Noname manuscript No.
(will be inserted by the editor)

On-line Object Detection: a Robotics Challenge

Elisa Maiettini[1,2,3] · **Giulia Pasquale**[1,2] · **Lorenzo Rosasco**[2,3] · **Lorenzo Natale**[1]

Received: date / Accepted: date

Abstract Object detection is a fundamental ability for robots interacting within an environment. While stunningly effective, state-of-the-art deep learning methods require huge amounts of labeled images and hours of training which does not favour such scenarios.

This work presents a novel pipeline resulting from integrating (Maiettini et al., 2017) and (Maiettini et al., 2018), which naturally trains a robot to detect novel objects in few seconds. Moreover, we report on an extended empirical evaluation of the learning method, justifying that the proposed hybrid architecture is key in leveraging powerful deep representations while maintaining fast training time of large scale Kernel methods.

We validate our approach on the Pascal VOC benchmark (Everingham et al., 2010), and on a challenging robotic scenario (iCubWorld Transformations (Pasquale et al., 2019)). We address real world use-cases and show how to tune the method for different speed/accuracy trades-off. Lastly, we discuss limitations and directions for future development.

This material is based upon work supported by the Center for Brains, Minds and Machines (CBMM), funded by NSF STC award CCF-1231216. We gratefully acknowledge the support of NVIDIA Corporation for the donation of the Titan Xp GPUs and the Tesla k40 GPU used for this research. L. R. acknowledges the financial support of the AFOSR projects FA9550-17-1-0390, BAA-AFRL-AFOSR-2016-0007 (European Office of Aerospace Research and Development), the EU H2020-MSCA-RISE project NoMADS - DLV-777826 and Axpo Italia SpA.

[1] Humanoid Sensing and Perception, Istituto Italiano di Tecnologia, Genoa, Italy

[2] Laboratory for Computational and Statistical Learning, Istituto Italiano di Tecnologia and Massachusetts Institute of Technology, Cambridge, MA

[3] Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi, University of Genoa, Genoa, Italy

E. Maiettini
E-mail: elisa.maiettini@iit.it

Keywords Computer Vision · Robot Vision · Object Detection · Deep Learning · Humanoid robots

1 Introduction

Perception is the first and fundamental task for robots which are supposed to interact within an environment. While typically robotic platforms, especially humanoids (Metta et al., 2010), are provided with multiple sensory modalities, latest results in computer vision, pushed by deep learning advances (Russakovsky et al., 2015; He et al., 2015; Redmon and Farhadi, 2016; He et al., 2017; Shelhamer et al., 2017), motivate a particular interest in visual perception systems.

Robots might be asked to accomplish very different tasks and each of them could require a specific level of cognition of the environment. In this paper we focus on the problem of object detection, i.e. the task of localizing the bounding box around an object and its label. This capability is clearly at the basis of more sophisticated tasks such as robot navigation, object grasping and manipulation.

State-of-the-art solutions for this task based on deep learning perform stunningly well (Redmon and Farhadi, 2016; Dai et al., 2016; He et al., 2017) in computer vision benchmarks (Everingham et al., 2015; Russakovsky et al., 2015; Lin et al., 2014). However, their applicability to robotics comes with difficulties (Sunderhauf et al., 2018). First, 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., 2017) 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. Both types of solutions are time consuming, if applied in canonical ways. In this work we propose a solution which relies on a fast approximation of the latter state-of-the-art approach.

While recent research on object detection and segmentation for robotics mainly focuses on improving robustness in challenging scenarios, like occlusions in clutter (Zeng et al., 2018; Georgakis et al., 2017; Schwarz et al., 2018; Patten et al., 2018) we suggest that an online learning pipeline which can be trained and adapted in few seconds, accounting for the issues described above, represents a fundamental step towards any further improvement. With this perspective in mind, in our previous work (Maiettini et al., 2017) we first demonstrated that an automatic procedure (Pasquale et al., 2016a; Pasquale et al., 2016b) to acquire annotated images by interacting with a robot allows to train detection algorithms (e.g. Faster R-CNN (Ren et al., 2015)) with good accuracy. However, in (Maiettini et al., 2017) training was still performed off-line (as in (Ren et al., 2015)).

