supervised learning*. I cover this topic in Chapter 5.

3.1 Supervised learning

The *supervised learning* problem Rosasco and Poggio (2015) is defined as finding an input-output relation, given data in the form of a set of input-output pairs as examples.

3.1.1 Basics of Statistical Learning Theory

More precisely, the goal of supervised learning is that of identifying a function $f : \chi \to Y$ given a set $S_n = \{(x_1, y_1), ..., (x_n, y_n)\}$ (the so called *training set*) where $x_i \in \chi$ and $y_i \in Y$. This target function f should be such that, given a new input point $x_{new} \in \chi$, $f(x_{new})$ is a good prediction of the real correspondent $y_{new} \in Y$.

Training set description

We call χ the *input space*, Y the *output space* and $Z = \chi \times Y$ the *data space*. In the classical statistical learning setting, it is assumed that the data space Z is a probability space with a probability distribution ρ and the training set S_n is assumed to be sampled identically and independently according to ρ, (the *i.i.d.* assumption). The probability distribution ρ describes the uncertainty in the data (e.g. noise).

While usually $\chi \subseteq \mathbb{R}^d$ (with $d \in \mathbb{N}$), according to the different choices for the output space, different variants of the supervised learning problem can be identified. For example, if $Y = \mathbb{R}$, it is called *regression*, if instead, Y is a set of integers, it is called *classification*. Specifically, if $Y = \{-1, 1\}$, it is *binary classification* while if $Y = \{1, 2, ..., T\}, T \in \mathbb{N}$, we can have two different cases: (i) the output is one of the T categories, (*multi-class classification*), (ii) the output is a subset of them (*multi-label classification*).

Loss function

Since the objective in supervised learning is to obtain a function f such that $f(x_{new})$ is a good estimation (prediction) of the ground-truth y_{new} correspondent to x_{new}, a quantity to assess the error of considering the first in place of the second is needed. This quantity is called *loss function*.

In the case of regression this has the aim of computing the distance of the prediction from the actual correspondent value and it has the form of

$$L : Y \times \mathbb{R} \to [0, \infty) \tag{3.1}$$

Some examples of popular loss functions for regression are:

- *Mean Square Error (MSE) (or L_2 Loss)*. It is concerned with the average magnitude of error, regardless of its direction. However, as a result of using squaring, predictions that are far from the actual values are penalized heavier than the closer ones. It also allows for easy computation. It can be written as:

$$L(y, f(x)) = \frac{\sum_{i=1}^{n} (y_i - f(x_i))^2}{n} \tag{3.2}$$

- *Mean Absolute Error (MAE) (or L_1 Loss)*. Like the MSE, it measures the magnitude of the error without considering its direction, however it is more robust to outliers since it does not use the square. Nevertheless, it is not differentiable, thus it might cause more complicated computation. It can be formalized as:

$$L(y, f(x)) = \frac{\sum_{i=1}^{n} |y_i - f(x_i)|}{n} \tag{3.3}$$

In both cases, n is the number of samples in S_n and y_i and $f(x_i)$ are, respectively, the ground truth and the prediction correspondent to x_i.

In the case of classification, instead, the aim of the loss function is to evaluate the number of misclassified samples. Some examples of loss functions for classification are:

- *Misclassification Loss*. It counts the number of errors in the predictions. It is defined as:

$$L(y, f(x)) = \sum_{i=1}^{n} \mathbb{1}_{y_i \neq f(x_i)} \tag{3.4}$$

where the function $\mathbb{1}_{y_i \neq f(x_i)}$ is defined as:

$$\mathbb{1}_{y_i \neq f(x_i)} = \begin{cases} 1, & if\ y_i \neq f(x_i) \\ 0, & otherwise \end{cases} \tag{3.5}$$

- *Hinge Loss*. In order for a prediction to be considered correct, the score for the real category (s_{y_i}) should be greater than the sum of the scores of all the incorrect categories (s_j) by some safety margin (usually one). This loss is usually used for maximum-margin classification (e.g. in Support Vector Machines Hearst et al. (1998); Vapnik (1998)). It is defined as:

$$L(y, f(x)) = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \tag{3.6}$$

- *Cross Entropy Loss (or Negative Log Likelihood)*. This loss function is based on the cross-entropy which is a measure of the difference between two probability distributions. For a binary classification problem, the Cross Entropy Loss is defined as:

$$L(y, f(x)) = -(y_i \log f(x_i) + (1 - y_i) \log (1 - f(x_i))) \tag{3.7}$$

  When the actual label is 1 ($y_i = 1$), the second half of the function disappears whereas, in case the ground truth label is 0 ($y_i = 0$), the first half disappears. This means that we are multiplying the log of the actual predicted probability for the ground truth class. Cross entropy loss heavily penalizes the predictions that are confident but wrong.

3.1.2 Learning from data

With the notation introduced above, a learning algorithm can be defined as a map $S_n \rightarrow f$, returning for each training set S_n an estimator function f. A good learning algorithm should allow to obtain a function f that guarantees a low error (i.e. a low value for the loss function) on the examples of the training set and, more importantly, on future, unknown, examples. This idea can be formalized by defining the so called *expected risk* as

$$E(f) = \int L(y, f(x)) d\rho(x, y) \tag{3.8}$$

The expected risk, for a function f, represents the average error with respect to the probability distribution ρ (which is unknown) according to which the data are drawn. Thus, intuitively, a learning algorithm is good if the output function f has a low expected risk. I.e., we are

aiming to find the function f^* that solves the following optimization problem

$$f^* = \arg\inf_{f \in F} E(f) \tag{3.9}$$

Where F is the *target space* which is the space of functions for which the expected error is well defined. Note that, this minimization problem cannot be solved since the probability distribution ρ is unknown. However, we have access to the training set S_n, which samples, being drawn from ρ, can be used to approximate the above optimization problem with the so called *empirical risk minimization*, which is defined as

$$\hat{f} = \arg\inf_{f \in H} \hat{E}(f) \tag{3.10}$$

where $H \subset F$ is called the *hypothesis space* and $\hat{E}(f)$ is the *empirical risk* and it has the following form

$$\hat{E}(f) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i)) \tag{3.11}$$

H should be large enough such that

$$\inf_{f \in H} E(f) \approx \inf_{f \in F} E(f) \tag{3.12}$$

However, a too wide H can also cause

$$\inf_{f \in H} \hat{E}(f) \neq \inf_{f \in H} E(f) \tag{3.13}$$

and specifically, it can happen that the empirical risk $\hat{E}(f)$ is arbitrarily small while the expected risk $E(f)$ is large, meaning that the learned function (the solution to the optimization problem in Equation 3.10, $\hat{f}$) describes the training set very well but it has poor generalization properties, failing to predict new examples drawn from ρ but not belonging to S_n. This issue is known as *overfitting* and it can be addressed through the so called *regularization networks*. An example is the Tikhonov Tikhonov (1943) regularization framework that substitutes the above minimization problem with

$$\hat{f} = \arg\inf_{f \in H} \hat{E}(f) + \lambda R(f) \tag{3.14}$$

$\lambda R(f)$ is a regularization term, where $\lambda > 0$ is the regularization parameter and R is a functional such that $R : H \to [0, \infty)$. It depends on f and should penalize having an extreme fitting on the training data.

Linear models

One of the simplest choices for the space H is the space of linear functions, such that $f(x) =< w,x >= w^T x$, i.e., by considering $x \in \mathbb{R}^d$, H is defined as:

$$H = \{f : \mathbb{R}^d \to \mathbb{R} \mid f(x) = w^T x\} \tag{3.15}$$

where the vector of parameters $w \in \mathbb{R}^d$ (or *weights*) represents the values to estimate for learning the function f. In this case the problem in Equation 3.14 can be rewritten as

$$\hat{f} = \arg\min_{f \in H} \frac{1}{n} \sum_{i=1}^{n} L(y_i, w^T x_i) + \lambda \|w\|^2 \tag{3.16}$$

Even if it is computationally easy to solve these linear problems, they might not lead to effective solutions, especially for complex data with a non linear structure. For this reason, sometimes, non linear models need to be used.

Non linear models

If data with non linear structure is used, we need to consider richer spaces of functions. For instance, we can consider the class of functions that can be written as

$$f(x) = \sum_{j=1}^{D} w_j \varphi_j(x) \tag{3.17}$$

where $D \in \mathbb{N}$, $\varphi_j(x) : \mathbb{R}^d \to \mathbb{R}$ is a non linear function, with $j \in \{1,...,D\}$ and usually $D >> d$.
By defining $\Phi : \mathbb{R}^d \to \mathbb{R}^D$ as

$$\Phi(x) = (\varphi_1(x), ..., \varphi_D(x))^T \tag{3.18}$$

we can rewrite Equation 3.17 as

$$f(x) = w^T \Phi(x) \tag{3.19}$$

Finally, the minimization problem in Equation 3.14, becomes

$$\hat{f} = \arg\min_{f \in H} \frac{1}{n} \sum_{i=1}^{n} L(y_i, w^T \Phi(x_i)) + \lambda \|w\|^2 \tag{3.20}$$

where now $w \in \mathbb{R}^D$. Φ is also called *feature map*. By choosing non linear models, we usually gain in accuracy, by paying a price in terms of computation needed. Indeed, typically, both d

and D are in the order of magnitude of 100, 1000 or greater and obtaining Φ may require a high computational budget.

3.1.3 Regularized Least Squares, a remarkable example

The *Regularized least squares (or RLS)* is a regression algorithm that, with respect to the framework presented above, assumes that:

- χ is an euclidean space in d dimensions, i.e. $x_i \in \mathbb{R}^d$
- The loss function considered is the MSE
- H is the space of linear functions, such that,

$$H = \{f : \mathbb{R}^d \to \mathbb{R} \mid f(x) =< w,x >\} \tag{3.21}$$

Therefore, the problem presented in Equation 3.14 can be rewritten as

$$\hat{f} = \arg\min_{f \in H} \frac{1}{n} \sum_{i=1}^{n} \left\| y_i - w^T x_i \right\|^2 + \lambda \|w\|^2 \tag{3.22}$$

that can be further rewritten in a more compact form, exploiting vectorial representation of the dataset, as

$$\hat{f} = \arg\min_{w} \frac{1}{n} \|Y - Xw\|^2 + \lambda \|w\|^2 \tag{3.23}$$

where $Y \in \mathbb{R}^n$ is the vector containing the output samples of the training set, i.e., $Y = [y_1, ..., y_n]$ and $X \in \mathbb{R}^{n \times d}$ is a matrix where each row is an input sample of the dataset.

One of the main advantages of RLS is that it allows for a closed form solution. Indeed, it is possible to compute the following gradient

$$\bigtriangledown \left(\frac{1}{n} \|Y - Xw\|^2 + \lambda \|w\|^2 \right) = -\frac{2}{n} X^T (Y - Xw) + 2\lambda w \tag{3.24}$$

and, equating it to 0, we can find the following solution for w

$$w = (X^T X + n\lambda I)^{-1} X^T Y \tag{3.25}$$

Kernel Regularized Least squares

The extension to a non linear input space, described in the presentation of the general framework (see Sec. 3.1.2), can be applied to RLS as well. Specifically, with the definitions

of φ and Φ presented above, we can define the following matrix

$$\hat{\Phi} = (\Phi(x_1), ..., \Phi(x_n))^T \in \mathbb{R}^{n \times D} \tag{3.26}$$

and the problem in Equation 3.23 becomes

$$\hat{f} = \arg\min_w \frac{1}{n} \left\| Y - \hat{\Phi} w \right\|^2 + \lambda \left\| w \right\|^2 \tag{3.27}$$

which is solved by

$$w = (\hat{\Phi}^T \hat{\Phi} + n\lambda I)^{-1} \hat{\Phi}^T Y \tag{3.28}$$

Note that, obtaining $\hat{\Phi}^T \hat{\Phi}$ might be computationally really hard (in many practical cases even impossible) since $\hat{\Phi}^T \hat{\Phi} \in \mathbb{R}^{D \times D}$ and D is usually large. To improve on this computational aspect, we can use the result demonstrated in the *Representer Theorem* Schölkopf et al. (2001). Note that, this theorem is here mentioned for the RLS problem but it also holds for more general loss functions. This theorem proves that, if w solves the RLS problem of Equation 3.27, then it can be rewritten as $w = \hat{\Phi}^T c$ where $c = (\hat{\Phi}\hat{\Phi}^T + \lambda n I)^{-1} Y$ and $\hat{\Phi}\hat{\Phi}^T \in \mathbb{R}^{n \times n}$. The proof of this theorem is out of the scope of this thesis (we refer the reader to Schölkopf et al. (2001) for the demonstration), however the result is important since it allows to substitute the computation of w with the computation of c, in the solution of the problem. This consents to relate the computational load of the learning algorithm to the size of the used training sets (i.e., n) instead of the dimension of the feature vector (i.e., D). This represents still a big problem, especially considering the size of state-of-the-art benchmarks in many research fields (see e.g. the image datasets Deng et al. (2009); Everingham et al. (2010); Lin et al. (2014); Pasquale et al. (2019)), however latest research in this domain has specifically worked on this computational aspect, obtaining impressive results Rudi et al. (2017) (more details on this solution will be provided in Sec. 7.1.3).

3.2 Learning from images

One of the main application domains of machine learning techniques is computer vision. Indeed, recently, one of the main trends is to address the majority of computer vision's tasks through the supervised learning framework. While in the next chapter, I introduce the specific task of object detection, describing how this framework can be applied to address it and some main solutions, in this section I explain, in general terms, how it is possible to apply supervised learning techniques to images, by using, as an example scenario, the *image classification* task, i.e. the task of recognizing the category of the subject of an image.

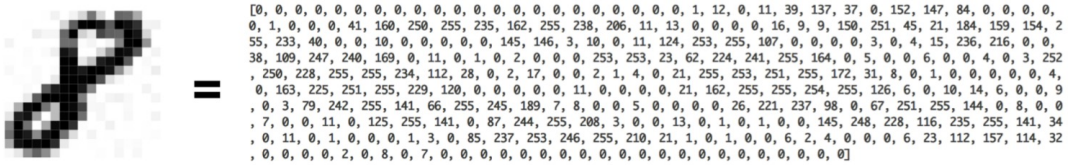


Figure 3.1: This figure[1] shows the matrix representation of an image, depicting the number eight. While it is relatively easy to recognize it in the picture on the left, it is almost impossible to do the same with the one on the right.

3.2.1 Problem definition

While for humans the task of classifying the content of an image might appear trivial, this is not true in the case of a machine. Indeed we, as humans, are able to recognize the subject of a picture with just a brief glance. However, this is far more challenging for robots or machines in general, since an image is typically encoded and stored in computers as a matrix of numbers. The reader can refer to Fig. 3.1 as a pictorial example of this issue. This picture shows that, while it is easy to recognize the number represented in the picture on the left, it is almost impossible to do the same with the matrix representation on the right. In the case represented in Fig. 3.1, the image is represented by a 2 dimensional matrix (with the so called *gray scale* encoding), where the intensity of each pixel is expressed by an integer from 0 (white) to 255 (black). However, another very common image representation uses a 3-channel encoding, which describes each pixel with a set of three numbers representing the intensity of the three colors: *red*, *green* and *blue* (the so called *RGB* encoding). This last encoding ends up in a $w \times h \times 3$ matrix, where w and h are respectively the width and the height of the image, expressed in terms of pixels, meaning that an image is an element of $\mathbb{R}^{w \times h \times 3}$.

3.2.2 Image representation

Given this matrix representation, which comes for free due to the computer's encoding, a naïve way to apply the supervised learning framework to perform image classification is to "unroll" the matrices in vectors (of $w * h * 3$ elements each) and consider $\chi \subset \mathbb{R}^{w \times h \times 3}$. Considering the classification task, the Y will be a set of integers, encoding the different classes to be identified in the images (e.g. $\{"dog" = 0, "cat" = 1, "bird" = 2\}$ if the task is to recognize dogs, cats and birds). At this point, by gathering a set of associations between

[1] taken from: https://medium.com/@ageitgey/machine-learning-is-fun-part-3-deep-\learning-and-convolutional-neural-networks-f40359318721

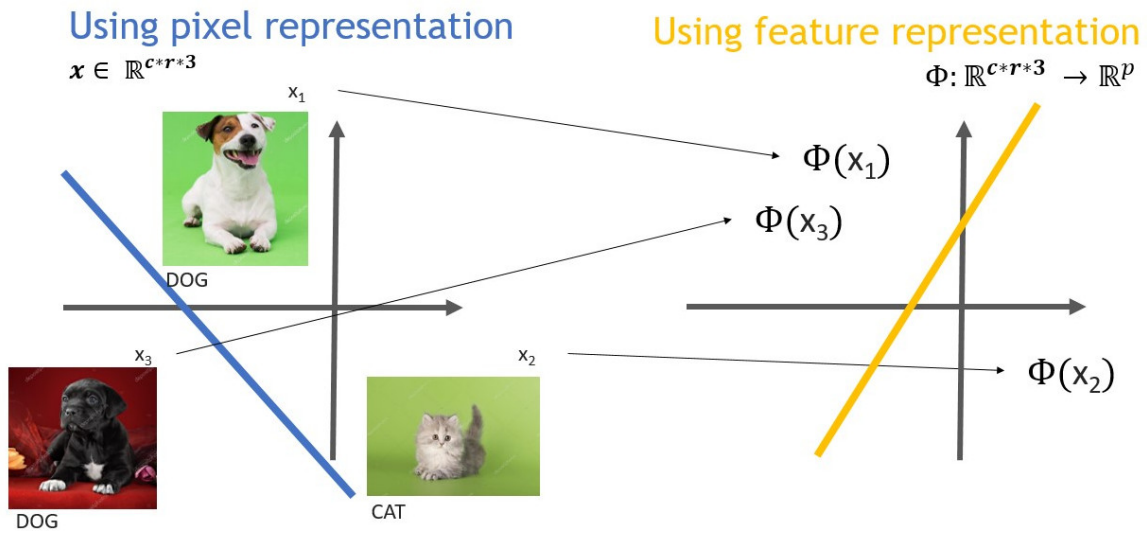


Figure 3.2: This figure[2]shows the comparison between using the RGB pixel representation (situation on the left) and a feature descriptor representation (situation on the right). This latter concept is left abstract in the picture, as it should represent a relevant feature for the considered task.

images and represented classes, we will end up with a training set S_n and we will be able to apply all the considerations described in Sec. 3.1.

However, classifying the content of an image only by considering the intensity (the color) of its pixels might be misleading. Indeed, the key feature to discriminate between two classes might not be related to colors. Refer to Fig. 3.2 for a pictorial example of this concept. In the situation described in the figure, using only colors would lead to mis-classify the two images with green backgrounds as they do not belong to the same class. For this reason, a more sophisticated image representation is usually used, the so called image descriptors or feature maps.

For a long period of time, a large part of the computer vision research community had focused on designing and engineering hand-crafted image descriptors, to improve performance of machine learning algorithms Bay et al. (2006); Dalal and Triggs (2005); Enzweiler and Gavrila (2011); Lampert et al. (2008); Sabzmeydani and Mori (2007); Tuzel et al. (2008); Walk et al. (2010); Wang et al. (2009). The intuition is that, the more informative the chosen features are with respect to the desired task, the better the algorithm will perform. The usual steps Bay et al. (2006) for the definition of new image descriptors are:

1. Definition of the *interest points*. These points represent distinctive locations in the image with important visual properties with respect to the considered task (e.g. corners).

[2]Figure inspired by http://www.robots.ox.ac.uk/ vedaldi//assets/teach/vedaldi14bmvc-tutorial.pdf

Detecting the interest points in an image is like focusing the feature extractor's attention on specific, important points.

2. Definition of a *feature extraction* criterion in the neighborhoods of the interest points. This represents the actual step of the descriptor computation. This descriptor has to be distinctive and, at the same time, robust to noise and geometric and photometric deformations. The output of this step is a feature vector which *describes* the considered image. More formally, an image descriptor can be seen as a *representation function* Φ that maps data samples $x \in \chi$ to vectors $\Phi(x) \in \mathbb{R}^D$.

3. Definition of a *distance metric* to compare two descriptor vectors. This comparison is often implemented by a distance between the vectors, like e.g., the euclidean distance.

A remarkable example of image descriptor, that has been widely used in the past, is the *Histogram of oriented gradients (HOG)* Dalal and Triggs (2005). The main idea behind this method is that the local shape of the objects in an image, can be described by the distribution of intensity gradients or edge directions. The HOG descriptor of an image is computed by first, dividing the image into a grid of small connected regions (*cells* in Dalal and Triggs (2005)). Then, for the pixels within each cell, a histogram of gradient directions is computed. The final descriptor of the image is obtained by concatenating these histograms. While methods, like the HOG, have been very popular in the past, the most used feature extractors at the moment are the so called *Convolutional neural networks (CNNs)* LeCun et al. (1989). I describe main aspects of this approach in the next section.

3.2.3 Convolutional feature maps

One of the most popular way, in recent days, to encode images into feature descriptors is using a concatenation of *convolutional layers*, also named as *convolutional neural network (CNN)* LeCun et al. (1989), obtaining so called *convolutional feature maps*.

Convolutional layer

A CNN processes images, gradually transforming the low-level RGB data, into progressively more abstract representations, capturing concepts such as edges, boundaries, textures, etc. More formally, a CNN can be represented as a concatenation of convolutional layers. A convolutional layer is a function that takes a tensor as input and gives another tensor as output, obtained by *convolving* a set (*bank*) of filters with the input tensor. In this setting, a tensor is a multi-dimensional vector which associates to each spatial location a vector of

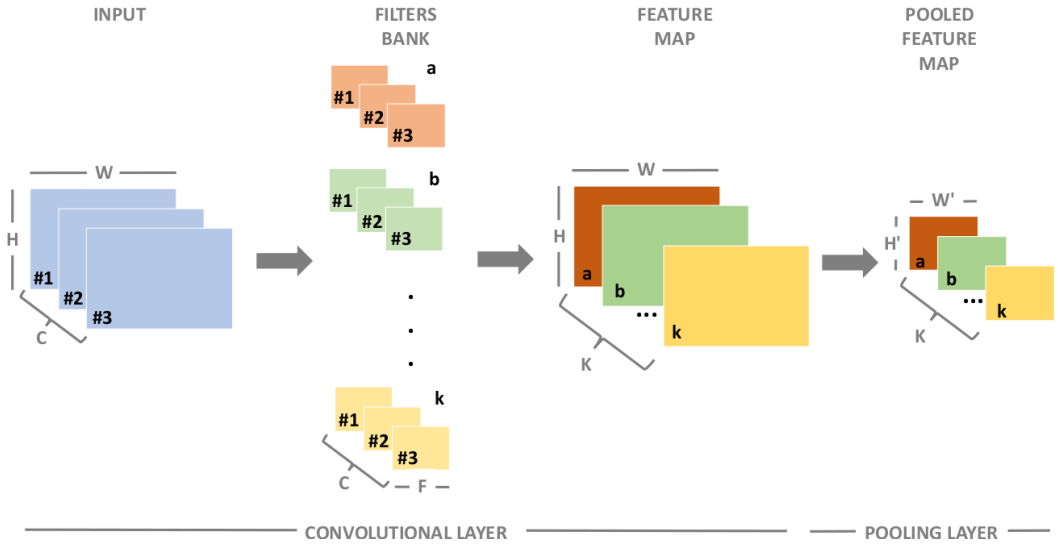


Figure 3.3: Representation of the concatenation between a *convolutional layer* and a *pooling layer*. The input is a tensor in $\mathbb{R}^{W\times H\times C}$, the filters bank is composed by K filters $F\times F$ of C channels each. The feature map is a tensor in $\mathbb{R}^{W\times H\times K}$ where the i^{th} matrix, out of the K, is computed by convolving the input with the i^{th} filter. Finally, the pooled feature map is a tensor in $\mathbb{R}^{W'\times H'\times K}$, where W' and H' are the new dimensions after the resize done by the pooling layer.

features. In this perspective, an RGB image is a tensor: the pixels are the different spatial locations, with three feature channels associated (one for each primary color).

More formally, we can define the integers W, H and C as respectively the width, the height and the number of channels of the input tensor and K and F respectively the number and the dimension of the filters (refer to Fig. 3.3 for a pictorial representation). Given these quantities, the input tensor of the convolutional layer is $x\in\mathbb{R}^{W\times H\times C}$, and the bank of filters can be represented as $\mathbf{f}\in\mathbb{R}^{K\times C\times F}$. Finally, the output tensor, also called *feature map*, $\mathbf{y}$, after the convolution is composed of:

$$y_{whk}=\sum_{c=0}^{C-1}\sum_{u=0}^{F-1} f_{kcu}\cdot x_{w,c,h+u} \tag{3.29}$$

where y_{whk} is the element of the tensor $\mathbf{y}$ in the column w, row y and k^{th} channel. This can be thought as applying to the input image the K filters, which are spatially small (in terms of height and width) with respect to the dimensions of the input but they extend through the full depth of the input volume (the channels). These filters are convolved (slided) across the width and height of the input volume and dot products between the entries of the filter and the input are computed at any position. The sliding step is called *stride* and it determines the

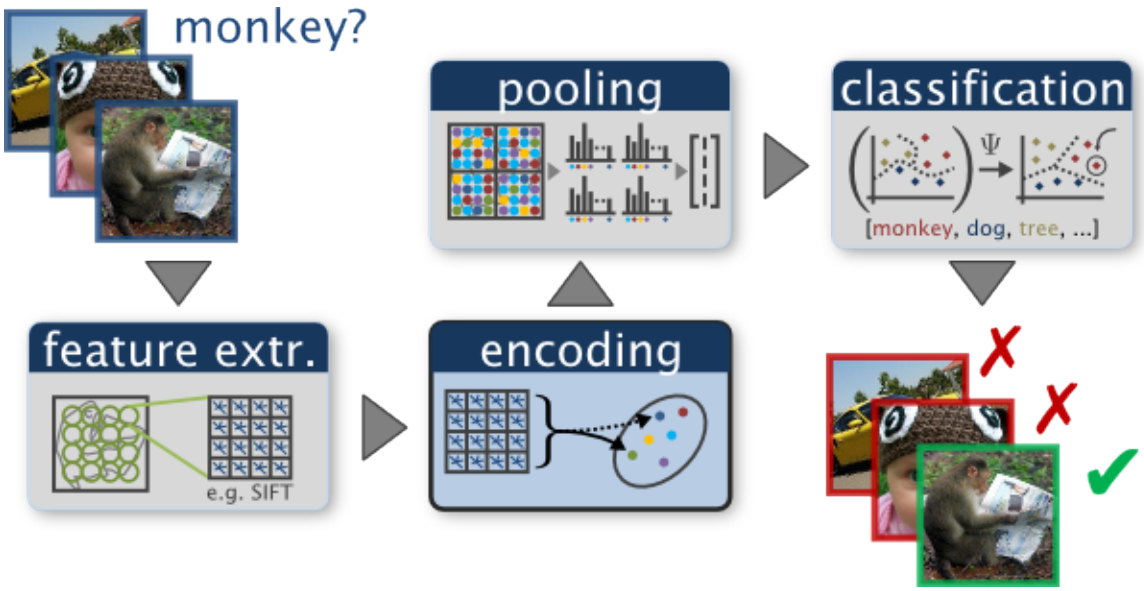


Figure 3.4: This picture[3] shows the usual pipeline to perform image classification, involving a step of feature extraction, encoding, possible pooling and finally classification. See Sec. 3.2.4 for more details.

final dimension of the output. If it is 1, the output width and height are the same as the input. Please, refer to Fig. 3.3 for a pictorial representation of this process.

In a typical CNN, the parameters of the convolutional filters are usually learned from data. Note that, while in classic image representation algorithms the feature descriptors were hand-crafted and engineered by humans, in this case they are learned from the training set.

Pooling layer

A typical CNN is composed by a concatenation of convolutional layers interspersed with *pooling layers*. Their function is to progressively reduce the spatial size of the feature map to reduce the amount of parameters and computation in the network, and hence to also control overfitting. The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, by sliding a fixed size window (typically 2×2) and applying, usually, a max or an average operation, substituting the elements from the input in the window with the resulting element.

3.2.4 Image classification pipeline

Given the tools described above, the pipeline to perform image classification can be represented as in Fig 3.4. Firstly, a feature extractor algorithm is applied to the image to compute

an image descriptor and encode the image into descriptive features. Possibly, if needed, a step of further pooling can be applied and finally, the feature vector is fed into the classifier, which has to discriminate the class represented in the image.

A very common trend in state-of-the-art solutions is to substitute the whole pipeline with a CNN. The concatenation of convolutional and pooling layers repeatedly performs the operations of feature extraction, encoding and pooling. Finally, the classification operation is usually performed by the so called *fully connected (FC) layers* on the extracted convolutional feature map. An FC layer is a function which first "unrolls" the feature map provided by the CNN into a flat vector, then it performs an inner product between this obtained vector and a set of weights. A bias term is finally added to the result. The vectors of weights and biases are learned from data and the objective that is usually used for the learning is to obtain as output a set of P values, representing the probabilities that the image has to represent all the P classes from the classification problem. The main trend is to concatenate the FC layers to the CNN and learn them together, *end-to-end*, through a process that is called *back-propagation*.

[3]Taken from: http://www.robots.ox.ac.uk/~vgg/research/encoding_eval/

Chapter 4

Object Detection

Object detection is the computer vision task that deals with localizing and recognizing instances of visual objects of certain classes in digital images. This is a crucial skill for robots that are designed to interact in a natural way with humans and the surrounding environment. Indeed, detecting objects is fundamental for a large variety of complex tasks such as robot navigation, object grasping and manipulation. Moreover, solving this problem can represent a first step towards more sophisticated perception tasks, such as object segmentation, 3D pose estimation or semantic scene understanding. Furthermore, latest results in this field, pushed by deep learning approaches, have gathered particular attention.

Considering the above reasons, one of the main objectives of this PhD project has been to bring the stunning performance achieved in this field to robotic applications. This implies specific requirements and limitations in terms of training time, data availability and computational resources.

In this chapter, I introduce the problem of object detection. I first give a formal definition of the task, relating it to the other main computer vision problems (Sec. 4.1) and specifying the metrics used to evaluate performance (Sec. 4.2). Subsequently, I overview object detection literature, mentioning the main historical approaches (Sec. 4.3), then focusing on the current main trends of the state-of-the-art (Sec. 4.4) and giving details of one of the main problems related to this task (Sec. 4.5). Furthermore, I introduce the object segmentation problem, which is a natural extension of the object detection and that has been of interest during this PhD project (Sec. 4.6). Finally, I describe the characteristics that an object detection dataset should have (Sec. 4.7) and I give a glance of the main research trends in robotics regarding this field (Sec. 4.8).

4.1 Problem definition

Computer vision literature does not provide a universally accepted definition of the object detection problem. Nevertheless, it is usually defined as the task of localizing and recognizing all the instances of given categories in digital images Agarwal et al. (2018); Everingham et al. (2015, 2010); Huang et al. (2016). More precisely, following the formalization reported in Agarwal et al. (2018), let's denote as C the set of categories that need to be detected, as I an image and as $O(I)$ the set of N_I^* objects that are present in the image I (also called the *ground truth* of I), with

$$O(I) = \left\{(Y_1^*, Z_1^*), ..., (Y_i^*, Z_i^*), ..., (Y_{N_I^*}^*, Z_{N_I^*}^*)\right\} \tag{4.1}$$

where $Y_i^* \in C$ is the category of the i^{th} object and $Z_i^* \in \mathscr{Z}$ is a representation of its location in I. $\mathscr{Z}$ is the space of possible locations that can be represented by one of the following options (or their combinations):

- The center of mass of the objects $(x_c, y_c) \in \mathbb{R}^2$
- A 2D bounding box, tightly containing the object, usually represented either by the coordinates of the top-left and bottom-right corners ($(x_{min}, y_{min}, x_{max}, y_{max}) \in \mathbb{R}^4$) or by its top left corner and width and height ($(x_{min}, y_{min}, w, h) \in \mathbb{R}^4$).

Using this notations, object detection can be defined as the function that associates an image I with a set of detection $D(I)$ with:

$$D(I) = \left\{(Y_1, Z_1, s_1), ..., (Y_j, Z_j, s_j), ..., (Y_{N_I}, Z_{N_I}, s_{N_I})\right\} \tag{4.2}$$

where (Y_j, Z_j, s_j) is the j^{th} detection of the image I, represented by the couple of the predicted location Z_j and category Y_j and s_j, a score representing the confidence on that detection. Note that, by virtue of the confidence score s_j, it is possible to define a ranking among the detections in $D(I)$ and a threshold T_s to discriminate which detections are valid (if their score is higher than T_s) and which are not.

In traditional computer vision literature Agarwal et al. (2018); Everingham et al. (2015, 2010); Huang et al. (2016), it is usually assumed that the location of an object is represented by the 2D bounding box that tightly surrounds it and the category is represented by a string label of the name of the object. Therefore, if not differently specified, hereafter, I will refer to this definition of the object detection problem.

Note that, the problem of object detection is related but different from other tasks of computer vision. For instance, it is different from (i) *object recognition* which is usually defined as giving the name of the category of the object contained in the image or from (ii) *object instance segmentation*, which aims at classifying all the pixels in an image, identifying different instances of the categories of interest. Indeed, while for solving the first one (*object recognition*), one can consider the assumption that there is always one and only one object in the image and the localization component is not tackled, the latter (*object segmentation*) can be considered as an extension of object detection, with a finer localization at the level of the pixels.

Solving object detection problem entails specific challenges due to the nature of the task itself. For instance, objects of interest can be represented at different scales (*scale variance*) or angles (*rotational variance*) or they can be depicted in cluttered environments partially overlapped with distractors or other objects of interest (*occlusion*). Moreover, the objects that needs to be detected might be really small, reducing the number of pixels (and thus the information) that are used to represent them and thus requiring a more precise localization. Finally, and this applies especially for learning based methods, an object detection algorithm should deal with a large variety of different backgrounds and light conditions (*domain adaptation*).

4.2 Evaluation metrics

Object detection algorithms are evaluated both in terms of (i) the *accuracy* of the computed detections (considering both the precision of the boxes and of the labels) and in terms of (i) the *time performance*. In the following sections, I describe the main metrics used to evaluate the methods of the state-of-the-art.

4.2.1 Accuracy metrics

A good metric to evaluate accuracy of an object detection algorithm needs to consider both the precision of the bounding box and correctness of the associated label. Following on the aforementioned formalization, we need to associate each detection from $D(I)$ to one and only object of the ground truth from $O(I)$. To perform this association, following Agarwal et al. (2018), we need to define:

- A **Geometric Compatibility Function** which defines the conditions to meet for associating two bounding boxes, $G : (Z, Z^*) \in \mathscr{Z}^2 \rightarrow \{0, 1\}$.

- An **Association Matrix** $A \in \{0,1\}^{N_I \times N_I^*}$ which defines a bipartite graph between the detections and the ground truth objects, accordingly to the Geometric compatibility function. Specifically,

$$A_{i,j} = \begin{cases} 1 & if\ G(Z_j, Z_i^*) = 1 \ \ AND \ \ Y_j = Y_i^* \ \ AND \\ & \nexists (Z_k, Y_k, s_k) \in D(I) \ \ s.t. \ \ G(Z_k, Z_i^*) = 1, \ \ Y_k = Y_i^*, \ \ s_k > s_j \\ 0 & otherwise \end{cases} \tag{4.3}$$

meaning that, the detection (Z_j, Y_j, s_j) can be associated to the ground truth box (Z_i^*, Y_i^*) only if the two bounding boxes and their labels correspond and a more confident detection with the same characteristics does not exist.

Each detection can be associated with only one ground truth object and vice versa. This leads to

$$\sum_{j=1}^{N_I} A_{i,j} \leq 1 \tag{4.4}$$

$$\sum_{i=1}^{N_I^*} A_{i,j} \leq 1 \tag{4.5}$$

Therefore, an evaluation metric is defined by its Geometric compatibility function. In the remainder of this section I provide details of two of the main evaluation metrics commonly used in the computer vision literature, provided along with two object detection challenges, i.e. the Pascal VOC Everingham et al. (2015) and the MS COCO Lin et al. (2014).

Before describing the two metrics, some notations need to be introduced. In both cases, the Geometric compatibility function is defined considering an *Intersection over Union* (IoU in the following) evaluating the overlap between the two bounding boxes Z_j and Z_i^*. Specifically, it is defined as

$$IoU = \frac{area(Z_j \bigcap Z_i^*)}{area(Z_j \bigcup Z_i^*)} = \frac{area\ of\ overlap}{area\ of\ union} \tag{4.6}$$

The Geometric compatibility function is then defined as

$$G(Z_j, Z_i^*) = \begin{cases} 1 & if\ IoU \geq T_{IoU} \\ 0 & otherwise \end{cases} \tag{4.7}$$

where T_{IoU} is a threshold that specifies the minimum IoU required to consider two boxes equivalent and it is set differently in Pascal VOC Everingham et al. (2015) and MS COCO Lin

et al. (2014).
At this point, the following concepts can be defined:

- A **True Positive (TP)** is a correct detection.
  (Z_j, Y_j, s_j) is defined as **TP** if $\exists (Z_i^*, Y_i^*) \;\; s.t. \;\; A_{i,j} = 1$.
- A **False Positive (FP)** is a wrong detection.
  (Z_j, Y_j, s_j) is defined as **FP** if $\nexists (Z_i^*, Y_i^*) \;\; s.t. \;\; A_{i,j} = 1$.
- A **False Negative (FN)** is a ground truth not detected.
  (Z_i^*, Y_i^*) is a **FN** if $\nexists (Z_j, Y_j, s_j) \;\; s.t. \;\; A_{i,j} = 1$

Note that, with respect to metrics used in classification tasks, the concept of True Negative, which represents negative class correctly recognized, does not apply, since, in object detection, there are many possible bounding boxes that should not be detected within an image. Thus, True Negative would be all possible bounding boxes that were correctly not detected (a huge amount of boxes within an image). That is the main reason for not using it.

Using the concepts described above, it is possible to define for *each* of the class in C, the following metrics:

- **Precision (p)**, which measures the percentage of the correct positive predictions over all the detections in $D(I)$. I.e.,

$$Precision = \frac{TP}{TP + FP} \tag{4.8}$$

- **Recall (r)**, which measures the percentage of the correct positive predictions over all the ground truths in $O(I)$. I.e.,

$$Recall = \frac{TP}{TP + FN} \tag{4.9}$$

- **Precision-Recall curve**, which is a curve that shows how precision changes for different values of recall. Note that, it is possible to consider different precision-recall combinations by varying the threshold T_s and considering more or less detections as valid ones. Usually, precision decreases for higher recall values. In fact, intuitively, a detector needs to increase the number of detected objects (and thus at the same time the number of False Positives, losing *Precision*), in order to retrieve a higher number of ground truth objects (achieving higher *Recall*).

- **Average Precision (AP)**, which is defined, in general, as the area under the *Precision-Recall curve*.

$$AP = \int_0^1 p(r)dr \tag{4.10}$$

Finally, a **Mean Average Precision (mAP)** can be defined as

$$mAP = \frac{1}{|C|} \sum_{c \in C} AP_c \tag{4.11}$$

where AP_c is the *Average Precision* of the class c.

In the remaining part of this section, the two main metrics currently used in computer vision from the Pascal VOC Everingham et al. (2015) and the MS COCO Lin et al. (2014) datasets are presented. Note that, considering that for the experimental evaluation of the methods proposed for this project, I considered the Pascal VOC as benchmark for comparing to the state-of-the-art, when not differently specified, the definition of the *mAP* provided in Everingham et al. (2015) and described in the next paragraph, will be considered.

Pascal VOC metric Everingham et al. (2015)

In the Pascal VOC devkit[1], the authors

1. set the $T_{IoU} = 0.5$
2. approximate the *AP* with an Interpolated AP by considering only 11 recall values (from 0 to 1.0 with a step size of 0.1). Thus the AP is computed as

$$AP = \frac{1}{11} \sum_{i \in [1,11]} p(i) \tag{4.12}$$

The motivation of using this 11-interpolation, according to the authors Everingham et al. (2015), is "*the intention in interpolating the precision/recall curve in this way is to reduce the impact of the "wiggles" in the precision/recall curve, caused by small variations in the ranking of examples.*"

COCO metric Lin et al. (2014)

In the evaluation metric used by COCO[2] the AP is redefined as the average of the area under the 101-point interpolated precisions curve (i) for *all* the categories of the dataset (thus,

[1] http://host.robots.ox.ac.uk/pascal/VOC/voc2007/
[2] http://cocodataset.org/#detection-eval

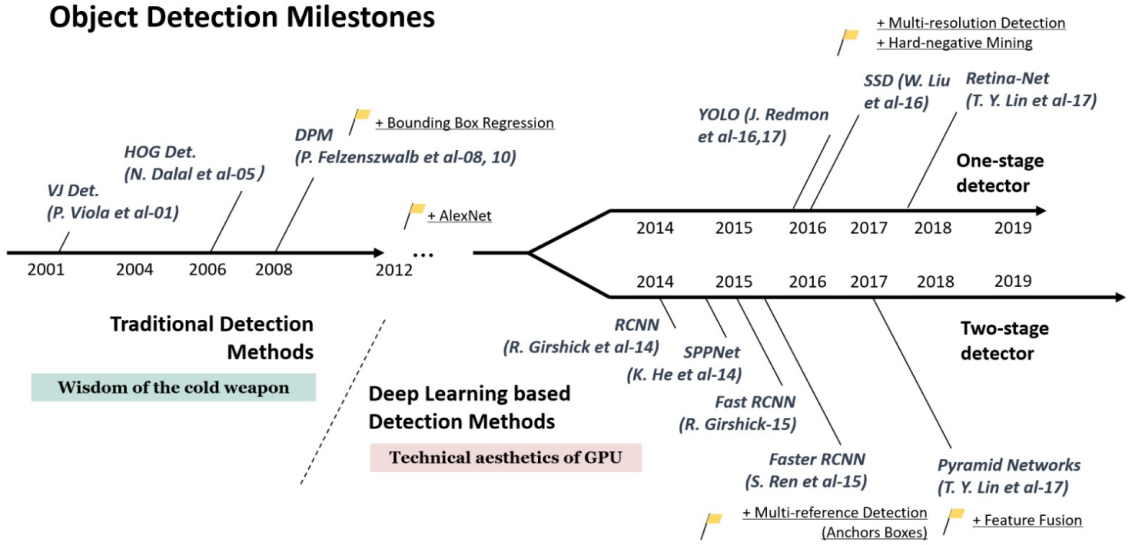


Figure 4.1: Historical overview of main progress in the object detection literature (picture taken from Agarwal et al. (2018)).

getting rid of the concept of *mAP*) and (ii) over a set of 10 IoU thresholds T_{IoU}, starting from 0.5 to 0.95 with a step size of 0.05. I.e.,

$$AP = \frac{1}{|C| \times 11 \times 10} \sum_{t_{IoU} \in [.5,.95]} \sum_{c \in C} \sum_{i \in [1,11]} p_{c,t_{IoU}}(i) \tag{4.13}$$

4.2.2 Time performance metrics

Other two important metrics to evaluate object detection models Huang et al. (2016), which plays a fundamental role in robotics Sunderhauf et al. (2018), are the *Fitting time* and the *Inference time*. The *Fitting time* represents the time required by the considered object detection method for being ready to be used. When using machine learning or deep learning methods, it is usually referred to as *Learning time* since it corresponds to the time required to train the model with the training dataset and it is usually measured in minutes or hours. The *Inference time*, instead, is the time required to predict all the detections in an image and it is usually measured in frame/seconds (*fps*), i.e. the number of images that the algorithm can process in one second. In this thesis, I mainly focused on the *Learning time* for methods evaluation since system's adaptation time is crucial in the considered robotic scenario.