Recently, in (Maiettini et al., 2018) we proposed an on-line learning pipeline that also speeds-up the training. We exploited generality of features provided by deep CNNs (Sharif Razavian et al., 2014; Donahue et al., 2014; Jia et al., 2014; Girshick et al., 2014; Pasquale et al., 2019) and designed our method in two main modules: (i) a per-region feature extractor, trained once off-line on a certain task, and (ii) a region classifier which can be trained quickly and on-line on a different task, the target task. Specifically, for this latter module we adopt FALKON (Rudi et al., 2017), a recently proposed kernel-based method for large-scale datasets, and we leverage on the stochastic sampling of the kernel centers that it performs. We, then, propose an approximated version of the Hard Negative Mining procedure, to efficiently re-balance the training set.

In this work we present a detailed description of the pipeline resulting by integrating (Maiettini et al., 2017) and (Maiettini et al., 2018), which allows to naturally train the robot to detect novel objects. In addition we present an extended study of the learning method, providing the following major contributions:

1. We rigorously benchmark the method on the official Pascal VOC (Everingham et al., 2010) and achieve state-of-the-art performance by integrating Resnet101 (He et al., 2015) as CNN backbone;
2. We test the approach on a challenging robotic dataset, namely the iCubWorld Transformations (Pasquale et al., 2019). Within this scenario, we analyze the components of the pipeline, providing interesting insights for each one of them. Specifically, we consider performance in terms of accuracy and train time. Our 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 our choice; (iii) the demonstration of how to tune the main parameters of the method in order to obtain a speed/accuracy trade-off.
3. In the same scenario we propose some experiments designed to challenge the pipeline in real world conditions, showing limitations and considering possible solutions.

The paper is organized as follows: in Sec. 2 we give an overview of the literature with a deeper insight on how the main problem in object detection (i.e., the large amount of background regions in an image) is tackled by state-of-the-art solutions. Then, in Sec. 3 we describe the proposed on-line detection pipeline. We provide results and considerations from our extended empirical analysis of the resulting system in Sec. 4, 5 and 6. Finally in Sec. 7 and 8 we conclude and draw lines for future work.

2 Related Work

In this section we give an overview of the most recent deep learning based methods for object detection and we analyze some current research for this field in robotics. While discussing the state-of-the-art, we illustrate our contribution.

Deep Learning for Object Detection. Approaches to address the object detection task can be grouped into (i) grid-based methods like SSD (Single-Shot MultiBox Detector) (Liu et al., 2015) and YOLO (You Only Look Once) (Redmon et al., 2016; Redmon and Farhadi, 2016) and (ii) region-based methods (see e.g., Region-CNN (R-CNN) (Girshick et al., 2014) and its optimizations: Fast R-CNN (Girshick, 2015), Faster R-CNN (Ren et al., 2015), Region-FCN (Dai et al., 2016) and Mask R-CNN (He et al., 2017)). In algorithms from the first group, for each image, classifiers are applied over a regular, dense sampling of possible object locations, scales, and aspect ratios, while in algorithms from the second group, a previous step of region proposal generation is performed to predict a sparse set of candidate object locations (see, e.g., Selective Search (Uijlings et al., 2013), EdgeBoxes (Zitnick and Dollár, 2014), DeepMask (Pinheiro et al., 2015; Pinheiro et al., 2016), Region Proposal Network (RPN) (Ren et al., 2015)). In these region-based approaches, the detection pipeline can be divided into (i) generation of region candidates, (ii) proposals encoding into deep features and (iii) classification and refinement. Even though the common trend in recent works is to integrate the three stages into "monolithic" models, learned end-to-end via backpropagation (Dai et al., 2016; He et al., 2017; Ren et al., 2015), another natural approach to address the task is to consider multi-stage architectures, which allow to tackle each step of the pipeline separately (Girshick et al., 2014; Girshick, 2015).

We propose a detailed empirical evaluation of a pipeline that exploits the benefits of an RPN for Regions of Interest prediction (Ren et al., 2015), while relying on a two-stage training procedure to allow fast model adaptation to new tasks.

The problem of Negatives Selection. The issue with all types of approaches is that, in order to predict the locations of the objects of interest, a huge amount of candidate regions needs to be visited. Each candidate region is treated as a positive or negative example, that is used to train classifiers. Since most candidate regions typically originate from background areas, 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.

In the literature, different solutions have been proposed to deal with these issues. In particular, (Lin et al., 2017) 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 rebalanced, in case their number is large. Such a loss is designed for end-to-end training of grid-based detectors and has been showed 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 some 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), uses only high-loss region proposals for the SGD iteration, rather than a heuristically sampled subset.

Finally, works that rely on a "cascade" of detectors (Viola et al., 2001; Felzenszwalb et al., 2010a) 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, afterwards they apply increasingly more complex classifiers on regions that are most likely to contain objects.

These solutions are not designed for on-line learning, because either they require to iteratively visit all nega-

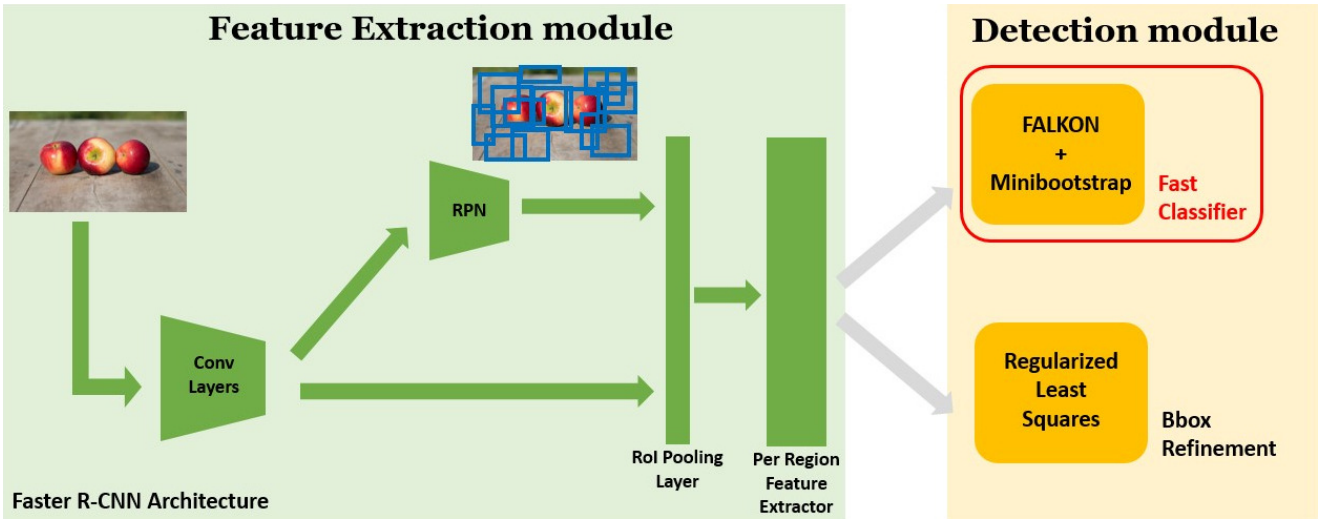

Fig. 1: Overview of the proposed on-line object detection pipeline. *Feature Extraction module* relies on Faster R-CNN architecture to extract deep features and predict RoIs (Regions of Interest) from each input image. *Detection module* performs RoIs classification and refinement, providing as output the actual detections for the input image. For a detailed description see Sec. 3.

tive examples in the training set (Girshick et al., 2014), or they rely on end-to-end backpropagation (Shrivastava et al., 2016).