4.3 Origins and historical approaches

Historically, the progress of object detection can be divided into two main periods of time, i.e. before and after the revival of Deep Learning in 2012 Agarwal et al. (2018); Huang et al. (2016); Zou et al. (2019) (see Fig. 4.1 to have a glance of this progress).

One of the most natural way of addressing the problem of object detection is to perform a dense search of the objects of interest, over all the possible locations in the image. This can be implemented by using (i) a sliding window approach for producing a set of regions of different dimensions and aspect ratio and (ii) a classifier to discriminate which of these regions contain the objects of interest Sung (1996); Viola and Jones (2001). The earliest proposed methods were indeed based on this principle and they were mainly employed for human face detection in images (which is a special case of object detection) Sung (1996); Viola and Jones (2001).

One of the first methods that achieved real-time performance was the so called *Viola-Jones (VJ) Detector* Viola and Jones (2001); Viola and Jones (2004). This approach was based on (i) a sliding window, (ii) a Haar wavelet as feature representation for the image (as it was the common trend at that time Mohan et al. (2001); Papageorgiou and Poggio (2000); Papageorgiou et al. (1998)) and (iii) a variant of the Adaboost Freund et al. (1999); Freund and Schapire (1997) to both select a small set of features and train the classifier to discriminate which windows contain the faces. The importance of the VJ Detector relies on the fact that it has dramatically improved the *Inference time* with respect to contemporary methods, by integrating the three following techniques in the sliding window approach:

1. *Integral Image* Viola and Jones (2001). The integral image is a computational method to speed up box filtering. Specifically, it makes the computational complexity related to each window independent from their sizes.

2. *Feature Selection*. In the VJ Detector the Haar basis are not manually set but the more "meaningful" are selected using the Adaboost algorithm Freund et al. (1999); Freund and Schapire (1997).

3. *Detection cascades*. The concept of "detection cascades" has been introduced in the VJ Detector Viola and Jones (2001). In this kind of approaches, as a first step, lightweight classifiers are applied to discard easy background regions, subsequently, increasingly more complex classifiers are applied on regions that are most likely to contain objects. The main idea is to reduce computational overhead for background windows.

Another important milestone in early object detection techniques has been the introduction of the *Histogram of Oriented Gradients* (HOG) feature descriptor Dalal and Triggs (2005).

This descriptor was designed to improve precision against translations, scale and light variations. To improve robustness for objects of different sizes, the HOG detector re-scales the input image multiple times, keeping the size of the detection window unchanged. This feature descriptor has represented an important component for many object detectors before the deep learning advent Felzenszwalb et al. (2008); Felzenszwalb et al. (2010b).

However, HOG and the Haar wavelets represent just two examples of image descriptors among all the others that were proposed in those years, as preliminary representation before classification. Indeed, one of the most crucial aspects of the early object detectors, before the advent of deep learning, was image description. Another feature extraction method that is worth mentioning is the so called bag-of-words method Lampert et al. (2008). This approach has been originally proposed in the field of text analysis and it was used to provide a simplifying representation of a text document, by using a histogram of the frequencies of the words it is composed of. Afterwards, it has been adopted in Lampert et al. (2008) and then extensively used in the computer vision domain (in Shen et al. (2012); Uijlings et al. (2013)) by considering histograms of "visual" words to describe an image, represented by keypoints or image descriptors like, e.g., so called SIFT descriptors Lampert et al. (2008). Finally, Edgelets Wu and Nevatia (2005), shapelets Sabzmeydani and Mori (2007), integral histograms Porikli (2005), color histograms Walk et al. (2010), covariance descriptors Tuzel et al. (2008, 2007) and linear binary patterns Enzweiler and Gavrila (2011); Wang et al. (2009) represent only a few other examples of the literature produced at that time on hand crafted features used for object detection.

The peak of traditional object detection methods is represented by the *Deformable Part-based Model (DPM)*. Methods based on DPM's principles won the Pascal VOC detection competition in 2007, 2008 and 2009. DPM was originally proposed in Felzenszwalb et al. (2008) as an extension of the HOG detector and then improved in later works Felzenszwalb et al. (2010a,b); Girshick (2012); Girshick et al. (2011). In these methods, detectors focus on localizing and recognizing different objects parts. Specifically, the training phase can be considered as the learning of the best way of decomposing an object, while the inference can be considered as the localization and gathering of the object's parts. The so called *part-filters* are responsible for the localization of the parts, which configurations are typically learned as latent variables through weakly supervised learning methods (see Chapter 5 for a definition of weakly supervised learning). Along with DPM, other common practices that are currently used in state-of-the-art approaches, have been introduced, like for instance the *bounding box regression* and *hard negative mining*. These concepts will be tackled later in this chapter (see respectively, Sec. 4.4.1 and 4.5).

After 2010, hand-crafted features saturated their performance and so did object detection Girshick (2012). This plateau lasted until 2012, the year that is considered to sanction the revival of *Deep Learning* Goodfellow et al. (2016).

4.4 Deep learning for object detection

Deep learning has gained an incredible popularity nowadays and it is currently used in different fields to address a large variety of tasks Cao et al. (2017); He et al. (2017, 2015); Redmon and Farhadi (2018). However, even though the expression "*deep learning*" is relatively new, this class of methods dates back to the 1940s Goodfellow et al. (2016). It has been re-branded multiple times through the years, having been called *Cybernetics* in the 1940s-1960s McCulloch and Pitts (1943); Widrow and Hoff (1960), *Connectionism* (or *Parallel distributed processing*) in the 1980s-1990s McClelland et al. (1986); Rumelhart et al. (1986) and finally, *Deep learning* (or *Artificial Neural Networks*) from the 2000s to the current days.

While at the beginning these methods were more related to the neuroscientific field Rumelhart et al. (1986); Widrow and Hoff (1960), being, e.g., inspired by the brain's neurons for the design of the first perceptrons Rumelhart et al. (1986), now "*It* [deep learning], *appeals to a more general principle of learning multiple levels of composition* [of functions], *which can be applied in machine learning frameworks that are not necessarily neurally inspired*" Goodfellow et al. (2016). Given this definition, many recently proposed algorithms can be numbered as belonging to the deep learning area. However, in this thesis I mainly focus on *Convolutional neural networks (CNNs)* LeCun et al. (1989) as they are at the basis of most of the latest object detection algorithms (see Sec. 3.2 for an introduction to these networks).

CNNs have been originally introduced to address the task of handwritten zip code recognition LeCun et al. (1989) in 1989 and after that, in the 1990s-2000s, they have been applied to a wider variety of tasks, such as face Garcia and Delakis (2002); Osadchy et al. (2007); Vaillant et al. (1994), hands Nowlan and Platt (1995), text Delakis and Garcia (2008) and pedestrian Sermanet et al. (2013) detection. However, the real breakthrough is usually dated back to 2012 when the network Alexnet Krizhevsky et al. (2012a), named after his author, won the ImageNet competition Deng et al. (2009), outclassing previous winner models of 10% points or more (see Fig. 4.2). From that moment on, deep learning based approaches surpassed hand crafted features and standard machine learning approaches in most of the applications and computer vision challenges, including object detection, gathering the attention of the scientific community. This incredible breakthrough can be in

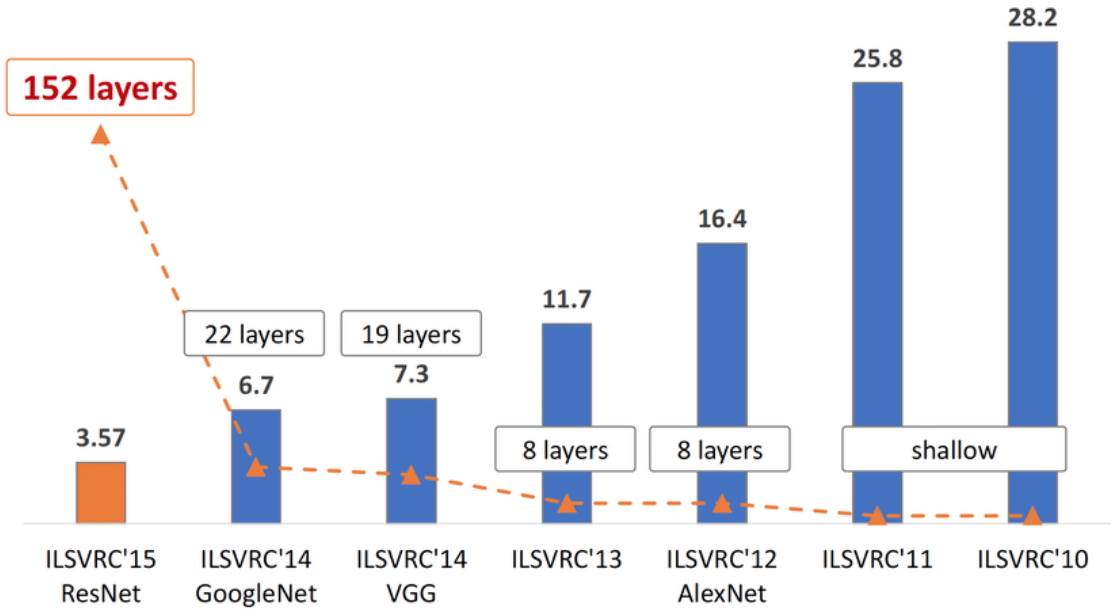


Figure 4.2: The evolution of the winning entries on the ImageNet Large Scale Visual Recognition Challenge from 2010 to 2015 (picture taken from Nguyen et al. (2017)).

fact motivated by two main factors Goodfellow et al. (2016): (i) the availability of larger annotated datasets Deng et al. (2009); Everingham et al. (2010); Lin et al. (2014) and (ii) the extremely rapid growth of machine capabilities in terms of memory and computation of the last years.

Regarding the object detection task, after 2012, CNNs have been widely adopted and used as feature extractors integrated into a more complex "meta-architecture" Huang et al. (2016). Since the earliest CNN based approaches after 2012, two main trends can be identified: the (i) *Single-stage detectors* and the (ii) *Multi-stages detectors*. In the remaining of this section, I describe main features of both approaches, providing significant examples (see Sec. 4.4.2 and 4.4.1).

4.4.1 Multi-stage object detectors

Algorithms belonging to this group perform detection by splitting the problem into three sub-tasks: (i) the image features extraction (by the *Feature extractor*) (ii) the region proposals (also called *Regions of Interest (RoIs)*) generation and finally (ii) the region classification and refinement (by the so called *Detection network*). The previously mentioned work LeCun et al. (1989) can be counted as the first deep learning based multi-stages object detector as it relied on (i) a first step of candidate regions generation, in a sliding window fashion and on (ii) a convolutional neural network used for per region feature extraction and classification.

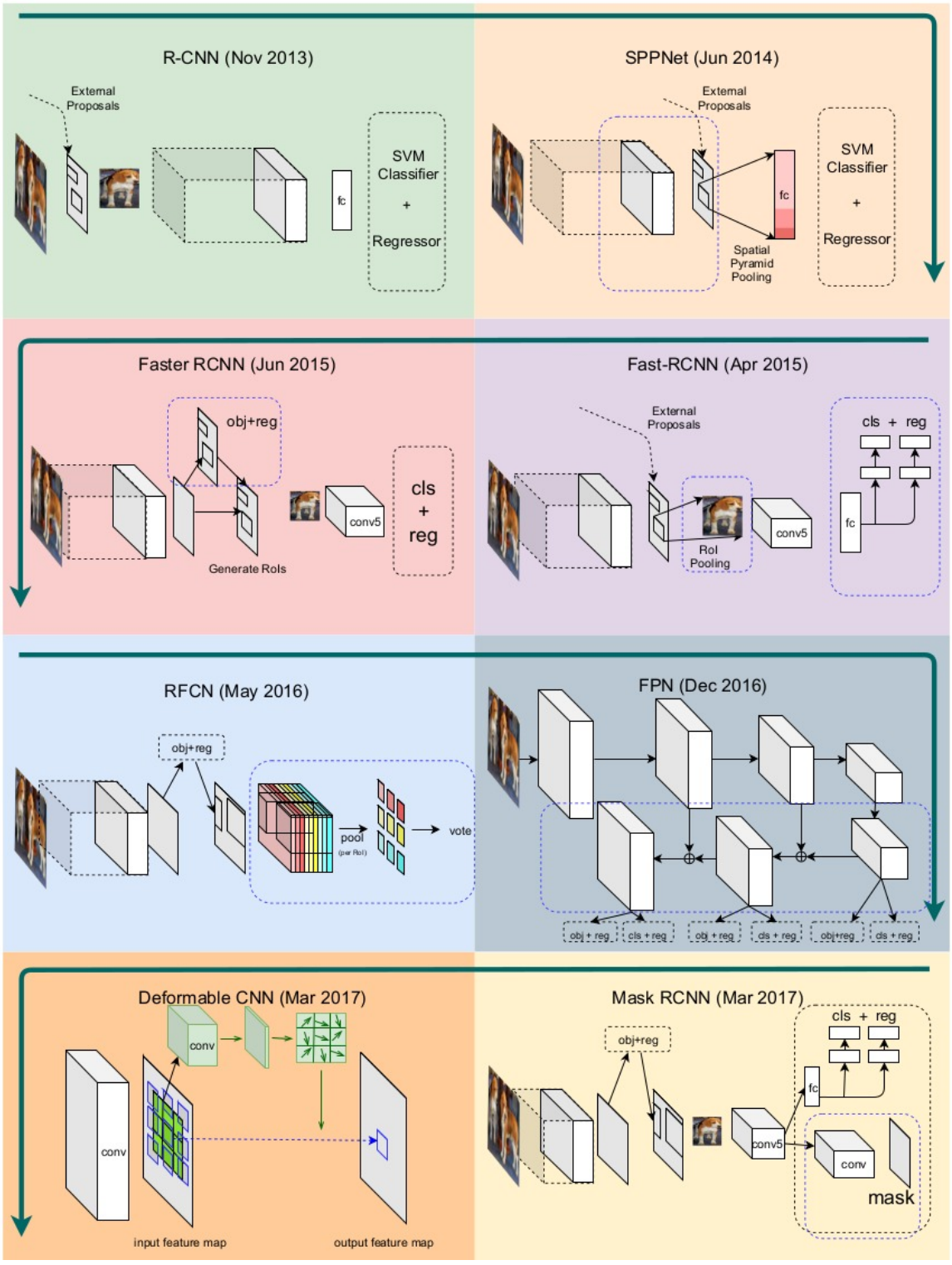


Figure 4.3: Overview of main multi-stages object detectors (picture taken from Agarwal et al. (2018)) (more details about specific architectures in Sec. 4.4.1).

In the subsequent paragraphs, I provide detailed descriptions of some of the main multi-stage object detectors, please refer to Fig. 4.3, for a more complete road-map of the main algorithms belonging to this group and for a glance of their architectures.

Region-CNN

A step closer to more recent approaches, with respect to LeCun et al. (1989), is represented by the *R-CNN* Girshick et al. (2014). This approach replaced the sliding window with more sophisticated algorithms for region proposals generation so to exclude as many background regions as possible and reduce computation. Originally, the proposed algorithm relied on the Selective Search Method Uijlings et al. (2013) for this preliminary step, but other equivalent approaches have been subsequently attempted, e.g. EdgeBoxes Zitnick and Dollár (2014), DeepMask Pinheiro et al. (2015, 2016) etc.

In R-CNN (please, refer to Fig. 4.3 for a pictorial representation), a region proposal system finds a set of regions in the image which have a high probability of containing one of the objects of interest. These regions are then cropped from the image, warped to a 7×7 matrix and fed into a CNN which is usually composed by a set of convolutional layers and a final set of fully connected used to generate a feature vector from the convolutional map. This vector is finally fed into two separate methods, that are also called *heads*, one for predicting the class of the region (*classification head*) and one for predicting per-class offsets for refining the RoIs to better fit the object (*regression head*). The training process of this architecture is, therefore, split in two phases: (i) the fine-tuning of the CNN used for feature extraction and (ii) the training of the two final heads. In the original version, the classification head is implemented with a set of SVMs Hearst et al. (1998), while the regression head is implemented with a regressor, trained with a L_2 loss. Note that, since every region is first fed into a CNN and then classified, the per-image computation at inference time is quite high since typically the number of regions of interest per image predicted by Selective Search Uijlings et al. (2013) can be of the order of thousands.

Fast R-CNN

In Fast R-CNN, this computation problem is addressed by introducing the so-called RoI pooling layer which allows to encode each proposed region into a deep representation, using only one forward pass of the convolutional feature extraction layers. This new layer takes the coordinates of the RoIs and directly crops them out from the feature map of the original image, allowing for computation to be shared for all regions as the convolutional base is not run for each region but only once. The other major difference with respect to the R-CNN is

the introduction of a multitask loss, enabling the classification and regression heads to be trained simultaneously, together with the CNN. This new multitask loss is given by

$$L_{tot} = L_c + \alpha \times \lambda \times L_r \tag{4.14}$$

where, L_c is the classification loss and is the cross-entropy loss, while L_r is the regression loss for bounding boxes refinement, it is defined as the so called Smooth L_1 loss and it is built to be less sensitive to outliers.

$$Smooth_{L_1} = \begin{cases} 0.5x^2 & if \quad |x| < 1 \\ |x| - 0.5 & otherwise \end{cases} \tag{4.15}$$

Finally, $\lambda \in \mathbb{R}$ controls the balance between the two task losses (usually set to 1) and $\alpha \in \{1, 0\}$ is equal to 1 when the predicted class is not the background.

Faster R-CNN

As a result of these advancements, Fast R-CNN's training and inference times improved dramatically with respect to R-CNN. However, due to the external, and often really time consuming, region proposal step, inference time in both R-CNN and Fast R-CNN was still far from real time.

The introduction of the *Region Proposal Network (RPN)* in *Faster R-CNN* Ren et al. (2015) leaded to a significant speed-up. The RPN is a double-layer Convolutional Neural Network, trained on the data to discriminate background from potential objects. It outputs a set of RoIs and *objectness* scores, one for each RoI, which represent the probability of each region to contain an object. Note that, since it is not trained to discriminate between different object classes, the RPN learns to propose locations which are class agnostic. The main advantage of this approach is that it provides a limited number of predicted RoIs with low computation time Ren et al. (2015). Moreover, being trained on images depicting the objects of interest, the proposals are also generally more precise than the one obtained with other general purpose methods, like e.g. Selective Search. Another important concept introduced with the RPN is the one of anchors, that are reference boxes of different sizes and ratios associated to single locations. Predicted RoIs are expressed in terms of offsets of these anchors.

In Faster R-CNN, initially, the input image is fed into a CNN which extracts a convolutional feature map. Then, this feature map is fed as input to the RPN, which predicts a set of RoIs and objectness probabilities. These RoIs play the same role of the ones generated by the Selective Search, being fed, together with the feature map, to the RoI pooling layer. From

that layer up to the last ones, the architecture is the same as in Fast R-CNN, i.e. the per-region feature maps are fed to the fully connected layers and the resulting feature vectors are used for classification and bounding boxes refinement by the two final sibling layers. With respect to its previous versions, Faster R-CNN allows for a joint training of all its components (the region proposal, the feature extractor and the detection network).

Extensions and improvements of Faster R-CNN

Through the years, Faster R-CNN proved to be a very powerful baseline Agarwal et al. (2018); Huang et al. (2016); Zou et al. (2019), difficult to beat and easy to extend. For this reason, it attracted the attention of researchers to improve some of its parts or build upon it.

In Faster R-CNN, after the RPN stage, each region proposal is cropped from the feature map by the RoI pooling layer and resized to be fed to the fully connected layers in the detection network, which performs computation separately for each region, leading to a computing overhead. To avoid this problem, the *Region-FCN (Fully Convolutional Network)* architecture Dai et al. (2016b) was proposed. As the name suggests, this method got rid of the fully connected-layers, making Faster R-CNN a fully convolutional network. Instead of cropping each region proposal out of the feature map, the R-FCN model inputs the entire feature map into the regression and classification heads. These are implemented in this case as two convolutional layers with z_r and z_c channels, respectively. $z_c = k \times k \times C$, where C represents the number of classes and k is the size of a side of the $k \times k$ spatial grid describing relative positions in the image. For example, with $k \times k = 3 \times 3$, the 9 score maps encode the cases of top-left, top-center, top-right, ..., bottom-right of an object category (please, refer to Fig. 4.3 for a pictorial representation). $z_r = k \times k \times 4$, where 4 is the number of box offsets. Beyond the significant computational speed-up, using a fully convolutional network also brought an improvement against scale variation. Indeed, images of various sizes can be fed into the network without modifying the architecture due to the absence of the fully connected layers.

The problem of scale variation, partially addressed with the R-FCN, was originally tackled in the first version of Faster R-CNN, by feeding the network with various re-scaled versions of the same image. This approach was not ideal as the network ran through one image multiple times, making the object detector slower. Another proposed approach to address this problem was the *Feature Pyramid Network (FPN)* Lin et al. (2017b). In the original Faster RCNN architecture, a single feature map is computed and fed to the classification and regression heads, whilst in the FPN, multiple feature maps taken at different levels of the CNN are used. This should allow to consider different scaled descriptors of the same image.

Specifically, this is implemented by a procedure that is called *lateral connection* (see Lin et al. (2017b) for more details).

Finally, the last architecture that is worth mentioning is *Mask R-CNN* He et al. (2017). Mask R-CNN is the latest evolution of Faster R-CNN, extending it to additionally address the object segmentation task. The major improvements, regarding the object detection performance, were the integration with the FPN and the substitution of the RoI pooling layer with a more precise process called *RoI align* He et al. (2017). More details about this architecture are given in Sec. 4.6.

4.4.2 Single-stage object detectors

In so called *Single-stage object detectors* (or alternately called *Grid-based detectors*), for each image, classifiers are applied over a regular, dense sampling of possible object locations, scales, and aspect ratios. Their distinctive characteristic is that algorithms of this group solve the detection problem without performing a preliminary step of region proposals generation. This typically leads to faster inference times but also possibly lower accuracy Huang et al. (2016). The previously mentioned VJ Detector Viola and Jones (2001) was one of the first single stage object detectors. More recently, other grid-based approaches have been proposed like *YOLO* (You Only Look Once) Redmon et al. (2016); Redmon and Farhadi (2016, 2018) and *SSD* (Single-Shot MultiBox Detector) Liu et al. (2015). These approaches are described in the following paragraphs, while another single-stage detector, i.e. RetinaNet Lin et al. (2017b), will be introduced in Sec. 4.5.

YOLO

YOLO Redmon et al. (2016); Redmon and Farhadi (2016, 2018) was the first single-stage detector in the deep learning era. It applies a single neural network to the full image. The network is built, in the original version, as the composition of 24 convolutional and 2 fully connected layers. The image is divided into a squared grid of cells. A grid cell is supposed to detect an object of interest if it contains its center. Moreover, each cell predicts a number B of bounding boxes and confidence scores and C class probabilities conditioned on the *objectness* of that cell (where C is the number of classes and the *objectness* is the probability of that cell containing any object). The original version of YOLO Redmon et al. (2016) suffers from a drop of the localization accuracy compared with two-stage detectors, especially for small objects. Later versions Redmon and Farhadi (2016, 2018) and the later proposed SSD Liu et al. (2015), tried to address this problem.

SSD

Taking inspiration from Faster R-CNN's anchors Ren et al. (2015), *SSD* Liu et al. (2015) used reference boxes of various sizes and aspect ratios to predict object instances. Additionally, the method combines predictions from multiple feature maps, taken from different levels of the network, with different resolutions. These two main differences, with respect to YOLO, help in increasing invariance to various object sizes, addressing the problem affecting the original version of YOLO, with small objects.

4.4.3 Learning strategies

A common trend in recent works is to integrate all the architecture's modules into "monolithic" models, learned end-to-end via backpropagation Dai et al. (2016a); He et al. (2017); Ren et al. (2015). This presents obvious advantages Glasmachers (2017) such as the possibility of updating, with a single backpropagation pass, all the weights from both the backbones and the heads of the network, leading to a relatively small amount of hand-engineered steps, exploiting just the structure of the data to learn the weights. Moreover, from an application point of view, at inference time, it allows to consider the network as a system with only unique points of input and output, making the design of the experiments easier. Nevertheless, reducing the amount of hand-engineered components might lead to the necessity of bigger quantities of data and longer training time, to automatically optimize all the parameters Glasmachers (2017). Moreover, considering just one source of input and one source of output, might reduce, in some cases, working flexibility and control, since usually, guarantees of convergence are not provided.

One possibility to reduce the requirements in terms of data and training time, also used in robotics Pasquale et al. (2019); Schwarz et al. (2015), is to *fine-tune* the network, applying some *transfer learning techniques*. These strategies usually allow to start the learning process on the task at hand, from pre-trained weights with a *warm re-start*, instead of randomly initializing them and learning them from scratch. It has been shown, indeed, that starting from a pre-existent knowledge, allows to drastically reduce the amount of data required to optimize all the parameters of deep networks Pasquale et al. (2019); Schwarz et al. (2015). For this pre-train the available datasets from the computer vision (like e.g. ImageNet Deng et al. (2009), Pascal VOC Everingham et al. (2010), COCO Lin et al. (2014), etc.) or robotics communities (like e.g.iCubWorld Transformations Pasquale et al. (2019),YCB Calli et al. (2015), IGLU MHRI datasetAzagra et al. (2017), etc.) can be considered. A well established procedure, in standard computer vision but also in robotic vision settings, is to start the training from networks pre-trained on the ImageNet Large-Scale Visual Recognition

Challenge (ILSVRC) 2012 Deng et al. (2009). However, even though fine-tuning deep networks allows to save time and data, the learning process can be still quite heavy.

End-to-end learning represents quite a straightforward strategy for one-stage detectors, however, another natural approach, especially considering multi-stages architectures (Sec. 4.4.1), is to train each step of the pipeline separately Girshick (2015); Girshick et al. (2014). This allows to have more flexibility and control over the learning of each part. The two major recent examples of multi-step training are R-CNN and Faster R-CNN (see Sec. 4.4.1 for a description of the two architectures).

In R-CNN, the learning is split into two phases: (i) firstly, the CNN feature extractor is fine-tuned on the task at hand, (ii) then, the classifiers and the bounding boxes regressors are optimized on the same task. Faster R-CNN, instead, even allowing for an end-to-end learning, was originally proposed with the so called *4-Steps Alternating Training procedure*. With this process, the network is fine-tuned with a 4-steps pipeline that alternates the optimization of the RPN and the Detection network. Specifically, in the first two steps respectively, the RPN and the Detection network are learned from scratch, while the shared convolutional layers and the fully-connected layers are fine-tuned, starting from weights previously trained on ImageNet Deng et al. (2009). In the two latter steps, instead, the shared convolutional layers are frozen and the weights of the RPN and of the Detection network, obtained during the previous steps, are fine-tuned on the target detection task.

For the design of the detection system proposed in this project, I considered a multi-stage architecture, trained with a two-step pipeline: a first off-line step on the available dataset and a second on-line step on the task at hand. This procedure allows to (i) learn (during the off-line step) the weights of the CNN and of the RPN to extract general features and region proposals and to (ii) train (during the on-line step) the kernel based detection network on the specific requested task when the data comes, allowing for faster adaptation.

4.5 Foreground-background imbalance, a statistical and computational problem

The issue with all types of approaches and learning procedures is that, in order to predict the locations of the objects of interest, a huge amount of candidate regions are required to be visited. Each candidate region is treated as a positive or negative example, and is used to train the classifiers. Since most candidate regions typically originate from background areas (indeed, usually an image contains mostly background), the training set associated to object detection tasks is typically very large and unbalanced. The size of the training set

makes training a computationally intensive task, while the fact that positive examples are underrepresented may bias the prediction if not treated properly Oksuz et al. (2019); Pang et al. (2019).

In the literature, different solutions have been proposed to deal with these issues. In particular, Lin et al. (2017b) recently proposed a novel loss function, called Focal Loss, which can be adopted to down-weight easy negative examples so that their contribution to the total loss is re-balanced, in case their number is large. Such a loss is designed for end-to-end training of one-stage detectors (the authors proposed the one-stage *RetinaNet* detector to show its effectiveness Lin et al. (2017b)) and has been demonstrated to improve performance not only for image (2D) inputs, but also 3D data Yun et al. (2019).

Solutions for region-based approaches, instead, are based on the idea of shrinking the set of negative examples by keeping only the hard ones (i.e., the ones that are difficult to classify). Early solutions were based on Bootstrapping Sung (1996), which is still used in modern detection methods, under the name of Hard Negatives Mining Felzenszwalb et al. (2010b); Girshick et al. (2014). This is an iterative approach that alternates between training the detection model given a current set of examples, and using that model to find new hard negatives to add to the bootstrapped training set. Lately, Shrivastava et al. (2016) proposed to "embed" the selection of hard negatives into the Stochastic Gradient Descent (SGD) method used to train Fast R-CNN Girshick (2015). This technique, called On-line Hard Example Mining (OHEM), only uses high-loss region proposals for the SGD iteration, rather than a heuristically sampled subset.

Finally, works that rely on a "cascade" of detectors Felzenszwalb et al. (2010a); Viola and Jones (2001) are based on the same concept of reducing the set of negative regions to be processed in an image. These methods, first apply lightweight classifiers to discard easy background regions, subsequently they apply increasingly more complex classifiers only on regions that are most likely to contain objects.

All these solutions are not designed for on-line learning, because they either require to iteratively visit all negative examples in the training set Girshick et al. (2014), or they rely on end-to-end backpropagation Shrivastava et al. (2016). For the design of the detection system of this thesis project a novel approach Maiettini et al. (2018) has been proposed which main aim is to (i) select hard negatives, by implementing an approximated speeded-up bootstrapping procedure, and (ii) account for the imbalance between positive and (hard) negative regions, by relying on a recently proposed scalable Kernel approach, namely FALKON Rudi et al. (2017).

4.6 From object detection to instance segmentation

The task of *object segmentation* aims at classifying all the pixels in an image identifying different instances of the categories of interest. Thus it can be interpreted as a finer object detection where localization is not at the level of a surrounding bounding box but of pixels belonging to the object. Note that, this task is different from the *semantic segmentation*, which has the similar objective of classifying the single pixels of an image but not being aware of the different instances.

While the field of semantic segmentation is mainly dominated by approaches based on fully convolutional networks (FCNs) Long et al. (2015); Shelhamer et al. (2017), a common trend for object segmentation Dai et al. (2015, 2016a); Ranjan et al. (2017) is to extend existing multi-stage object detection pipelines, splitting the task in four main stages: (i) a convolutional feature map is generated by an FCN applied on the whole image, (ii) a list of RoIs is generated by a region proposal method, (iii) an RoI pooling layer warps each RoI and extracts per-RoI features from the convolutional map and finally, (iv) fully connected layers predict a mask for each feature map. However, in such methods the feature warping and resizing, happening during the RoI pooling step, can destroy spatial details. This is due to the two quantizations preformed in the RoI pooling layer which first aligns the proposal to the feature map (first quantization) and then pools (with a max pooling or an average pooling layer) the features (second quantization). Note that, some information might be lost because the original proposal may not be a multiple of the final map's dimensions. This may not affect the classification and only partially affect coarser bounding box localization, but it has a large negative effect on mask prediction, which is measured on a pixel level.

This problem is addressed in *Mask R-CNN* He et al. (2017), that substitutes the RoI pooling layer with the so called *RoI Align* which directly samples the N final cells from the original map (where N is the dimension of the size of regions warped from the RoI pooling layer), avoiding any lossy quantization. Mask R-CNN extends the two-stage architecture of Faster R-CNN, tackling the object segmentation task by adding, in parallel to the final layers for classification and bounding box regression, a further output layer for binary mask prediction. Note that, this is in contrast to latest approach Dai et al. (2016a); Li et al. (2017); Pinheiro et al. (2015) where, instead, the classification depends on the prediction of the masks. The multi-task loss presented in Ren et al. (2015) is extended to also consider the loss computed on the predicted masks, L_{mask}. Note that, L_{mask} is such that, for an RoI associated with a ground-truth class k, the considered contribution to the loss is only defined on the predicted k^{th} mask, decoupling masks and classes prediction. Finally, another important difference with respect to the original Faster R-CNN architecture is the stable integration of the FPN in the feature extractor.

I contributed in a master thesis project with the aim of extending the on-line object detection system proposed in this thesis to also perform object segmentation (please refer to Sec. 7.5 for further details).

4.7 Object detection datasets

Datasets releases during the last years went alongside with the improvements achieved by the object detection algorithms in a virtuous circle. Indeed, the availability of shared data collections is fundamental in the research community and the reason is twofold. First of all, sharing training and test sets makes methods comparison easier, allowing for new baselines establishment. Moreover, data availability reduces the algorithms development time, especially considering the ones that fall in the supervised learning framework, allowing the research community to only focus on the design of the methods. This aspect is fundamental, especially for the tasks for which creating and annotating a new dataset might be complicated or expensive. For example, in the case of object detection, both the phases of collection and annotation are challenging.

Regarding the data collection, one main aspect to consider is that the samples in a training set, used for supervised learning algorithms, need to be representative of the distribution they are extracted from. This means that in the case of multiple objects detection the dataset needs to be balanced, i.e., all the objects of interest have to be represented by a comparable number of samples. Moreover, this means that the available samples should cover the majority of the possible appearance conditions for each object (e.g., different view poses, backgrounds, scales, light conditions, etc.). This might lead to an enormous effort in the phase of data collection as all these appearance conditions need to be produced and recorded.

During the annotation phase, instead, the annotator has to provide both the label and the location for each object in each image of the collected dataset. While providing the label of an object might be relatively easy and fast, adding the information of the location leads to longer annotation times per image and entails some aspects to disambiguate. In fact, the objects might be represented in cluttered scenes or just partially visible, therefore an annotation policy has to be established, which should assess the percentage of an object that needs to be visible in order to consider it as a true positive in that image. Moreover, sometimes it might be unclear how specific objects have to be labeled and necessitates to be clarified (e.g., whether or not to include the stem of a flower in the bounding box, or the shadow of an object). Finally, the tightness of the bounding boxes around the objects of interest needs to be established as well, in order to have an uniformly annotated dataset.

For the aforementioned reasons, the availability of shared dataset is critical for achieving fast research advancements. The most commonly used object detection datasets in the computer vision community are huge general purpose images collections. They depict everyday life objects, aiming at representing, as much as possible, all the various aspects of reality. The major ones that fall in this category are: (i) the Pascal VOC Everingham et al. (2010), (ii) the MS Common Objects in Context (COCO) Lin et al. (2014), (iii) the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) Deng et al. (2009), (iv) OpenImages Kuznetsova et al. (2020) and the Cityscapes Dataset Cordts et al. (2016). Moreover, lately, domain specific datasets have been released, like e.g., the ones for the aerial imagery Heitz and Koller (2008); Xia et al. (2018), faces Jain and Learned-Miller (2010), pedestrians Dalal and Triggs (2005); Geiger et al. (2012), logos Eggert et al. (2017) and traffic signs Houben et al. (2013), cars Geiger et al. (2012) and domestic objects Damen et al. (2018). A special mention is worth for the robotics datasets which are often acquired from a "robo-centric" point of view and are typically smaller, representing a reduced number of object instances but also providing richer views and perspectives of the objects of interest Calli et al. (2017, 2015); Lai et al. (2014); Pasquale et al. (2019); Singh et al. (2014).

Aside the great contribution of these available data collections, they also entails two major issues. First of all, needing an application specific object detector, these datasets, even thought for certain domains, close to the one of the required application, might still be too generic and thus either unusable or not sufficient to train a robust object detector. This topic will be covered more in details in Chapter 5. Moreover, datasets of all types are affected by the well known problem of dataset bias Mancini et al. (2018); Tommasi et al. (2017); Torralba and Efros (2011). As the name suggests, this issue concerns with the fact that each dataset, being a finite set of samples, ends up representing just a part of the real world, leading to train models that are biased and that will fail or drop in performance on test sets significantly different with respect to the training sets Torralba and Efros (2011). Therefore, even by exploiting the available datasets for training detection models, a further effort is required to maintain optimal performance in different scenarios by either modifying the model to be as general as possible or, if possible, by collecting new data to refine the models on the setting at hand.

4.8 Object detection, the robotics perspective

Perceiving the objects in the surrounding environment represents the first step of many, more sophisticated, robotic tasks Hernández-González et al. (2016); Schwarz et al. (2017); Vezzani et al. (2018); Zeng et al. (2017); Zhu et al. (2014). For this reason, studying and improving

object detection systems plays a fundamental role in the development of more complex robotic pipelines.

The class of deep learning based algorithms for object detection, described in the previous sections, represents an appetizing list of possibilities roboticists could choose from, resulting from the stunning performance achieved on computer vision benchmarks. Nevertheless, obtaining very good accuracy on general purpose datasets like COCO or Pascal VOC, is not a sufficient guaranty that these methods can be used as building blocks in a robotic application, more application specific tests should be performed Sunderhauf et al. (2018). Indeed, it is often the case that deep learning models downloaded from the web need to be adapted to the task at hand to perform well Pasquale et al. (2019).

4.8.1 State-of-the-art in the Amazon Picking Challenge

Examples of deep learning based methods applied to real robotic scenarios can be found in works proposed for existing challenges, like the Amazon Picking Challenge [3](APC), where the robots are required to pick objects in a cluttered bin, a classical problem in robotics. These works mainly focus on improving robustness and precision in particular scenarios like occlusions in clutter Georgakis et al. (2017); Harada et al. (2016); Holz et al. (2015); Kaipa et al. (2016); Schwarz et al. (2018); Tobin et al. (2017); Zeng et al. (2018). The main proposed solutions follow a standard two-steps approach: (i) object detection and pose estimation followed by (ii) model-based grasp planning. For instance, Jonschkowski et al. (2016) proposed a two-step method where object segmentation is performed over hand-crafted image features to compute grasping proposals for picking objects with a vacuum. However, more recent approaches widely use deep learning based algorithms for the first step of generation of bounding box proposals or segmented masks Hernández-González et al. (2016); Schwarz et al. (2017); Zeng et al. (2017), usually followed by an object pose estimation that allow to compute the best grasp to pick the objects. Moreover, a common hypothesis is that 3D models of the objects are known and available during training and/or inference. These can be used to refine both the localization and the recognition and help at grasping time Holz et al. (2015); Pretto et al. (2013).

However, these hypotheses can work for robots functioning in specific, highly constrained scenarios, with a limited set of objects, meeting specifications known beforehand (regarding volume, size and weight) and in controlled workspaces. However, these hypotheses might not hold for robots that are supposed to interact in unconstrained and continuously changing scenarios, which are requested to adapt quickly to novel, possibly very different tasks, like

[3]http://amazonpickingchallenge.org/

e.g. humanoids Metta et al. (2010); Parmiggiani et al. (2017). Therefore, different, more flexible, solutions are needed.

In this thesis, I tackle this problem, devising solutions for the realization of a flexible and easy to adapt detection system. Note that, the same unconstrained setting has been considered, in the past, for the application of object recognition systems Camoriano et al. (2017); Pasquale et al. (2019, 2016a). However, extending those solutions to the multi object detection case is a difficult problem, due to the differences between the two tasks and the intrinsic challenges entailed by the localization (see Sec. 4.1 and 4.5). Thus ad-hoc solutions are required to be designed.

4.8.2 Active perception

With respect to standard computer vision, having a robotic platform allows to exploit its embodiment in the surrounding scenario, opening pathways for interaction and autonomous exploration. While in Sec. 5.4.1, I describe how this aspect can be exploited for training data acquisition, in this section, I discuss about another related research field, regarded as *Active Perception* (also referred to as *Active Vision*) Aloimonos et al. (1988); Bajcsy (1988); Bajcsy et al. (2018); Chen et al. (2011). Active Perception is defined as the task of determining the pose and configuration for the visual sensor to improve object detection performance by refining or gaining confidence on the predictions at inference time. It clearly plays an important role in robotics since a robot's sensor is able to see just a portion of a scene from a single viewpoint, having typically a limited field of view. Moreover, generally, it is not possible to reconstruct a global description of objects from just one point of view. However, robots, as autonomous agents, can take intentional actions in order to explore the environment, according to a goal such as obtaining additional information about an object. Nevertheless, difficulties can arise due to sensor noise and the presence of obstacles in the workplace, that make a strategic plan necessary for e.g., navigating through an office or moving own body parts to get different views of an unknown object.

In the past years, a great amount of research has been proposed in this field Bajcsy et al. (2018); Browatzki et al. (2012); Chen et al. (2011); Grotz et al. (2019). Specifically, for humanoids, the literature present different solutions to solve the related problem of the search of the so called *Next best view (NBV)* Connolly (1985). For instance, Kahn et al. (2015) proposed a trajectory optimization method to explore new and occluded regions for robotics grasping. Moreover, Potthast and Sukhatme (2014) presented a general next best view system for a humanoid to explore a scene, specifically designed for occluded environments. While in this thesis the embodiment of the robot has been exploited mostly for acquiring training images, the Active Perception field might be of interest for future research directions.

Chapter 5

Weakly Supervised Learning

The supervised learning framework presented in Chapter 3 and used by a great majority of the object detection methods described in Chapter 4, turned out to be impressively effective in many research fields. However, the assumption behind this framework is the availability of a training set S_n, which elements have been sampled from a data space Z, accordingly to an unknown probability distribution ρ. Moreover, this dataset has to be large enough to be representative of the whole data space, allowing the learned model to generalize on new samples from the same distribution. This is quite a strong assumption, especially considering domains such as robotics.

Available annotated datasets can be found on the web for various research fields, but while they can be perfectly used as benchmarks or to train general purpose models, they are not enough for specific applications where very specific tasks may be required (see Sec. 4.7 for an overview of the main object detection datasets). For instance, while the ImageNet dataset Deng et al. (2009) might contain plenty of images depicting mugs, it probably lacks frames of the specific mug that we want our robot to identify, among all the others, and interact with it. One possibility to address this problem with "brute force", is to collect the necessary dataset and create correspondent annotations manually. In the case of object detection, this means acquiring images depicting the objects of interest from different appearance conditions. Then, manually draw a bounding box surrounding each object of interest in the acquired images, specifying the exact label. This approach, however, does not scale well in cases of big tasks, as the process of manual annotation takes very long time. Moreover, it is not applicable to all those situations where the task at hand is not decided a priori, but it can change in time, due to the fact that the acquisition and annotation process should be done off-line before training the model. For these reasons, the "brute force" approach cannot be applied in many cases and other solutions need to be devised.

Successful research directions to overcome this problem can be either (i) to rely on images taken from the web Mancini et al. (2019); Molinari et al. (2019) or (ii) to use synthetically created training sets, i.e. by augmenting the available information, synthesizing new images Dwibedi et al. (2017). Recent advancements in this latter field have brought interesting results addressing the main issue of these methods, i.e., how to transfer the knowledge learned from synthetic data to real images. Nonetheless, considering the robotic setting, all the aforementioned approaches, while being effective, do not explicitly exploit the robot's capability of actively and autonomously acquiring training samples, selecting those that are more relevant for the task at hand.

A possible solution, investigated in this thesis, is the so called *weakly supervised learning* problem. Being the subject of interest in many applied fields, like e.g. robotics, this framework has attracted the attention of current research. In the remainder of this chapter, I start by formalizing the definition of this problem (Sec. 5.1) then, I present an overview of some approaches which are relevant for robotics and specifically for this thesis (Sec. 5.2, 5.3 and 5.4).

5.1 Problem definition

The supervised learning framework, presented in Sec. 3.1, assumes the availability of a training set $S_n = \{(x_1, y_1), ..., (x_n, y_n)\}$, typically composed by n couples of values (the x) and corresponding labels (the y). However, this assumption may not be always fully satisfied, especially in the scenarios where acquiring a sufficient annotated dataset is difficult, expensive or even impossible and only a partial (weak) supervision might be available. The framework that deals with such scenarios is the *weakly supervised learning* Hernández-González et al. (2016); Zhou (2017) and, depending on the type and characteristics of the supervision available, it can assume different shapes. More specifically, a first distinction that we can find in the literature about weak supervision Hernández-González et al. (2016) regards the phase of the learning pipeline presenting a partial labeling: (i) at learning time or (ii) at inference time.

5.1.1 Weak supervision during inference

In real world applications it is often the case that additional information (even if incomplete or inexact) can be provided to an already trained model, in order to help the prediction. For instance, for object or person identification tasks from images in real settings the important assumption of uniqueness holds Kuncheva (2010) or in some scenarios it is

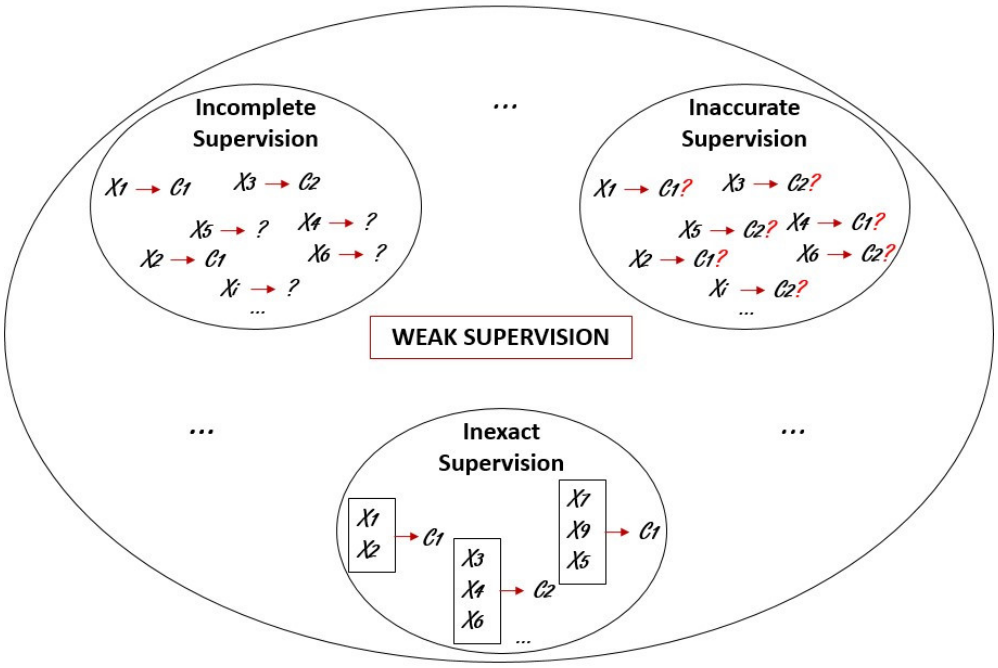


Figure 5.1: This figure shows a pictorial overview of three possible weakly supervised scenarios. It is worth mentioning that this partition is not exhaustive and combinations of these settings can have a practical relevance.

possible to reduce or even set to zero the probability of some labels by reasoning on contextual information Cour et al. (2011). While working on this thesis, I mainly focused on the problem of lack of supervision at learning time, however, extending the work by exploiting additional information at inference time is surely worth to be considered for future research.

5.1.2 Weak supervision during training

By following the distinction made by Zhou (2017), we can distinguish between three classes of weakly supervised learning problems which are detailed below (see Fig. 5.1 for a pictorial overview).

- *Inaccurate Supervision*. A weakly supervised learning problem is called inaccurate supervision when the supervision information is not always ground-truth and some labels are corrupted by noise. The training set S_n is the same as in the supervised learning problem, but the labels y_i may be incorrect due to the noise.
  A practical example of incorrect supervision in object detection, is represented e.g., in situations where annotated bounding boxes of the training set might be imprecise or labels might be incorrect.

- *Inexact Supervision*. A weakly supervised learning problem is called inexact supervision when some supervision information is given, but not as exact as desired. Typically, only coarse-grained label information is available. More formally, the training set for this problem can be written as $S_n = \{(X_1, y_1), ..., (X_m, y_m)\}$, where $X_i = \{x_{i,1}, ..., x_{i,t_i}\} \subseteq \chi$ is called *bag*, $x_{i,j} \in \chi$ (with $j \in 1, ..., t_i$) is a single instance and t_i is the number of instances in the bag X_i. Finally, $y_i \in Y$. In the standard definition of this problem, X_i is associated to the label y_i if there exists an $x_{i,p}$ in X_i which is positive for that class, where $p \in \{1, ..., t_i\}$ is unknown (in Sec. 5.3 other variants of this problem are introduced). Note that if $t_i = 1 \; \forall i$, then the problem falls in the supervised learning framework. The problem of inexact supervision is also called *multi-instance learning* as the training set S_n can be also seen as the association between a bag of instances (each X_i) with a label.
  A practical example of inexact supervision in object detection, is represented for instance, by those situations where no bounding boxes annotations are provided, but the images are annotated only with the list of labels of the objects which depict. In this case, X_i represents the i^{th} image, $x_{i,j}$ is the j^{th} location in the i^{th} image and y_i is the list of labels of the objects which are depicted in a subset of all those regions.

- *Incomplete Supervision*. A weakly supervised learning problem is called incomplete supervision when the available training set is large, but only a small portion is labeled, which is not sufficient to train a good model. More formally, the training set for this problem can be written as $S_n = L \bigcup U$, where $L = \{(x_1, y_1), ..., (x_l, y_l)\}$ is the labeled subset of the training set and $U = \{x_1, ..., x_u\}$ is the unlabeled part. It usually happens that $|L| << |U|$. Methods addressing this problem concern on how to use the unlabeled part of the dataset. Note that, an extreme case of incomplete supervision is when $L = \emptyset$ where no labels are provided, while if $U = \emptyset$ then it falls in the supervised learning framework.
  A practical example of incomplete supervision in the case of object detection is represented by those situations where only a part of the images are annotated, while a second part is available but not labeled.

The literature about weak supervision considers either one of the aforementioned scenarios or their combinations. In the following sections, I describe some state-of-the-art approaches to address them, by focusing particularly on their application to the task of object detection and robotics.

5.2 Learning from inaccurate supervision

Learning from inaccurate supervision, i.e. from noisy data, has attracted both theoretical and practical studies. From the theoretical point of view, many works assume random classification noise, i.e., labels are subject to random noise Angluin and Laird (1988); Blum et al. (2003); Gao et al. (2016) and they prove results that ensure convergence properties in those conditions.

From a more practical point of view, this problem assumes a great importance. Indeed, it might often happen that errors during data acquisition are inserted by systematic or random noise of the employed sensors or by human's mistakes due to, e.g., inattention or inexperience. Specifically, this latter case has become more relevant in recent days since it is a growing practice to rely on *crowdsourcing* to gather annotated data Brabham (2008); Kovashka et al. (2016). This paradigm is used as a cost-saving way to collect labels for training sets and it is commonly exploited in the field of object detection too Su et al. (2012). It is based on the idea of outsourcing unlabeled instances to a large group of "labelers". A popular crowdsourcing system is *Amazon Mechanical Turk (AMT)*[1] which is a market where the user can submit a task to be completed by workers in exchange for monetary payments. These crowdsourcing tools are quite useful for those who need to obtain annotated data saving costs (both in terms of money and time). However, people working as labelers can be "spammers", assigning almost random labels, "adversaries" producing deliberately incorrect annotations or simply "inexpert" on the domain of the submitted task, assigning labels inaccurately.

A basic and general idea to address these problems is to devise methods that attempt to identify the mislabeled samples Brodley and Friedl (1999) and then try to make corrections by either removing or modifying the ones that are wrong Muhlenbach et al. (2004). Other approaches, instead, have been specifically proposed to handle noise in data obtained with crowdsourcing. For instance, some popular solutions Sheng et al. (2008); Snow et al. (2008) are based on majority voting strategies, finding theoretical support in the so called *ensemble methods* Zhou (2012). Ensemble methods (or *committee-based learning* or *multiple classifier systems*) train multiple learners to solve the same problem. Compared to standard learning approaches that try to obtain one learner from training data, ensemble methods try to obtain a set of learners, trained under different conditions, and combine them. By considering multiple models trained differently with the noisy data, the noise should be averaged out.

Specifically, for the object detection task, many approaches have been proposed to deal with noisy data. For instance, in Reed et al. (2014), the authors propose a simple approach to handle noisy labeling by adding to the usual prediction objective, used for learning a CNN,

[1] https://www.mturk.com/

the notion of *perceptual consistency*. Thus, the learner makes use of its representation of the world (implicit in the network parameters learned at each step) to predict categories for incoming inputs. This provides the learner justification to "agree" or "disagree" with the provided training label, allowing to decide whether maintaining or re-labeling the data while training. This process is structured as a bootstrap where a first batch of "reliable" annotations are used to build a seed model which is then used to evaluate other labels. Note that, more accurate annotations may lead to a better model, allowing further label clean-up in a virtuous loop, whereas discarding too many labels from the original noisy dataset, may lead to poor results. Therefore, it is important to balance the trade-off between labels and the learner's perceptual consistency.

5.3 Learning from inexact supervision

The field of inexact supervision, or multi-instance learning Foulds and Frank (2010), has a practical relevance, since gathering annotated data for complex tasks (such as object detection or segmentation) might be difficult and expensive. For instance, in computer vision it is much easier and less time consuming obtaining image-level information instead of bounding boxes or pixel-level annotations.

This field of research is mainly focused on studying algorithms and applications, while theoretical results on multi-instance learning are very rare because the analysis is quite cumbersome Foulds and Frank (2010). However, a formalization of this problem can be found in the literature Foulds and Frank (2010) and an important distinction on different approaches can be made considering the fundamental assumption on the existence of a key instance (the so called *MI assumption* Foulds and Frank (2010)). Indeed, standard approaches of multi-instance learning Dietterich et al. (1997) assume that each positive bag for a certain class, must contain an instance belonging to that class (the key instance), i.e., the standard MI assumption states that each instance has a hidden class label. However, it is noteworthy that there are other studies which assume that key instances do not exist. For instance, two common assumptions are that either (i) every instance in the bag contributes to the bag's label Chen et al. (2006); Xu and Frank (2004) or (ii) that different concepts exist, being represented by some of the instances in the bags, and all of them contribute to the definition of the label of a bag Whitehill et al. (2009).

An application of the latter case, which does not assume the presence of a key instance, is the text categorization domain. In that case, the text to analyze represents the bag and the words in the text represent the instances. It is evident that for identifying the topic of a text it might not be sufficient to find only a key word but the combination of more words

make it possible. Conversely, an important application that assumes the existence of key instances in the bags is the one mentioned above, regarding inexact supervision for object detection. In this case the only available supervision is provided in the form of image-level annotations Foulds and Frank (2010); Tang et al. (2018); Zhang et al. (2018). Some solutions to this problem are based on iterative bootstrapping processes in which the current classifier is used to select the highest-confidence boxes in each image, which are treated as pseudo-ground truth in the next training iteration. Other interesting approaches, instead, improve this iterative process by integrating it with either *self-paced learning* paradigms Kumar et al. (2010) or *curriculum learning* techniques Bengio et al. (2009). These should guide the selection of the pseudo ground-truth with the aim of avoiding model drifting due to errors in early model predictions. Similarly, another practical scenario is that of having a considerable amount of videos annotated only with the label of the contained object Prest et al. (2012). In this case, the temporal information can be exploited to extract spatio-temporal "boxes" (namely, *tubes* in Prest et al. (2012)) based on motion segmentation Brox and Malik (2010), which are likely to contain the object of interest. Afterwards, these candidates are evaluated, based on boxes similarity measures, and only the more confident ones are chosen as training samples for the object detection learning.

5.3.1 Combining inexact and incomplete supervision

A case of inexact supervision that plays an important role in applied computer vision, especially in the robotics domain, is when it is combined with an incomplete supervision. This combination depicts the very common situation where only a part of the dataset is exactly annotated on the task at hand but there is an abundant part of the training set which is labeled at more coarse-grained level. This combination is specifically important, e.g., in robotics where small amounts of labeled data for the task at hand are available but contextual information about the environment around the robot can be easily extracted with its sensors. This information is not enough to get the fine-grained labels necessary for the task at hand, however, it can be used as inexact supervision.

For this reason, the literature describes methods that explore ways to extract useful forms of supervision from the contextual information available in real world scenarios (e.g., the spatio-temporal coherence on a sequence of images). Among these, one approach is to exploit this natural structure of the visual data by training a CNN to solve an “alternative” (also called *pretext*) visual tasks, the ones provided by the available information Agrawal et al. (2015); Jayaraman and Grauman (2015); Pathak et al. (2017); Pinto et al. (2016); Pinto and Gupta (2016). These methods demonstrate that the CNN, which is trained with this

inexact supervision, can provide good features that can be subsequently fine-tuned on the available supervised training set for the detection tasks at hand.

For example, in Agrawal et al. (2015), the authors pre-train the CNN to predict the robot's ego-motion. In Jayaraman and Grauman (2015); Pathak et al. (2017), instead, the authors exploit motion clues to segment moving objects and gather a ground-truth to train a CNN to segment objects in images. Finally, in Pinto et al. (2016); Pinto and Gupta (2016) the authors devise an autonomous trial-and-error behavior for training a CNN for the task of grasp detection. In all these works, the learned CNN is shown to result into a good feature extractor for object recognition/detection tasks.

5.4 Learning from incomplete supervision

In real world applications, it is more common to have rich unlabeled dataset on which to rely on, due to the pervasive presence of cameras and sensors on robotic platforms and everyday life devices. However, without proper labeling, this data cannot be used within the fully supervised learning setting and needs to be treated properly in order to be exploited. The framework of incomplete supervision helps in this situation, handling the scenarios where only a small portion of the training set is labeled while the majority is unlabeled.

In the next sections, I overview some common techniques within this framework. Specifically, first, I describe some methods to automatically produce annotated data by either exploiting the unlabeled or the labeled part of the available training set (Sec. 5.4.1). Then, I cover two major weakly supervised learning frameworks, namely the *semi-supervised learning* (Sec. 5.4.2) and the *active learning* (Sec. 5.4.3).

5.4.1 Automatic data annotation

A possibility to address this annotation problem is to exploit the available label information to augment the dataset or to exploit some further information (if available) to obtain new data. In the following paragraphs, I give more details on these two types of solutions by overviewing some methods to synthesize datasets and by explaining opportunities offered by robotics.

Synthesizing datasets

A successful research direction to overcome this annotation problem is to rely on synthetically created datasets, i.e. by augmenting the available labeled data, synthesizing new

images Dwibedi et al. (2017); Georgakis et al. (2017); Karsch et al. (2011); Movshovitz-Attias et al. (2016); Su et al. (2015). Two major approaches are recently gaining momentum: one is based on rendering both scenes and objects Karsch et al. (2011); Movshovitz-Attias et al. (2016); Su et al. (2015) and the second one, instead, relies on the combination of images and patches from real frames to augment the data Dwibedi et al. (2017); Gupta et al. (2016). Approaches from the first group put effort on rendering the scenes and the objects as much realistic as possible, trying to ensure high quality global and local consistency. Models trained on such synthetic data struggled for a long time, to generalize to real data Chen et al. (2016); Peng et al. (2015), while recent research brought interesting results Tremblay et al. (2018). Approaches from the second group Dwibedi et al. (2017); Gupta et al. (2016), instead, focus on composing realistic images by "pasting" masks of real objects on real images. The challenge in this case is to avoid creating artifacts due to a too coarse object segmentation or an imprecise patching, which would affect local consistency of the created image and thus the learning of the model.

While these methods are becoming always more reliable, they do not make use of the unlabeled part of the dataset, which instead in domains like robotics, might represent an opportunity to select the data that is more relevant for the task at hand.

Automatic data collection in robotics

Real world use-cases challenge machine learning frameworks with specific tasks to learn with scarcity (or lack) of data. However, they may also represent a source of opportunities. Indeed, with respect to standard computer vision, having e.g., a robotic platform allows to exploit its embodiment in the surrounding scenario, opening pathways for interaction and autonomous exploration. Moreover, real world data posses a series of intrinsic properties that can be exploited, as e.g., in videos, each frame can be considered correlated to the previous and next ones with a temporal coherence.

These assumptions can be exploited to overcome the problem of only having a small portion of annotated data, for acquiring automatically labeled data in constrained and controlled settings. For instance, in Pasquale et al. (2016b), the authors relies on the interaction between the robot and a human teacher. Object labels are obtained through a speech interface, while motion and depth cues allow the robot to follow with the gaze and segment the objects and automatically assign bounding boxes. Specifically, the fundamental assumption used in the pipeline is that the shown object is represented by the closest blob of pixels. This approach was originally used to deal with the lack of annotations for the object classification problem, by considering as training samples the cropped bounding boxes Pasquale et al. (2019).

A part of the experimental analysis carried out during this thesis (see Sec. 6) shows that this kind of data can be successfully used to train object detection models as well. However, results show that the constrained scenarios used for acquisition may lead to generalization problems that can be addressed with techniques that exploit the large amount of unlabeled data that the robot is continuously exposed to. These techniques are described in the following sections as they belong to the semi-supervised and active learning frameworks.

5.4.2 Semi-supervised learning

In the incomplete supervision problem introduced in Sec. 5.1.2, the semi-supervised learning (SSL) framework Chapelle et al. (2009, 2006); Zhou and Li (2010); Zhu (2005) attempts to exploit the unlabeled instances from $U = \{x_1, ..., x_u\}$ without querying human experts.

The intuition of the benefits from using active learning techniques (see Sec. 5.4.3) in incomplete supervision scenarios, might appear clear since new data, annotated by a human, is added to the dataset. However, the reasons why SSL, i.e. using data without labels, should help training predictive models is less intuitive. Fig. 5.2 provides a grasp of the idea behind many SSL algorithms. In the situation on the left side of the image, i.e., where only the little amount of labeled data is considered, taking a decision on the test point is much harder as it lies exactly in the middle of the two distributions. However, being able to observe the unlabeled data (situation on the right side of the image in Fig. 5.2) we can predict the test data point as belonging to the second class with relatively high confidence. This is because, although the unlabeled data points are not explicitly associated to a label, they implicitly convey information about the data distribution. This is particularly appetizing, especially from a practical point of view, since it would avoid involving human effort to exploit the unlabeled data. Nevertheless, this assumption might not be always satisfied. Indeed, while the expectation is to improve model's performance due to the additional (even if unlabeled) data, in some cases the semi-supervised learning can degenerate. The intuition is that a poor initial model (trained on the labeled data) can commit errors in predictions, leading to model drifts Cozman et al. (2002); Li and Zhou (2014).

Semi-supervised learning algorithms relies on the so called *cluster assumption* and *manifold assumption* Zhou (2017). The first one assumes that data have inherent cluster structure, and thus, instances falling into the same cluster have the same class label. The second one, instead, assumes that data lie on a manifold, and thus, nearby instances have similar predictions. The idea behind both assumptions is the belief that similar data points should have similar outputs.

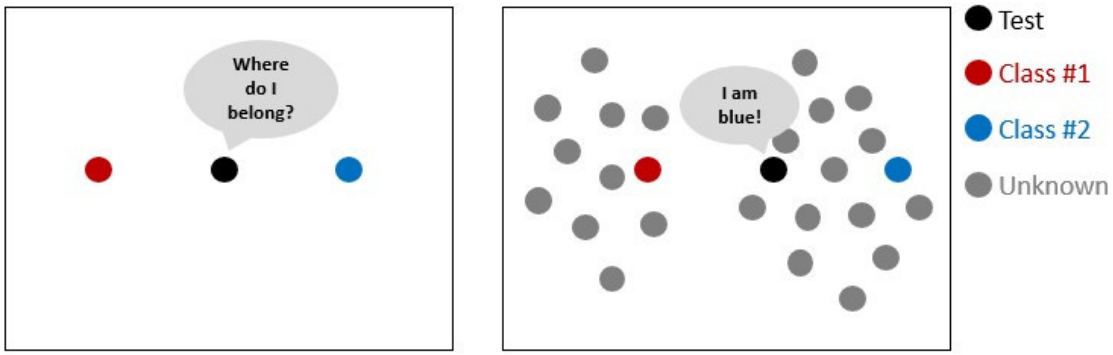


Figure 5.2: This figure shows the intuition behind the utility of the unlabeled data. Specifically, it lies in the implicit information carried about the data distribution. It gives a pictorial idea of the comparison between the two situations of considering (case on the right) or not considering (case on the left) the unlabeled training points.

The literature Zhou (2017) identifies four different categories of SSL approaches: (i) generative methods, (ii) graph-based methods, (iii) low-density separation methods and (iv) disagreement-based methods. In the next paragraphs, I briefly describe each of them.

Generative methods

This family of methods Miller and Uyar (1997); Nigam et al. (2000) is based on the assumption that both labeled and unlabeled data are generated from the same inherent model. Thus, annotation values of unlabeled instances can be seen as missing values of model parameters, and estimated by approaches such as the *expectation-maximization* algorithm Dempster et al. (1977). An important aspect of these approaches is that they need a good knowledge of the data domain, in order to compute a sufficient generative model. This assumption is fundamental in order to obtain good performance but it is not easy to obtain in real world cases.

Graph-based methods

Methods from this family Blum and Chawla (2001); Zhou et al. (2004); Zhu et al. (2003) usually encode the training set with a graph structure where each node is a training instance and each edge represents a relation between two instance (e.g. some measures of similarity). Then, the information on the available labels are propagated on the graph to the unlabeled samples. For example, in Blum and Chawla (2001) the labels are propagated within different subgraphs which are separated by minimum cut.

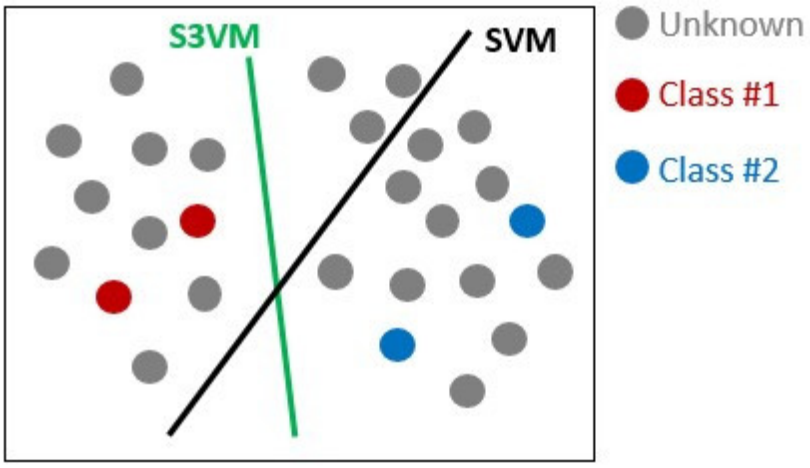


Figure 5.3: This figure shows the comparison between a standard SVM approach, which relies only on the annotated data and a low-density separation method, namely the S3VM which instead also considers the unlabeled samples.

Low-density separation methods

Methods from this family Chapelle and Zien (2005); Li et al. (2013) work with the concept of classification boundary. Specifically, they condition the creation of the classification boundary to cross the region of the input space which are less dense of data points (either labeled or unlabeled). Fig. 5.3 reports a simplified example which compares a standard supervised SVM (relying only on the labeled part of the dataset) (black line in Fig. 5.3) with one of the most representative low-density separation method, the *S3VM* Chapelle and Zien (2005); Li et al. (2013) (green line in Fig. 5.3).

Disagreement-based methods

Methods from this family Blum and Mitchell (1998); Zhou and Li (2005, 2010) train different versions of a model with the available labeled data and subsequently use them to evaluate the unlabeled one. The idea is that the more the different models "disagree" on the instances from U, the more information can be extracted. For example, in Blum and Mitchell (1998), the authors train two different versions of a model using different feature sets (so called "views" Blum and Mitchell (1998)) and use them in a cyclic process where, at each iteration, each model chooses its most confidently predicted unlabeled instances, and assigns its predictions as pseudo-labels that are then used for the training of the other model. This technique is also called *co-training*.

5.4.3 Active learning

In the incomplete supervision problem described in Sec. 5.1.2, the active learning (AL) framework Settles (2009, 2012) assumes that the ground-truth labels of the unlabeled instances from $U = \{x_1, ..., x_u\}$ can be queried to an oracle (e.g., a human). A common assumption is that each query has the same cost (e.g. in terms of human effort or time). The goal of active learning is to minimize the total cost for training a good model, in terms of labeling (i.e., human effort). While not suffering from model degradation, these methods still require some human effort, even if significantly lower than a full dataset annotation.

The usual workflow of an active learning algorithm is reported in Fig. 5.4. It is typically structured as a cyclic process where, at each iteration, a new model is trained with the available labeled data, this model is then used, according to a *query strategy*, to decide which samples from the unlabeled data to query to the oracle. The intuition is that the query strategy should be such that the queried samples are the most "meaningful", allowing to reduce the total number of annotations required and thus to minimize the human effort.

While this workflow is typically shared by most of the AL algorithms Settles (2009), they can be grouped according to two main characteristics: (i) the assumption on the availability of the unlabeled part of the training set, which allows defining different *AL scenarios* and (ii) the *query strategies* used for decision. In the following paragraphs, I describe common cases for both aspects.

AL scenarios

There are several different problem scenarios in which the learner may be able to ask queries. The literature Settles (2009, 2012) presents three main settings: (i) *membership query synthesis*, (ii) *stream-based selective sampling*, and (iii) *pool-based sampling*.

Membership query synthesis is one of the first AL scenarios that has been investigated Angluin and Laird (1988). In this setting, the model typically can pick any instance from the given input space χ. This includes the case where the model generates samples, rather than sampling them from some underlying natural distribution. This idea of synthesizing queries, extended in the regression problem, has been used e.g., in robotics, for the task of predicting the position of the hand-effector of a robot, given the joint angles of the arm as inputs Cohn et al. (1996).

However, even if this kind of approaches can be considered for a large amount of problems, labeling arbitrary instances might not be an easy task for a human annotator. For instance, in Baum and Lang (1992), the authors used membership query learning to train a neural network

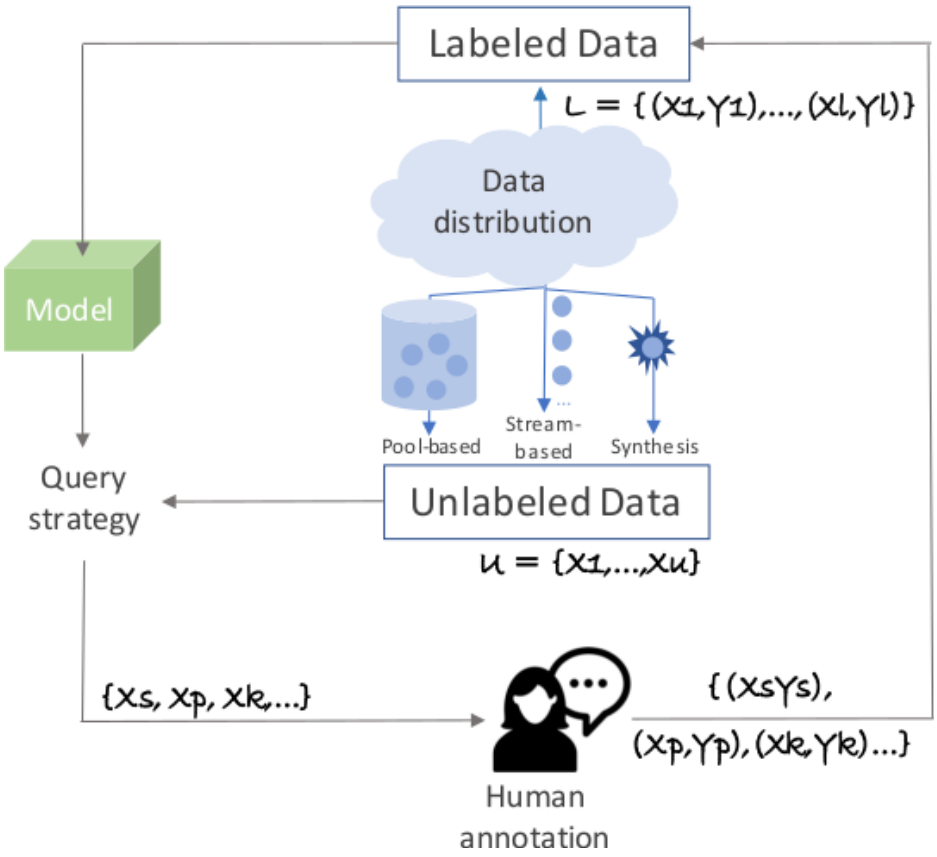


Figure 5.4: This figures shows a simplification of the usual work flow of an AL algorithm, reporting the main differences between the three main AL scenarios, i.e., membership query synthesis, stream-based selective sampling and pool-based sampling (more details in Sec. 5.4.3).

to classify handwritten characters. However, they discovered that many of the query images generated by the AL algorithm did not contain recognizable symbols, but only artificial hybrid characters with no semantic meaning, making the labeling an impossible task for humans.

Nevertheless, in domains where labels do not come from human oracles, but as results of experiments like e.g. in King et al. (2009, 2004), query synthesis may represent a promising direction for generation of automated experiments.

Stream-based selective sampling is an AL scenario where the unlabeled instances are drawn one at a time from U, and the AL algorithm should decide whether to query the oracle for its label or discard it Atlas et al. (1990); Cohn et al. (1994). One of the key assumptions for this framework is that there is no cost in terms of time or human effort in obtaining an unlabeled instance from U. Therefore, a sample can first be sampled from the actual distribution, and

then the AL algorithm can decide whether to request the label or not. The decision is taken according to a *query strategy* that can be based on some measures of either informativeness or uncertainty Cohn et al. (1994).

Stream based AL finds important applications in real world scenarios, like e.g. in Narr et al. (2016) where the authors propose an AL approach for classifying objects from streams of 3D point cloud data. In that case the sampling from the data source is guided by the stream provided by the 3D camera and one of the main problems is that, in the scenario considered, class instances occur non uniformly also producing an unbalanced dataset. In the proposed approach, the authors use the Mondrian forests (MF) Lakshminarayanan et al. (2014), with a twofold benefit: (i) it is independent on the data order, overcoming the problem generated by the non uniform occurrence of classes and (ii) in general, it is less overconfident on wrong predictions, which is a clear advantage for those AL algorithms that use query strategies based on the predicted score confidence.

Pool-based sampling is the AL scenario where it is assumed that the instances of the unlabeled part of the training set U are all available, like in a *pool* Lewis and Gale (1994). Queries are drawn from this pool, which is assumed to be closed (i.e., static). The permanent availability of U allows for greedy query strategies according to an informativeness or uncertainty measure used to evaluate all (or a subset of) the instances in the pool.

This scenario attracted researchers from many real world settings, like e.g. text classification Hoi et al. (2006); Lewis and Gale (1994), image classification and retrieval Zhang and Chen (2002), speech recognition Tur et al. (2005), and cancer diagnosis Liu (2004).

The main difference between stream-based and pool-based active learning is that algorithms from the first group, receiving a stream of samples, must decide whether to query the label for an instance without accessing the rest of the data, while algorithms from the latter group can evaluate and rank all the instances in U before querying the oracle. Even if pool-based approaches are very common in the literature, stream-based ones find practical applications in many real world scenarios, like e.g. in those cases where U is not immediately entirely available or when memory or processing power may be limited, not allowing for computation on huge amount of samples at the same time.

Query strategies

All the aforementioned scenarios involve evaluating a measure to decide which is the *best next query* (identified by x_A^* in the following, where A is the considered query selection policy), among the available unlabeled instances (which can either be generated, accessed

serially or from a pool). AL literature Settles (2009, 2012) proposes a large variety of query strategies. In the remaining of this section, I provide a synopsis of the most common and general ones.

Uncertainty sampling is the simplest and most commonly used framework. It is based on the natural assumption that a learning model should query the samples from U which it is least certain on how to label. This concept can assume different shapes according to the situation. For instance, some common examples are reported below.

1. For *binary classification* with probabilistic models, it is considered straightforward as it can be implemented by querying the instances which predicted probability of being positive is close to 0.5 (i.e. the more ambiguous).

2. The generalization to *multi-class classification* problems is obtained by querying the sample with the *least confident* prediction. It can be formalized as follows:

$$x^*_{LC} = \arg\max_x \left(1 - P_\theta(\hat{y}|x)\right) \tag{5.1}$$

   where, $\hat{y} = \arg\max_y P_\theta(y|x)$ is the prediction for x with the highest probability, considering the model represented by θ. This criterion, however, only considers information about the most probable label, discarding the relation with the others.

3. This problem is addressed with the *margin sampling* technique which instead is based on the intuition that instances with larger margins between the two labels that are considered most probable by the model are easy, since the classifier can easily distinguish between them. Instances with smaller margins, instead, are more ambiguous and thus need to be labeled. More formally:

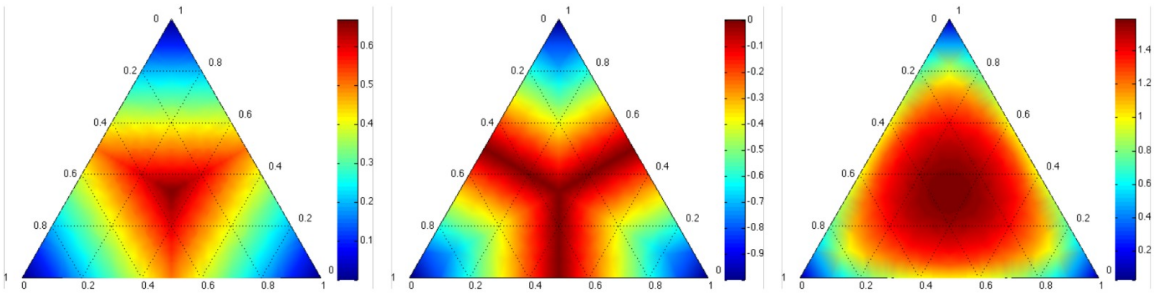
$$x^*_M = \arg\min_x \left(P_\theta(\hat{y_1}|x) - P_\theta(\hat{y_2}|x)\right) \tag{5.2}$$

   where y_1 and y_2 are the two most probable predicted labels for the sample x.

4. The most popular and general sampling strategy is based on the concept of *entropy* as measure of uncertainty. Formally,

$$x^*_H = \arg\max_x -\left(\sum_i P_\theta(\hat{y_i}|x) \log P_\theta(\hat{y_i}|x)\right) \tag{5.3}$$

   Entropy, in information theory, represents the amount of information needed to "encode" a certain distribution. Thus, it can be considered as a measure of uncertainty.



(a) least confident (b) margin (c) entropy

Figure 5.5: This picture, taken from Settles (2009), shows three heat-maps illustrating the query behaviors, in a three class classification problem, of respectively *least confident* (a), *margin sampling* (b) and *entropy based* (c). In all the cases, the corners indicate where one label has very high probability. The most informative query region for each strategy is shown in dark red.

> For binary classification, this criterion reduces to the aforementioned margin and least confident strategies.

Fig. 5.5 shows differences between the uncertainty measures described above. More specifically, it allows to visualize which are the most informative data regions (represented in dark red) for the *least confident*, *margin sampling* and *entropy based* methods, for a three class classification task. The corners of the triangles indicate where one label has very high probability.

Query-by-committee is another query selection strategy which is theoretically motivated Burbidge et al. (2007); Seung et al. (1992). With respect to the incomplete supervision scenario described in Sec. 5.1.2, methods based on this framework train a set (*committee*) of models on the available labeled dataset L (each model should be trained with different competing hypotheses). For each unlabeled instance from U, each member of the committee can "vote" on the possibility of asking its label.

The literature does not provide a minimum number regarding the committee size, however in some practical scenarios committees of small sizes have proved to be effective. A common measure of disagreement that we can find in the literature is the *vote entropy* which associates the concept of entropy to the disagreement between the members of the committee. More formally:

$$x^*_{VE} = \arg\max_x -\left(\sum_i \frac{V(y_i)}{C} \log \frac{V(y_i)}{C}\right) \tag{5.4}$$

where $V(y_i)$ is the number of "votes" that the label y_i receives from the committee for the sample x.

Note that, query-by-committee methods can be easily combined with SSL disagreement-based methods. Indeed, the instances on which the committee members are confident but contradictive can be selected to query while the models could also be co-trained as explained in Sec. 5.4.2.

Expected model change is a query selection strategy which aims to select the instance from U that is expected to change the current model the most, *if we knew its label*. An example that implements this idea is the *extected gradient length (EGL)* Settles et al. (2008). This method relates the impact of a sample on the current model to the length of the training gradient (i.e., the vector used to re-estimate parameter values) in all the methods that are trained using gradient-based optimization. More formally:

$$x^*_{EGL} = \arg\max_x \left(\sum_i P_\theta(y_i|x) \left\| \bigtriangledown l_\theta(L \bigcup < x, y_i >) \right\| \right) \tag{5.5}$$

where $\bigtriangledown l_\theta(L)$ is the gradient of the objective function l with respect to the model parameters θ and $\bigtriangledown l_\theta(L \bigcup < x, y_i >)$ is the new gradient that would be obtained by adding the training sample $< x, y >$ to L.

Active learning, practical use cases

Efficiency in terms of training data represents a very important feature for real world applications Sunderhauf et al. (2018), therefore the weakly supervised framework and, specifically, the active learning one gathered researchers attention.

A recent work in robotics Conkey and Hermans (2019) applies an AL algorithm, together with a learning by demonstration approach, with the aim of minimizing the number of samples required to learn how to grasp in a constrained workspace. Specifically, the authors use the *Probabilitic movements primitives (ProMP)* to define a distribution over the possible arm's trajectories, representing the total working space of the robot with a mixture of these ProMPs. When the robot needs to perform a grasp, this is mapped to one of these ProMPs and the correspondent task is accomplished. This mapping is learned through a learning by demonstration method, where an AL algorithm generates the samples to be demonstrated and a human shows them, by moving the robotic arm in the proper way.

AL can be also applied to the task of object detection with promising results Kyu Rhee et al. (2017); Sivaraman and Trivedi (2014); Vijayanarasimhan and Grauman (2014); Wang

et al. (2019); Wang et al. (2018). A first problem that needs to be addressed to apply AL techniques to an object detection learning algorithm is to decide whether to ask for annotation of entire images or just specific regions. While, indeed, the first case might lead to more accurate models, the second one can represent a cost-effective solution since labeling a single region is easier than labeling an entire image. In the next section (Sec. 5.4.4), I describe a recent AL approach which being integrated with a SSL method, brings interesting results in terms of annotations efficiency. However, some other approaches purely based on AL are worth mentioning. In Vijayanarasimhan and Grauman (2014), for instance, the authors proposed to refine object detectors by actively requesting crowd-sourced image annotations from the web, while in Kyu Rhee et al. (2017), the authors propose a method to improve object detection performance by leveraging the concept of diversity for the active learning policy. Finally, in Sivaraman and Trivedi (2014), the authors show an interesting comparison analysis between different query strategies for the task of object detection. This study considers detection algorithms based on hand-crafted features (i.e. before the revival of deep learning algorithms on this field) but the results also hold for more recent detectors.

5.4.4 Combining semi-supervised and active learning

Both the active and semi-supervised learning frameworks provide advantages and disadvantages. Indeed, while SSL approaches completely avoid human intervention with respect to AL, which still requires some human labeling, early errors in the self supervision may produce model drifts which instead are not considered a risk in AL. Moreover, both frameworks consider just a part of the unlabeled dataset. Specifically, the AL makes use of those data that do not allow for a confident prediction, while the SSL only considers the remaining "easy" part of the dataset.

For these reasons, a possible solution is to integrate the two frameworks. The literature shows many cases where both strategies have been successfully applied Lin et al. (2017a); Wang et al. (2019); Wang et al. (2018, 2016). For instance, in Wang et al. (2016), both SSL and AL are used for efficiently improving object classification accuracy, while in Lin et al. (2017a), they are jointly used for face identification. A recent approach that does this for the object detection is the *Self-Supervised Sample Mining* method Wang et al. (2019); Wang et al. (2018) (SSM), a weakly supervised approach, which combines (i) a SSL technique to generate pseudo ground truth, with (ii) an AL strategy to select the hardest unlabeled images to be requested for annotation. The SSM method was proposed as an end-to-end deep architecture, where the AL and SS processes alternated with the fine-tuning of Region-FCN Dai et al. (2016b). Specifically, the learning process is divided in two phases: a *Supervised Phase* and

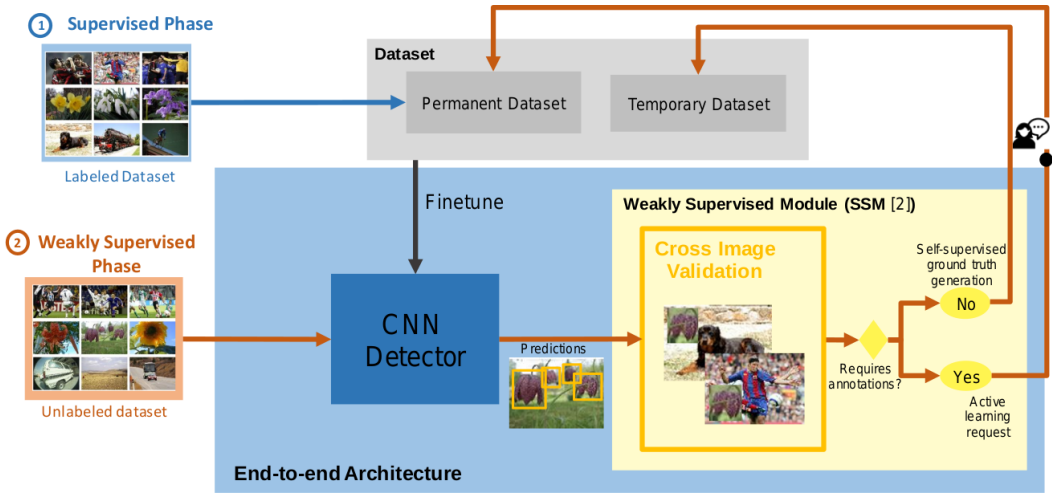


Figure 5.6: This picture shows a scheme of the SSM architecture. Please refer to 8.1 for more details.

a *Weakly Supervised Phase* (see Fig. 5.6 for a pictorial representation of these two phases). During the *Supervised Phase* the available labeled data is used to train a first instance of the detection model, while during the *Weakly Supervised Phase* the unlabeled part of the dataset is used to refine it by using the AL and SSL strategies in an iterative process. Please, refer to Fig. 5.6 for a pictorial representation of the system and to Sec. 8.1 for a more detailed description of the method.