In this work we analyze a novel approach (Maiettini et al., 2018) 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).

Object Detection for Robotics. State-of-the-art research on object detection for robotics mainly focuses on improving robustness and precision in particular scenarios like occlusions in clutter (Zeng et al., 2018; Georgakis et al., 2017; Schwarz et al., 2018; Tobin et al., 2017). This finds one of the main motivations in the existing challenges, like the Amazon Picking Challenge [1], where the robots are asked to pick objects in a cluttered bin.

Another important research field related to Object Detection in Robotics is regarded as Active Perception (Bajcsy et al., 2018). In the past years, many works have been proposed to improve object detection and recognition performance by exploiting the robot's active exploration of the environment at inference time, in order to refine or gain confidence on the predictions (Bajcsy et al., 2018; Browatzki et al., 2012).

The aim of this work is, instead, to propose and analyze a learning pipeline for object detection, targeting a scenario where a humanoid robot is required to quickly learn novel objects. To the best of our knowledge this is the first work focusing on providing a robotic visual system for object detection trainable in few seconds.

[1] `http://amazonpickingchallenge.org/`

3 On-line Object Detection Pipeline

In the scenario considered for this work, 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, we 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 we describe the proposed approach. We give an overview of the pipeline in Sec. 3.1. In Sec. 3.2, we explain how each component is learned. Finally, in Sec. 3.3 we provide details on the on-line learning method, describing how the issues of the dataset size and imbalance are addressed.

3.1 Overview

The proposed pipeline is composed of two stages (see Fig. 1): (i) per-region feature extraction (*Feature Extraction module*) and (ii) region classification and refinement (*Detection module*).

For the first stage we 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. Specifically, in this work we adopt ResNet50 and ResNet101 (He et al., 2015), as CNN backbones, integrated in Faster R-CNN as explained in Sec. 3.2.

For the second stage we replace the last two output layers of Faster R-CNN for class prediction and bounding box refinement (namely *cls* and *bbox*), with our *Detection module*. Specifically, for the classification of the proposed RoIs, we design a novel method which employs a set of FALKON binary classifiers; for the bounding box refinement, instead, we simply rely on the Regularized Least Squares (RLS) regression approach proposed in Region-CNN (Girshick et al., 2014).

Note that, in splitting feature extraction and classification, we take inspiration from the Region-CNN architecture originally proposed in (Girshick et al., 2014). However, differently to R-CNN, we fine-tune off-line the weights for feature extraction and region proposal on one task (the FEATURE-TASK), while we train the classifiers and the bounding box regressors on-line on the task at hand (the TARGET-TASK). More details about this two-stage learning procedure are provided in Sec. 3.2.

3.2 Learning

Within the scenario previously described, in this work we propose to learn the two modules of the pipeline 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). We refer the reader to (Ren et al., 2015) for a detailed explanation of this architecture and training procedure. As previously mentioned, we adopt ResNet50 or ResNet101 as CNN backbone for Faster R-CNN. As suggested in (He et al., 2015), we 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 we 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 we used the method of Region-CNN (Girshick et al., 2014), keeping the same learning objective and loss function, while the novelty of our approach is in the proposals classification. Specifically, we 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, we collect the training set by selecting and labeling candidate RoIs as either positive examples, i.e., belonging to the class (we indicate this set as P) or negative ones, i.e., with an Intersection over Union (IoU) with the ground truth smaller than 0.3 (we indicate this set as N). The resulting dataset is used to train a binary classifier and it is usually large and strongly unbalanced. E.g., 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, we 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.

3.3 Fast Training of Classifiers

Our 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.

Algorithm 1 Bootstrapping Iteration. Core iteration of proposed Minibootstrap. See Appendix A 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} iterations, 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:
$N_{chosen\_i} \leftarrow \text{PruneEasy}(M_i, N_{chosen\_i-1} \bigcup N_i^H)$

In this section, we 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 (Sung, 1996; Felzenszwalb et al., 2010b; Girshick et al., 2014) 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, we 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 represents the set of the negatives of the i^{th} batch (the complete Minibootstrap algorithm is reported in Appendix A).

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 our evaluation we empirically set them respectively to -0.7 and -0.9.

As we will show in Sec. 4, the proposed Minibootstrap is effective as the original Hard Negatives Mining method and preserves accuracy, while performing the selection over a subset of the dataset, hence allowing much faster train time.

FALKON: Speeding-up Classifier Training while Rebalancing. As classification algorithm for our approach we 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 (Williams and Seeger, 2001; Smola and Schökopf, 2000) 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.

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 we 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. We refer the reader to (Rudi et al., 2017) for a detailed description of the algorithm. In our pipeline, we adopt the publicly available FALKON implementation[2].

In this work, we propose to 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, we take a number of P' positives with $P' = \min\left(P, \frac{M}{2}\right)$, while we 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 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

[2] `https://github.com/LCSL/FALKON_paper`

and, in the worst case, discarding all positives from the sampled Nÿstrom centers.

3.3.1 Hyper-parameters

The main parameters of FALKON are (i) the Kernel parameter (where not specified, we 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.

We 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 we define two different ranges, respectively for σ and λ, and we search for the best values by considering every possible combination and performing the Minibootstrap procedure for each of them. Furthermore, in Sec. 4, we provide experimental evaluation of the other three parameters characterizing our approach. Notably, we show how by setting their values, it is possible to tune the procedure to subsample the training set more or less extensively, depending on the desired speed/accuracy trade-off.

4 Results

In this section we first provide details about the setup of the experiments (Sec. 4.1) and then we present the performance achieved by the proposed on-line object detection pipeline on two different benchmarks (Sec. 4.2 and Sec. 4.3).

4.1 Experimental Setup

In our experiments we compare to Faster R-CNN, as baseline. For a fair comparison, we consider the weights learned by training Faster R-CNN on the FEATURE-TASK (as in Sec. 3.2, Off-line Stage) and use them for both (i) the *Feature Extraction module* of our pipeline (Fig. 1), and (ii) as a warm restart for fine-tuning the output layers of Faster R-CNN on the TARGET-TASK (i.e. we set the learning rate to 0 for all the layers of the network except the output ones). We consider ResNet50 and ResNet101 (He et al., 2015) as different backbones for feature extraction in Faster R-CNN. We report in Appendix D the results of the cross validation for the number of epochs for the fine-tuning of the baselines on the TARGET-TASK. We used those results for our stopping criterion, that consists in choosing the model at the epoch achieving the highest mAP on the validation set (we stopped when no mAP gain was observed).

In the following, we 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 we consider the implementation of (Girshick et al., 2014) and we re-train the classifier only when a number of 2000 new negative examples has been accumulated to the training set. We 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.

We evaluate the method on two very different datasets: (i) Pascal VOC (Everingham et al., 2010) (Sec. 4.2) and (ii) ICUBWORLD TRANSFORMATIONS (Pasquale et al., 2019) (Sec. 4.3) and we report performance, in both cases, in terms of (i) mAP (mean Average Precision), as defined for Pascal VOC 2007, and (ii) training time.

All experiments reported in this paper 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, we limit the RAM usage of FALKON to at most 10GB.

4.2 Benchmark on The Pascal VOC

We first evaluate the performance of the proposed method on the Pascal VOC dataset, a standard benchmark for object detection.

We consider, for training and validation, the union set of Pascal VOC 2007 and 2012 trainval sets, gathering ∼16k images (*voc07++12* in the following). We use the available Pascal VOC 2007 test set, which consists of about ∼5k images, for testing. We 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 ICUBWORLD TRANSFORMATIONS, here we cannot split the object categories to be used for the FEATURE-TASK 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, we use the standard training and test splits

Table 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**

Table 2: Benchmark on iCWT. We compare 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**

(specified above) both for the FEATURE-TASK and the TARGET-TASK.