This thesis leverages on the AL and SSL presented in Wang et al. (2018), to propose a weakly supervised strategy that integrates the SSM with the on-line learning pipeline proposed in this project for object detection. Specifically, the AL and SSL processes have been isolated from the Region-FCN and combined with the fast on-line learning pipeline described in Chapter 7. In the proposed system, a new model is trained at every adaptation iteration (rather than fine-tuning or modifying the previous one). Please refer to Sec. 8.1 for more details.

Part III

Methodologies

Chapter 6

Automatic Data Collection

Recent Deep Learning methods Krizhevsky et al. (2012b); Ren et al. (2015); Sermanet et al. (2014), obtained remarkable performance on difficult tasks such as the ImageNet Large-Scale Visual Recognition Russakovsky et al. (2015), the MS COCO Lin et al. (2014) and the Pascal VOC Everingham et al. (2010) challenges. One of the problems in the adoption of these methods in robotics, is that they require a large dataset of images carefully annotated. Image annotation is particularly demanding for training object detection systems as this process requires not only object labels but also bounding boxes (see Sec. 4.7 for more details). In addition, it implies an off-line process, which is unfeasible for a system that learns on-line.

One of the contributions of this Ph.D. project focused on this problem. Specifically, I demonstrated that object detectors could be trained using an automatic data annotation pipeline. The considered approach has been previously validated for the task of object recognition Pasquale et al. (2019), and adopted to acquire large-scale annotated image datasets (i.e. the iCubWorld Transformations[1]). This method exploits the depth estimation, using the robot's cameras presented in Pasquale et al. (2016b) and a human robot interaction. The first part of this thesis project focused on assessing whether or not this approach could be adopted to effectively train deep object detectors as Faster Region-CNN Ren et al. (2015). Thus, the resulting proposed pipeline allows to first acquire an automatically annotated dataset and then train a state-of-the-art method for object detection.

The rest of the chapter is organized as follows: Sec. 6.1 describes the proposed pipeline in detail and Sec. 6.2 presents results from the considered benchmark, specifically analyzing performance on a human-robot interaction setting (see Sec. 6.2.2) and generalization capabilities to different scenarios (see Sec. 6.2.3).

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

6.1 Description of the pipeline

The proposed pipeline is an automatic procedure for extracting and labeling example images for training an object detection network. The robot used for the experiments is the iCub humanoid Metta et al. (2010). In the following sections, I describe the two main steps in the pipeline: i) data acquisition and ii) model training.

6.1.1 Data acquisition method

For the first step of data acquisition, the pipeline relies on the method used to collect the iCubWorld Transformations dataset[2] Pasquale et al. (2019) (shortened to iCWT for simplicity in the following). This method exploits depth information and human-robot interaction to collect labeled images. The procedure is as follows: the teacher shows the object in front of the cameras of the iCub. A tracking routine Pasquale et al. (2016b), uses stereo vision Geiger et al. (2010), selecting the pixels from the depth map that are closer to the robot, thus segmenting the object from the background. A bounding box is estimated around it, and stored as annotation along with the label of the objects provided verbally by the teacher.

6.1.2 Object detection architecture

For the step of Object Detection the pipeline relies on Faster R-CNN Ren et al. (2015) as a representative architecture among the ones recently proposed in the deep learning literature for the same task (please, refer to Sec. 4.4 for a detailed description of the method). As feature extraction CNN, I evaluated and compared two models, whose integration in the Faster R-CNN meta-architecture is publicly available[3]:

- the ZF network proposed by Zeiler and Fergus (2013),
- the VGG_CNN_M_1024 network proposed by Chatfield et al. (2014).

For both networks, the training process is initialized by adopting for the shared convolutional layers, the weights trained on the image classification task of the ImageNet Deng et al. (2009). Then, the network is fine-tuned by following the 4-Steps Alternating Training pipeline proposed by Ren et al. (2015) and described in Sec. 4.4.1. Specifically, for fine-tuning, the number of training epochs is set to 6 for two steps where the RPN is learned while

[2] iCubWorld website: https://robotology.github.io/iCubWorld/
[3] https://github.com/rbgirshick/py-faster-rcnn/tree/master/models/pascal_voc/ZF

Figure 6.1: Examples images from ICWT, depicting 20 objects shown to the robot by a human teacher. Please, refer to Appendix A for further details about the used dataset.

it is set to 2 for the other two phases. All other parameters are left unchanged with respect to Ren et al. (2015).

6.2 Experimental evaluation

This section reports on the experiments performed to quantitatively assess the effectiveness of the proposed approach.

6.2.1 Experimental setup

As mentioned in Sec. 6.1.2, Faster R-CNN Ren et al. (2015) is adopted with either the ZF or the VGG_CNN_M_1024 feature extractor CNNs. An object identification task among 20 objects is considered by randomly choosing one object instance for each of the 20 categories available in the ICWT. Figure 6.1 shows an example image for each selected object. Please, refer to Appendix A for further details about the used dataset. As training set for the considered task, we used the union of the 4 image sequences available in ICWT for each object, corresponding to the 2D ROT, 3D ROT, BKG and SCALE viewpoint transformations. I considered the two available acquisition days and only images from the left camera, overall leading to a training set of ∼27K images. Please, refer to Appendix A for further details.
The system has been evaluated on two settings, respectively in Sec. 6.2.2 and 6.2.3:

1. First, I assessed the effectiveness of the approach in the same HRI setting used for learning: a human holds the object to be detected in the hand. For this test, the MIX sequence available in ICWT is used, leading to a test set of ∼13k images (see Sec. 6.2.2). Testing on these sequences is the best way to evaluate the robustness of the predictions, since, the object is shown naturally to the robot and can appear in any configuration.

Figure 6.2: Example frames where the trained detector (either ZF or VGG_CNN_M_1024) outputs a correct bounding box around the object (red), while the depth segmentation fails (yellow).

	ovenglove	hairclip	hairbrush	book	ringbinder	flower	cellphone	glass	mouse	wallet	bodylotion	pencilcase	soapdispenser	sprayer	sodabottle	mug	sunglasses	perfume	remote	squeezer	mAP
ZF	0.59	0.78	0.71	0.59	0.52	0.86	0.69	0.83	0.92	0.79	0.84	0.41	0.81	0.61	0.79	0.85	0.61	0.75	0.59	0.77	0.72
VGG_CNN_M_1024	0.53	0.76	0.72	0.55	0.51	0.86	0.62	0.84	0.92	0.77	0.83	0.44	0.75	0.63	0.78	0.85	0.64	0.77	0.64	0.79	0.71

Table 6.1: AP for each class and mAP (last column) reported by the two Faster-RCNN models considered in this work when tested on the MIX sequences of ICWT. The reference ground truth is the automatic bounding box provided by the depth segmentation routine.

2. Then, I evaluated the ability of the detection system to generalize to a different setting. To this end, I acquired and manually annotated three new image sequences, representing the considered 20 objects randomly positioned on the floor, on a table or on a shelf (see Sec. 6.2.3 for further details). These sequences have been made already available at the same dataset website.

6.2.2 Evaluating object detection in a HRI setting

In this experiment, I evaluate the detection performance of the network on a setting similar to the one used for training. Since the training data is segmented and labeled automatically by the robot, it inevitably contains some imperfections or errors in the bounding box. This evaluation is therefore important to determine the extent to which the noise in the training set affects the detection network. To this end, I considered the two ZF and VGG_CNN_M_1024 models, trained to detect 20 objects from ICWT as described in Sec. 6.1 and Sec. 6.2, and started testing them in the same HRI setting, using the MIX sequences of the considered 20 objects.

In Table 6.1, I report, for the two network models, the Average Precision (AP) for each object, with the mAP over all objects (last column). Performance is computed against the ground truth bounding boxes provided by the depth segmentation routine.

What first comes to light is that the reported performance is good overall, in line with the state-of-the-art of deep learning detection systems on other benchmarks (see, e.g., results

	ZF	**VGG_CNN_M_1024**
depth's ground truth	0.71	0.69
manual ground truth	0.75	0.73

Table 6.2: mAP reported by the two models on a 3K subset of images sampled from the test set of Table 6.1 and manually annotated. It can be noticed that the mAP with respect to the manual ground truth is even higher than the one with respect to the depth's automatic ground truth.

achieved by Faster RCNN Ren et al. (2015) on the Pascal VOC Dataset Everingham et al. (2010), which consists as well in a 20-class discrimination task). This result is a first important achievement due to the fact that it suggests that these network models, trained with the proposed method, are successful in localizing and identifying objects in RGB images. Note that, the testing scenario is more challenging, as, the object is localized without the use of depth information. As a matter of fact, the depth's bounding boxes are only used to teach the robot to detect the object and, once the training is completed, the robot does not need depth to localize it. As will be shown in Sec. 6.2.3, this approach expands the range of possible applications also to settings where the depth cannot be used to localize objects, or even not available at all.

In this first test, I used as ground-truth the bounding boxes computed automatically using depth information. Since these bounding boxes may contain errors, I further evaluated the system against bounding boxes computed with manual annotation on a subset of the images. In fact, a high AP against the depth's ground truth implies that the model learned to predict bounding boxes which are "similar" to the ones provided by the automatic annotation procedure, which, however, may contain noise, be biased or less precise with respect to "ideal" bounding boxes provided by a human supervisor.

To evaluate this, I manually annotated a subset of images from the MIX sequences used in the previous test. I adopted the *labelImg* tool[4] and fixed an annotating policy such that an object must be annotated if at least a 50-25% of its total shape is visible (i.e. not cut out from the image or occluded). I annotated 150 frames from each MIX sequence, gathering a test set of 3K images for all the 20 objects that have been made available at the iCWT website. Therefore, I evaluated the two models on this test set, computing their performance both against the depth's ground truth and the manual ground truth. In Table 6.2, I report the mAP of the predicted bounding boxes against automatic ones (first row) and manually annotated ones (second row). Since the mAP evaluated on the manual ground truth is even higher, it can be inferred that, not only the automatic annotations are sufficient to train good

[4]https://github.com/tzutalin/labelImg

detectors, but these networks also learned to "average out" possible noise in the ground truth, performing thus better than the depth segmentation procedure. In Figure 6.2, I show some example frames where this effect is evident: while the depth segmentation routine fails to segment the object (yellow), the prediction of the model (red) provides a substantially correct bounding box around it.

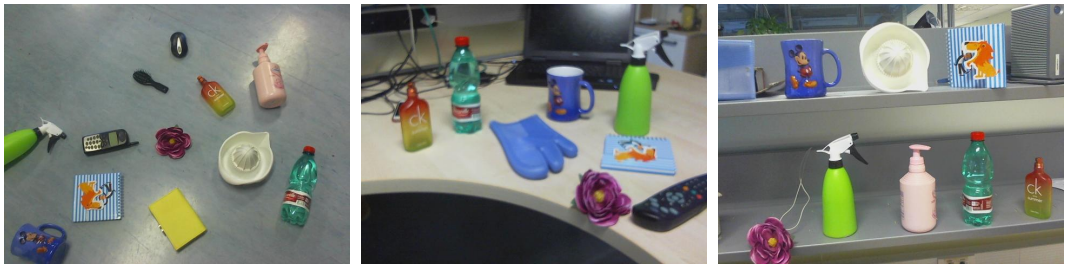
6.2.3 Evaluating generalization to other settings

In the evaluation described in the previous section, training and testing took place in similar settings. A further evaluation has been carried out to determine to what extent the learning system can generalize to a different scenario. This is important due to the fact that the training uses images that have (i) the constant presence of a human, holding the object in the hand, possibly generating occlusions, and (ii) the presence of a single object of interest, mostly centered (because the robot was tracking it). In this Section, I investigate if this bias Torralba and Efros (2011) affects the generalization properties of the network (please, refer to Sec. 4.7 for further details about dataset bias).

Therefore, the question is whether the networks learned to detect the objects only in these conditions, or are able to generalize to other settings. To this end, I collected and manually annotated three new image sequences, representing three scenes where the objects are randomly positioned respectively on a table, on the floor and on a shelf. These sequences remarkably differ from the ones in ICWT as (i) there is no human presence in the scene and (ii) they contain a variable number of objects, at multiple locations in the image. In addition, the light and background are different from the ones represented in the dataset. Figure 6.3 shows an example frame for each sequence (comprising ~ 300 frames):

- The FLOOR sequence depicts 14 out of the 20 objects, lying on the floor.
- The TABLE sequence shows 11 objects on a table with others that are not part of the dataset (like a laptop or a monitor), and hence not to be detected.
- In the SHELF sequence 10 objects are placed on two shelves. The one below is partially shadowed by the one above and, as in the previous sequence, it may contain objects not to be detected.

Table 6.3 reports the performance of the two models tested on the three sequences separately, while Fig. 6.4 shows the predictions of ZF (we obtained similar results with the VGG_CNN_M_1024 model) for two randomly sampled frames from each sequence.



(a) (b) (c)

Figure 6.3: Example images from the three sequences (respectively FLOOR (6.3a), TABLE (6.3b) and SHELF (6.3c)) collected and manually annotated to evaluate the generalization performance of the learned detectors with respect to different indoor settings.

	ZF	**VGG_CNN_M_1024**
FLOOR	0.55	0.47
TABLE	0.66	0.32
SHELF	0.53	0.78

Table 6.3: Performance (mAP) of the two models considered in this work when tested on the three image sequences described in Sec. 6.2.3.

These results show that the performance on these testing sequences remains of good quality, with an average mAPs over the three sequences of 0.58 for the ZF model and 0.52 for VGG_CNN_M_1024. This indicates that the networks succeeded in learning to detect the objects even when these are not hand-held and that the proposed approach is a feasible solution to quickly obtain robust and accurate enough object detectors to be used in a general indoor setting. However, the performance reported presents a drop with respect to the one obtained testing on sequences of handheld objects (see Tab. 6.2). This means that, even if the network learned to detect these objects, there is still room for improvement in terms of generalization to different scenarios. Specifically, this point has been tackled in the last part of this project and it is reported in Chapter 8.

The proposed approach has been developed in a robotic application and videos are available as results[5][6].

[5]https://www.dropbox.com/s/973uyayq1xy9b1i/0164_VI.mp4?dl=0
[6]https://www.dropbox.com/s/njykynrdnrbovqz/ElisaDetectionFinal.mp4?dl=0

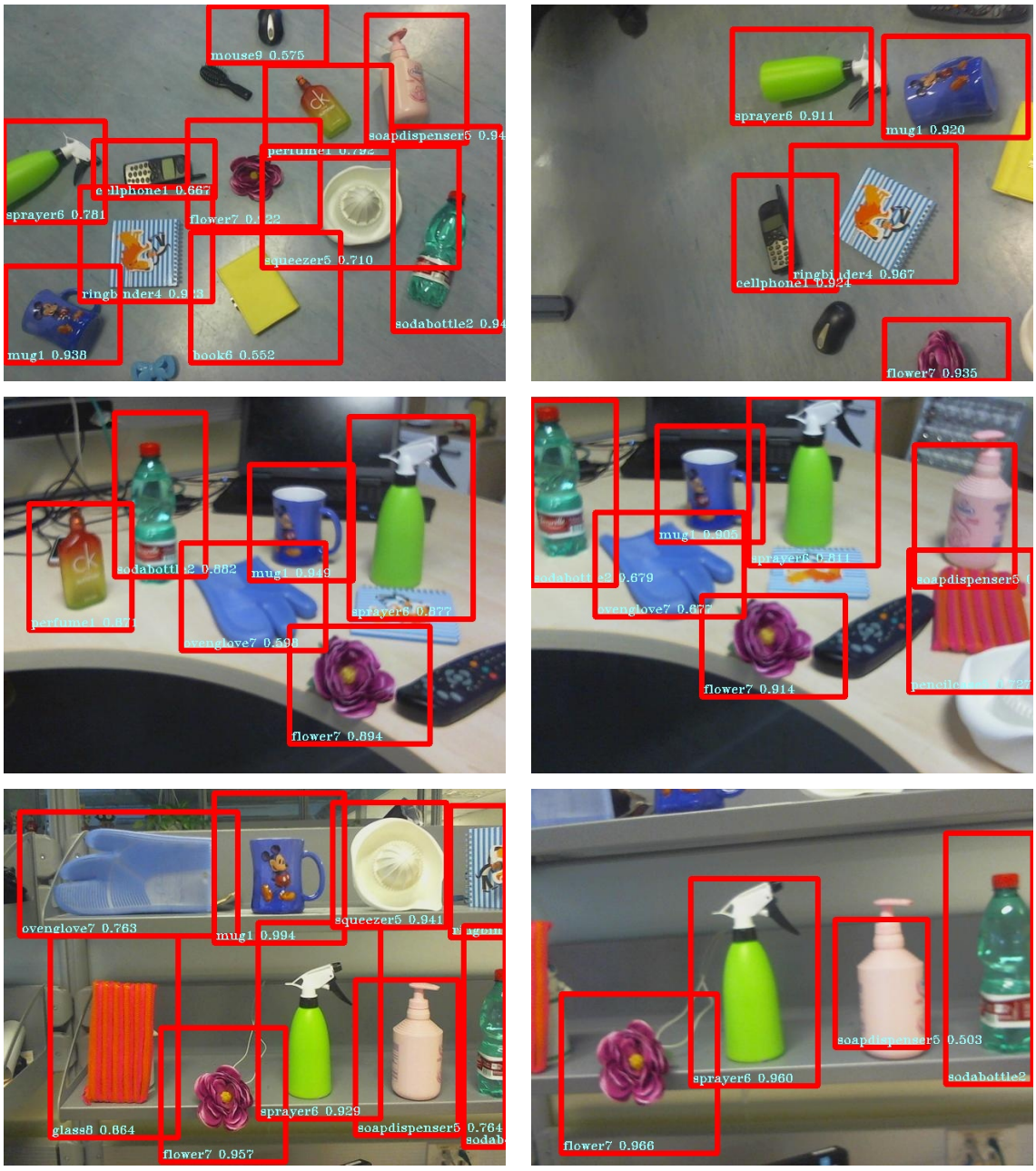


Figure 6.4: Example frames, randomly sampled from each sequence, showing the predictions reported by the trained detector (using ZF as feature extractor).

Chapter 7

On-line Object Detection Learning

Training neural models for object detection is computationally demanding Dai et al. (2016b); He et al. (2017); Redmon and Farhadi (2016) and this makes their applicability to robotics difficult Sunderhauf et al. (2018). Firstly, the enormous amount of parameters that typically characterizes deep learning models makes them generally slow to train and impressively data hungry. Moreover, training a model for object detection comes with a further complication which is due to the large number of regions per image that may contain the objects of interest. More specifically, these objects are represented only in very few of such regions while all the others are treated as negative examples. This leads to a training set that is large and highly unbalanced. State-of-the-art solutions are either based on (i) ad-hoc loss functions Lin et al. (2017b) or modifications of the standard stochastic gradient descent iteration Shrivastava et al. (2016) or on (ii) Hard Negatives Mining Methods Girshick et al. (2014), applied to train the system only on the most difficult negative examples (please refer to Sec. 4.4 for more details about state-of-the-art solutions to this problem). Both types of solutions are time consuming, if applied in canonical ways.

As a part of my project, in this perspective, I proposed a solution Maiettini et al. (2018, 2019) which relies on a fast approximation of the latter state-of-the-art approach and that allows to train a detection model in few minutes or seconds. In this chapter, I present a detailed description and an extensive evaluation of this method. Note that, by integrating this fast learning strategy with the automatic data collection pipeline described in Chapter 6, the resulting application allows to naturally train a robot to detect novel objects. The major contributions, that will be discussed in this chapter, are the following:

1. The design and implementation of an on-line learning pipeline that speeds-up the training of an object detection model, based on two main modules: (i) a per-region feature extractor, trained once, off-line, on the available task, and (ii) a region classifier which can be trained quickly and on-line on a different task, the target task.

2. The rigorous benchmark of the method on the official Pascal VOC Everingham et al. (2010), achieving state-of-the-art performance by integrating Resnet101 He et al. (2015) as CNN backbone;

3. The test of the approach on a challenging robotic dataset, namely the iCubWorld Transformations Pasquale et al. (2019). Within this scenario, I analyze the components of the pipeline, providing interesting insights for each one of them. Specifically, I consider performance in terms of accuracy and train time. This analysis comprises: (i) the comparison between different feature extraction modules, investigating various domains and configurations; (ii) the comparison between various options for the classifiers, motivating the choice done for the proposed pipeline; (iii) the demonstration of how to tune the main parameters of the method in order to obtain a speed/accuracy trade-off.

4. A further analysis, considering the same scenario, with experiments designed to challenge the pipeline in real world conditions, showing limitations and considering possible solutions.

The remainder of this chapter is organized as follows: in Sec. 7.1, I describe the proposed on-line detection pipeline. The results and considerations from the extended empirical analysis of the resulting system are provided in Sec. 7.2, 7.3 and 7.4. Finally, in Sec. 7.5, I report on the preliminary results obtained by extending the pipeline to also perform segmentation.

7.1 Description of the on-line learning pipeline

In the scenario considered for this pipeline, a robot is asked to learn to detect a set of novel object instances (TARGET-TASK in the following) during a few seconds of interaction with a human. To this end, I propose and analyze an object detection method that can be trained on-line on the TARGET-TASK, by exploiting some components previously trained on a different task (FEATURE-TASK in the following).

In this section, I describe the proposed approach. Specifically, in Sec. 7.1.1 I give an overview of the pipeline. Afterwards, in Sec. 7.1.2, I explain how each component is learned. Finally, in Sec. 7.1.3, I provide details on the on-line learning method, describing how the issues of the dataset size and imbalance are addressed.

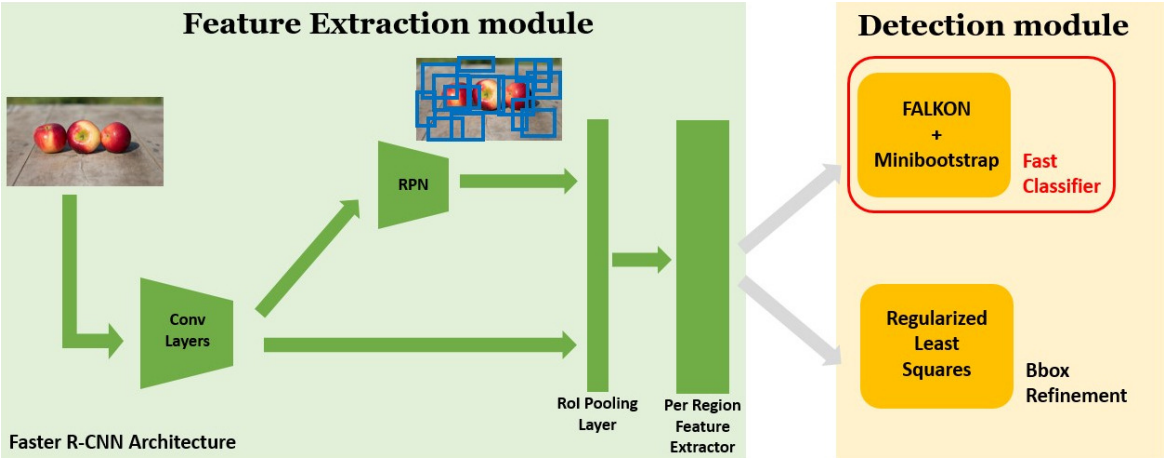


Figure 7.1: Overview of the proposed on-line object detection pipeline. The *Feature Extraction module* relies on Faster R-CNN architecture to extract deep features and predict RoIs (Regions of Interest) from each input image. The *Detection module* performs RoIs classification and refinement, providing as output the actual detections for the input image. For a detailed description, refer to Sec. 7.1.

7.1.1 Overview of the pipeline

The proposed pipeline is composed of two stages (see Fig. 7.1): (i) per-region feature extraction (*Feature Extraction module*) and (ii) region classification and refinement (*Detection module*).

For the first stage, I rely on the architecture of Faster R-CNN Ren et al. (2015), which uses the class-agnostic Region Proposal Network (RPN) to predict a set of candidate Regions of Interest (RoIs). Each RoI is then encoded into a deep feature map by means of the so-called RoI pooling layer Girshick (2015), which spatially aggregates the activations of a convolutional feature map within the area defined by each proposed RoI (please, refer to Sec 4.4 for a detailed description of the method). Specifically, for the evaluation reported in Sec. 7.2, 7.3 and 7.4, I adopt ResNet50 and ResNet101 He et al. (2015), as CNN backbones, integrated in Faster R-CNN as explained in Sec. 7.1.2.

For the second stage, the last two output layers of Faster R-CNN for class prediction and bounding box refinement (namely *cls* and *bbox*), have been replaced with the proposed *Detection module*. Specifically, for the classification of the proposed RoIs, a novel method has been designed, which employs a set of binary classifiers. As classification algorithm a novel approach optimized for large scale datasets is used, namely FALKON Rudi et al. (2017) (please, refer to Sec. 7.1.3 for further details about this method). For the bounding box refinement, instead, the Regularized Least Squares (RLS) regression approach proposed in Region-CNN Girshick et al. (2014) is adopted.

Note that, in splitting feature extraction and classification, I take inspiration from the Region-CNN architecture originally proposed in Girshick et al. (2014). However, differently to R-CNN, the weights for feature extraction and region proposal are fine-tuned off-line on one task (the FEATURE-TASK), while the classifiers and the bounding box regressors are trained on-line on the task at hand (the TARGET-TASK). More details about this two-stage learning procedure are provided in Sec. 7.1.2.

7.1.2 Learning

Within the scenario previously described, the two modules of the pipeline are learned separately. The *Feature Extraction module* is trained off-line on the FEATURE-TASK, so that, when the robot is asked to learn the TARGET-TASK, only the final *Detection module* must be trained on-line.

Off-line Stage

The off-line step is performed by training Faster R-CNN on the FEATURE-TASK, following the 4-Steps alternating training procedure proposed in Ren et al. (2015). Please, refer to Sec. 4.4 for a description of the architecture and of this training procedure. As previously mentioned, I adopt ResNet50 or ResNet101 as CNN backbone for Faster R-CNN. As suggested in He et al. (2015), I use layers from *conv*1 to the last layer of *conv*4_*x* to compute the shared convolutional feature map used by the RPN, performing RoI pooling before *conv*5_1. Thereupon, all layers of *conv*5_*x* and up are used to extract per-region features, so that I extract $1 \times 1 \times 2048$ feature vectors from layer *pool*5 and use them as input for the classifiers and bounding boxes regressors. The weights learned during this off-line stage (specifically the RPN, the convolutional and fully-connected layers) are then used in the on-line learning stage to extract region proposals and encode them into deep features for learning the task at hand (the TARGET-TASK).

On-line Stage

The features provided by the *Feature Extraction module* are training examples for FALKON classifiers and the RLS regressors, for proposals classification and refinement respectively. For the RLS regressors, I used the method of Region-CNN Girshick et al. (2014), keeping the same learning objective and loss function, while the novelty of the proposed approach

is in the proposals classification. Specifically, I consider the one-vs-all approach so that a multi-class problem reduces to a collection of n binary classifiers (where n is the number of classes). For each class, the training set is collected by selecting and labeling candidate RoIs as either positive examples, i.e., belonging to the class (indicated as P in the following) or negative ones, i.e., with an Intersection over Union (IoU) with the ground truth smaller than 0.3 (indicated as N in the following). The resulting dataset is used to train a binary classifier and it is usually large and strongly unbalanced. For instance, considering a single object detection task and setting the RPN to produce at most 300 regions per image, the number of elements in N would be 300 times larger than the number of elements in P ($\sim$300k negatives vs $\sim$1k positives for a dataset of 1k images).

In the next section, I explain the proposed fast training method, which accounts for the large size and imbalance of this dataset, while allowing to learn a model in only a few seconds.

7.1.3 Fast training of classifiers

The proposed protocol for training a binary classifier combines the recently proposed FALKON Rudi et al. (2017), with an approximation of the Hard Negatives Mining procedure adopted in Region-CNN Girshick et al. (2014) and in Felzenszwalb et al. (2010b) (originally proposed in Sung (1996)), which we call Minibootstrap. In this section, I first describe the Minibootstrap procedure and then give an overview of FALKON, explaining how the combination of these two ideas allows to achieve remarkable speedups.

Minibootstrap: Approximated Hard Negatives Mining.

The core idea behind the original Hard Negatives Mining method Felzenszwalb et al. (2010b); Girshick et al. (2014); Sung (1996) is to gradually grow (bootstrap) the set of negative examples by repeatedly training and testing a classifier and including in the training set only those samples which are hard to predict. This idea is implemented with an iterative procedure that visits all images in the training set and, for each image i, performs the steps showed in Alg. 1, where the variable N_i represents the set of all the negative examples in the i^{th} image. The output of this procedure is a set of N_{chosen_final} (hard) negative examples, which, jointly with the P positives, are used to train the final classifier.

Such an approach is time consuming, as it iterates over all the images in the training set and processes *all* regions proposed by the RPN. Therefore, I propose to approximate it by

first considering a random subset of regions proposed by the RPN from all training images. The selected regions are then split into a number n_B of batches of size BS. Finally, the hard negatives are selected by iterating over the batches, following the steps in Alg. 1, where N_i, in this case, represents the set of the negatives of the i^{th} batch (the complete Minibootstrap algorithm is reported in Appendix B).

Algorithm 1 Bootstrapping Iteration. Core iteration of proposed Minibootstrap. See Appendix B for the complete procedure.

Input: N_i = {Set of negative examples at the i^{th} iteration}, P = {Set of all positive examples in the dataset}, M_{i-1} = Classifier trained at previous iteration

Output: M_i = classifier trained at the i^{th} iteration, N_{chosen_i} = hard negatives selected after i iterations

1) Select hard negatives from N_i using M_{i-1} and add them to the train set:

$N_i^H \leftarrow \text{SelectHard}(M_{i-1}, N_i)$

$D_i \leftarrow P \bigcup N_{chosen\_i-1} \bigcup N_i^H$

2) Train classifier with the new dataset:

$M_i \leftarrow \text{TrainClassifier}(D_i)$

3) Prune easy negatives from D_i using M_i:

$N_{chosen\_i} \leftarrow \text{PruneEasy}(M_i, N_{chosen\_i-1} \bigcup N_i^H)$

Note that, for the selection of the negatives, the scores produced by the classifiers, that represent the confidence on the predictions, are thresholded. Thus, two values need to be set, one for the selection of the hard negatives and one for the pruning of the easy ones and they depend on the type of the classifier chosen. In the proposed evaluation, I empirically set them respectively to -0.7 and -0.9.

As will be shown in Sec. 7.2, the proposed Minibootstrap preserves state-of-the-art accuracy, while performing the selection over a subset of the dataset, hence allowing much faster train time. Moreover, by setting the parameters n_B and BS, it is possible to sub-sample more or less the training set, allowing to achieve different speed/accuracy trades-off.

FALKON: Speeding-up Classifier Training while Rebalancing.

As classification algorithm for the proposed approach I opted for the recently proposed FALKON Rudi et al. (2017). This combines (i) a suitable preconditioning (of the linear system associated with Kernel methods), with (ii) an iterative solution via conjugate gradient Saad (2003), and finally (iii) a Nÿstrom-based sampling Smola and Schökopf (2000);

Williams and Seeger (2001) for both the preconditioning and kernel calculation. More specifically, this latter aspect allows to stochastically sample a subset of $M \ll n$ training points as Kernel centers (where n is the size of the dataset).

Training standard kernel methods for classification on large datasets can be prohibitive, because it requires to solve the linear system (also known as Kernel Ridge Regression or KRR): $(K_{nn} + \lambda nI)\,\alpha = \widehat{y}$, where n is the number of training points $\{(x_1, y_1), ..., (x_n, y_n)\}$, $(K_{nn})_{ij} = K(x_i, x_j)$ is the Kernel matrix and λ is a regularization parameter (please, refer to Sec. 3.1 for more details).

It can be easily observed that this problem does not scale well with the number of samples n. Just storing K requires $O(n^2)$ in memory space while computing and inverting K_{nn} (i.e. learning phase) requires $O(n^2c + n^3)$ in time (where c is the kernel evaluation cost). FALKON Rudi et al. (2017) approximates the KRR problem using a Nÿstrom method Smola and Schökopf (2000); Williams and Seeger (2001) by stochastically sampling a subset of $M \ll n$ training points as Kernel centers. In addition it uses the conjugate gradient method Saad (2003) associated with a preliminary preconditioning, for an iterative and faster solution of the associated linear system.

FALKON requires $O(M^2)$ in memory space and $O(nMt + M^3)$ for the kernel computation and inversion, where t is the number of iterations required. Since it has been shown that choices of $M \ll n$ (like e.g., $M = \sqrt{n}$) preserve statistical properties Rudi et al. (2017), FALKON allows to drastically reduce training time of Kernel-based classifiers, gaining a factor of $O(\sqrt{n})$ with respect to other Nÿstrom-based approaches and a factor $O(n\sqrt{n})$ with respect to standard Kernel-based methods. This notable gain in learning time is fundamental in order to apply Kernel methods for the considered detection task. During every Minibootstrap's iteration a new model is trained with a dataset of thousands of points and a standard Kernel based classifier would not allow to accomplish the procedure in the required time. In the experimental section, I will show that a Kernel based classifier is fundamental to obtain the best accuracy and that FALKON allows to do it while maintaining a training time comparable to the ones of linear classifiers. I refer the reader to Rudi et al. (2017) for a detailed description of the algorithm. In the proposed pipeline, I adopt the publicly available FALKON implementation[1].

For the proposed approach, I modify the stochastic sampling of the M Nÿstrom centers performed when training FALKON at each iteration of the Minibootstrap, to account for the positive-negative imbalance of the dataset. In particular, I take a number of P' positives with $P' = \min\left(P, \frac{M}{2}\right)$, while I randomly choose the remaining $(M - P')$ among the $N_{chosen\_i-1} \bigcup N_i^H$ negatives obtained at the i^{th} iteration. This step is fundamental because

[1] https://github.com/LCSL/FALKON_paper

when $P \ll (N_{chosen\_i-1} \bigcup N_i^H)$, randomly sampling the M centers in $P \bigcup N_{chosen\_i-1} \bigcup N_i^H$ might lead, reasonably, to further reducing the number of positives with respect to the number of negatives and, in the worst case, discarding all positives from the sampled Nÿstrom centers.

Hyper-parameters

The main parameters of FALKON are (i) the Kernel parameter (where not specified, I use a Gaussian with variance σ), (ii) the regularization parameter, λ, and (iii) the number of Nÿstrom centers, M. The parameters of the Minibootstrap are (i) the number (n_B) and (ii) the size (BS) of the selected batches.

I cross-validated σ and λ using a standard one-fold cross-validation strategy, considering as validation set a subset of 20% of the training set. For doing this, I define two different ranges, respectively for σ and λ, and I search for the best values by considering every possible combination and performing the Minibootstrap procedure for each of them. Furthermore, in Sec. 7.2, I provide experimental evaluation of the other three parameters characterizing our approach. Notably, I show how by setting their values, it is possible to tune the procedure to sub-sample the training set more or less extensively, depending on the desired speed/accuracy trade-off.

7.2 Experimental evaluation

In this section, I first provide details about the setup of the experiments (Sec. 7.2.1) and then I present the performance achieved by the proposed on-line object detection pipeline on two different benchmarks (Sec. 7.2.2 and Sec. 7.2.3).

7.2.1 Experimental Setup

In the presented experiments, the proposed method is compared to Faster R-CNN, as baseline. For a fair comparison, I consider the weights learned by training Faster R-CNN on the FEATURE-TASK (as in Sec. 7.1.2, Off-line Stage) and use them for both (i) the *Feature Extraction module* of the proposed pipeline (Fig. 7.1), and (ii) as a warm restart for fine-tuning the output layers of Faster R-CNN on the TARGET-TASK (i.e. I set the learning rate to 0 for all the layers of the network except the output ones). I consider ResNet50 and ResNet101 He et al. (2015) as different backbones for feature extraction in Faster R-CNN. I report in Appendix C the results of the cross validation for the number of epochs for the

Table 7.1: Benchmark on PASCAL VOC. Models have been trained on *voc07++12* set and tested on PASCAL VOC 2007 test set.

	mAP (%)	**Train Time**
Faster R-CNN (last layers)	73.5	2h 20m
FALKON + FULLBOOTSTRAP	**75.1**	55m
FALKON + MINIBOOTSTRAP 10X2000	70.4	**1m 40s**

fine-tuning of the baselines on the TARGET-TASK. I used those results as stopping criterion, that consists in choosing the model at the epoch achieving the highest mAP on the validation set (I stopped when no mAP gain was observed).

In the following sections, I indicate as FULLBOOTSTRAP the procedure of performing as many bootstrapping iterations (Alg. 1) as the number of training images (namely, the Hard Negatives Mining method of Girshick et al. (2014) and Felzenszwalb et al. (2010b)). Note that, in this case I consider the implementation of Girshick et al. (2014) where the classifier is re-trained only when a number of 2000 new negative examples has been accumulated to the training set. I use instead MINIBOOTSTRAP $n_B \times BS$ to indicate the proposed approximated bootstrapping procedure, where n_B and BS represent respectively the number and the size of selected batches of negatives.

I evaluate the method on two very different datasets: (i) Pascal VOC Everingham et al. (2010) (Sec. 7.2.2) and (ii) ICWT Pasquale et al. (2019) (Sec. 7.2.3) and I report performance, in both cases, in terms of (i) mAP, as defined for Pascal VOC 2007, and (ii) training time (please, refer to Sec. 4.2 for a description of these metrics).

All experiments reported in this experimental analysis have been performed on a machine equipped with Intel(R) Xeon(R) E5-2690 v4 CPUs @2.60GHz, and a single NVIDIA(R) Tesla P100 GPU. Furthermore, I limit the RAM usage of FALKON to at most 10GB.

7.2.2 Benchmark on the Pascal VOC

I first evaluate the performance of the proposed method on the Pascal VOC dataset, a standard benchmark for object detection. I consider, for training and validation, the union set of Pascal VOC 2007 and 2012 trainval sets, gathering $\sim$16k images (*voc07++12* in the following). I use the available Pascal VOC 2007 test set, which consists of about $\sim$5k images, for testing. I consider Resnet101 as convolutional backbone for Faster R-CNN.

The aim of this first experiment is to compare our pipeline with the state-of-the-art on a well-known object detection benchmark. For this reason, differently from the experiments on ICWT, here I cannot split the object categories to be used for the FEATURE-TASK

Table 7.2: Benchmark on ICWT. The models compared are Faster R-CNN's last layers (**First row**), FALKON + Minibootstrap 100x1500 (**Second row**) and FALKON + Minibootstrap 10x2000 (**Third row**).

	mAP (%)	**Train Time**
Faster R-CNN (last layers)	73.5	2h 16m
FALKON + Minibootstrap 100x1500	**73.6**	3m
FALKON + Minibootstrap 10x2000	71.2	**40s**

and the TARGET-TASK, because the TARGET-TASK addresses the common benchmark of the 20-class categorization task of the official Pascal VOC, which includes *all available* categories in the dataset (see, e.g., Ren et al. (2015)). Hence, I use the standard training and test splits (specified above) both for the FEATURE-TASK and the TARGET-TASK.

For learning the FEATURE-TASK, the number of iterations are set to 80k when learning the RPN and to 40k when learning the detection network. As a baseline, the output layers of Faster R-CNN is trained from scratch for 32k iterations (4 epochs, with batch size 2). In Appendix C, I report the results of the cross validation for the number of epochs of the fine-tuning of Faster R-CNN's last layers on the TARGET-TASK, that I used for the stopping criterion.

I compare: FALKON + FULLBOOTSTRAP, which in this case performs ∼16k bootstrapping iterations, processing a batch of 300 regions for each visited image, against FALKON + MINIBOOTSTRAP 10X2000. In both cases, the number of Nÿstrom centers are set to 2000 (the influence of this parameter is investigated in Sec. 7.3.3).

As can be observed from Table 7.1, a detection model can be trained in less than 2 minutes with a performance gap of 3.1% with respect to the mAP provided by training Faster R-CNN output layers (which requires 2 hours and 20 minutes). Moreover, the state-of-the-art performance is reproduced (and outperformed of 1.6%) in less than a half of the time. Examples of detections predicted by the FALKON + MINIBOOTSTRAP 10X2000 are reported in Appendix D.

7.2.3 A robotic scenario: iCubWorld

In this section, I evaluate the proposed method in a robotic scenario, considering the ICWT dataset[2]. A description of the dataset and details regarding how I use it for the presented experiments are reported in Appendix A.

Within the scenario described in Sec. 7.1, I define a FEATURE-TASK as an identification task among 100 objects comprising all available instances (10 per class) of 10

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

out of 20 categories in ICWT. I then define a TARGET-TASK considering 3 objects for each of the remaining 10 categories of ICWT, i.e., an identification task among 30 objects. For each task, I considered, as training set a subset of the union of the 4 image sequences available in ICWT for each object, corresponding to the 2D ROT, 3D ROT, BKG and SCALE viewpoint transformations, using both acquisition days (see Appendix A). Overall, this leads to a training set of $\sim$55k and $\sim$8k images for respectively the FEATURE-TASK and the TARGET-TASK.

As test set I used a subset of 150 images from the first day of acquisition of the MIX sequence for each object, manually annotated adopting the *labelImg* tool[3]. Please, refer to the Appendix A for further details.

In the *Feature Extraction module*, I used ResNet50 as convolutional backbone for Faster R-CNN, which is trained end-to-end on the FEATURE-TASK, by setting the number of iterations to 165k when learning the RPN and to 110k when learning the detection network.

I report results for two different configurations of the Minibootstrap, namely in Table 7.2, (i) FALKON + MINIBOOTSTRAP 10X2000 and (ii) FALKON + MINIBOOTSTRAP 100X1500, that, as will be shown in Sec. 7.3.3, turn out to represent two of the best speed/accuracy trades-off, giving priority respectively to train time and to accuracy (note that in Sec. 7.3.3 other more extreme trades-off are presented).

From Table 7.2 it can be observed that the proposed method reproduces in just 3 minutes of training the same accuracy as the one obtained by fine-tuning the Faster R-CNN's last layers for 48k iterations (12 epochs, with batch size 2) in more than 2 hours. In Appendix C, I report the results of the cross validation for the number of epochs of the fine-tuning of Faster R-CNN's last layers on the TARGET-TASK, that I used for the stopping criterion. Moreover, I show that 40 seconds were enough to train a 30 objects detection model with a mAP gap of 2.3% with respect to the baseline. Examples of detections predicted by the FALKON + MINIBOOTSTRAP 10X2000 are reported in Appendix D.