For learning the FEATURE-TASK we set the number of iterations to 80k when learning the RPN and to 40k when learning the detection network. As a baseline, we train the output layers of Faster R-CNN from scratch for 32k iterations (4 epochs, with batch size 2). In Appendix D, we 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 we used for our stopping criterion.

We 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, we set the number of Nÿstrom centers to 2000 (the influence of this parameter is investigated in Sec. 5.3.1).

As can be observed from Table 1 we can train a detection model 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, we are able to reproduce the state of the art performance (outperforming it of 1.6%) in less than a half of the time. Examples of detections predicted by the FALKON + Minibootstrap 10x2000 are reported in Appendix C.

4.3 A robotic scenario: iCubWorld

In this section, we evaluate the proposed method in a robotic scenario, considering the iCubWorld Transformations dataset[3] (iCWT in the following). A description of the dataset and details regarding how we use it for our experiments are reported in Appendix B.

Within the scenario described in Sec. 3, we define a FEATURE-TASK as an identification task among 100 objects comprising all available instances (10 per class) of 10 out of 20 categories in iCWT. We 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, we 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 B). Overall, this leads to a training set of ∼55k and ∼8k images for respectively the FEATURE-TASK and the TARGET-TASK.

As test set we 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[4]. We refer to Appendix B for further details.

In the *Feature Extraction module* we used ResNet50 as convolutional backbone for Faster R-CNN, which we 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.

We report results for two different configurations of the Minibootstrap, namely in Table 2, (i) FALKON + Minibootstrap 10x2000 and (ii) FALKON + Minibootstrap 100x1500, that, as we will show in Sec. 5.3, 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. 5.3 other more extreme trades-off are presented).

From Table 2 it can be observed that we were able to reproduce 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 D, we

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

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

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 we used for our stopping criterion. Moreover, we 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 C.

Towards Real World Robotic Applications. We deployed the proposed algorithm 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. We relied on the YARP (Metta et al., 2006) middleware for integrating different modules. A video of the resulting application is publicly available [5].

5 Ablation Study

In Sec. 4 we showed results of the proposed method on two benchmarks, demonstrating its effectiveness and gain in train time. In this section we propose an ablation study, analyzing in detail the main components of the proposed on-line object detection pipeline.

First, we present a study of the *Feature Extraction module* (Sec. 5.1). Then, we compare different options for the classification algorithm, motivating the choice of FALKON (Sec. 5.2). Finally, we investigate the main hyper-parameters of the method, providing guidelines on how to tune the training procedure (Sec. 5.3).

If not specified, we consider (i) the same 100 objects FEATURE-TASK and 30 objects TARGET-TASK as in Sec. 4.3, (ii) the Minibootstrap configuration $n_B = 10$ and $B = 2000$ and (iii) the number of Nÿstrom centers to $M = 2000$.

5.1 How to Learn the Feature Extraction Module?

In our pipeline we train the *Feature Extraction module* only once off-line and on a FEATURE-TASK, which is different from the TARGET-TASK. By doing so we are able to adapt the feature extractor 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, we analyze the impact of the *Feature Extraction module* on the proposed on-line learning pipeline. In particular, we first show the accuracy loss that originates when features are tuned on a task that is different from the TARGET-TASK. We 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 we evaluate the performance loss we obtain when the FEATURE-TASK differs from the TARGET-TASK. We 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 D, we report the results of the cross validation for the number of epochs for learning Faster R-CNN on the TARGET-TASK, that we used as model selection criterion.

As it can be noticed from Tab. 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, we 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, we consider as FEATURE-TASK, the categorization task on *voc07++12* and we set the number of iterations to 160k when learning the RPN and to 80k when learning the detection network. For ICWT, we consider the same feature extractor as in Sec. 4.3 (learned on the FEATURE-TASK of 100 objects identification). We use ResNet50 as backbone.

In Table 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

[5] `https://youtu.be/eT-2v6-xoSs`

Table 3: We compare: 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

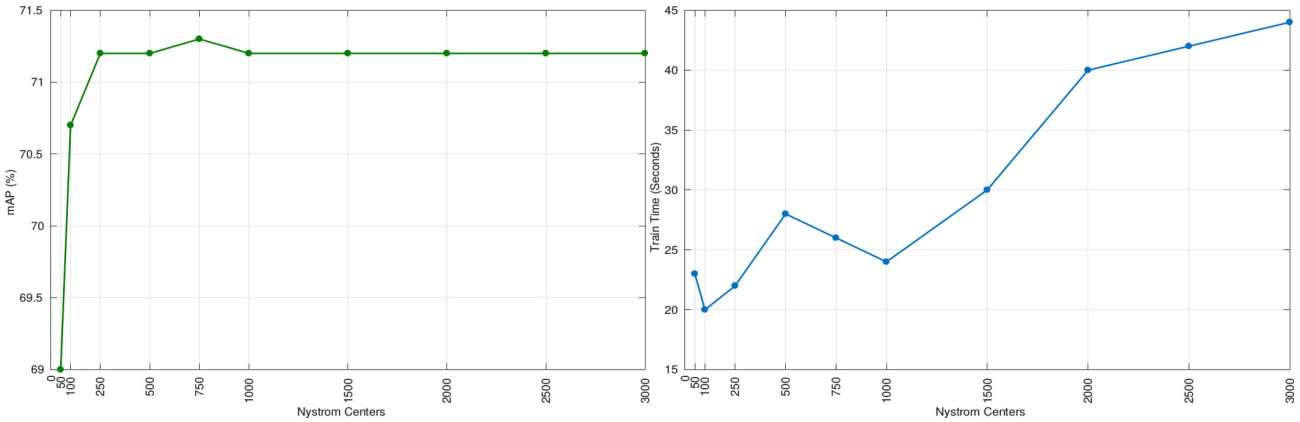

Fig. 2: Train Time (**Right**) and mAP (**Left**) trends for varying number of Nÿstrom centers when training FALKON within the Minibootstrap (see Sec. 5.3.1 for details).

Table 4: We compare: 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. 5.1).

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

performance than using a more general, though richer, one.

Choosing the task. We demonstrated that adapting the feature extractor to the domain of the TARGET-TASK remarkably improves performance. With this experiment we 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, we evaluate the impact of the number of classes used to define the FEATURE-TASK. We decrease the classes from 100 to 10, considering respectively all the classes in the FEATURE-TASK of Sec. 4.3 in the first case and selecting 1 instance per category from it in the second case (see Appendix B for details).

From Table 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.

5.2 Classification Module: Is FALKON Key to Performance?

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

In Tab. 6 we compare as classifiers (i) FALKON with Gaussian Kernel (FALKON-gauss + MINIBOOTSTRAP), (ii) FALKON with Linear Kernel (FALKON-linear + MINIBOOTSTRAP) and (iii) linear SVMs (SVMs-linear + MINIBOOTSTRAP). For the latter one, we use the LIBLINEAR implementation[6]. Note that, instead, for the proposed pipeline, we used the available FALKON Matlab [7] 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 we consider it as the best choice for the proposed learning pipeline since, even if the corresponding train time is slightly longer,

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

Table 5: We compare results considering two different FEATURE-TASKs from iCWT to train off-line the *Feature Extraction module*. Specifically, we compare the train time and mAP obtained by the *Detection module* on the same TARGET-TASK, using 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 6: We compare Train Time and mAP of different classifiers within the Minibootstrap, 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**

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.