7.2.4 Towards Real World Robotic Applications

The proposed algorithm has been deployed into a prototypical application, which allows to naturally train in few seconds humanoids as R1 Parmiggiani et al. (2017) and iCub Metta et al. (2010) to detect novel objects. I relied on the YARP Metta et al. (2006) middleware for integrating different modules. A video of the resulting application is publicly available [4]. Further details of the resulting application are provided in Chapter 9 and 10.

[3]https://github.com/tzutalin/labelImg
[4]https://youtu.be/eT-2v6-xoSs

7.3 Ablation study

In Sec. 7.2, I showed results of the proposed method on two benchmarks, demonstrating its effectiveness and gain in train time. In this section, I propose an ablation study, analyzing in detail the main components of the proposed on-line object detection pipeline.

First, I present a study of the *Feature Extraction module* (Sec. 7.3.1). Then, I compare different options for the classification algorithm, motivating the choice of FALKON (Sec. 7.3.2). Finally, I investigate the main hyper-parameters of the method, providing guidelines on how to tune the training procedure (Sec. 7.3.3).

If not specified, I consider (i) the same 100 objects FEATURE-TASK and 30 objects TARGET-TASK as in Sec. 7.2.3, (ii) the Minibootstrap configuration $n_B = 10$ and $B = 2000$ and (iii) the number of Nÿstrom centers to $M = 2000$.

7.3.1 How to learn the Feature Extraction module?

In this pipeline the *Feature Extraction module* is trained only once off-line and on a FEATURE-TASK, which is different from the TARGET-TASK. By doing so, the feature extractor can be adapted to the domain of the TARGET-TASK, while maintaining the two modules decoupled, thus allowing to train the detector only on the task at hand, in just few seconds.

In this section, I analyze the impact of the *Feature Extraction module* on the proposed on-line learning pipeline. In particular, I first show the accuracy loss that originates when features are tuned on a task that is different from the TARGET-TASK. I then evaluate performance of different FEATURE-TASKs, to show how to recover from this loss.

How much do we lose splitting?

Key for the proposed pipeline is the decoupling of the *Feature Extraction module* and the *Detection module*. In this experiment, I evaluate the performance loss I obtain when the FEATURE-TASK differs from the TARGET-TASK. I compare the mAP obtained by the proposed method with the one obtained by training Faster R-CNN as in Ren et al. (2015), on the TARGET-TASK (i.e. optimizing convolutional layers, RPN, feature extractor and output layers). In Appendix C, I report the results of the cross validation for the number of epochs for learning Faster R-CNN on the TARGET-TASK, that I used as model selection criterion.

Table 7.3: The models compared are FALKON + Minibootstrap 100x1500 (**First row**) and Faster R-CNN fully trained on the TARGET-TASK (**Second row**).

	mAP (%)	**Train Time**
FALKON + Minibootstrap 100x1500	73.6	**3m**
Faster R-CNN (full train)	**78.6**	4h 30m

Table 7.4: The table reports a comparison between the two cases of using, for the proposed on-line learning method, a *Feature Extraction module* trained on the set *voc07++12* (**First row**) and on the 100 objects identification task of ICWT (**Second row**) (see Sec. 7.3.1).

	mAP (%)	**Train Time**
Pascal VOC features	42.4	64s
iCWT features	**71.2**	**40s**

As it can be noticed from Tab. 7.3 the lack of feature adaptation on the task at hand produces a gap of 5% in mAP. However the consistent gain in train time (3 minutes instead of 4 hours and 30 minutes) motivates the choice of the proposed method for a robotic application with strict time constraints.

Moreover, as showed in following experiments, this gap can be recovered, partially or completely, either by tuning the procedure or by considering a larger FEATURE-TASK, which could better generalize to the novel TARGET-TASK.

Choosing the Domain

The objective of this experiment is to show, for the scenario proposed in this work, which is the more convenient domain for learning the weights of the *Feature Extraction module*. To this aim, I compare performance obtained by training on either Pascal VOC or ICWT. The first one, being a richer and general purpose dataset for categorization, should allow to learn features and region proposals that can generalize better to novel tasks. The second one should allow instead to learn application-specific, but also possibly, more limited weights.

For Pascal VOC, I consider as FEATURE-TASK, the categorization task on *voc07++12* and I set the number of iterations to 160k when learning the RPN and to 80k when learning the detection network. For ICWT, I consider the same feature extractor as in Sec. 7.2.3 (learned on the FEATURE-TASK of 100 objects identification). I use ResNet50 as backbone.

In Table 7.4 it can be observed that an adaptation of the feature extractor to the same setting of the TARGET-TASK (i.e. considering the 100 objects identification task on ICWT) provides a significant boost in performance than using a more general, though richer, one.

Table 7.5: The table reports a comparison between two different FEATURE-TASKs from ICWT to train off-line the *Feature Extraction module*. Specifically, I compare the train time and mAP obtained by the *Detection module* on the same TARGET-TASK, using feature extractors learned on 10 (**First row**) or 100 (**Second row**) objects as FEATURE-TASKs.

	mAP (%)	**Train Time**
iCWT features (10 objects)	59	58s
iCWT features (100 objects)	**71.2**	**40s**

Table 7.6: The table reports train time and mAP of different classifiers within the Miniboot-strap, respectively, FALKON with Gaussian kernel (**First row**), FALKON with Linear kernel (**Second row**) and linear SVMs (**Third row**).

	mAP (%)	**Train Time**
FALKON-gauss + MINIBOOTSTRAP	**71.2**	40s
FALKON-linear + MINIBOOTSTRAP	68.7	34s
SVMs-linear + MINIBOOTSTRAP	65.7	**19s**

Choosing the task

I demonstrated that adapting the feature extractor to the domain of the TARGET-TASK remarkably improves performance. With this experiment, I illustrate how, in the chosen domain, it is possible to learn feature and region proposals, which can better generalize from the FEATURE-TASK to the TARGET-TASK.

To this end, I evaluate the impact of the number of classes used to define the FEATURE-TASK. I decrease the classes from 100 to 10, considering respectively all the classes in the FEATURE-TASK of Sec. 7.2.3 in the first case and selecting 1 instance per category from it in the second case (see Appendix A for details).

From Table 7.5, it can be noticed that considering a FEATURE-TASK of a smaller number of classes (second row) leads to notably lower performance on the TARGET-TASK, demonstrating that with a bigger FEATURE-TASK, learned features can better generalize to a new TARGET-TASK.

7.3.2 Classification module: is FALKON key to performance?

In this section, I evaluate different choices for the classification algorithm, comparing them to the one finally adopted for our system, that is, FALKON with Gaussian Kernel.

In Table 7.6, I compare as classifiers (i) FALKON with Gaussian Kernel (FALKON-gauss + MINIBOOTSTRAP), (ii) FALKON with Linear Kernel (FALKON-linear + MINIBOOT-STRAP) and (iii) linear SVMs (SVMs-linear + MINIBOOTSTRAP). For the latter one, I use

the LIBLINEAR implementation[5]. Note that, instead, for the proposed pipeline, I used the available FALKON Matlab [6] implementation, thus the train times reported have still large room for improvements.

As it can be observed, using FALKON with a Gaussian Kernel allows to achieve the best mAP, while considering linear classifiers produces a drop in performance, respectively of 2.5% with FALKON implementation and of 5.5% with SVMs. For this reason, it has been considered as the best choice for the proposed learning pipeline since, even if the corresponding train time is slightly longer, it is still in the order of magnitude of seconds (thanks to FALKON with Gaussian Kernel) while having the best mAP. Therefore, it represents a suitable choice for robotic applications.

Nevertheless, it is worth noticing that linear methods, in the Minibootstrap, train faster than the one with Gaussian Kernel. Hence, FALKON with Linear Kernel represents an interesting speed/accuracy trade-off.

7.3.3 Analysis of hyper-parameters

In this section, I show the role of the main parameters of the proposed method. I recall that I cross-validate σ and λ, namely the variance of the Gaussian Kernel and the regularization parameter, as explained in Sec. 7.1. In the following, instead, I study the influence of the parameters specific of the proposed method, namely the number of Nÿstrom centers (M in Sec 7.1.3) and the number and size of the batches of negatives in the Minibootstrap procedure (respectively n_B and BS in Sec7.1.3).

Nÿstrom Centers

In order to evaluate the impact of the number of Nÿstrom centers M in FALKON training, I report performance when varying it from 50 to 3000. Note that, as during the Minibootstrap, the number of training points may vary, depending on the negative samples selected at each iteration, the percentage of the Nÿstrom centers over the total may vary. However, I noticed that for the chosen tasks the number of the training points remains bounded in average between 3000 and 10000.

From the results reported in Fig. 7.2, it can be observed that, while, as expected, increasing the number of Nÿstrom centers leads to longer train times, the mAP saturates early, for quite small values of M. This allows to preserve accuracy while training much faster, by setting

[5]https://www.csie.ntu.edu.tw/~cjlin/liblinear/
[6]https://www.mathworks.com/

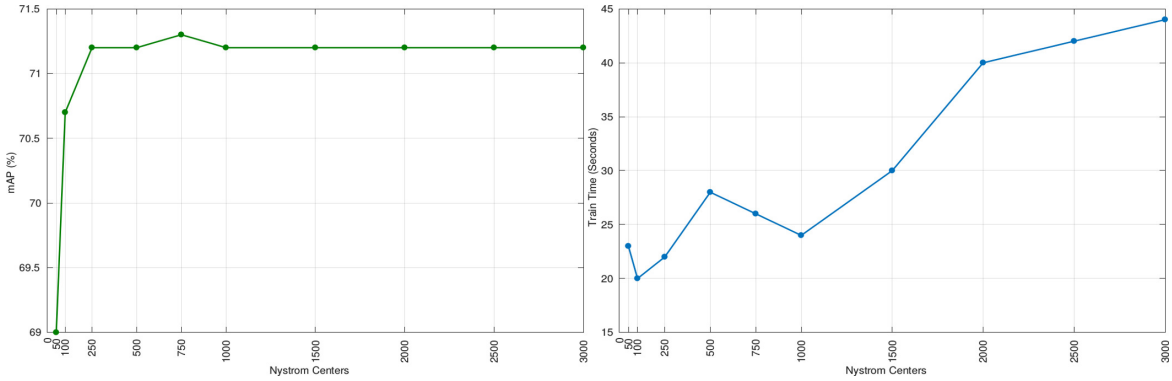


Figure 7.2: Train Time (**Right**) and mAP (**Left**) trends for varying number of Nÿstrom centers when training FALKON within the Minibootstrap (see Sec. 7.3.3 for details).

this parameter. In fact, the value of $M = 750$ is sufficient to obtain the best mAP with 27 seconds of training instead of 44 seconds. Moreover, a value of $M = 100$ allows to train a 30 objects detection model in $\sim$20 seconds, with a gap of only $\sim$1% with respect to the best mAP.

Minibootstrap Configuration

The defined Minibootstrap parameters (namely n_B and BS) allow to tune the proposed procedure to sub-sample more or less extensively the training set, depending on the desired computation time. In this section, I show the impact of varying n_B from 5 to 1000 and BS from 500 to 5000.

For this study, the TARGET-TASK is the same 30 objects identification task as in Sec. 7.2.3, but a training set of $\sim$16k images is considered, to show more extensively the effect of parameter tuning on a bigger dataset.

Results for this experiment are reported in Fig. 7.3 (note that the x axis is represented in logarithmic scale). I consider as baseline the performance achieved by training the output layers of Faster R-CNN, represented by the violet dot in Fig. 7.3 (mAP=72.7 in 4 hours and 35 minutes of training). Since the TARGET-TASK of this experiment is the same as the one in Sec. 7.2.3 (it differs only for the number of images), the validation for it is not repeated, training Faster R-CNN for 12 epochs (see Appendix C for further details).

It can be inferred a growing trend in accuracy by increasing the value of the batch size (color variation), which saturates for BS=5000. Moreover, Fig. 7.3 also shows that models trained with larger batches achieve better accuracy with smaller numbers of iterations and thus in shorter training time. For instance, accuracy of models trained with a batch size of

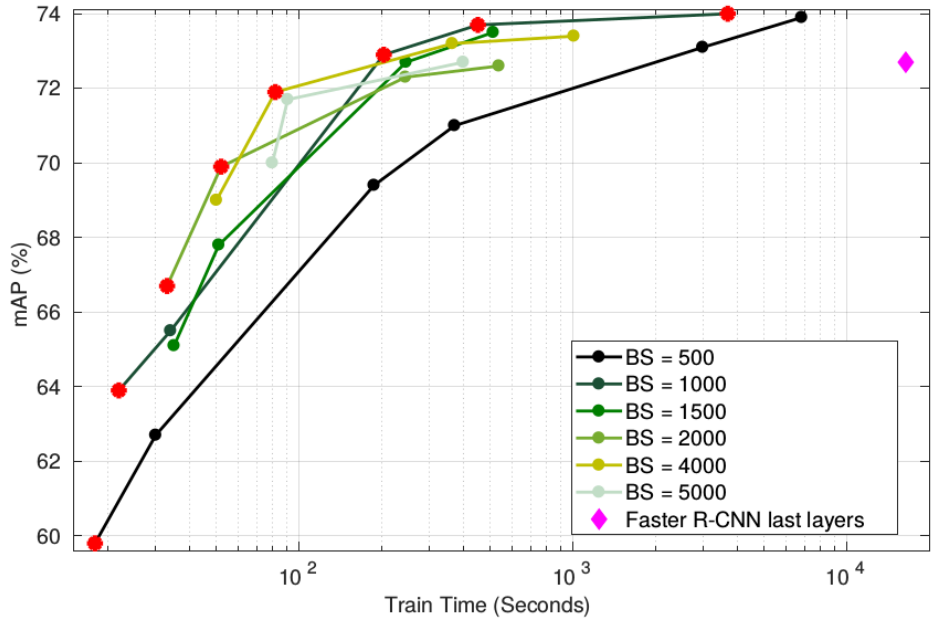


Figure 7.3: The table shows mAP and Train Time trends for n_B and BS variation (see Sec. 7.1.3) during Minibootstrap. Color code represents BS variation while each point of each plot represents a different n_B taken from the ordered set: $\{5, 10, 50, 100, 500, 1000\}$.

500 (black curve in Fig. 7.3) increases with lower slope than accuracy of models trained with larger batches (see e.g., BS = 2000 and BS = 4000).

More interestingly, by considering all plots simultaneously, an edge of points can be identified, displayed as red dots in Fig. 7.3, which represent the best speed/accuracy trades-off, going from a mAP of 59.8% achieved in $\sim$15 seconds to a mAP of 74% (surpassing the Faster R-CNN baseline) in $\sim$1 hour, with suitable middle solutions which allow to train models equivalent in accuracy to the Faster R-CNN baseline in few minutes or seconds.

7.4 Challenging the method for robotics

In this section, I propose a further evaluation of the performance of the method considering some scenarios specifically useful for robotic applications where the robot might be asked to learn continuously novel objects (see Sec. 7.4.1) or the number of available images decreases drastically (see Sec. 7.4.2).

To this aim, in the following experiments, I fix (i) the same 100 objects identification FEATURE-TASK as in Sec. 7.2.3, (ii) the Minibootstrap configuration to $n_B = 10$ and $B = 2000$ and (iii) the number of Nÿstrom centers to $M = 2000$.

7.4.1 Increasing number of objects for the TARGET-TASK.

In this section, I investigate how accuracy and training time are affected by increasing the number of classes of the TARGET-TASK. To this aim, I consider five different tasks as TARGET-TASK. In particular, while I fix the number of images for each object to 250, I vary the number of objects from 1 to 40. I consider the 10 categories left in ICWT by excluding the ones chosen for the FEATURE-TASK, and randomly sample, respectively, 1, 2, 3 and 4 instances from each category for the detection tasks of 10, 20, 30 and 40 objects (see Appendix A for further details). Then, I randomly select four objects instances to evaluate performance on four different 1-object detection tasks and report the average results. I compare with training the last layers of Faster R-CNN for 12 epochs.

Results are reported in Fig. 7.4 (note that the y axis of the train time graph is logarithmic). As it can be noticed, the mAP decreases reasonably, by increasing the number of objects with a trend similar to the one of Faster R-CNN. Moreover, Fig. 7.4 shows that the proposed method (green line), while naturally increasing training time when incrementing the number of objects, it is still able to learn a detection model in a time of the order of magnitude of seconds. On the contrary, by training the last layers of Faster R-CNN, the optimization time is in the order of magnitude of hours.

7.4.2 Decreasing the number of images for the TARGET-TASK

In this experiment, I evaluate the effect on mAP and training time of considering more or less example images for the TARGET-TASK. To this end, I consider the same TARGET-TASK as in Sec. 7.2.3 (30 objects), and vary the number of example images for each object from $\sim$16 (for a total of $\sim$500 training images) to $\sim$800 (for a total of $\sim$16k training images).

As can be noticed from Fig. 7.5, the number of samples for the same task does not affect notably the trend of accuracy, which fluctuates around 71% for both the baseline and the Minibootstrap, however it can be observed a slight decrease in accuracy for the proposed method when increasing the number of samples from 4k to 16k. This is due to the fact that, while for Faster R-CNN, the number of epochs for the optimization is kept constant, allowing the network to really train on more data, for the Minibootstrap, the n_B and BS parameters are fixed, leading to a more aggressive subsampling in the cases of bigger datasets, thus to a poorer representation of the negatives. As shown in Fig. 7.3, this loss can be completely recovered by considering a different Minibootstrap configuration in order to perform a less aggressive subsampling of the negatives. Specifically, by considering FALKON + MINIBOOTSTRAP 10X4000 and FALKON + MINIBOOTSTRAP 50X1000 it is possible to achieve, respectively, mAP of 71.9% in $\sim$80 seconds and 72.9% in $\sim$3 minutes.

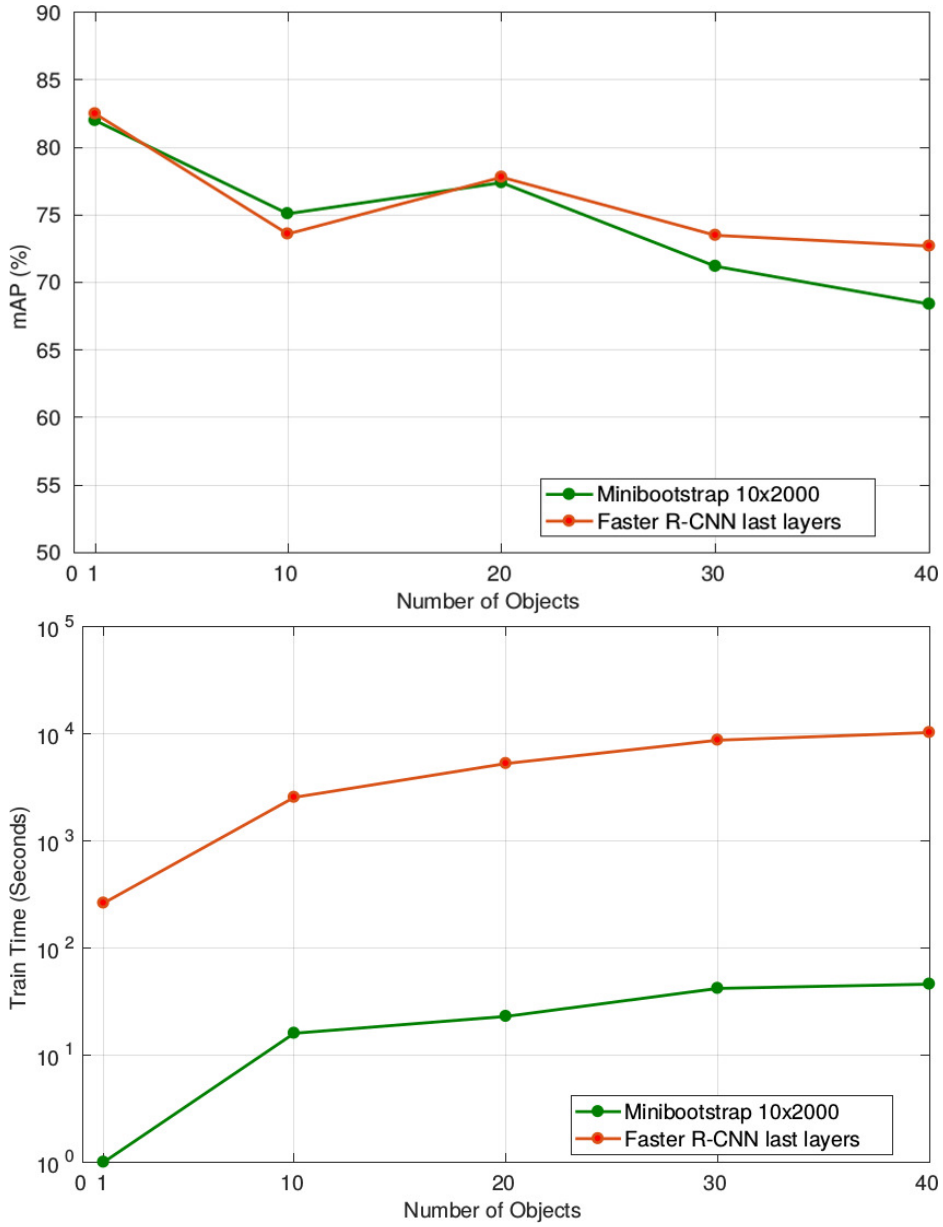


Figure 7.4: The table compares performance of FALKON + MINIBOOTSTRAP 10x2000 trained on different TARGET-TASKs, varying the number of objects in a range from 1 to 40. I show mAP (**Top**) and Train Time (**Bottom**), comparing results with fine-tuning Faster R-CNN last layers.

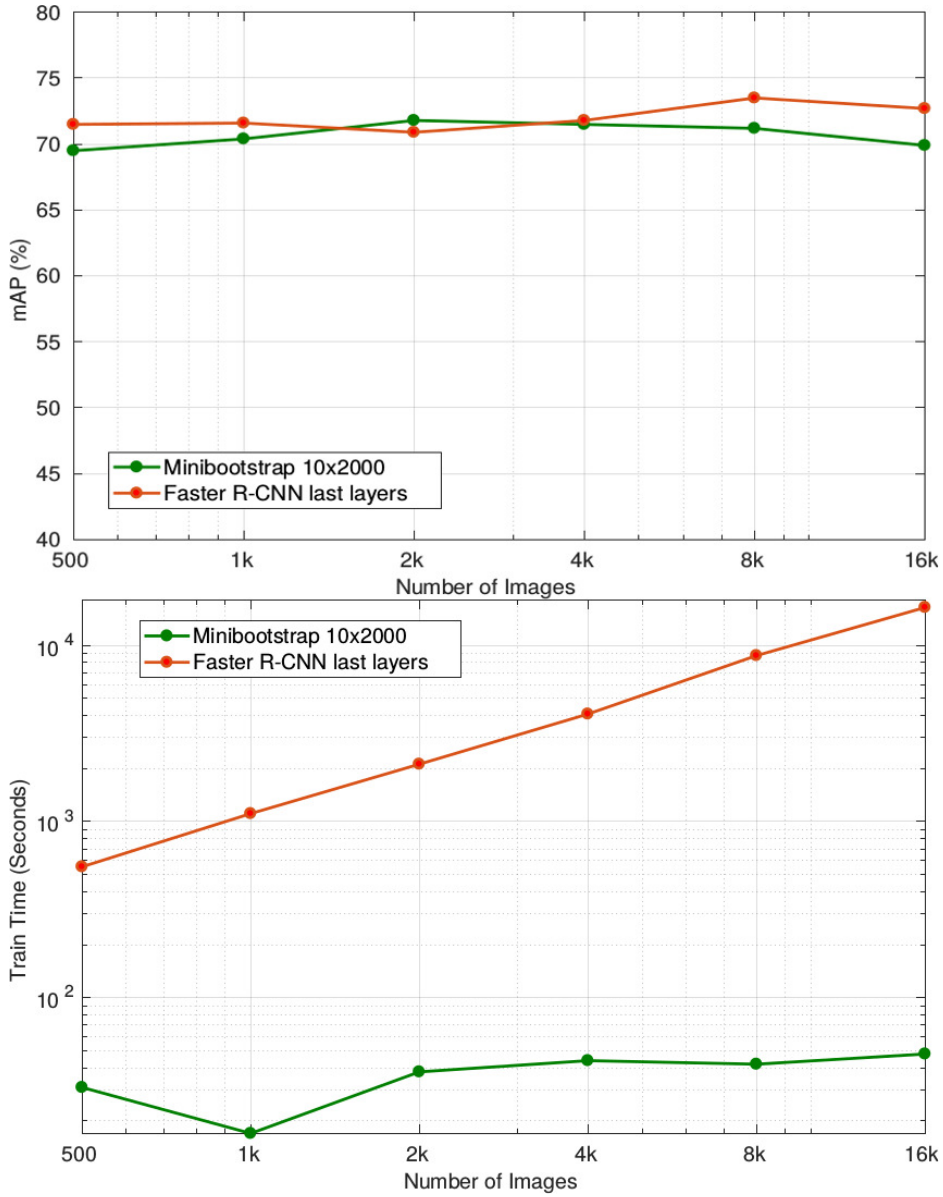


Figure 7.5: The table compares performance of FALKON + MINIBOOTSTRAP 10x2000 trained on a 30 objects identification TARGET-TASK considering different numbers of samples for each class. I show mAP (**Top**) and Train Time (**Bottom**), comparing results with fine-tuning Faster R-CNN last layers.

Moreover, Fig. 7.5 shows that while the time to fine-tune last layers of Faster R-CNN increases linearly with the number of images, as expected, the time necessary to accomplish Minibootstrap remains bounded in the order of magnitude of seconds. Note that this is true even considering the two more accurate configurations FALKON + MINIBOOTSTRAP 10X4000 and FALKON + MINIBOOTSTRAP 50X1000, mentioned before.

7.5 Towards on-line segmentation

In an additional activity, but relevant to the thesis, I collaborated in a master thesis project, which aim was to devise an on-line learning method for object segmentation Bunz (2019). Specifically, the main efforts of the project have been twofold:

1. Identifying a suitable learning strategy for training Mask R-CNN He et al. (2017) to perform object segmentation on the *YCB video* dataset Calli et al. (2017, 2015), by exploiting both real images from the dataset and synthetic images, generated by tuning the method proposed in Dwibedi et al. (2017).

2. Integrating the on-line object detection method with Mask-RCNN so to obtain an on-line learning pipeline which performs object segmentation.

In the following sections, I cover the work done in collaboration with the master student, in order to accomplish the second point listed above and I report on the obtained preliminary results.

7.5.1 Overview of the pipeline

The pipeline resulting from the integration of the On-line learning method for object detection and Mask R-CNN is represented in Fig. 7.6. Note that, the main differences with respect to the on-line object detection have been reported in blue. The first main difference is the substitution of the Faster R-CNN's first layers with Mask R-CNN's ones. This brings two main contributions: (i) the adoption of the *RoI Align* which avoids the quantizations happening in the *RoI pooling layer*, improving pixel level precision (see Sec. 4.6 for further details) and (ii) the introduction of the *FPN* in the Convolutional backbone (see Sec. 4.4.1 for further details). The other major change with respect to the on-line object detection method is the adoption of the Mask R-CNN's last output layer for binary mask prediction (*Mask predictor* in Fig. 7.6). This last layer receives as inputs the feature map computed by the *Feature Extraction module* and the detections predicted by the *Detection module* and gives as

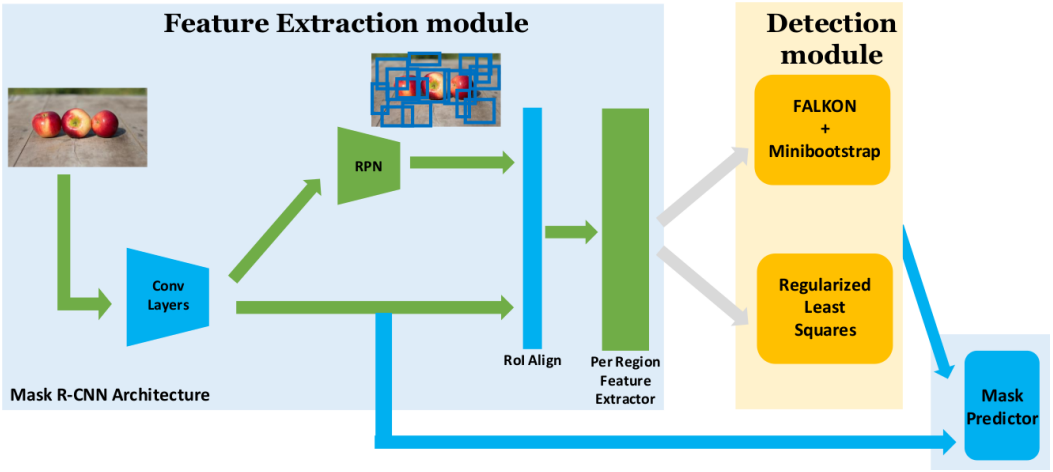


Figure 7.6: Overview of the pipeline resulting from the integration of the on-line object detection and Mask R-CNN. The improvements with respect to the on-line detection are reported in blue. Specifically, in the *Feature Extraction module*, the Faster R-CNN has been substituted with the first layers of Mask R-CNN. Moreover, the Mask predictor has been introduced to predict segmentation masks from the detections provided by the *Detection module*. For a detailed description see Sec. 7.5.1.

output a binary mask for each detection. Note that, the default Mask R-CNN's segmentation layer predicts class-specific masks, however the authors showed that using a class-agnostic segmentation is nearly as effective He et al. (2017). For a first attempt of this integration we opted for this latter solution since in the on-line learning scenario the classes are not known a-priori and a class-agnostic segmentation layer could facilitate the integration, while maintaining comparable performance.

The pipeline resulting from the integration takes as input an RGB image. The image is fed into the convolutional backbone that computes a feature map which is used to extract feature descriptors, through the *RoI align*, for each RoI predicted by the RPN. These regions are then classified and refined by the *Detection module* and finally, the *Mask Predictor* gives as output binary masks, taking as inputs the previously computed convolutional feature map and the predicted detections.

7.5.2 Learning the pipeline

Similarly to the learning process described in Sec. 7.1.2, the pipeline resulting from this integration is trained with a two-step method: (i) an *off-line stage* that is performed only once on a FEATURE-TASK and (ii) an *on-line stage* which can be performed every time a new TARGET-TASK needs to be learned.

The Off-line stage

During the off-line learning stage, the *Feature Extraction module* and the class-agnostic *Mask Predictor* are trained on the FEATURE-TASK. Specifically, this is achieved by optimizing the Mask R-CNN on the FEATURE-TASK, using the procedure described in He et al. (2017), considering the case of a class-agnostic segmentation layer. The *ResNet-50-FPN* has been used as backbone, integrated in Mask R-CNN as explained in He et al. (2017). Note that, during this stage, all the elements are fine-tuned on the FEATURE-TASK, which with respect to the datasets used previously in this chapter for the same purpose, needs to be annotated with objects locations at the pixel level, since segmentation ground-truth is required to train Mask R-CNN. Similarly to the original on-line object detection, this off-line stage needs to be performed only once. Afterwards, weights of the CNN backbone, the RPN and the Mask Predictor are retained and used during the on-line stage for feature extraction and mask prediction.

The On-line stage

The on-line learning stage is the same as in the on-line detection pipeline. Note that, this means that only the classifier and the bounding box refiner are trained on the TARGET-TASK while the *Mask Predictor* is not updated. We refer the reader to Sec. 7.1.2 for further details of this on-line stage.

7.5.3 Preliminary results

In this section, I will report on the preliminary experimental analysis carried out to understand the potential of the pipeline.

Experimental analysis

For the *Off-line stage*, the FEATURE-TASK has been generated from the YCB-video dataset Calli et al. (2017, 2015). Specifically, it is composed by a mixture of real frames taken from the dataset and synthetic images generated using the *Cut, paste and learn* strategy Dwibedi et al. (2017). For the *On-line stage*, the TARGET-TASK has been obtained by acquiring a set of sequences depicting 21 objects from the ICWT. Specifically, the resulting dataset contains a sequence for each object lying on a table and taken from different view points. For evaluating the performance of the on-line pipeline, a similar table-top dataset has been used which has been acquired with an analogous strategy. Performance

Figure 7.7: This picture is taken from Bunz (2019) and it shows randomly sampled examples of bounding boxes and segmentation predicted by the proposed pipeline.

has been evaluated considering the AP_{50} measure which is equivalent to the mAP used for object detection but considering an intersection over union at a pixel level to evaluate mask's precision.

The accuracy obtained in the aforementioned conditions is of 60.8% which, even though is not really high for the considered setting, is promising, showing the potential of the pipeline. Examples of segmented objects are reported in Fig. 7.7.

Ablation study

This first result motivated a further analysis to understand weaknesses and potential of the pipeline. For this reason, first of all, the contributions of the detection and of the segmentation have been decoupled. In order to achieve this, considering the previous experiment, the computation of the accuracy has been separated and the following performance values have been measured:

1. Performance of the task of object detection considering the output of the *Detection module*.

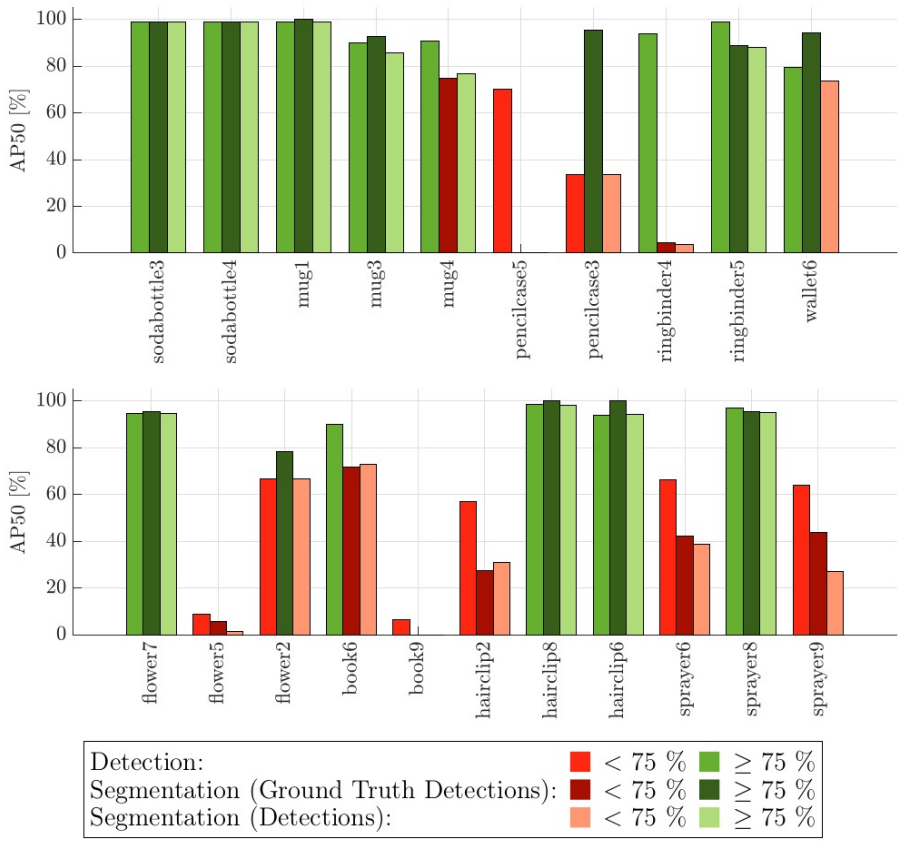


Figure 7.8: This figure is taken from Bunz (2019) and it shows the per-class accuracy obtained by the proposed pipeline. Specifically, for each object, the first bin represents the mAP achieved by the *Detection module*, while the second and the third bins represent the segmentation accuracy obtained by considering respectively the ground-truth boxes and the predicted detections as object locations.

2. Performance of the task of object segmentation. This has been done by using the ground-truth object locations instead of the bounding boxes predicted by the *Detection module* as input for the *Mask Predictor*.

The results are reported, for each class, in Fig. 7.8. For each object, the first bin represents the mAP achieved by the *Detection module*, while the second and the third bins represent the segmentation accuracy obtained by considering respectively the ground-truth boxes and the predicted detections as object locations.

Fig. 7.8 shows that, while for just four objects the results are aligned with the average, for the most of them, the accuracy is either around 90-100% or very low, around 10% or less. To study this discrepancy and understand its causes, the region proposals extracted by the RPN in the *Feature Extraction module* have been qualitatively evaluated. Fig. 7.9

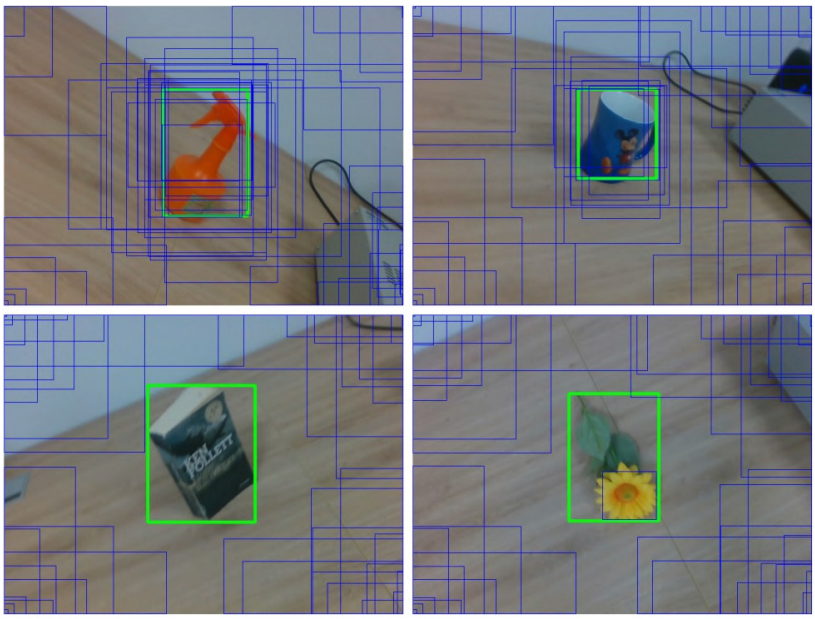


Figure 7.9: This figure is taken from Bunz (2019) and it shows four frames examples depicting the top 50 proposals (blue) and ground truth (green) for different objects (i.e. form the top left to the bottom right: *sprayer8*,*mug1*, *book9* and *flower5*).

shows some example frames used for this analysis, where the 50 top scoring region proposals have been drawn as blue boxes. What can be observed is that, while for the objects that achieved the best results, like e.g. *sprayer8* and *mug1* in Fig. 7.9, the object of interest is fully bounded with many proposals, this is not true for the objects that instead achieved the worst accuracy, like e.g. *flower5* and *book9* in Fig. 7.9. This means that when good features and region proposals are provided, the system can successfully perform object detection and segmentation, while the poor accuracy is caused by the lack (or absence in some cases) of proposals. These results are promising, showing the effectiveness of the pipeline, while a more accurate training of the *Feature Extraction module* is required.

Chapter 8

Weakly Supervised On-line Object Detection

State-of-the-art methods for object detection generally rely on architectures trained end-to-end on datasets carefully collected and annotated (once and off-line). This provides an effective baseline, but it represents a limit when considering their deployment on a humanoid robot to unconstrained environments. Adapting to these scenarios includes learning to recognize novel, specific object instances, as well as tuning to specific settings, by relying on data gathered during the robot's operation ("on-line"), which may be scarce or not annotated. Moreover, the training may be constrained in terms of computational resources and time.

In the first part of the project, I demonstrated that an automatic procedure, devised considering a constrained scenario, can be used to acquire a sufficient dataset to train a detection model. However, I also reported experiments showing a generalization problem when considering different settings (see Chapter 6). In the last part of the project I focus, therefore, on the problem of training and in particular *adapting* object detectors on-line on little, partially annotated data, to novel scenarios where automatic annotation is not possible.

To address this problem, I build on previous work Maiettini et al. (2017, 2018, 2019), where I proposed a method to train a humanoid robot to detect novel object instances with training time in the order of seconds and only a few hundred frames. In Maiettini et al. (2017, 2018), however, supervision originated from interaction with a human teacher, while generalization to different background and light conditions was limited by the small number of training examples. As a part of my project, I propose a strategy that allows the robot to adapt an object detector by acquiring new training samples with limited human intervention. The main idea is that, when faced with a new setting, the robot can iteratively adapt the object detector by parsing incoming images and either annotating them autonomously or

asking for human help. This weakly supervised strategy integrates the fast object detector proposed in Maiettini et al. (2018, 2019), described in Chapter 7, with an adapted version of the Self-Supervised Sample Mining, SSM Wang et al. (2019); Wang et al. (2018).

The resulting method shows successful results and allows the detection models to adapt to the new conditions, while limiting the amount of novel annotated images. The rest of the chapter is organized as follows: Sec. 8.1 describes the proposed pipeline in detail and Sec. 8.2 presents results from the considered benchmarks.

8.1 Description of the pipeline

In the scenario considered for this approach, similarly to the one considered in Chapter 7, a robot is asked to detect a set of object instances in an unconstrained environment (the TARGET-TASK).

Following the same paradigm as in Sec. 7.1, the assumption is that the detection system is initialized with a set of convolutional weights, previously trained off-line on a separate set of objects (the FEATURE-TASK), using the method described in Sec. 7.1.2. A first detection model is trained during a brief interaction with a human (as explained in Sec. 6), in a constrained scenario (the TARGET-TASK-LABELED). The robot then explores the environment autonomously, acquiring a stream of images in a new setting. These images are not labeled (TARGET-TASK-UNLABELED) and are used to adapt the detector.

The pipeline uses the on-line detection algorithm described in Chapter 7 and an adaptation of the weakly supervised approach of SSM Wang et al. (2019); Wang et al. (2018). The detector is adapted thanks to the additional training data which is either automatically labeled by the robot (called pseudo ground truth in the following) or labeled with human supervision.

8.1.1 Pipeline description

The proposed pipeline is divided into two main modules (see Fig. 8.1): an (i) *On-line Object Detection Module* (OOD) and a (ii) *Weakly Supervised Module* (SSM). The first one predicts bounding boxes and labels and can be trained in few seconds as a new dataset is available, while the second one processes the predictions generated by the former one on a stream of (unlabeled) images in order to generate their annotations. While the *On-line Object Detection Module* has been described in details in Sec. 7.1, I cover the description of the *Weakly Supervised Module* in this section.

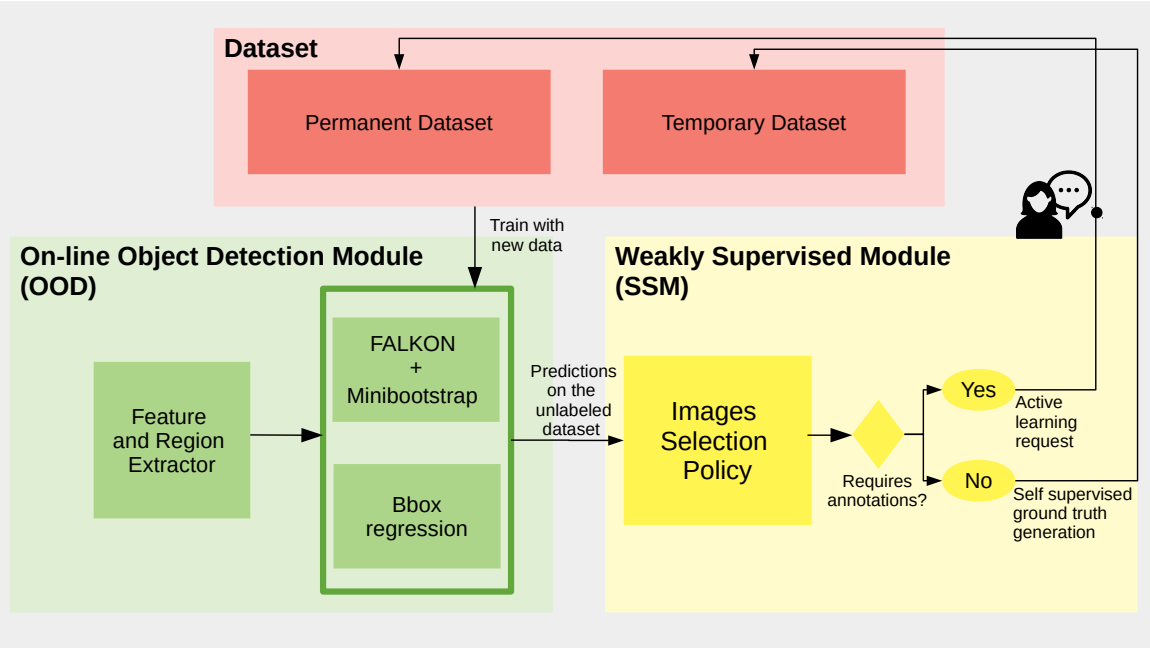


Figure 8.1: Overview of the proposed pipeline. The on-line detection system proposed in Maiettini et al. (2018) (green block) is integrated with a weakly supervised method Wang et al. (2018) (yellow block) that combines a self supervised technique to generate pseudo ground truth (*Temporary Dataset*), with an active learning strategy to select the hardest unlabeled images to be asked for annotation and added to a growing database (*Permanent Dataset*). Please, refer to Sec. 8.1 for further details.

Weakly Supervised Module

The aim of this module (yellow block in Fig. 8.1) is to generate a new training set by combining images annotated by the robot autonomously, with those annotated with human supervision. This is achieved with an iterative process Wang et al. (2018). For each iteration, the predictions of the current detection model on the images acquired by the robot (the TARGET-TASK-UNLABELED) are evaluated in order to identify (i) those detections that can be used as training set (pseudo ground truth) and (ii) those that need to be labeled with a human intervention. The dataset resulting from this process is used to train a refined version of the model with the *On-line Object Detection Module*.

For this module, the proposed approach relies on the weakly supervised method presented in Wang et al. (2018). It combines a self supervision based on a *Cross Image Validation* to select a reliable pseudo ground truth, with an active learning policy to pick the most informative unlabeled samples and ask for their annotation. Specifically, the *Cross Image Validation* is performed for each unlabeled image of the TARGET-TASK-UNLABELED and is designed as follows: the current detection model is tested on an unlabeled image, then,

the consistency of the predicted detections is evaluated by (i) pasting them into different annotated images and (ii) using the current detection model to predict them. If the detection is confirmed for the majority of the cases, it is considered consistent (the reliability is measured by a *Consistency score*), and thus usable as pseudo ground truth.

Instead, for the active learning process, the selection criteria is based on the classical uncertainty-based strategy Lewis and Gale (1994) where the policy is to ask for annotations of the least confident samples, (the *Consistency score* computed previously is used as measure of confidence of the image).

8.1.2 Training the pipeline

The learning process of the proposed method is divided into three phases:

1. An *Off-line learning stage* for the feature extractor, on a FEATURE-TASK.
2. A *Fully supervised learning stage* with a few seconds of interaction with a human, on the TARGET-TASK-LABELED, in order to get a first detection model, performed on-line.
3. A *Weakly supervised learning stage*, where the previously trained detector is used to generate pseudo ground truth, or queries for image annotations, on the TARGET-TASK-UNLABELED

For the first two stages the reader can refer to Sec. 7.1 for a detailed description. Specifically in this pipeline, I adopted ResNet50 He et al. (2015) as a CNN backbone for Faster R-CNN for the off-line stage and the combination of FALKON, the Minibootstrap and the RLS regressors as presented in Sec. 7.1, is used to train a detector on the TARGET-TASK-LABELED. This model will be, consequently, used as a seed model for the weakly supervised learning phase on the TARGET-TASK-UNLABELED. The third learning stage is described in the following paragraph.

Weakly Supervised Phase

After the first supervised learning phase, the weakly supervised process on the TARGET-TASK-UNLABELED starts. For this phase, the pipeline relies on the protocol proposed in Wang et al. (2018). Specifically, this is a process that iterates on the TARGET-TASK-UNLABELED to progressively refine the detection model. Each iteration is structured

as follows: the images of the unlabeled dataset are predicted with the current model and the consistency of the predictions is evaluated with the *Image Cross Validation* procedure illustrated above. The images with a high *Consistency score* are added as pseudo ground truth while the ones with a low *Consistency score* or the ones ambiguous for the detector (specifically, the images where the same region is predicted with two positive categories) are added to the set that needs to be asked for labeling.

The dataset composition at each iteration is controlled by a parameter that limits the number of images to be added to both sets, which is defined as a percentage of the TARGET-TASK-LABELED. The strategy adopted to set this parameter in Wang et al. (2018) is to allow, for early iterations, a higher number of images to be labeled, while, in subsequent iterations, an increasing number of pseudo labeled images can be added.

After this pruning, the images considered as pseudo ground truth are added to a *Temporary Dataset*, while the ones that need annotation are asked to be labeled and then added to a *Permanent Dataset* (see Fig. 8.1). Note that, while at the beginning of this iterative procedure the first one is empty, the latter one already contains the TARGET-TASK-LABELED. At the end of each iteration, while the *Permanent Dataset* is retained (it thus grows at each iteration), the *Temporary Dataset* is cleaned. For further details on this weakly supervised approach refer to Wang et al. (2018).

Note that, I adopted the protocol of Wang et al. (2018), but I replaced the fine-tuning of Region-FCN Dai et al. (2016b) with the fast learning method proposed in Maiettini et al. (2018), thus reducing the training time at each iteration from minutes/hours to a few seconds, allowing to use the pipeline in an on-line scenario. Another important distinction with respect to the original SSM algorithm is that, in the proposed pipeline, at each iteration, the detector is trained from scratch on the composed image set, while in SSM the Region-FCN is fine-tuned with a warm restart from the weights obtained at the previous iteration.

8.2 Experimental evaluation

In this section, I first describe the datasets used for evaluation (Sec. 8.2.1), then I provide details about the setup used for the experiments (Sec. 8.2.2) and finally I present the performance achieved by the proposed pipeline on two different scenarios (Sec. 8.2.3 and Sec. 8.2.4). Note that, a preliminary experimental analysis of the SSM method is reported in Appendix E. This analysis has been carried out with the aim of verifying the effectiveness of the weakly supervised strategy proposed in SSM, in the robotic scenario considered in this project.

8.2.1 Datasets description

For evaluating the proposed method, I considered the ICWT, as benchmark and a new set of frames sequences, depicting table top scenarios acquired with the R1 robot Parmiggiani et al. (2017). Please refer to Appendix A for a detailed description of the ICWT, while a description of the table top dataset is reported below.

Table Top Dataset

To prove the generalization capabilities of the proposed integration to different settings, I collected a table top dataset (that is publicly available[1]) by using the R1 robot Parmiggiani et al. (2017). For this dataset, I selected 21 objects from ICWT.

The data acquired is split in 2 sets of sequences. In each set, I considered a different table cloth: (i) pink/white pois (hereinafter referred to as POIS) and (ii) white (hereinafter referred to as WHITE). For each set, I split the 21 objects in 5 groups, and I acquire 2 sequences for each group for the WHITE set, and 1 sequence for each group for the POIS set, gathering a total of 2K images for the WHITE set and 1K images for the POIS set.

For each sequence, the robot is placed in front of the objects and executes a set of pre-scripted exploratory movements to acquire images depicting the objects from different perspectives, scales, and viewpoints (please, refer to Appendix F for further details about the acquisition procedure). I used a table top segmentation procedure to gather the ground truth of the object locations and labels, and I manually refined them using the *labelImg* tool[2]. See Fig. 8.2 (second and third rows) for some example images.

8.2.2 Experimental setup

To show the effectiveness of the proposed integration, I present results on two different experiments. I firstly validate the pipeline on ICWT, then, I consider the scenario of a robot trained with human interaction to detect a set of objects, which needs to adapt and refine the detection model in order to generalize to a different setting. Specifically, as proof of concept, the table top dataset, described above, is considered as the new setting. This is a challenging task as the robot is trained by a human demonstrator while holding the objects in the hand and it is later required to detect objects when they are placed on a table (see Fig. 8.2 to compare the two settings). Fast adaptation is required to avoid large performance drop as demonstrated by the presented experiment.

[1] https://robotology.github.io/iCubWorld/#icubworld-transformations-modal/
[2] https://github.com/tzutalin/labelImg

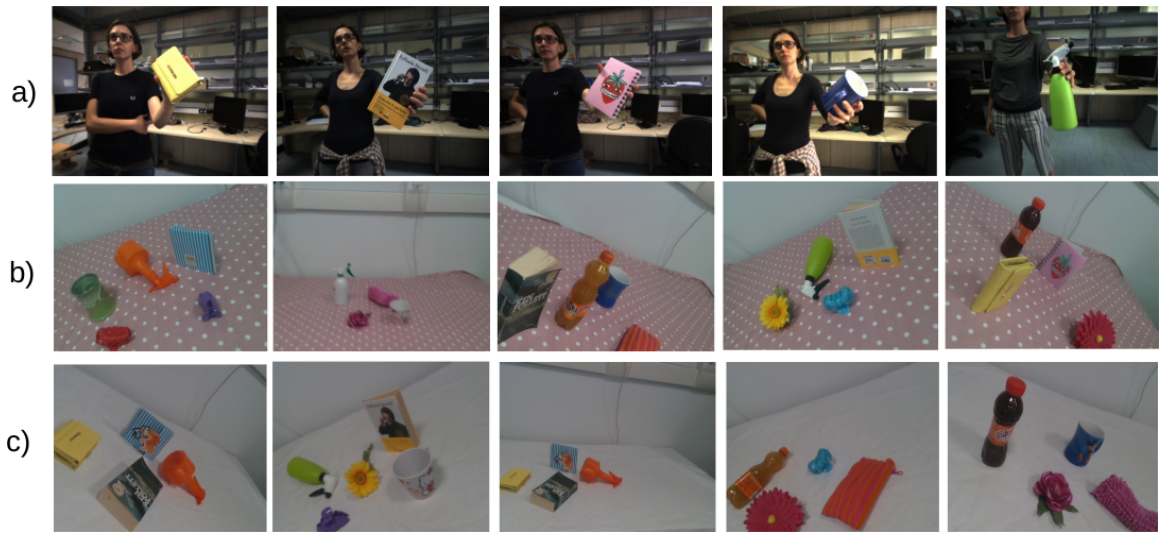


Figure 8.2: Examples images of the datasets used for this work: **a)** ICWT dataset; **b)** POIS cloth in the table top dataset; **c)** WHITE cloth in the table top dataset.

In both experiments, within the scenario described in Sec. 8.1, I defined as FEATURE-TASK the same identification task of 100 objects considered for the experimental analysis reported in Sec. 7.2.3, and I trained Faster R-CNN end-to-end on this task by setting the number of iterations to 165K and to 110K when learning respectively the RPN and the detection network.

Note that, when considering the TARGET-TASK-UNLABELED, I simulate the human intervention for providing annotations, by fetching the actual ground truth from the dataset. Performance are reported in terms of mAP as defined for Pascal VOC 2007 Everingham et al. (2010).

All experiments reported have been performed on a machine equipped with Intel(R) Xeon(R) E5-2690 v4 CPUs @2.60GHz, and a single NVIDIA(R) Tesla P100 GPU. Furthermore, the RAM usage of FALKON has been limited to at most 10GB.

8.2.3 Experiments on the iCubWorld Transformations dataset

For this experiment, I define as TARGET-TASK a 30-object identification task, considering the same object instances as in Sec. 7.2.3 (please, refer to Appendix A for further details about the used dataset). However, for each object, I use the TRANSL sequence (for a total of $\sim$2K images) as TARGET-TASK-LABELED and the union of the 2D ROT, 3D ROT and SCALE sequences (for a total of $\sim$6K images) as the TARGET-TASK-UNLABELED. This simulates a situation where only a simple sequence is fully annotated

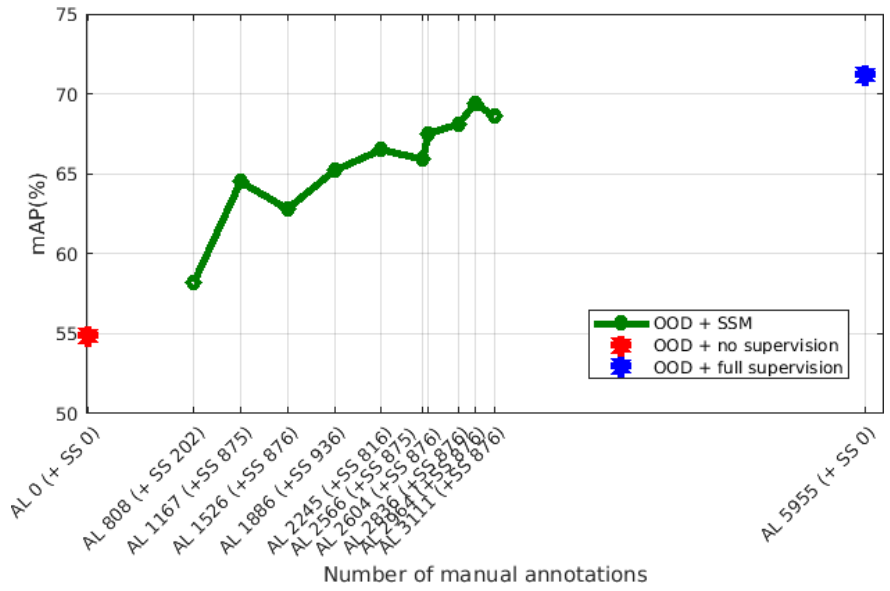


Figure 8.3: Benchmark on ICWT. The figure shows (i) the mAP trend of the proposed pipeline, as the number of annotations required on the TARGET-TASK-UNLABELED grows (*OOD + SSM*), compared to (ii) the accuracy of a model trained only on the TARGET-TASK-LABELED (*OOD + no supervision*) and to (iii) the mAP of a model trained with full supervision on the TARGET-TASK-UNLABELED (*OOD + full supervision*). The number in parenthesis reported in the x axis represents the number of images selected by the self supervision process at each iteration.

and other sequences are not. As a test set, I used 150 images from the MIX sequence of each object, which annotations have been manually refined adopting the *labelImg* tool[3].

In Fig. 8.3, I report the mAP trend (**green line**) with respect to the total number of images asked for annotation in the TARGET-TASK-UNLABELED (in parenthesis the number of samples selected by the self supervision process is specified). Note that, as the images get accumulated at every iteration, in order to calculate how many images are required by the robot, one has to take the difference of the indicated number with the one at the previous iteration.

The **red point** shows the mAP on the considered test set, achieved after the supervised learning phase, i.e., after training the detection module on the TARGET-TASK-LABELED. Thus, it can be considered as the lower-bound for this experiment. The **blue point** represents the mAP achieved by training the detection module on the union set of the TARGET-TASK-LABELED and TARGET-TASK-UNLABELED (fully manually annotated). Thus, it can be considered as the upper-bound of this experiment. As it can be observed, nearly half of

[3]https://github.com/tzutalin/labelImg

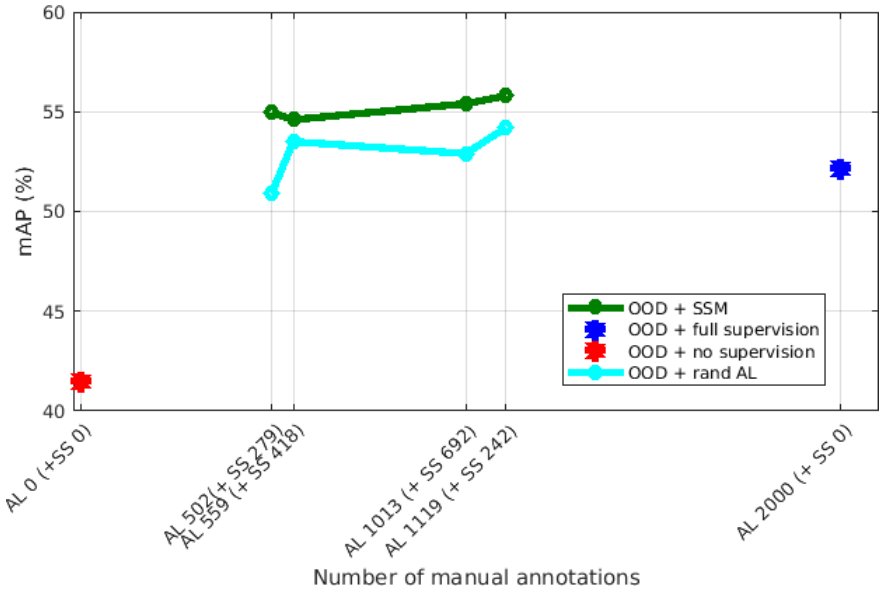


Figure 8.4: Benchmark on the table top dataset. The figure shows (i) the mAP trend of the proposed pipeline, as the number of annotations required grows (*OOD + SSM*), compared to (ii) the mAP of a model trained on the TARGET-TASK-LABELED (*OOD + no supervision*) and to (iii) the mAP of a model trained with full supervision on the TARGET-TASK-UNLABELED (*OOD + full supervision*). In this experiment, I also compare with the mAP of a model trained only on annotated images randomly selected (*OOD + rand AL*). The number in parenthesis in the x axis represents the number of images selected by the self supervision process at each iteration.

the images of TARGET-TASK-UNLABELED are enough to obtain $\sim$70% of mAP with a drop in performance of $\sim$1.2% with respect to the fully supervised case.

Each point of the green line has been obtained by retraining a new set of 30 FALKON classifiers, with the Minibootstrap, on the data accumulated after the weakly supervised iteration. As the dataset increases, the training time increments from $\sim$40 seconds to $\sim$60 seconds (overall average of $\sim$55 seconds for each step).

8.2.4 Experiments on Table-Top scenario

For this experiment, I define as TARGET-TASK an identification task among 21 object instances chosen from the ICWT –excluding those used as FEATURE-TASK (please refer to Appendix A for details about the tasks). As TARGET-TASK-LABELED, a subset of the available images from the TRANSL, 2D ROT, 3D ROT and SCALE sequences (for a total of $\sim$ 5600 images) is selected, while the 2K images of the WHITE table top set (see

Sec. 8.2.1) are considered as TARGET-TASK-UNLABELED and the POIS table top set is chosen as test set.

Fig. 8.4 shows the result of this experiment. As before, the **green line** reports the mAP with respect to the increasing number of images asked for annotation (in parenthesis the number of self-annotated images at each iteration).

Similarly, the **red point** shows the mAP on the considered test set, achieved after the supervised learning phase on the TARGET-TASK-LABELED, while the **blue point** represents the mAP obtained by training the on-line detection module on the union set of the TARGET-TASK-LABELED and TARGET-TASK-UNLABELED (fully annotated).

As it can be observed, just 1/4 of the full TARGET-TASK-UNLABELED dataset was enough to train a model with even a higher accuracy ($\sim$55%) than the one obtained with full supervision ($\sim$52%). This may be due to the fact that, by using all images from the TARGET-TASK-UNLABELED, the model may overfit the scenario of the white table cloth, which causes a poorer performance when testing on images depicting a different table cloth. Our findings suggest that AL algorithms may help reducing overfitting, confirming what has been previously reported in the literature (see, e.g., Burbidge et al. (2007)).

One may argue that, in order to avoid the overfitting caused by considering all the images in the TARGET-TASK-UNLABELED (**blue point**), a random sub-sampling of the images to label would suffice. To this end, in Fig. 8.4, I also compare the proposed approach with a model trained on the same number of images as the ones selected by the AL process, but randomly sampled (**cyan line**). It can be noticed that, while the mAP obtained is relatively high, it also presents a gap with respect to the performance achieved with the proposed integration, demonstrating the effectiveness of the active learning and self supervision processes in choosing the more meaningful samples.

As for the previous experiment, each point of the green line has been obtained by retraining a new set of 21 FALKON classifiers, with the Minibootstrap, on the data accumulated after the weakly supervised iteration. As the dataset increases, the training time increments from $\sim$35 seconds to $\sim$47 seconds (overall average of $\sim$42 seconds for each step).

Part IV

Robotic Application

Chapter 9

Deployment on Humanoid Robots

As part of this Ph.D. project, after the extended experimental evaluation, the on-line object detection pipeline (presented in Chapter 7), has been brought to run on robotic platforms to test its capabilities in real world scenarios. Ad-hoc modules have been developed to wrap the on-line learning algorithm, to orchestrate the application and to create interfaces with the robots and communication between the different modules. These new modules have been integrated with existing ones, taken from the Robotology[1] repository. The application has been developed to run on both the R1 Parmiggiani et al. (2017) and the iCub Metta et al. (2010) humanoid robots, but it can be easily deployed on any robotic platform with an RGB-D or a stereo vision camera. This has been made possible partly due to the fact that it has been developed using the YARP[2] middleware that allows communication between the different modules.

The outcome is an application that allows to train a robot to detect novel objects in just a few seconds in a natural student-teacher interaction scenario. In more details, the human needs to show the novel object to the robot, which in turn is able to track and follow it while acquiring the stream of images coming from its camera. At that point, a detection model is trained using the method explained in Chapter 7, with the new acquired data. Finally, when the robot is not learning a new object, i.e. at inference time, it is able to detect objects in the scene, relying on the RGB image only, without human intervention.

In the following sections, I provide an overview of the pipeline, describing the main modules (see Sec.9.1) and I describe the main functioning states (see Sec. 9.2 and 9.3). A video of the application is publicly available[3] and Fig. 9.2 and 9.3 are two snapshots depicting different phases of the applications.

[1] https://github.com/robotology
[2] https://github.com/robotology/yarp
[3] https://www.youtube.com/watch?v=eT-2v6-xoSs

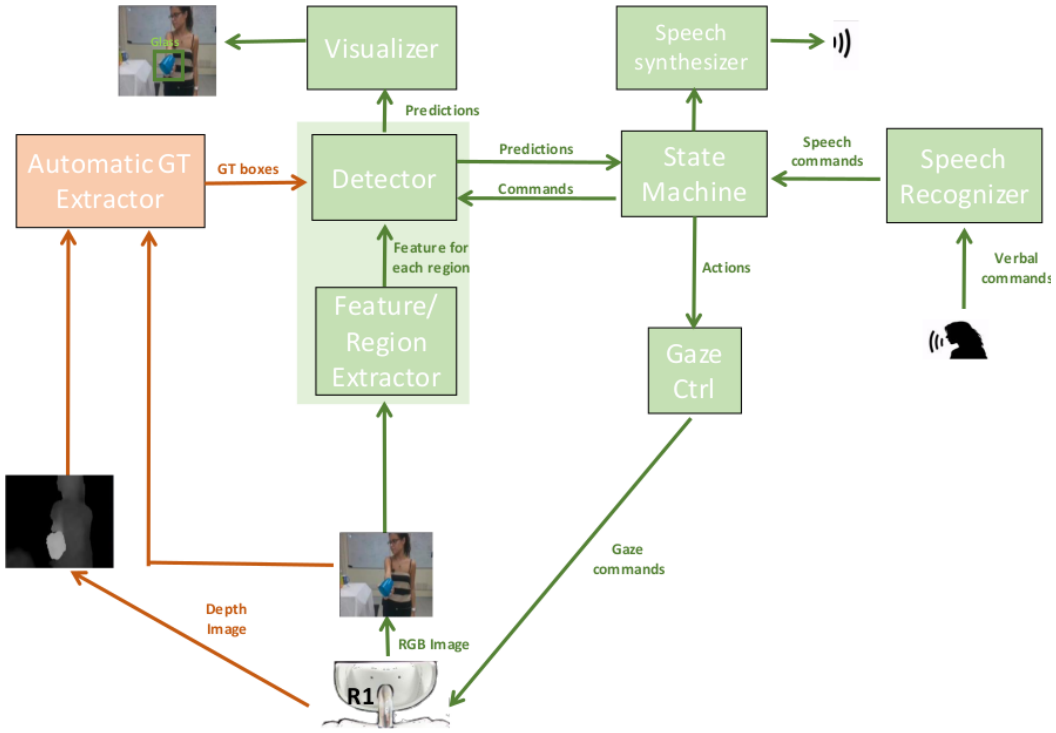


Figure 9.1: This picture shows an overview of the on-line detection application (more details can be found in Sec. 9.1).

9.1 Pipeline overview

In this section, I overview the main components of the proposed application. Please, refer to Fig. 9.1 for a pictorial representation of the main input/output connections between the modules. The input signals for the application are: (i) an RGB image, (ii) a depth image and (iii) the verbal commands or questions about the scene from the user, while the outputs are (i) the list of predictions, both in the form of a YARP Bottle and of drawn boxes and labels on the original image and (ii) the answers to user's questions.

State machine

The application is composed of several modules which are orchestrated by a *State machine*[4] (implemented in LUA[5] and rFSM[6]). There are 4 main states of the application: (i) *train*, (ii) *forget*, (iii) *inference* (which is the default state) and (iv) *interaction*. The different states are triggered by the user's commands or questions and activate different behaviors of the modules

[4]Implemented by Vadim Tikhanoff
[5]https://www.lua.org/
[6]https://github.com/kmarkus/rFSM

which are connected via YARP's ports. During the *train* and *forget* states the detection model is modified for either adding or removing the knowledge of an object, while during the *inference* and the *interaction* states the detection model's predictions are used in the first case to show the detected objects and in the second one, to reply to the user's questions about objects locations. More details about the different modalities are reported in the next sections (see Sec. 9.2 and 9.3).

Detection system

The *Feature and region extractor* and the *Detector* are the basis of the *Detection system*. The first one is a wrapper of the *Feature Extraction module* described in Sec. 7.1. This takes an image as input and extracts regions of interest, encoding them into convolutional features. The *Detector*, instead, implements and adapts the regions classifier and refiner, proposed as a contribution to this thesis, i.e., the set of FALKON classifiers, the Minibootstrap and the RLS for bounding box refinement (namely, the *Detection module* described in Chapter 7.1). It contains an internal state machine that allows to switch between the different behaviors, according to the different states of the application. When the application is in state *train*, the Detector processes the incoming annotated data and trains a new FALKON classifier and a new RLS refiner, using the Minibootstrap procedure, as explained in Sec. 7.1. When, instead, the state is *forget*, the Detector removes the classifier and the rls refiner responsible for the specified classes. Finally, during the states *inference* and *interaction*, the detection system receives the stream of images as input and provides a list of predicted detection as output. I implemented the two modules that constitute the Detection system in MATLAB[7].

Gaze controller

The *Gaze controller*[8] is the module which creates an interface between the robot's head controller (head and neck) and the application. During the *train* state, it is used to make the robot track and follow the object of interest, using the gaze. During the *interaction* state, instead, it is used to make the robot fixate on specific points containing predicted detections that are then used to reply to the user's questions.

Automatic GT Extractor

The *Automatic GT Extractor*[8] is the module that implements the procedure for the automatic data acquisition explained in Sec. 6.1.1. It exploits the depth information in order to segment

[7] https://it.mathworks.com/products/matlab.html

[8] This module has been taken from the Robotology repository

the blob of pixels belonging to the closest object. It computes a bounding box surrounding this particular blob and sends it to the detection system and to the *Gaze Controller*. The detection system uses the bounding box as ground truth for training a new detection model while the *Gaze Controller* uses this information to generate a target point to follow.

Speech recognizer and synthesizer

The *Speech recognizer*[9] is the module that takes as input the voice signals from the user and translates it into a transcript that is then interpreted and sent to the *State machine* as command. The *Speech synthesizer*[9], instead, is the module that converts the answers elaborated from the application into sounds that construct the vocal reply from the robot.

Visualizer

The *Visualizer* is the module that receives the list of predicted detection from the Detection system and draws them on the original image for a visual representation. I implemented this module using Python[10].

9.2 Learning or forgetting an object

In the proposed application, the detection model can be modified on-line. Specifically, the time required to update the model to either add or delete the knowledge of an object is in the matter of a few seconds.

Learning a new object

The *Train* state of the application is triggered by the user command *"Have a look at the *object_name*"*. The robot replies "Let me have a look at the *object_name*" and the learning phase begins. There are a few (usually around 30) seconds of images acquisition, where the user needs to show and hold the object in his/her hand and show it to the robot under different view points and perspectives. During this phase, the *Automatic GT extractor* computes the automatic ground truth bounding boxes and sends it to the *Detection system* and to the *Gaze Controller*. While the latter one uses this information to let the robot track the object, the first one gathers it together with the label provided verbally and the stream of images as an automatically acquired dataset. During the acquisition, the regions of interest

[9] This module has been taken from the Robotology repository
[10] https://www.python.org/

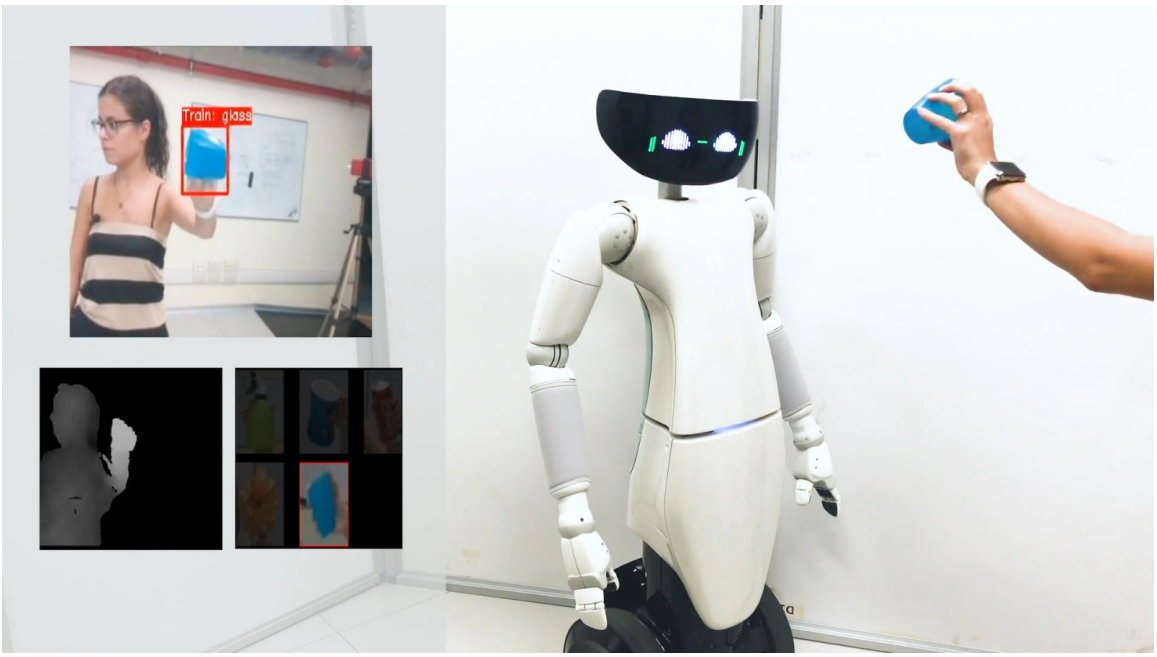


Figure 9.2: This picture is a snapshot taken from the video of the application during the *Train* state.

and features are extracted from the collected images and stored in the memory. At the end of the acquisition (the actual number of frames can be set and modified by the user) a new detection model is trained by the *Detector*, as explained in Sec. 7.1 (i.e., a new FALKON classifier and a new rls for bounding box refinement are trained to detect the novel object). Fig. 9.2 shows a snapshot of the learning phase. When this phase is completed, the application returns to the *inference* state.

Forgetting objects

The state *Forget*, instead, can be triggered by one of the following commands:

- *"Forget the *object_name*"*, which makes the model delete the classifier and the rls of that particular object.
- *"Forget all objects"*, which delete all the models.

When a "forget" command is given, the *Detection system* removes the classifiers and the rls relative to the objects mentioned in the command. When this phase is completed, the application returns to the *inference* state.

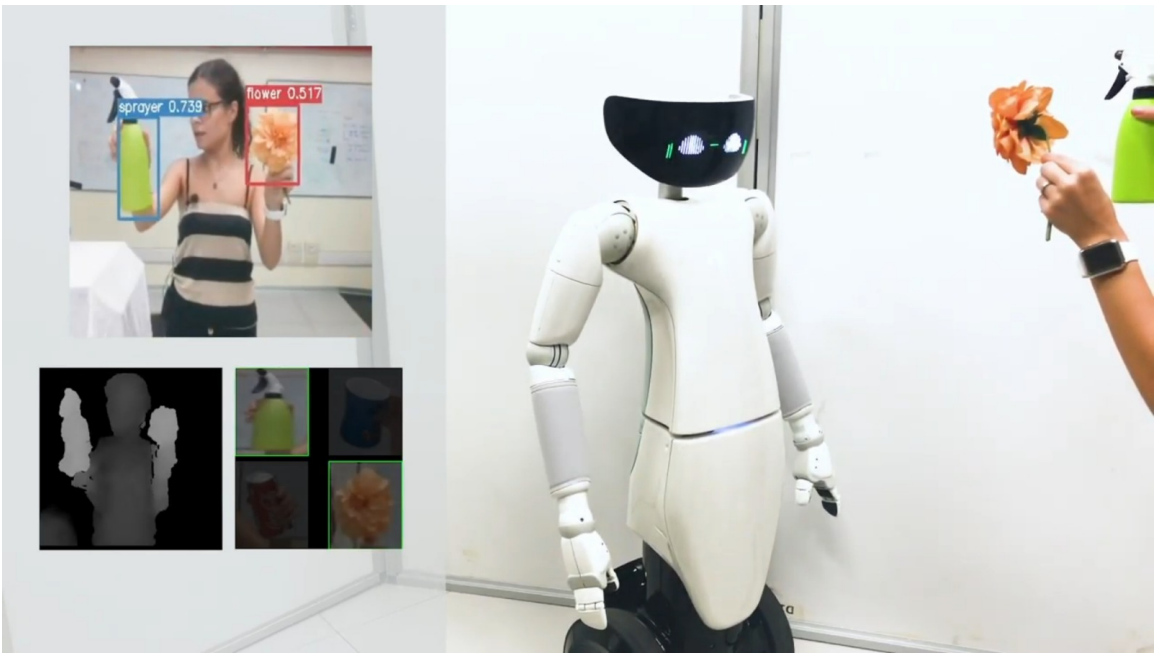


Figure 9.3: This picture is a snapshot taken from the video of the application during the *Inference* state.

9.3 Detecting objects in the scene

The application's default state is the *Inference* state, where the *Detection system* receives a stream of images as input and it provides as output the list of predicted detections for each frame. For each image, RoIs and features are extracted and then processed by the *Detector*. At that point, the output is visualized with the *Visualizer*. Fig. 9.3 shows a snapshot of the inference phase.

While the system is in state *"Inference"*, the user can ask questions to the robot about the detected objects. This switches the state of the application to *"Interaction"*. The application can deal with various question/commands which can easily be extended. The one that have been implemented are listed below:

- *"Look at the *object_name*"*: to accomplish this command the robot should look at the desired object if it is present in the list of the detected objects.
- *"Look around"*: to accomplish this command the robot should change periodically the fixation point, moving the gaze, alternating randomly between the different detected objects in the scene.
- *"Where is the *object_name*"*: to accomplish this command the robot should reply with the list of objects that are close to the mentioned one (within a specified radius).

- *"What is close to the *object_name*"*: to accomplish this command the robot should reply with the name of the closest object to the one requested.

Please, refer to the available on-line video[11] for an example of this interaction. Note that, in the case represented in the video the model used was Faster R-CNN, resulting from the training procedure described in Sec. 6.1.

[11] https://www.dropbox.com/s/njykynrdnrbovqz/ElisaDetectionFinal.mp4?dl=0

Chapter 10

A real use-case: learning to detect from afar

The problem of detecting objects from distance and more generally of scale variance is a well known issue in computer vision domain Eggert et al. (2017), which plays an important role in applied fields. In robotics, e.g., where the platforms can navigate an environment, the view point might change significantly, dramatically modifying the size of the objects to be detected in the image, leading to a very poor accuracy if not properly handled.

As part of this project, I collaborated on improving the performance of the proposed detection application, described in Chapter 9, against scale variation. The use-case considered for this project was that of a robot which is asked to detect the objects of interest when entering in a room, from a distance of 2 meters or more. The solutions devised are simple yet demonstrated to be effective for the scenario considered. In the next sections, I start by introducing the problem (see Sec. 10.1), then I report on what has been done to improve the existing detection pipeline against the scale problem, showing qualitative results (see Sec. 10.2).

10.1 Problem definition

The problem of scale variation entails two main issues: (i) the different appearances that the objects have at different scales which makes the task of recognizing them more challenging and (ii) the poor pixels representation that an object has when seen from afar. Specifically, this latter issue can affect the recognition, since, for example, some patterns or textures are more evident from a closer view point and might disappear from afar. Moreover, it can affect

the detection even more heavily, since in more extreme cases the number of pixels used to represent an object might be not sufficient to "perceive" it.

This behavior is also accentuated when using CNN architectures for feature extraction. Specifically, CNNs, as explained in Sec. 3.2, progressively down-sample the extracted feature maps trough the alternating concatenation of convolutional and pooling layers. This is done to obtain a higher level of abstraction and to reduce the over-fitting in deeper layers. Moreover, it reduces the computational complexity of applying the model. Nevertheless, this produces a feature map with a lower resolution than the original image and this drawback is also emphasized as the depth of the CNN increases. A relationship has been formalized between the feature map resolution and the minimum size object which can be detected Eggert et al. (2017).

A naïve solution to this problem might be either to use images of higher resolutions or to up-sample the available ones if there is no possibility to acquire better data. This would lead to have richer information in the computed feature maps after the CNN's down-sampling Eggert et al. (2017). However, this does not actually solve the problem, but it works around it, since the down-sampling of the CNN is still happening. Moreover, this solution saturates. Indeed, there are intrinsic limitations in the hardware that does not allow to increase image resolution ad lib and after a certain threshold, up-sampling images does not provide additional information Eggert et al. (2017). Finally, higher resolution images might generate heavier computation, slowing down parts of the pipeline. This is not acceptable in some real world scenarios, as in our case, a robotic application.

More sophisticated solutions have been proposed to solve this problem, which are mainly based on improving the feature representation used for the classification. Specifically, in one kind of approaches lateral connections have been introduced Lin et al. (2017b), which is a procedure that allows to consider multiple feature maps taken at different levels of the CNN (and not only the last one). This should allow to consider different scaled descriptor of the same image enabling the classifier to "see" also far or small objects which representation would disappear at deeper levels. Another possible solution, instead, is based on the idea of using a set of features trained with the purpose of being robust to scale variation Eggert et al. (2017). For instance, a way to improve Faster R-CNN's accuracy against scale is to dynamically re-scale some input images, during training. The improvements brought have been proved empirically Eggert et al. (2016) and this is of interest, since down-sampling or up-sampling images cannot introduce new information. This has already been noted in Eggert et al. (2016), where the authors attribute this property to the receptive field of the network.

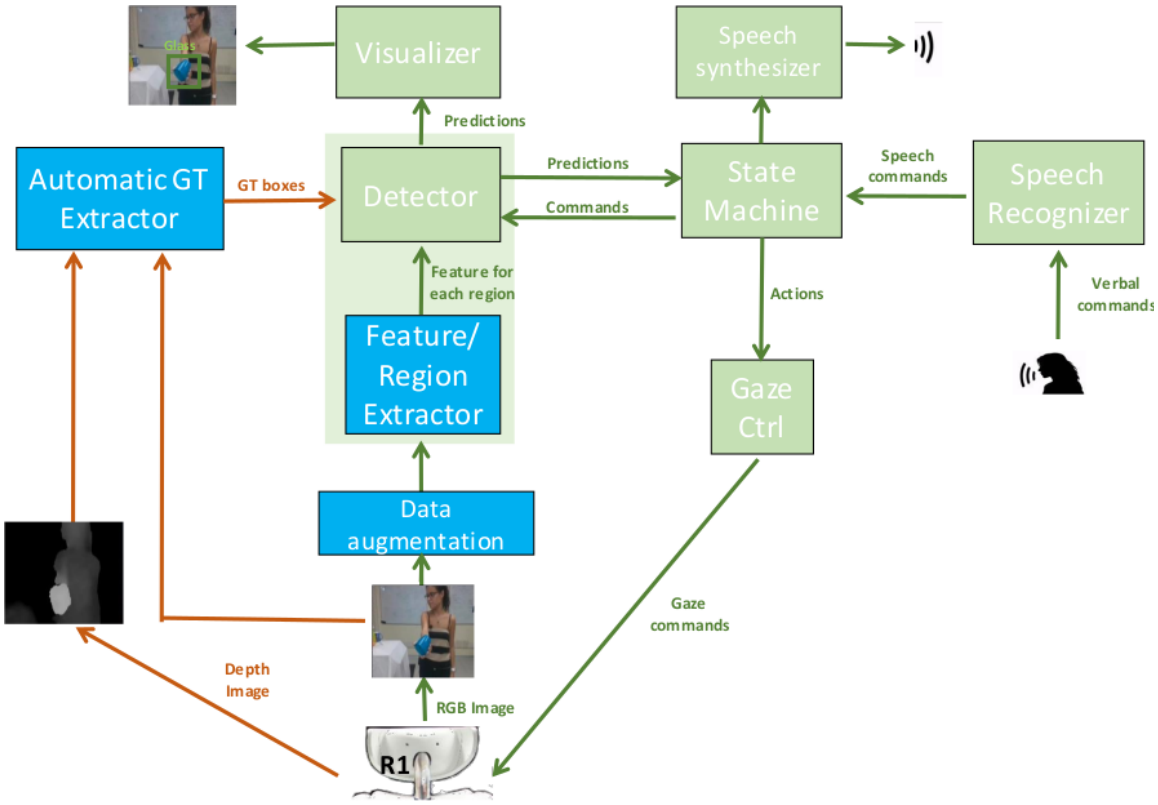


Figure 10.1: This picture shows the architecture of the improved on-line object detection application. The improvements are reported in blue and they involve the *Automatic GT extractor*, the *Feature and Region extractor* and a new module for *Data augmentation* (more details about these improvements in Sec. 10.2).

10.2 Improving the pipeline against scale variation

The on-line object detection pipeline described in Chapter 9 demonstrated to be effective for the setting it was developed in, but it presents limitations when trying to generalize to different settings (see Sec. 6.2.3) and for the scale problem, since no specific expedients are used against these issues.

More specifically, in the pipeline described in Chapter 9, the *Automatic GT extractor* which works using the depth information from the robot's sensor, functions properly in a range that is between 0.5 m and 1 m. The reason for this is that, considering relatively small objects, the blob of pixels depicting them from a distance that is further than 1 m is so small that the noise from the depth sensor might generate problems in the segmentation. This short range of movements limits the possibilities that the teacher has to show the objects at different scales, conditioning consequently the accuracy of the detector from further distances. Moreover, the feature extractor considered in Chapter 9 was trained on a *FEATURE-TASK*

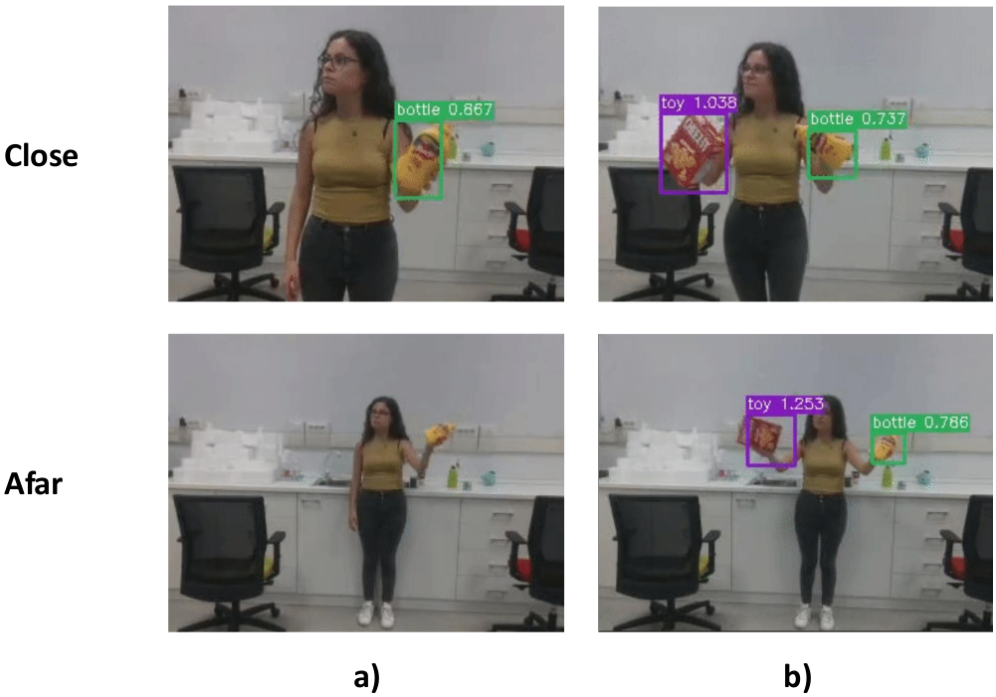


Figure 10.2: This figure shows snapshots of a qualitative comparison between the original system (a) and the improved system (b), reporting examples of detections from a close (*Close*) and afar (*Afar*) distances. Videos of the comparison can be found online.

from the *iCWT*, without any specific procedure devised against scale variations. Finally, the constant presence of the human in the image and of the hand around the handheld object produced the generalization problems described in Sec. 6.2.3.

In this part of the project, I collaborated to make the application effective along with objects seen from distance, working on all these aspects. Fig. 10.1 shows the diagram of the application resulting from the improvements. New or improved modules, with respect to the original application, are reported as blue blocks. As for the development of the original application, the platform used for testing was the R1 humanoid, but the improved application can run on any other robotic platform.

The resulting system has been tested qualitatively and has proved to work in the considered use-case scenario. While in the remaining of this section I explain the main enhancements brought to the system, videos which qualitatively show the improvements can be found online. They compare performance of the original system[1] and of the improved one[2] in experiments carried out on the R1. Finally Fig. 10.2 shows the comparison between snapshots of the two systems.

[1]https://www.dropbox.com/s/85yus21dwli59vg/afar_before.gif?dl=0
[2]https://www.dropbox.com/s/av3j4n8pmi7w3pw/afar_after.gif?dl=0

Figure 10.3: Example output frame of the *Skeleton retriever* module, based on the Openpose Cao et al. (2017) algorithm, taken using the Intel RealSense camera[3].

Improving feature extraction for scale variation

As a first step, I worked on the region and feature extractor, specifically adopting the technique of dynamically and randomly re-scaling some input images while training the feature extractor. I maintained the same *FEATURE TASK* as in Chapter 9, but allowed Faster R-CNN to be trained with some of the images randomly re-scaled by setting the width to six different values, i.e. [300, 400, 500, 600, 700, 800, 1000]. This brought to an improved feature extractor, which allowed to propose regions and extracting features more robust to scale variation.

Improving the automatic GT collection

In order to allow for an automatic collection of annotated data from afar, we needed to improve the *Automatic GT collector* to widen the user's range of action. With this aim, we integrated in the application a *Skeleton retriever* module[4]. Specifically, this module is a YARP wrapper of the Openpose Cao et al. (2017) algorithm, which adapts it to be used in a robotic application. The Openpose uses a non-parametric representation, referred to as Part Affinity Fields (PAFs) Cao et al. (2017), to learn to associate body parts with individuals in the image. Thus, this module allows for a multi-person 2D pose estimation, in the form of 25 body key-points identification. Fig. 10.3 shows an example output frame.

[3]https://www.intel.it/content/www/it/it/architecture-and-technology/realsense-overview.html
[4]This module has been taken from the Robotology repository

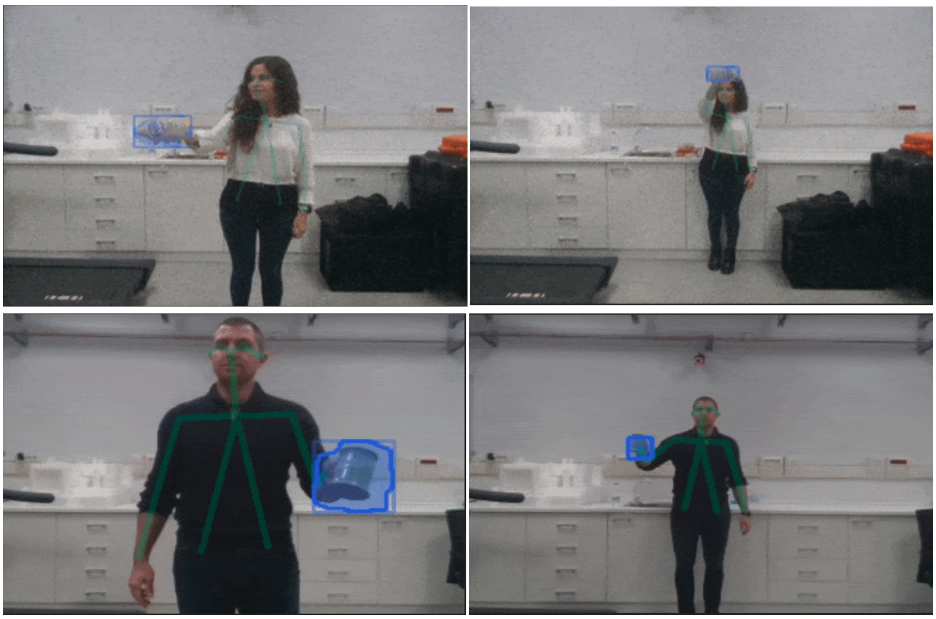


Figure 10.4: Example output frames of the resulting *Automatic GT collector*. The green lines overlapping the user's body highlights the upper body key-points output by the *Skeleton retriever*, while the blue box and boundary show the extracted ground truth, using the method described in Sec. 10.2.

The information of the positions of the human body's key-points is used in the new *Automatic GT collector*[5] to drive robot's attention toward the user that is teaching a novel object. Specifically, as a first step the robot identifies the teacher by looking for a specific predefined movement of the hand such as, in this case, the lifting of the user arm. Subsequently, it focuses on what is held and demonstrated by the teacher. Knowing the precise position of the object, the robot can segment it, using the depth information, considering the assumption that the shown object's 3D points are all close to the teacher's hand.

Data augmentation

Finally, to mitigate the generalization problem caused by the constant human presence in the training frames, a *Data augmentation*'s module[4] has been introduced. This new module introduces variability in the training set by patching the blob of pixels of the object, segmented by the *Automatic GT extractor* from a portion of the training set, on other real pictures depicting novel possible different backgrounds.

Note that, this approach mitigate the generalization problem in the setting considered for this application, taking inspiration by state-of-the-art approaches like Dwibedi et al. (2017);

[5]Implemented by Vadim Tikhanoff

Gupta et al. (2016), however it does not use the available (not annotated) data during the functioning of the robot, nor it allows for a more precise adaptation to the environment around the robot. For this reason, I am planning to also integrate the weakly supervised strategy described in Chapter 8 in the pipeline.

Part V

Conclusions

Chapter 11

Contributions

The purpose and outcome of this Ph.D thesis is threefold. Firstly, from a methodological point of view, new strategies have been proposed to maintain performance of state-of-the-art approaches for object detection within a robotic scenario. Secondly, the proposed strategies have been deployed in robotic applications proving their effectiveness in a real world use-case. Thirdly, an outcome of this project has been to contribute to the iCubWorld project with new, manually annotated sequences of images that can be used as benchmarks. The description and the analysis of the contributions have been collected in a list of papers and publications, publicly available.

11.1 Methodological contributions

From a methodological point of view, the main endeavor of this project have been concentrated in devising algorithmic solutions which maintain performance of state-of-the-art methods for object detection, within the constraints imposed by robotics. Specifically, with respect to the learning pipeline represented in Fig. 1.2, the main contributions focused on improving the first two stages of annotated data collection and training.

As a first step, I first demonstrated that a human robot interaction procedure Pasquale et al. (2016a,b) can be used to acquire automatically annotated images, allowing to train detection algorithms (e.g. Faster R-CNN Ren et al. (2015)). An experimental analysis, carried out using the *iCWT* dataset, showed that this method allows to achieve good accuracy within the human robot interaction scenario used for learning, while being reasonably slightly less accurate, but still effective, in different settings (see Chapter 6).

Subsequently, regarding the learning phase, an on-line learning pipeline has been proposed that speeds-up the training (see Chapter 7). It exploits generality of features provided

by deep CNNs being composed of two main modules: (i) a per-region feature extractor, trained once off-line on a certain task (the *Feature Extraction module*), and (ii) a region classifier which can be trained quickly and on-line on a different task, the target task (the *Detection module*). Specifically, for this latter module, FALKON has been adopted, a recently proposed kernel-based method for large-scale datasets. The region classifier additionally accounts for the foreground-background unbalance by (i) leveraging on the stochastic sampling of the kernel centers performed by FALKON and by (ii) using an approximated version of the Hard Negative Mining procedure, to efficiently re-balance the training set specifically designed for this system.

I evaluated the proposed method through an extensive empirical analysis, demonstrating the benefits of its adoption in real world robotic applications. The validation has been carried out on two different benchmarks, considering the Pascal VOC and ICWT datasets, to prove its effectiveness in both a general-purpose computer vision scenario and in a robotic setup. Then, a detailed evaluation of the main components of the method has been performed, with the aims of (i) trying different variants for learning the *Feature Extraction module*, (ii) motivating the choice of FALKON as classification method and (iii) showing the influence of the three main hyper-parameters of the approach. Finally, the method has been put to the test by considering typical real robotic scenarios. This method proved to be effective and I believe that it can be used as baseline for future work on detection systems in robotics, since it allows to be quickly adapted while preserving state-of-the-art accuracy. In fact, as a part of an external project, I collaborated in demonstrating that this pipeline could be extended and integrated with an instance segmentation method, with promising preliminary results.

Finally, I integrated the on-line detection system with the SSM method Wang et al. (2018) (see Chapter 8). This latter combines a self-supervision process to generate pseudo ground-truth for the most confident predictions, with an active learning strategy to select the hardest images to be asked for annotation. The system resulting from this integration has been tested on the iCWT dataset and on a table-top image sequences acquired with the R1 robot, showing promising performance.

11.2 Technological contributions

The pipeline resulting from integrating the automatic data collection and the on-line learning algorithm for object detection allows to naturally train a humanoid robot to detect novel objects in just a few seconds. As a contribution of this Ph.D. project, this pipeline has been deployed in an application on the R1 and the iCub humanoid robots. It allows for a natural human robot interaction, where the user shows the robot the novel object to learn.

The robot tracks it, gathering images and corresponding ground-truth and, lastly, the object detection model is updated with the proposed on-line learning method. At inference time, the application shows the output of the algorithm, overlapping the predicted detections to the images coming from the cameras of the robot. This latter can exploit detection information to reply to the user's questions. This application demonstrates the effectiveness of the proposed algorithmic results in a real world application. Moreover, it represents a baseline for more specific use-case scenarios. For instance, the application has been extended to be more robust to scale variations for the purpose of being used to detect objects while R1 navigates in an environment, thus being not necessarily near the objects.

While videos of different stages of the deployment of the application have been already released[1], the code will be made publicly available on the Robotology repository[2].

11.3 Data contributions

For the experimental analysis carried out during this Ph.D. project, the predominantly used datasets have been the Pascal VOC benchmark and the *iCWT*. Note that, *iCWT* has been originally released as a dataset for object recognition. Object detection annotations have been made available as a further contribution of this thesis. Moreover, in order to further analyze and validate the proposed methods in different real world settings, I collected and manually annotated new image sequences, depicting subsets of the objects in the *iCWT* dataset, randomly placed on floors, tables and shelves. The sequences have been acquired using the iCub and the R1 humanoid robots. For the manual annotation, I adopted the *labelImg*[3] tool, establishing a policy such that an object must be annotated if at least a 50-25% of its total shape is visible (i.e. not cut out from the image or occluded).
More specifically, the contribution consists in:

- Release of object detection annotations in ImageNet-like format of the ICWT.
- Manual annotation of a subset of 150 images from the first day of acquisition of the MIX sequence of 40 objects of the iCWT. These images have been used in the experimental analysis of the on-line object detection system, reported in Chapter 7.
- Three new sequences used for the experimental analysis reported in Chapter 6 respectively showing: (i) 14 objects lying on a an empty floor (*floor* sequence), (ii) 11 objects

[1] https://www.dropbox.com/s/njykynrdnrbovqz/ElisaDetectionFinal.mp4?dl=0 and https://www.youtube.com/watch?v=eT-2v6-xoSs
[2] https://github.com/robotology
[3] https://github.com/tzutalin/labelImg

placed on a table with other distractors (*table* sequence) and (iii) 10 objects placed on two shelves, containing also distractors and shadows (*shelf* sequence).

- 2 sets of sequences depicting 21 different objects from the iCWT. In each set, I considered a different table cloth: (i) pink/white pois (POIS set) and (ii) white (WHITE set). For each set, I split the 21 objects in groups and I acquired different sequences of each group. Please, refer to Sec. 8.2 for further details about the data acquisition procedure.

These image sequences have been made available on the iCubWorld project website[4] and can be used jointly with the *iCWT* sequences where the objects are handheld and automatically annotated to test generalization capabilities of detection systems in real world robotic settings.

11.4 Publications and dissemination

Part of the work carried out during this project has been collected in the following list of publications and dissemination.

The results of applying an automatic data collection procedure to the object detection task has been presented in:

- "*Interactive Data Collection for Deep Learning Object Detectors on Humanoid Robots*"
  E. Maiettini, G. Pasquale, L. Rosasco, and L. Natale, in 2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids), Nov 2017.

- "*Natural, Interactive Training of Service Robots to Detect Novel Objects.*"
  talk presented at the NVIDIA's GPU Technology Conference, 2017.

The on-line learning pipeline for object detection and the detailed experimental analysis have been presented in:

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

- "*Speeding-up Object Detection Training for Robotics with FALKON*"
  E. Maiettini, G. Pasquale, L. Rosasco, and L. Natale, in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Oct 2018.

- "*Object Detection Training: An Online Learning Pipeline for Humanoid Robots.*"
  talk presented at the NVIDIA's GPU Technology Conference, 2018,

- "*On-line Object Detection: a Robotics Challenge*"
  E. Maiettini, G. Pasquale, L. Rosasco, and L. Natale, in the Autonomous Robots journal, Nov 2019.

Finally, the integration of the on-line learning pipeline with the weakly supervised strategy has been presented in:

- "*A Weakly Supervised Strategy for Learning Object Detection on a Humanoid Robot*"
  E. Maiettini, G. Pasquale, V. Tikhanoff, L. Rosasco, and L. Natale, in 2019 IEEE-RAS 19th International Conference on Humanoid Robotics (Humanoids), Oct 2019.

Chapter 12

Discussion

The common thread of the works carried out during this Ph.D project has been that of finding solutions bridging robotics requirements and opportunities with computer vision state-of-the-art performance, considering the task of object detection. Specifically, main efforts have been put in realizing methods and applications that demonstrated to be effective and that can be either used in real world applications or considered as baselines for future research. In the following paragraphs, I discuss on the particular findings of this Ph.D while providing examples of possible future research directions to pursue.

Improving the robotic application

The pipeline resulting from the integration of the automatic data collection and the on-line object detection algorithm has been widely tested in a robotic application with good results. However, its modular structure allows an easy integration of extensions. For instance, the data acquisition procedure can be extended in order to allow the user for a wider range of movements during the object demonstration. A step toward this direction has been already taken, during this Ph.D project, integrating a human body parts extractor in the pipeline so to let the robot focus attention on the user's hand, expanding teacher's range of motion. However, other improvements can be easily integrated like e.g., the possibility of acquiring data for more than one object at a time or adding new interaction modalities Azagra et al. (2017).

Deploying the weakly supervised strategy on R1

The empirical analysis reported in Chapter 8 demonstrated the effectiveness of the integration of the on-line learning detection method with the SSM Wang et al. (2018). This motivates the

interest in devising an interactive application to to be deployed on a robotic platform. In the carried out off-line experiments the action of asking for human supervision was simulated with a process that reads annotations from a database. While designing the robotic application, one should devise an interaction so that for example, a user can provide annotations through pointing at objects. Moreover, spatial and temporal cues can be exploited to propagate labels and further reduce the requested human supervision.

Active exploration

The integration between the on-line learning detection method and the SSM Wang et al. (2018) proved to be effective on the considered datasets (see Chapter 8). Specifically, the pipeline has been tested, as a proof of concept, on a table-top dataset acquired with R1 executing a pre-scripted exploration procedure. I believe that the results can significantly improve with the integration of an active exploration to allow the robot to push, pick up and rotate objects acquiring new views, linking the movements with the learning process, exploiting a *Next best view* strategy Connolly (1985).

Coupling the active learning with the Minibootstrap

From an algorithmic point of view, the results obtained by integrating the on-line detection method with the SSM are promising, offering an interesting starting point for improvements. As a matter of fact, the self-supervision, the active learning and the *Minibootstrap* all iterate on the dataset in order to extract an effective training set. A more efficient and robust sample selection process could be obtained by integrating them in a tighter way.

Exploiting additional sources of labeling

During this Ph.D project the main focus has been in addressing object detection with the aim of distinguishing different instances of the same categories (namely *object identification*). This is motivated by the fact that the robot, in many practical cases, might need to identify a specific object instance. For this reason, available datasets were not enough, and, as a contribution of this thesis, a human robot interaction has been used for the automatic data annotation of images. Nevertheless, I believe that strategies that can exploit the abundant amount of data, available in the web, can be considered as integration within the proposed system. Indeed, the literature shows evidence Mancini et al. (2019); Molinari et al. (2019) of the effectiveness of the approach for the object recognition task. Moreover, in Prest et al. (2012), a method to exploit spatio-temporal coherence in web videos has been proposed to

learn, in a weakly supervised fashion, an object detector. I believe that it might be of interest trying to use similar strategies to integrate web data exploitation in the proposed application for, e.g., improving generalization capabilities during feature extraction.

Further exploiting 3D information

For the object detection learning pipeline proposed in this thesis, the 3D information, coming from the sensors of the robot, has been used to acquire an automatically annotated dataset to learn novel tasks. However, the 3D information can be further exploited by more tightly involving it in the learning algorithm. Indeed, the literature Porzi et al. (2016); Schwarz et al. (2015) shows evidences that it can be used to obtain more powerful CNN-based feature representations.

Going further in collecting automatic ground-truth

One of the contribution of this thesis proved that the automatic data collection procedure proposed in Pasquale et al. (2016b) for acquiring datasets for object classification can be used for object detection as well. However, while developing this thesis project, the need of acquiring new data, with multiple objects depicted in different scenes, arose multiple times (see e.g. the new sequences of images acquired for the experimental analysis reported in Chapter 6 and 8). Even if we consider a simple table top acquisition scenario, where objects are well separated, a considerable amount of human manual labeling is required. I believe that devising a procedure to acquire automatic ground-truth from the robot point of view in different scenarios could lead to an important speed-up in the experimental process.

Going towards a continuous learning scenario

In the experimental analysis carried out in Sec. 8.2, the objective was to assess the ability of the proposed integration to efficiently refine the robotic vision system to generalize to a new setting (table top), different from the one used for the original training (handheld objects). To this aim, performance has been measured only on the task at hand, i.e., only on the table top scenario. However, it is of interest for the design of a robust detection method to also preserve the performance on the original task (handheld), while adapting to the newer one. This aspect is considered in two main machine learning frameworks: (i) the *continual learning* framework Ring (1994), where the assumption is that the data distribution and the learning objective might change in time and (ii) the *lifelong learning*, which focuses on developing versatile systems that accumulate and refine their knowledge over time Schmidt and Fox (2020).

Going towards on-line learning of object segmentation

In the experimental analysis carried out in Sec. 7.5, promising results have been shown regarding the integration of an object segmentation method with the on-line detection pipeline proposed in this project. This can motivate a future work to further benchmark this integration to understand its effectiveness, eventually, going towards the fast segmentation mask learning and the integration of the resulting method in the on-line detection application.

Reinforcement learning for object detection

The aim of using a weakly supervised strategy Wang et al. (2018) within the proposed object detection system Maiettini et al. (2018) was towards the direction of avoiding the brute force approach of labeling manually all the images in a dataset, but choosing only the ones more "meaningful", in an efficient way. This data selection is based on the active learning and self supervised learning principles and specifically on the definition of a policy to decide, in an iterative process, whether to ask for the label of an image or automatically annotate it by considering the current state of the model. The reinforcement learning framework Kaelbling et al. (1996) lies not far from these concepts. Latest works König et al. (2020); Mathe et al. (2016) in this area, indeed, show promising results for object detection and I think it is worth considering this approaches as possible integration to the current system.

Exploiting data augmentation

Although the experiments reported in Chapter 7 show high accuracy, detection in highly cluttered scene is still a challenging task, especially when training and testing conditions are different. This is mainly true in the considered scenario, because training sequences may contain insufficient variability (in terms of background, illumination or partial occlusions). Possible directions for addressing this problem and improving the system robustness is to consider some data augmentation techniques, like the creation of synthetic datasets Georgakis et al. (2017) or relying on simulation data Tobin et al. (2017), for both feature and classification learning. The qualitative results obtained in Chapter 10 in integrating these techniques, motivate the interest in carrying out a more systematic analysis of these approaches in the considered robotic setting.

Multi task learning

During this Ph.D. project, no relations between the objects have been considered in the learning process, however they can represent an important source of additional information

to exploit. Indeed, it is widely acknowledged Ciliberto et al. (2015) that leveraging on tasks similarities can reduce the requirement on supervised data. This idea is at the basis of so called *Multi-task learning*, where the joint solution of different tasks exploits the inherent structure of the data to improve learning accuracy Ciliberto et al. (2015).

One-shot learning

In Sec. 7.4.2, a noticeable fast training time has been achieved by dramatically decreasing the number of samples used for training the classifier. In this perspective, *One-shot learning* techniques Fei-Fei et al. (2006); Kaiser et al. (2017) can be considered to bring the speed/accuracy trade-off to the extreme.

Bibliography

Agarwal, S., du Terrail, J. O., and Jurie, F. (2018). Recent advances in object detection in the age of deep convolutional neural networks. *CoRR*, abs/1809.03193.

Agrawal, P., Carreira, J., and Malik, J. (2015). Learning to see by moving. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 37–45.

Aloimonos, J., Weiss, I., and Bandyopadhyay, A. (1988). Active vision. *International journal of computer vision*, 1(4):333–356.

Angluin, D. and Laird, P. (1988). Learning from noisy examples. *Machine Learning*, 2(4):343–370.

Atlas, L. E., Cohn, D. A., and Ladner, R. E. (1990). Training connectionist networks with queries and selective sampling. In *Advances in neural information processing systems*, pages 566–573.

Azagra, P., Golemo, F., Mollard, Y., Lopes, M., Civera, J., and Murillo, A. C. (2017). A multimodal dataset for object model learning from natural human-robot interaction. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6134–6141. IEEE.

Bajcsy, R. (1988). Active perception. *Proceedings of the IEEE*, 76(8):966–1005.

Bajcsy, R., Aloimonos, Y., and Tsotsos, J. K. (2018). Revisiting active perception. *Autonomous Robots*, 42(2):177–196.

Baum, E. B. and Lang, K. (1992). Query learning can work poorly when a human oracle is used. In *International joint conference on neural networks*, volume 8, page 8.

Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM.

Blum, A. and Chawla, S. (2001). Learning from labeled and unlabeled data using graph mincuts.

Blum, A., Kalai, A., and Wasserman, H. (2003). Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM (JACM)*, 50(4):506–519.

Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM.

Brabham, D. C. (2008). Crowdsourcing as a model for problem solving: An introduction and cases. *Convergence*, 14(1):75–90.

Brodley, C. E. and Friedl, M. A. (1999). Identifying mislabeled training data. *Journal of artificial intelligence research*, 11:131–167.

Browatzki, B., Tikhanoff, V., Metta, G., Bülthoff, H. H., and Wallraven, C. (2012). Active object recognition on a humanoid robot. In *2012 IEEE International Conference on Robotics and Automation*, pages 2021–2028.

Brox, T. and Malik, J. (2010). Object segmentation by long term analysis of point trajectories. In *European conference on computer vision*, pages 282–295. Springer.

Bunz, E. (2019). On-the-fly learning of object segmentation with deep neural networks.

Burbidge, R., Rowland, J. J., and King, R. D. (2007). Active learning for regression based on query by committee. In Yin, H., Tino, P., Corchado, E., Byrne, W., and Yao, X., editors, *Intelligent Data Engineering and Automated Learning - IDEAL 2007*, pages 209–218, Berlin, Heidelberg. Springer Berlin Heidelberg.

Calli, B., Singh, A., Bruce, J., Walsman, A., Konolige, K., Srinivasa, S., Abbeel, P., and Dollar, A. M. (2017). Yale-cmu-berkeley dataset for robotic manipulation research. *The International Journal of Robotics Research*, 36(3):261–268.

Calli, B., Walsman, A., Singh, A., Srinivasa, S., Abbeel, P., and Dollar, A. M. (2015). Benchmarking in manipulation research: The ycb object and model set and benchmarking protocols. *IEEE Robotics and Automation Magazine*.

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)*.

Cao, Z., Simon, T., Wei, S.-E., and Sheikh, Y. (2017). Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7291–7299.

Chapelle, O., Scholkopf, B., and Zien, A. (2009). Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542.

Chapelle, O., Schölkopf, B., Zien, A., et al. (2006). Semi-supervised learning, vol. 2. *Cambridge: MIT Press. Cortes, C., & Mohri, M.(2014). Domain adaptation and sample bias correction theory and algorithm for regression. Theoretical Computer Science*, 519:103126.

Chapelle, O. and Zien, A. (2005). Semi-supervised classification by low density separation. In *AISTATS*, volume 2005, pages 57–64. Citeseer.

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*.

Chen, S., Li, Y., and Kwok, N. M. (2011). Active vision in robotic systems: A survey of recent developments. *International Journal of Robotics Research*, 30(11):1343–1377.

Chen, W., Wang, H., Li, Y., Su, H., Wang, Z., Tu, C., Lischinski, D., Cohen-Or, D., and Chen, B. (2016). Synthesizing training images for boosting human 3d pose estimation. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 479–488. IEEE.

Chen, Y., Bi, J., and Wang, J. Z. (2006). Miles: Multiple-instance learning via embedded instance selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):1931–1947.

Ciliberto, C., Mroueh, Y., Poggio, T., and Rosasco, L. (2015). Convex learning of multiple tasks and their structure. In *International Conference on Machine Learning*, pages 1548–1557.

Cohn, D., Atlas, L., and Ladner, R. (1994). Improving generalization with active learning. *Machine learning*, 15(2):201–221.

Cohn, D. A., Ghahramani, Z., and Jordan, M. I. (1996). Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145.

Conkey, A. and Hermans, T. (2019). Active learning of probabilistic movement primitives. *2019 IEEE-RAS 19th International Conference on Humanoid Robotics (Humanoids)*.

Connolly, C. (1985). The determination of next best views. In *Proceedings. 1985 IEEE international conference on robotics and automation*, volume 2, pages 432–435. IEEE.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Cour, T., Sapp, B., and Taskar, B. (2011). Learning from partial labels. *Journal of Machine Learning Research*, 12:1501–1536. cited By 121.

Cozman, F. G., Cohen, I., and Cirelo, M. (2002). Unlabeled data can degrade classification performance of generative classifiers. In *Flairs conference*, pages 327–331.

Dai, J., He, K., and Sun, J. (2015). Convolutional feature masking for joint object and stuff segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3992–4000.

Dai, J., He, K., and Sun, J. (2016a). Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3150–3158.

Dai, j., Li, Y., He, K., and Sun, J. (2016b). R-fcn: Object detection via region-based fully convolutional networks. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 379–387. Curran Associates, Inc.

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.

Damen, D., Doughty, H., Farinella, G. M., Fidler, S., Furnari, A., Kazakos, E., Moltisanti, D., Munro, J., Perrett, T., Price, W., and Wray, M. (2018). Scaling egocentric vision: The epic-kitchens dataset. In *European Conference on Computer Vision (ECCV)*.

Delakis, M. and Garcia, C. (2008). text detection with convolutional neural networks. In *VISAPP (2)*, pages 290–294.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

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*.

Dietterich, T. G., Lathrop, R. H., and Lozano-Pérez, T. (1997). Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*, 89(1-2):31–71.

Dwibedi, D., Misra, I., and Hebert, M. (2017). Cut, paste and learn: Surprisingly easy synthesis for instance detection. In *The IEEE International Conference on Computer Vision (ICCV)*.

Eggert, C., Winschel, A., Zecha, D., and Lienhart, R. (2016). Saliency-guided selective magnification for company logo detection. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 651–656. IEEE.

Eggert, C., Zecha, D., Brehm, S., and Lienhart, R. (2017). Improving small object proposals for company logo detection. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, pages 167–174.

Enzweiler, M. and Gavrila, D. M. (2011). A multilevel mixture-of-experts framework for pedestrian classification. *IEEE Transactions on Image Processing*, 20(10):2967–2979.

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.

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.

Fei-Fei, L., Fergus, R., and Perona, P. (2006). One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611.

Felzenszwalb, P., McAllester, D., and Ramanan, D. (2008). A discriminatively trained, multiscale, deformable part model. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

Felzenszwalb, P. F., Girshick, R. B., and McAllester, D. (2010a). Cascade object detection with deformable part models. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2241–2248. IEEE.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2010b). Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645.

Foulds, J. and Frank, E. (2010). A review of multi-instance learning assumptions. *The Knowledge Engineering Review*, 25(1):1–25.

Freund, Y., Schapire, R., and Abe, N. (1999). A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119 – 139.

Gao, W., Wang, L., Zhou, Z.-H., et al. (2016). Risk minimization in the presence of label noise. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Garcia, C. and Delakis, M. (2002). A neural architecture for fast and robust face detection. In *Object recognition supported by user interaction for service robots*, volume 2, pages 44–47. IEEE.

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Geiger, A., Roser, M., and Urtasun, R. (2010). Efficient large-scale stereo matching. In *Asian Conference on Computer Vision (ACCV)*.

Georgakis, G., Mousavian, A., Berg, A. C., and Kosecka, J. (2017). Synthesizing training data for object detection in indoor scenes. *CoRR*, abs/1702.07836.

Girshick, R. (2015). Fast R-CNN. In *Proceedings of the International Conference on Computer Vision (ICCV)*.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Girshick, R. B. (2012). *From Rigid Templates to Grammars: Object Detection with Structured Models*. PhD thesis, Chicago, IL, USA. AAI3513455.

Girshick, R. B., Felzenszwalb, P. F., and McAllester, D. (2011). Object detection with grammar models. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, pages 442–450, USA. Curran Associates Inc.

Glasmachers, T. (2017). Limits of end-to-end learning. *arXiv preprint arXiv:1704.08305*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Grotz, M., Sippel, D., and Asfour, T. (2019). Active vision for extraction of physically plausible support relations.

Gupta, A., Vedaldi, A., and Zisserman, A. (2016). Synthetic data for text localisation in natural images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2315–2324.

Harada, K., Wan, W., Tsuji, T., Kikuchi, K., Nagata, K., and Onda, H. (2016). Iterative visual recognition for learning based randomized bin-picking. In *International Symposium on Experimental Robotics*, pages 646–655. Springer.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. B. (2017). Mask r-cnn. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*.

Hearst, M. A., Dumais, S. T., Osuna, E., Platt, J., and Scholkopf, B. (1998). Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28.

Heitz, G. and Koller, D. (2008). Learning spatial context: Using stuff to find things. In *European conference on computer vision*, pages 30–43. Springer.

Hernández-González, J., Inza, I., and Lozano, J. A. (2016). Weak supervision and other non-standard classification problems: a taxonomy. *Pattern Recognition Letters*, 69:49–55.

Hoi, S. C., Jin, R., and Lyu, M. R. (2006). Large-scale text categorization by batch mode active learning. In *Proceedings of the 15th international conference on World Wide Web*, pages 633–642. ACM.

Holz, D., Topalidou-Kyniazopoulou, A., Stückler, J., and Behnke, S. (2015). Real-time object detection, localization and verification for fast robotic depalletizing. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1459–1466. IEEE.

Houben, S., Stallkamp, J., Salmen, J., Schlipsing, M., and Igel, C. (2013). Detection of traffic signs in real-world images: The german traffic sign detection benchmark. In *The 2013 international joint conference on neural networks (IJCNN)*, pages 1–8. IEEE.

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., and Murphy, K. (2016). Speed/accuracy trade-offs for modern convolutional object detectors. *CoRR*, abs/1611.10012.

Jain, V. and Learned-Miller, E. (2010). Fddb: A benchmark for face detection in unconstrained settings. Technical report, UMass Amherst technical report.

Jayaraman, D. and Grauman, K. (2015). Learning image representations tied to ego-motion. In *The IEEE International Conference on Computer Vision (ICCV)*.

Jonschkowski, R., Eppner, C., Höfer, S., Martín-Martín, R., and Brock, O. (2016). Probabilistic multi-class segmentation for the amazon picking challenge. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–7. IEEE.

Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285.

Kahn, G., Sujan, P., Patil, S., Bopardikar, S., Ryde, J., Goldberg, K., and Abbeel, P. (2015). Active exploration using trajectory optimization for robotic grasping in the presence of occlusions. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4783–4790. IEEE.

Kaipa, K. N., Kankanhalli-Nagendra, A. S., Kumbla, N. B., Shriyam, S., Thevendria-Karthic, S. S., Marvel, J. A., and Gupta, S. K. (2016). Addressing perception uncertainty induced failure modes in robotic bin-picking. *Robotics and Computer-Integrated Manufacturing*, 42:17–38.

Kaiser, L., Nachum, O., Roy, A., and Bengio, S. (2017). Learning to remember rare events. *CoRR*, abs/1703.03129.

Karsch, K., Hedau, V., Forsyth, D., and Hoiem, D. (2011). Rendering synthetic objects into legacy photographs. In *ACM Transactions on Graphics (TOG)*, volume 30, page 157. ACM.

King, R. D., Rowland, J., Oliver, S. G., Young, M., Aubrey, W., Byrne, E., Liakata, M., Markham, M., Pir, P., Soldatova, L. N., et al. (2009). The automation of science. *Science*, 324(5923):85–89.

King, R. D., Whelan, K. E., Jones, F. M., Reiser, P. G., Bryant, C. H., Muggleton, S. H., Kell, D. B., and Oliver, S. G. (2004). Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427(6971):247.

König, J., Malberg, S., Martens, M., Niehaus, S., Krohn-Grimberghe, A., and Ramaswamy, A. (2020). Multi-stage reinforcement learning for object detection. In Arai, K. and Kapoor, S., editors, *Advances in Computer Vision*, pages 178–191, Cham. Springer International Publishing.

Kovashka, A., Russakovsky, O., Fei-Fei, L., and Grauman, K. (2016). Crowdsourcing in computer vision. *Foundations and Trends® in Computer Graphics and Vision*, 10(3):177–243.

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

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012b). 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.

Kumar, M. P., Packer, B., and Koller, D. (2010). Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, pages 1189–1197.

Kuncheva, L. I. (2010). Full-class set classification using the hungarian algorithm. *International Journal of Machine Learning and Cybernetics*, 1(1):53–61.

Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Kolesnikov, A., et al. (2020). The open images dataset v4. *International Journal of Computer Vision*, pages 1–26.

Kyu Rhee, P., Erdenee, E., Shin, D. K., Ahmed, M., and Jin, S. (2017). Active and semi-supervised learning for object detection with imperfect data. *Cognitive Systems Research*, 45.

Lai, K., Bo, L., and Fox, D. (2014). Unsupervised feature learning for 3d scene labeling. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3050–3057. IEEE.

Lakshminarayanan, B., Roy, D. M., and Teh, Y. W. (2014). Mondrian forests: Efficient online random forests. In *Advances in neural information processing systems*, pages 3140–3148.

Lampert, C. H., Blaschko, M. B., and Hofmann, T. (2008). Beyond sliding windows: Object localization by efficient subwindow search. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE.

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.

Lewis, D. D. and Gale, W. A. (1994). A sequential algorithm for training text classifiers. In Croft, B. W. and van Rijsbergen, C. J., editors, *SIGIR '94*, pages 3–12, London. Springer London.

Li, Y., Qi, H., Dai, J., Ji, X., and Wei, Y. (2017). Fully convolutional instance-aware semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2359–2367.

Li, Y.-F., Tsang, I. W., Kwok, J. T., and Zhou, Z.-H. (2013). Convex and scalable weakly labeled svms. *The Journal of Machine Learning Research*, 14(1):2151–2188.

Li, Y.-F. and Zhou, Z.-H. (2014). Towards making unlabeled data never hurt. *IEEE transactions on pattern analysis and machine intelligence*, 37(1):175–188.

Lin, L., Wang, K., Meng, D., Zuo, W., and Zhang, L. (2017a). Active self-paced learning for cost-effective and progressive face identification. *IEEE transactions on pattern analysis and machine intelligence*, 40(1):7–19.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017b). Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125.

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. In *European Conference on Computer Vision (ECCV)*, Zürich. Oral.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., and Reed, S. E. (2015). Ssd: Single shot multibox detector. *CoRR*, abs/1512.02325.

Liu, Y. (2004). Active learning with support vector machine applied to gene expression data for cancer classification. *Journal of chemical information and computer sciences*, 44(6):1936–1941.

Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440.

Maiettini, E., Pasquale, G., Rosasco, L., and Natale, L. (2017). Interactive data collection for deep learning object detectors on humanoid robots. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 862–868.

Maiettini, E., Pasquale, G., Rosasco, L., and Natale, L. (2018). Speeding-up object detection training for robotics with falkon. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Maiettini, E., Pasquale, G., Rosasco, L., and Natale, L. (2019). On-line object detection: a robotics challenge. *Autonomous Robots*.

Mancini, M., Bulò, S. R., Caputo, B., and Ricci, E. (2018). Best sources forward: domain generalization through source-specific nets. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 1353–1357. IEEE.

Mancini, M., Karaoguz, H., Ricci, E., Jensfelt, P., and Caputo, B. (2019). Knowledge is never enough: Towards web aided deep open world recognition. *arXiv preprint arXiv:1906.01258*.

Mathe, S., Pirinen, A., and Sminchisescu, C. (2016). Reinforcement learning for visual object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2894–2902.

McClelland, J. L., Rumelhart, D. E., and Hinton, G. E. (1986). Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter The Appeal of Parallel Distributed Processing, pages 3–44. MIT Press, Cambridge, MA, USA.

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.

Metta, G., Fitzpatrick, P., and Natale, L. (2006). Yarp: Yet another robot platform. *International Journal of Advanced Robotics Systems, special issue on Software Development and Integration in Robotics*, 3(1).

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.

Miller, D. J. and Uyar, H. S. (1997). A mixture of experts classifier with learning based on both labelled and unlabelled data. In *Advances in neural information processing systems*, pages 571–577.

Mitchell, T. M. (1997). Machine learning.

Mohan, A., Papageorgiou, C., and Poggio, T. (2001). Example-based object detection in images by components. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(4):349–361.

Molinari, D., Pasquale, G., Natale, L., and Caputo, B. (2019). Automatic creation of large scale object databases from web resources: A case study in robot vision. In Ricci, E., Rota Bulò, S., Snoek, C., Lanz, O., Messelodi, S., and Sebe, N., editors, *Image Analysis and Processing – ICIAP 2019*, pages 488–498, Cham. Springer International Publishing.

Movshovitz-Attias, Y., Kanade, T., and Sheikh, Y. (2016). How useful is photo-realistic rendering for visual learning? In *European Conference on Computer Vision*, pages 202–217. Springer.

Muhlenbach, F., Lallich, S., and Zighed, D. A. (2004). Identifying and handling mislabelled instances. *Journal of Intelligent Information Systems*, 22(1):89–109.

Narr, A., Triebel, R., and Cremers, D. (2016). Stream-based active learning for efficient and adaptive classification of 3d objects. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 227–233.

Nguyen, K., Fookes, C., Ross, A., and Sridharan, S. (2017). Iris recognition with off-the-shelf cnn features: A deep learning perspective. *IEEE Access*, PP:1–1.

Nigam, K., McCallum, A. K., Thrun, S., and Mitchell, T. (2000). Text classification from labeled and unlabeled documents using em. *Machine learning*, 39(2-3):103–134.

Nowlan, S. J. and Platt, J. C. (1995). A convolutional neural network hand tracker. *Advances in neural information processing systems*, pages 901–908.

Oksuz, K., Cam, B. C., Kalkan, S., and Akbas, E. (2019). Imbalance problems in object detection: A review. *arXiv preprint arXiv:1909.00169*.

Osadchy, M., Cun, Y. L., and Miller, M. L. (2007). Synergistic face detection and pose estimation with energy-based models. *Journal of Machine Learning Research*, 8(May):1197–1215.

Pang, J., Chen, K., Shi, J., Feng, H., Ouyang, W., and Lin, D. (2019). Libra r-cnn: Towards balanced learning for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 821–830.

Papageorgiou, C. and Poggio, T. (2000). A trainable system for object detection. *International Journal of Computer Vision*, 38(1):15–33.

Papageorgiou, C. P., Oren, M., and Poggio, T. (1998). A general framework for object detection. In *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, pages 555–562.

Parmiggiani, A., Fiorio, L., Scalzo, A., Sureshbabu, A. V., Randazzo, M., Maggiali, M., Pattacini, U., Lehmann, H., Tikhanoff, V., Domenichelli, D., Cardellino, A., Congiu, P., Pagnin, A., Cingolani, R., Natale, L., and Metta, G. (2017). The design and validation of the r1 personal humanoid. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 674–680.

Pasquale, G., Ciliberto, C., Odone, F., Rosasco, L., and Natale, L. (2019). Are we done with object recognition? the icub robot's perspective. *Robotics and Autonomous Systems*, 112:260 – 281.

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.

Pasquale, G., Mar, T., Ciliberto, C., Rosasco, L., 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.

Pathak, D., Girshick, R., Dollár, P., Darrell, T., and Hariharan, B. (2017). Learning features by watching objects move. In *CVPR*.

Peng, X., Sun, B., Ali, K., and Saenko, K. (2015). Learning deep object detectors from 3d models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1278–1286.

Pinheiro, P. O., Collobert, R., and Dollár, P. (2015). Learning to segment object candidates. In *Advances in Neural Information Processing Systems*, pages 1990–1998.

Pinheiro, P. O., Lin, T.-Y., Collobert, R., and Dollár, P. (2016). Learning to refine object segments. In *ECCV*.

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.

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.

Porikli, F. (2005). Integral histogram: A fast way to extract histograms in cartesian spaces. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 829–836. IEEE.

Porzi, L., Bulo, S. R., Penate-Sanchez, A., Ricci, E., and Moreno-Noguer, F. (2016). Learning depth-aware deep representations for robotic perception. *IEEE Robotics and Automation Letters*, 2(2):468–475.

Potthast, C. and Sukhatme, G. S. (2014). A probabilistic framework for next best view estimation in a cluttered environment. *Journal of Visual Communication and Image Representation*, 25(1):148–164.

Prest, A., Leistner, C., Civera, J., Schmid, C., and Ferrari, V. (2012). Learning object class detectors from weakly annotated video. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3282–3289. IEEE.

Pretto, A., Tonello, S., and Menegatti, E. (2013). Flexible 3d localization of planar objects for industrial bin-picking with monocamera vision system. In *2013 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 168–175. IEEE.

Ranjan, R., Patel, V. M., and Chellappa, R. (2017). Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(1):121–135.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Redmon, J. and Farhadi, A. (2016). Yolo9000: Better, faster, stronger. *arXiv preprint arXiv:1612.08242*.

Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *CoRR*, abs/1804.02767.

Reed, S., Lee, H., Anguelov, D., Szegedy, C., Erhan, D., and Rabinovich, A. (2014). Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In *Neural Information Processing Systems (NIPS)*.

Ring, M. B. (1994). *Continual learning in reinforcement environments*. PhD thesis, University of Texas at Austin Austin, Texas 78712.

Rosasco, L. and Poggio, T. (2015). Machine learning: a regularization approach, mit-9.520 lectures notes.

Rudi, A., Carratino, L., and Rosasco, L. (2017). Falkon: An optimal large scale kernel method. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 3888–3898. Curran Associates, Inc.

Rumelhart, D. E., McClelland, J. L., and PDP Research Group, C., editors (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. MIT Press, Cambridge, MA, USA.

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.

Saad, Y. (2003). *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edition.

Sabzmeydani, P. and Mori, G. (2007). Detecting pedestrians by learning shapelet features. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229.

Schmidt, T. and Fox, D. (2020). Self-directed lifelong learning for robot vision. In *Robotics Research*, pages 109–114. Springer.

Schölkopf, B., Herbrich, R., and Smola, A. J. (2001). A generalized representer theorem. In *International conference on computational learning theory*, pages 416–426. Springer.

Schwarz, M., Milan, A., Lenz, C., Munoz, A., Periyasamy, A. S., Schreiber, M., Schüller, S., and Behnke, S. (2017). Nimbro picking: Versatile part handling for warehouse automation. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3032–3039. IEEE.

Schwarz, M., Milan, A., Periyasamy, A. S., and Behnke, S. (2018). Rgb-d object detection and semantic segmentation for autonomous manipulation in clutter. *The International Journal of Robotics Research*, 37(4-5):437–451.

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. IEEE.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and Lecun, Y. (2014). *Overfeat: Integrated recognition, localization and detection using convolutional networks*.

Sermanet, P., Kavukcuoglu, K., Chintala, S., and LeCun, Y. (2013). Pedestrian detection with unsupervised multi-stage feature learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3626–3633.

Settles, B. (2009). Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences.

Settles, B. (2012). Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*.

Settles, B., Craven, M., and Ray, S. (2008). Multiple-instance active learning. In *Advances in neural information processing systems*, pages 1289–1296.

Seung, H. S., Opper, M., and Sompolinsky, H. (1992). Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294. ACM.

Shelhamer, E., Long, J., and Darrell, T. (2017). Fully convolutional networks for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):640–651.

Shen, X., Lin, Z., Brandt, J., Avidan, S., and Wu, Y. (2012). Object retrieval and localization with spatially-constrained similarity measure and k-nn re-ranking. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3013–3020. IEEE.

Sheng, V. S., Provost, F., and Ipeirotis, P. G. (2008). Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 614–622. ACM.

Shrivastava, A., Gupta, A., and Girshick, R. B. (2016). Training region-based object detectors with online hard example mining. In *CVPR*, pages 761–769. IEEE Computer Society.

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. IEEE.

Sivaraman, S. and Trivedi, M. M. (2014). Active learning for on-road vehicle detection: A comparative study. *Machine vision and applications*, 25(3):599–611.

Smola, A. J. and Schökopf, B. (2000). Sparse greedy matrix approximation for machine learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 911–918, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Snow, R., O'Connor, B., Jurafsky, D., and Ng, A. Y. (2008). Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 254–263. Association for Computational Linguistics.

Su, H., Deng, J., and Fei-Fei, L. (2012). Crowdsourcing annotations for visual object detection. In *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Su, H., Qi, C. R., Li, Y., and Guibas, L. J. (2015). Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2686–2694.

Sunderhauf, N., Brock, O., Scheirer, W., Hadsell, R., Fox, D., Leitner, J., Upcroft, B., Abbeel, P., Burgard, W., Milford, M., and Corke, P. (2018). The limits and potentials of deep learning for robotics. *The International Journal of Robotics Research*, 37(4-5):405–420.

Sung, K. K. (1996). *Learning and Example Selection for Object and Pattern Detection*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA. AAI0800657.

Tang, P., Wang, X., Wang, A., Yan, Y., Liu, W., Huang, J., and Yuille, A. (2018). Weakly supervised region proposal network and object detection. In *The European Conference on Computer Vision (ECCV)*.

Tikhonov, A. N. (1943). On the stability of inverse problems. In *Dokl. Akad. Nauk SSSR*, volume 39, pages 195–198.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. (2017). Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30.

Tommasi, T., Patricia, N., Caputo, B., and Tuytelaars, T. (2017). A deeper look at dataset bias. In *Domain adaptation in computer vision applications*, pages 37–55. Springer.

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.

Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., To, T., Cameracci, E., Boochoon, S., and Birchfield, S. (2018). Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 969–977.

Tur, G., Hakkani-Tür, D., and Schapire, R. E. (2005). Combining active and semi-supervised learning for spoken language understanding. *Speech Communication*, 45(2):171–186.

Tuzel, O., Porikli, F., and Meer, P. (2008). Pedestrian detection via classification on riemannian manifolds. *IEEE transactions on pattern analysis and machine intelligence*, 30(10):1713–1727.

Tuzel, O., Porikli, F., Meer, P., et al. (2007). Human detection via classification on riemannian manifolds. In *CVPR*, volume 1, page 4.

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.

Vaillant, R., Monrocq, C., and Le Cun, Y. (1994). Original approach for the localisation of objects in images. *IEE Proceedings-Vision, Image and Signal Processing*, 141(4):245–250.

Vapnik, V. (1998). *The Support Vector Method of Function Estimation*, pages 55–85. Springer US, Boston, MA.

Vezzani, G., Pattacini, U., Pasquale, G., and Natale, L. (2018). Improving superquadric modeling and grasping with prior on object shapes. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6875–6882.

Vijayanarasimhan, S. and Grauman, K. (2014). Large-scale live active learning: Training object detectors with crawled data and crowds. *International Journal of Computer Vision*, 108(1):97–114.

Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I.

Viola, P. and Jones, M. J. (2004). Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154.

Walk, S., Majer, N., Schindler, K., and Schiele, B. (2010). New features and insights for pedestrian detection. In *2010 IEEE Computer society conference on computer vision and pattern recognition*, pages 1030–1037. IEEE.

Wang, K., Lin, L., Yan, X., Chen, Z., Zhang, D., and Zhang, L. (2019). Cost-effective object detection: Active sample mining with switchable selection criteria. *IEEE Transactions on Neural Networks and Learning Systems*, 30(3):834–850.

Wang, K., Yan, X., Zhang, D., Zhang, L., and Lin, L. (2018). Towards human-machine cooperation: Self-supervised sample mining for object detection. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 1605–1613.

Wang, K., Zhang, D., Li, Y., Zhang, R., and Lin, L. (2016). Cost-effective active learning for deep image classification. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(12):2591–2600.

Wang, X., Han, T. X., and Yan, S. (2009). An hog-lbp human detector with partial occlusion handling. In *2009 IEEE 12th international conference on computer vision*, pages 32–39. IEEE.

Whitehill, J., Wu, T.-f., Bergsma, J., Movellan, J. R., and Ruvolo, P. L. (2009). Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in neural information processing systems*, pages 2035–2043.

Widrow, B. and Hoff, M. E. (1960). Adaptive switching circuits. Technical report, Stanford Univ Ca Stanford Electronics Labs.

Williams, C. K. I. and Seeger, M. (2001). Using the nyström method to speed up kernel machines. In Leen, T. K., Dietterich, T. G., and Tresp, V., editors, *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press.

Wu, B. and Nevatia, R. (2005). Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 1, pages 90–97. IEEE.

Xia, G.-S., Bai, X., Ding, J., Zhu, Z., Belongie, S., Luo, J., Datcu, M., Pelillo, M., and Zhang, L. (2018). Dota: A large-scale dataset for object detection in aerial images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3974–3983.

Xu, X. and Frank, E. (2004). Logistic regression and boosting for labeled bags of instances. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 272–281. Springer.

Yun, P., Tai, L., Wang, Y., Liu, C., and Liu, M. (2019). Focal loss in 3d object detection. *IEEE Robotics and Automation Letters*, 4(2):1263–1270.

Zeiler, M. D. and Fergus, R. (2013). Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901.

Zeng, A., Song, S., Yu, K., Donlon, E., Hogan, F. R., Bauza, M., Ma, D., Taylor, O., Liu, M., Romo, E., Fazeli, N., Alet, F., Dafle, N. C., Holladay, R., Morena, I., Nair, P. Q., Green, D., Taylor, I., Liu, W., Funkhouser, T., and Rodriguez, A. (2018). Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8.

Zeng, A., Yu, K., Song, S., Suo, D., Walker, E., Rodriguez, A., and Xiao, J. (2017). Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1386–1383.

Zhang, C. and Chen, T. (2002). An active learning framework for content-based information retrieval. *IEEE transactions on multimedia*, 4(2):260–268.

Zhang, Y., Bai, Y., Ding, M., Li, Y., and Ghanem, B. (2018). W2f: A weakly-supervised to fully-supervised framework for object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhou, D., Bousquet, O., Lal, T. N., Weston, J., and Schölkopf, B. (2004). Learning with local and global consistency. In *Advances in neural information processing systems*, pages 321–328.

Zhou, Z.-H. (2012). *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC.

Zhou, Z.-H. (2017). A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53.

Zhou, Z.-H. and Li, M. (2005). Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge & Data Engineering*, (11):1529–1541.

Zhou, Z.-H. and Li, M. (2010). Semi-supervised learning by disagreement. *Knowledge and Information Systems*, 24(3):415–439.

Zhu, M., Derpanis, K. G., Yang, Y., Brahmbhatt, S., Zhang, M., Phillips, C., Lecce, M., and Daniilidis, K. (2014). Single image 3d object detection and pose estimation for grasping. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3936–3943. IEEE.

Zhu, X., Ghahramani, Z., and Lafferty, J. D. (2003). Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919.

Zhu, X. J. (2005). Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences.

Zitnick, C. L. and Dollár, P. (2014). *Edge Boxes: Locating Object Proposals from Edges*, pages 391–405. Springer International Publishing, Cham.

Zou, Z., Shi, Z., Guo, Y., and Ye, J. (2019). Object detection in 20 years: A survey. *CoRR*, abs/1905.05055.

Appendix A

The iCubWorld Transformations Dataset

During this Ph.D. project, the iCubWorld Transformations Dataset (ICWT) has been considered as a benchmark for validating the proposed methods in a robotic scenario. This dataset is part of a robotic project, called *iCubWorld*[1] which main goal is to benchmark the development of the visual recognition capabilities of the iCub Humanoid Robot Metta et al. (2010). Datasets from the *iCubWorld* project are collections of images recording the visual experience of iCub, while observing objects in its typical environment, a laboratory or an office. ICWT is the last released dataset and the largest one of the project. Please, refer to Pasquale et al. (2019) for details about the acquisition setup.

A.1 Dataset description

ICWT contains images for 200 objects instances belonging to 20 different categories (10 instances for each category). Each object instance is acquired in two separate days with the HRI procedure described in Sec. 6.1.1, in a way that isolates, for each day, different viewpoints transformations: planar 2D rotation (2D ROT), generic rotation (3D ROT), translation with changing background (BKG) and scale (SCALE) and, finally, a sequence that contains all transformations (MIX).

While the dataset has been acquired as benchmark for object recognition, as a contribution of this thesis, I provided object detection annotations in an Imagenet-like format. Moreover, I manually annotated a subset of images, that could be used to validate object detection methods trained with automatically collected data, as it has been done in Maiettini et al. (2017, 2019).

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

A.2 Defining the tasks

For the experiments carried out during this Ph.D. project, where not differently specified, I used as training set, for the objects of the considered task, a subset of the union set of 2D ROT, 3D ROT, TRANSL and SCALE, while as test set, I used a subset of 150 images from the first day of acquisition of the MIX sequence for each object, manually annotated adopting the *labelImg* tool[2]. I fixed an annotating policy such that an object must be annotated if at least a 50-25% of its total shape is visible (i.e. not cut out from the image or occluded). In the remaining part of this section, I describe the details of the subsets of the ICWT used during this project.

Experiments for validating the automatic data collection

For defining the 20 objects identification task presented in Sec. 6.2.2, I considered the following instances from the ICWT:

'ringbinder4', 'flower7', 'perfume1', 'hairclip2', 'hairbrush3','sunglasses7', 'sodabottle2', 'soapdispenser5', 'ovenglove7', 'remote7', 'mug1', 'glass8', 'bodylotion8', 'book6', 'cellphone1', 'mouse9', 'pencilcase5', 'wallet6', 'sprayer6', 'squeezer5'

Experiments for validating the on-line learning for object detection

For defining the FEATURE-TASK presented in Sec. 7.2.3, I considered all the 10 instances of the categories: *'cellphone', 'mouse', 'perfume', 'remote', 'soapdispenser', 'sunglasses', 'glass', 'hairbrush', 'ovenglove', 'squeezer'*, while I defined the TARGET-TASKs presented in Sec. 7.2.3 and in Sec. 7.4 by considering the remaining 10 categories by choosing the instances as follows:

- **1 Object Task**: obtained by averaging results from considering *'sodabottle2', 'mug1', 'sprayer6', 'hairclip2'*
- **10 Objects Task**: *'sodabottle2', 'mug1', 'pencilcase5', 'ringbinder4', 'wallet6', 'flower7', 'book6', 'bodylotion8', 'hairclip2', 'sprayer6'*
- **20 Objects Task**: 10 Objects Task + *'sodabottle3', 'mug3', 'pencilcase3', 'ringbinder5', 'wallet7', 'flower5', 'book4', 'bodylotion2', 'hairclip8', 'sprayer8'*

[2]https://github.com/tzutalin/labelImg

- **30 Objects Task**: 20 Objects Task + *'sodabottle4', 'mug4' , 'pencilcase6', 'ringbinder6', 'wallet10', 'flower2', 'book9', 'bodylotion5', 'hairclip6', 'sprayer9'*
- **40 Objects Task**: 30 Objects Task + *'sodabottle5', 'mug9', 'pencilcase1', 'ringbinder7', 'wallet2', 'flower9', 'book1', 'bodylotion4', 'hairclip9', 'sprayer2'*

Experiments for validating the weakly supervised strategy

For defining the 30 objects identification task presented in Sec. 8.2.3, I considered the following instances from the ICWT:

'sodabottle2', 'mug1', 'pencilcase5', 'ringbinder4', 'wallet6', 'flower7', 'book6', 'bodylotion8', 'hairclip2', 'sprayer6', 'sodabottle3', 'mug3', 'pencilcase3', 'ringbinder5', 'wallet7', 'flower5', 'book4', 'bodylotion2', 'hairclip8', 'sprayer8', 'sodabottle4', 'mug4' , 'pencilcase6', 'ringbinder6', 'wallet10', 'flower2', 'book9', 'bodylotion5', 'hairclip6', 'sprayer9'

Appendix B

Complete Minibootstrap procedure

This section reports the complete pseudo-code (Alg. 2) for the Minibootstrap procedure described in Sec. 7.1.3.

Algorithm 2 Minibootstrap Complete pseudo-code for the Minibootstrap procedure. See Sec 7.3.2 for a detailed explanation of the pipeline.

Input: N = {Set of all negative examples in the dataset}, P = {Set of all positive examples in the dataset}, BS = size of bootstrap's batches, n_B = number of bootstrap's iterations
Output: M_{final} = trained classifier model

Stage 1: *Subsample dataset and create n_B batches of negatives of size BS*
$[N_1, ..., N_{n_B}] \leftarrow$ CreateRandNegativesBatches(N, BS, n_B)
Stage 2: *Train classifier using first batch*
$D_1 \leftarrow P \bigcup N_1$;
$M_1 \leftarrow$ TrainClassifier(D_1);
$N_{chosen\_1} \leftarrow N_1$
Stage 3: *Select hard negatives*
for all $i \in \{2, ..., n_B\}$ **do**
1) Select hard negatives from N_i using M_{i-1} and add them to the train set:
$N_i^H \leftarrow$ SelectHard(M_{i-1}, N_i)
$D_i \leftarrow P \bigcup N_{chosen\_i-1} \bigcup N_i^H$
2) Train classifier with the new dataset:
$M_i \leftarrow$ TrainClassifier(D_i)
3) Prune easy negatives from D_i using M_i:
$N_{chosen\_i} \leftarrow$ PruneEasy$(M_i, N_{chosen_i-1} \bigcup N_i^H)$
end for
Stage 4: *Train final model with selected dataset*
$D_{final} \leftarrow P \bigcup N_{chosen_{N_{n_B}}}$
$M_{final} \leftarrow$ TrainClassifier(D_{final})

Appendix C

Stopping Criterion for Faster R-CNN Fine-tuning

In this section, I report on the cross validation carried out to study the convergence of Faster R-CNN and to determine when to stop the learning for the tasks of the experiments presented in Chapter 7.

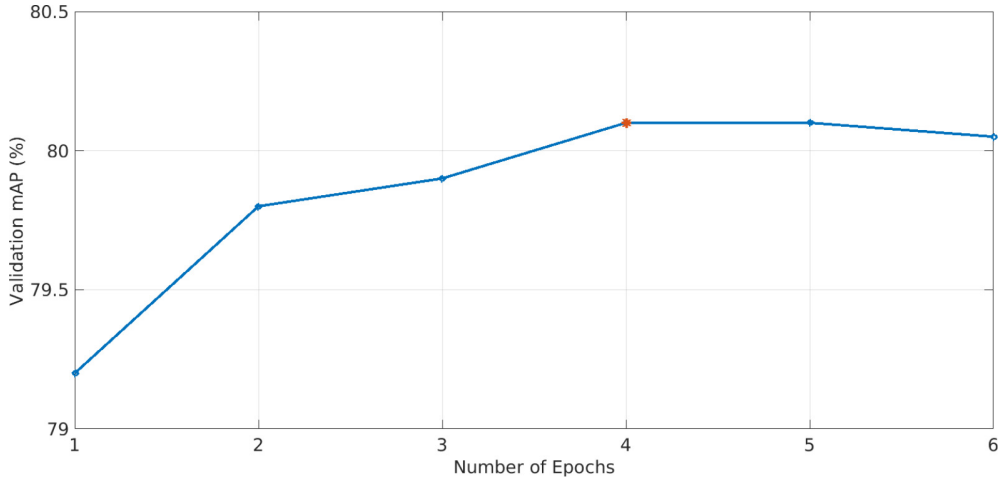


Figure C.1: This figure shows the validation accuracy trend with respect to the number of epochs for the Pascal VOC dataset (**blue line**). The **red star** highlights the number of epochs chosen to train the Faster R-CNN baseline, reported in Tab. 7.1).

In Fig. C.1, I report the validation accuracy trend on the Pascal VOC dataset, when learning the last layers of Faster R-CNN for increasing number of epochs. To this aim, within the Pascal VOC, I split the available images considering the union of the validation sets of Pascal 2007 and 2012 as validation set and the union of the training sets of Pascal 2007 and 2012 as training set.

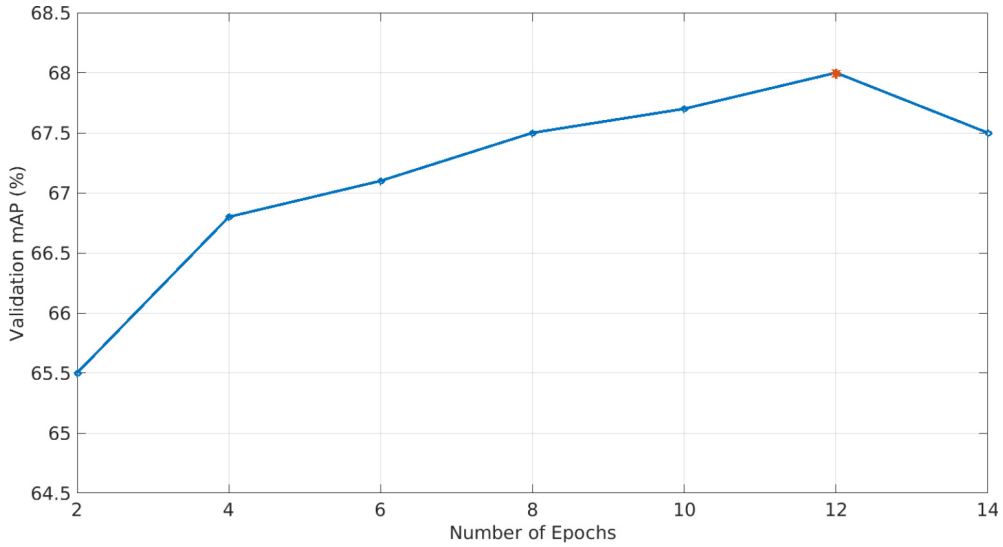


Figure C.2: This figure shows the validation accuracy trend with respect to the number of epochs for the ICWT dataset (**blue line**). The **red star** highlights the number of epochs chosen to train the Faster R-CNN baselines, reported in Tab. 7.2 and Fig. 7.3.

Similarly, in Fig. C.2 I report the validation accuracy trend with respect to the number of epochs for the ICWT dataset. In this case, I considered as train set the same ∼8k images used for the TARGET-TASK in Sec. 7.2, while I selected a different set of 4.5k images as validation set, considering the remaining images in the 2D ROT, 3D ROT, SCALE and TRANSL transformations.

Finally, in Fig. C.3, I show the validation accuracy trend of the full train of Faster R-CNN (i.e. the optimization of the convolutional layers, RPN, feature extractor and output layers on the TARGET-TASK). Specifically, in this case, since I used the 4-Steps alternating training procedure as in Ren et al. (2015), I report the mAP trend, considering different numbers of epochs, when learning the RPN and the Detection Network. Therefore, the two numbers reported for each tick of the horizontal axis, represent respectively the number of epochs (i) for learning the RPN during steps 1 and 3 of the procedure and (ii) for learning the Detection Network during steps 2 and 4 of the procedure. I consider the same training and validation splitting as in Fig. C.2.

Note that, I used these results for the stopping criterion, that consists in choosing the model at the epoch achieving the highest mAP on the validation set (I stopped when no mAP gain was observed in the three plots). I highlighted in red in the three plots, the configurations chosen to train the baselines on the tasks at hand, in Table 7.1, 7.2 and 7.3 and in Fig. 7.3.

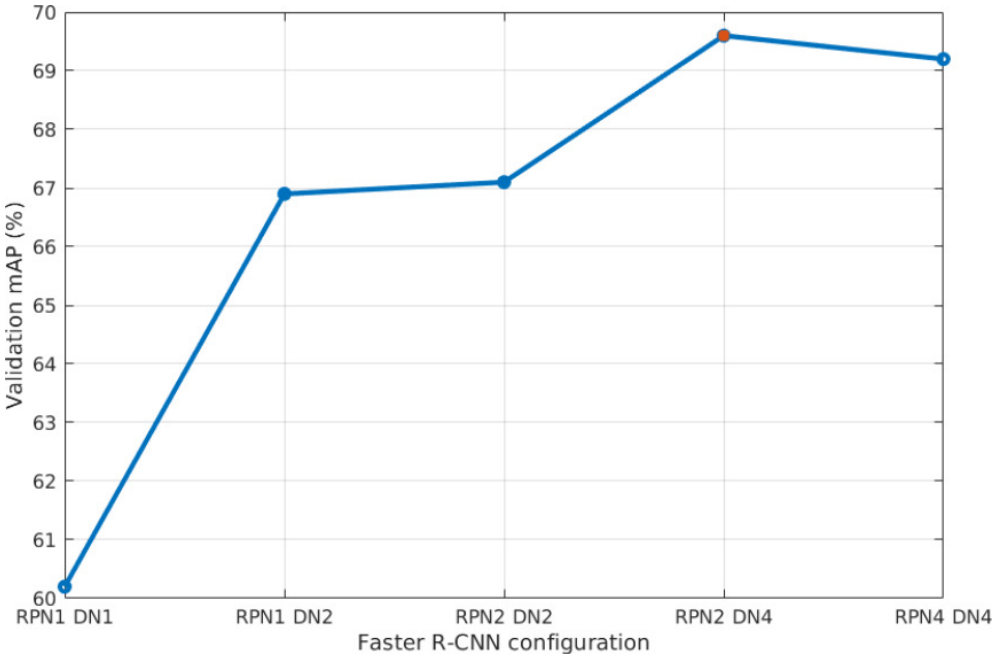


Figure C.3: This figure shows the validation accuracy trend for different configurations of epochs of the 4-Steps alternating training procedure Ren et al. (2015) for the ICWT dataset (**blue line**). Each tick of the horizontal axis represents a different configuration. The numbers of epochs used for the RPN and for the Detection Network are reported, respectively, after the labels *RPN* and *DN*. The **red point** highlights the configuration chosen to train the Faster R-CNN baseline, reported in Tab. 7.3.

Appendix D

Examples of detected images

Fig. D.1 and Fig. D.2 show examples of detections predicted by the FALKON + MINIBOOTSTRAP described in Sec. 7.1, on random sampled images from respectively the test sets of Pascal VOC Everingham et al. (2010) and of ICWT Pasquale et al. (2019).

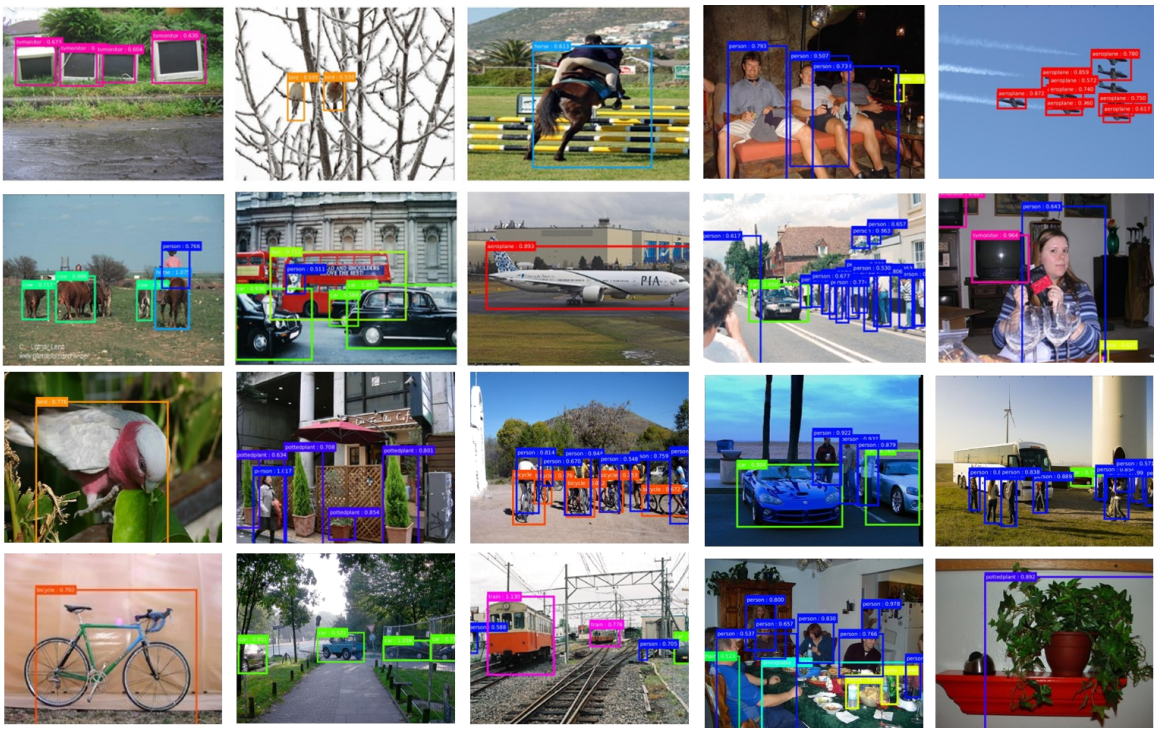


Figure D.1: Randomly sampled examples of detections on the PASCAL VOC 2007 test set, obtained with the on-line learning pipeline described in Sec. 7.1. Resnet101 is used as CNN backbone for the feature extractor. The training set is *voc07++12* and the model used is FALKON + MINIBOOTSTRAP 10x2000 (1 m and 40 s of **Train Time** and 70.4% of **mAP**).

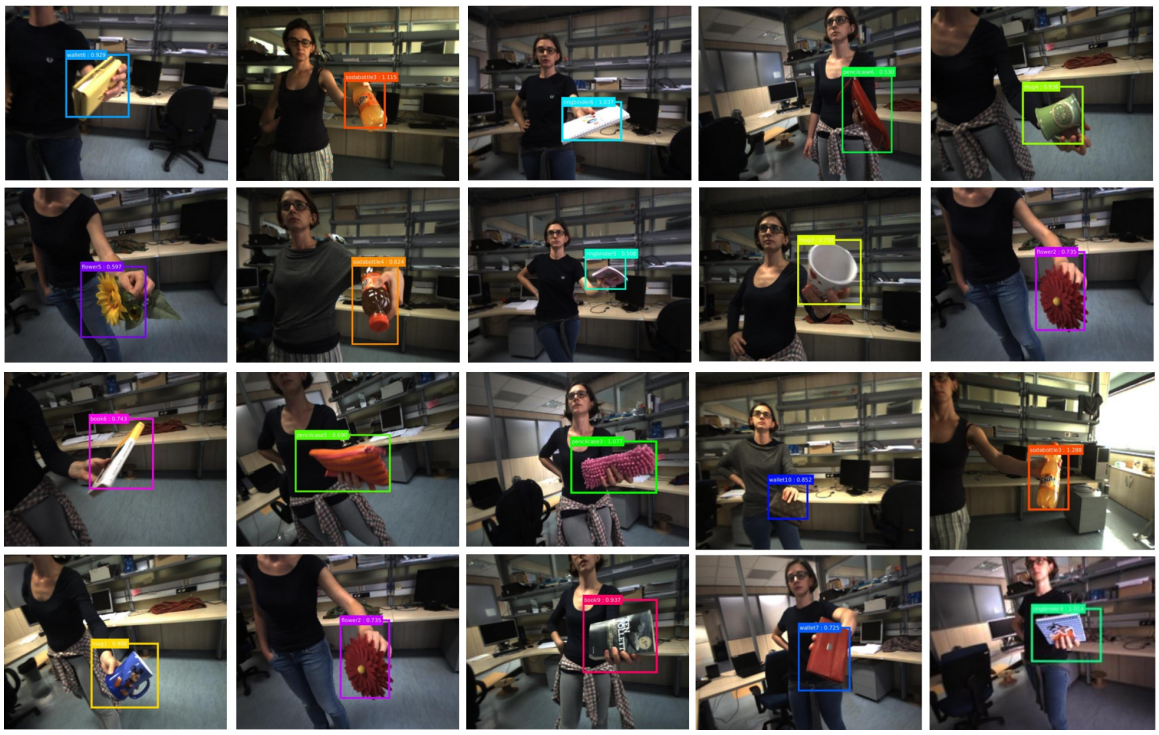


Figure D.2: Randomly sampled examples of detections on ICWT, obtained with the on-line learning pipeline described in Sec. 7.1. Resnet50 is used as CNN backbone for the feature extractor. FEATURE-TASK and TARGET-TASK are respectively the 100 and 30 objects tasks described in Sec. 7.2.3. The model used is FALKON + MINIBOOTSTRAP 10x2000 (40 s of **Train Time** and 71.2% of **mAP**).

Appendix E

Preliminary Analysis of the SSM method

Before integrating the on-line learning method for object detection described in Sec. 7.1 with the weakly supervised strategy proposed in the SSM method Wang et al. (2018), a preliminary analysis on this latter was carried out to understand feasibility. Specifically, the aim of this first experiments was intended to verify the effectiveness of the weakly supervised strategy, proposed in SSM, in the robotic scenario considered in this project. For this purpose, the ICWT dataset has been used.

E.1 Experimental setup

For this preliminary analysis, the SSM was trained as explained in Wang et al. (2018). Specifically, the detection algorithm used is the R-FCN Dai et al. (2016b) with Resnet101 He et al. (2015) as convolutional backbone. Moreover, all the settings of the method (e.g., the ones regarding the percentages of data chosen for the active learning and the self supervised learning processes) have been left unchanged with respect to Wang et al. (2018).

For the two learning phases considered in SSM, i.e., the *Supervised Phase* and the *Weakly Supervised Phase* (as defined in Sec. 5.4.4), two different datasets have been considered, respectively the LABELED-TASK and the UNLABELED-TASK. In both cases, a 30-object identification problem was considered, using the same object instances as in Sec. 7.2.3. However, for each object, I used the TRANSL sequence (for a total of $\sim$2K images) as LABELED-TASK and the union of the 2D ROT, 3D ROT and SCALE sequences (for a total of $\sim$6K images) as UNLABELED-TASK. This simulates a situation where only a simple sequence is fully annotated and other sequences are not. As a test set, I used

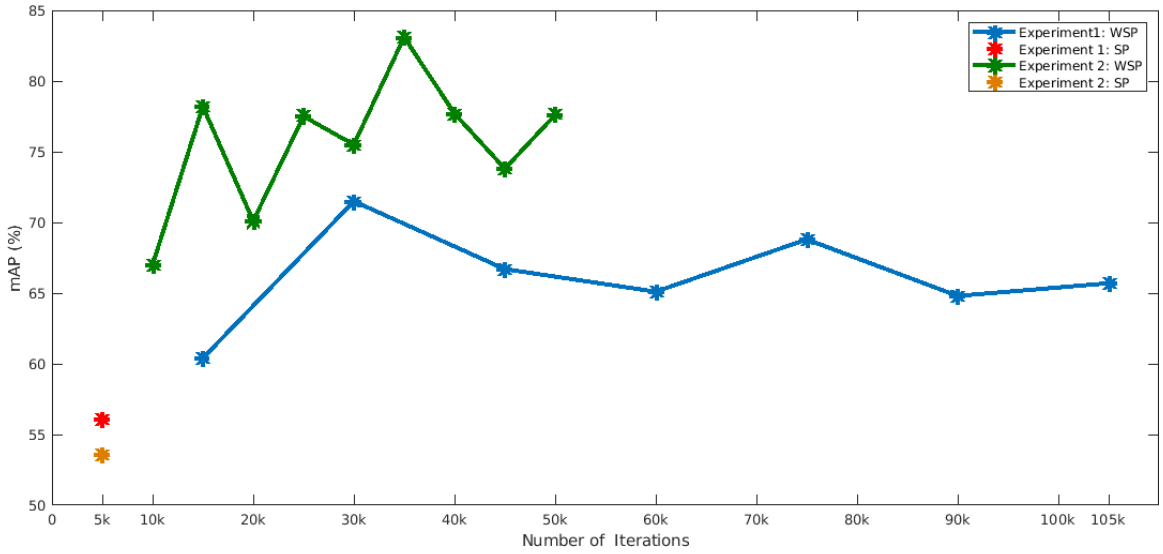


Figure E.1: The figure shows the mAP trends of SSM, as the number of training iterations on the UNLABELED-TASK grows for two different experiments. For both of them, the number of iterations for the *Supervised Phase* (SP in the plot) has been set to 5k. In **Experiment 1**, (red point and blue line), 15k iterations are used for each step of the *Weakly Supervised Phase* (WSP in the plot), while in **Experiment 2**, (orange point and green line), 5k iterations are used.

150 images from the MIX sequence of each object, which annotations have been manually refined adopting the *labelImg* tool[1].

E.2 Results

A part of the results of this preliminary analysis is reported in Fig. E.1, which shows the mAP trends with respect to the number of iterations used to train the detection model on the UNLABELED-TASK for two different cases. More specifically, Fig. E.1 reports on the comparison between two experiments with the same aforementioned setup conditions but where the detection model is trained with different numbers of iterations at each weakly supervised step. I identify in Fig. E.1 the two trials as **Experiment 1** and **Experiment 2**. In both cases, the number of iterations for the *Supervised Phase* has been set to 5k (represented in Fig. E.1 as a red point for **Experiment 1** and as an orange point for **Experiment 2**). For each step of the *Weakly Supervised Phase* instead, 15k iterations were considered for **Experiment 1** (blue line in Fig. E.1), following the default value proposed in Wang et al.

[1]https://github.com/tzutalin/labelImg

(2018) and 5k iterations for **Experiment 2** (green line in Fig. E.1). The reported points represent the accuracy after each weakly supervised step.

The results reported in the figure show that the default value for the number of iterations (i.e. 15k) brings to a drop of accuracy for growing number of iterations, probably due to overfitting. However, by considering a smaller number of iterations an increasing overall trend for the mAP is observed. This represented a promising result that motivated the integration reported in Chapter 8.

Note that, while the LABELED-TASK and UNLABELED-TASK correspond respectively to the TARKET-TASK-LABELED and TARKET-TASK-UNLABELED used in Sec. 8.2.3, the two experiments cannot be compared. In fact, in the analysis reported in Sec. 8.2.3, the feature extractor considered is composed by the first layers of Faster R-CNN, with Resnet50 used as CNN backbone while in this case R-FCN is used with Resnet101. Moreover, in Sec. 8.2.3, the first layers of the architecture (namely, the *Feature Extraction module*) are not updated on the TARKET-TASK-LABELED, retaining the weights optimized on the FEATURE-TASK, while in this case, no FEATURE-TASK is considered but a part of the backbone's layers in the R-FCN are trained on both the LABELED-TASK and UNLABELED-TASK. Finally, in this case, the R-FCN is trained using the technique of dynamically and randomly re-scaling some input training images, while the *Feature Extraction module* in Sec. 8.2.3 is trained with no specific technique against scale variations.

Appendix F

Pre-scripted autonomous exploration with R1

Since the main objective of the work reported in Sec. 8 is to assess the feasibility of the integration between the on-line learning object detection pipeline and the SSM, the exploration strategy used to acquire the table-top dataset described in Sec. 8.2.1 was pre-scripted (more sophisticated exploratory strategies will be of interest for future developments). Specifically, three degrees of freedom in the table top scenario have been considered (please, refer to Fig. F.1 for a pictorial representation):

1. The *Position* of R1 with respect to the table (in Fig. F.1: N: *North*, S: *South*, E: *East*, W: *West*, H: *Home*). The different positions can be reached by moving the base of the robot.

2. The *Elevation* of R1's torso. Three possible elongations are considered (in Fig. F.1: Up: *maximum elevation*, Down: *minimum elevation*, Medium: *medium elevation*).

3. The *Bending* of R1's torso. Three possible angles are considered (in Fig. F.1: Backwards: *maximum backwards bending*, Forwards: *maximum forwards bending*, No: *no bending*).

The sequence of movements used to acquire the table-top dataset described in Sec. 8.2.1 is obtained by combining the different possibilities for each degree of freedom represented in Fig. F.1 and it is reported is Fig. F.2. Note that, each step in the figure defines a target position, elevation and bending.

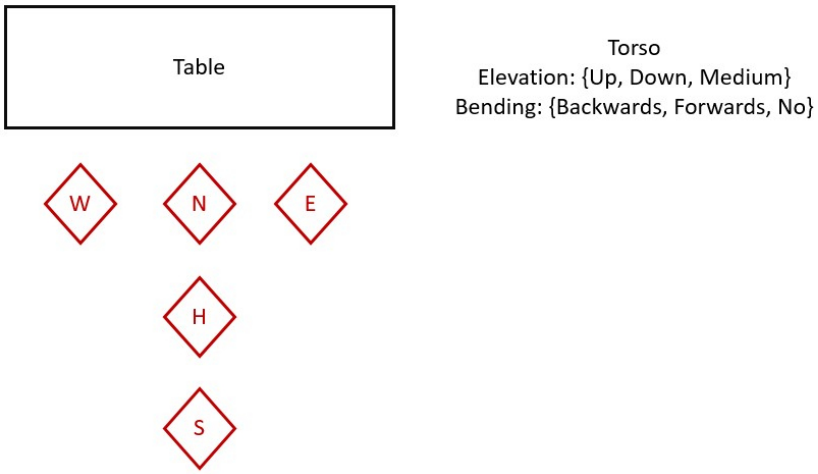


Figure F.1: This figure shows a pictorial representation of the map of all the possible movements that can be used, within the three degrees of freedom considered.

Steps	Position	Torso Elevation	Torso Bending
Step0	H	Medium	No
Step1	S	Up	Backwards
Step2	W	Down	No
Step3	N	Down	No
Step4	H	Up	No
Step5	E	Down	No
Step6	N	Up	Forwards
Step7	H	Down	No

Figure F.2: List of movements used to acquire the table-top dataset described in Sec. 8.2.1, with the R1 robot.