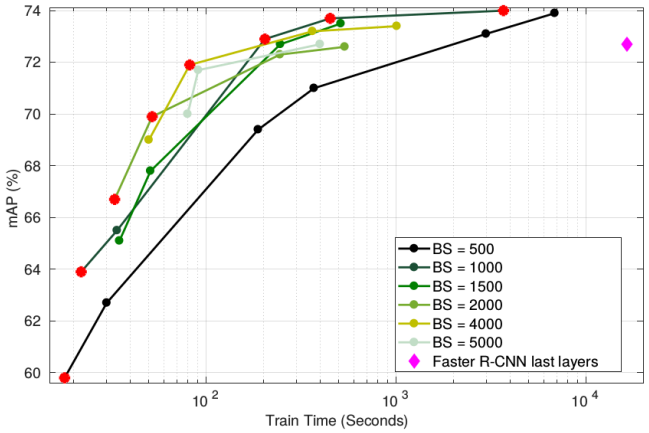

Fig. 3: We show mAP and Train Time trends for n_B and BS variation (see Sec. 3.3) during Minibootstrap. Color code represents BS variation while each point of each plot represents a different n_B taken from the ordered set: $\{5, 10, 50, 100, 500, 1000\}$.

5.3 Analysis of Hyper-parameters

In this section, we show the role of the main parameters of the proposed method. We recall that we cross-validate σ and λ, namely the variance of the Gaussian Kernel and the regularization parameter, as explained in Sec. 3. In the following, instead, we study the influence of the parameters specific of our method, namely the number of Nÿstrom centers (M in Sec 3.3) and the number and size of the batches of negatives in the Minibootstrap procedure (respectively n_B and BS in Sec3.3).

5.3.1 Nÿstrom Centers

In order to evaluate the impact of the number of Nÿstrom centers M in FALKON training, we 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 we 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. 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 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.

5.3.2 Minibootstrap Configuration

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

For this study, we fix the TARGET-TASK to the same 30 objects identification task as in Sec. 4.3, but we consider a training set of ∼16k images, to show more extensively the effect of parameter tuning on a bigger dataset.

Results for this experiment are reported in Fig. 3 (note that x axis is represented in logarithmic scale). We consider as baseline the performance achieved by training the output layers of Faster R-CNN, represented by the violet dot in Fig. 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. 4.3 (it differs only for the number of images), we do not repeat the validation for it, training Faster R-CNN for 12 epochs (see Appendix D 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. 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 500 (black curve in Fig. 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. 3, which represent the best speed/accuracy trades-off, going from a mAP of 59.8% achieved in ∼15 seconds to a mAP of 74% (surpassing the Faster R-CNN baseline) in ∼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.

6 Challenging the Method for Robotics

In this section, we 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. 6.1) or the number of available images decreases drastically (see Sec. 6.2).

To this aim, in the following experiments we fix (i) the same 100 objects identification FEATURE-TASK as in Sec. 4.3, (ii) the Minibootstrap configuration to $n_B = 10$ and $B = 2000$ and (iii) the number of Nÿstrom centers to $M = 2000$.

6.1 Increasing number of objects for the TARGET-TASK.

In this section, we investigate how accuracy and training time are affected by increasing the number of classes of the TARGET-TASK. To this aim, we consider five different tasks as TARGET-TASK. In particular, while we fix the number of images for each object to 250, we vary the number of objects from 1 to 40. We 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 task of 10, 20, 30 and 40 objects (see Appendix B for further details). Then, we randomly select four objects instances to evaluate performance on four different 1-object detection tasks and report the average results. We compare with training the last layers of Faster R-CNN for 12 epochs.

Results are reported in Fig. 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. 4 shows that our 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, optimization time is in the order of magnitude of hours.

6.2 Decreasing the number of images for the TARGET-TASK

In this experiment, we evaluate the effect on mAP and training time of considering more or less example images for the TARGET-TASK. To this end, we consider the same TARGET-TASK as in Sec. 4.3 (30 objects), and vary the number of example images for each object from ∼16 (for a total of ∼500 training images) to ∼800 (for a total of ∼16k training images).

As can be noticed from Fig. 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 we observe 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 we keep constant the number of epochs for the optimization, allowing the network to really train on more data, for the Minibootstrap we fix the n_B and BS parameters, leading to a more aggressive subsampling in

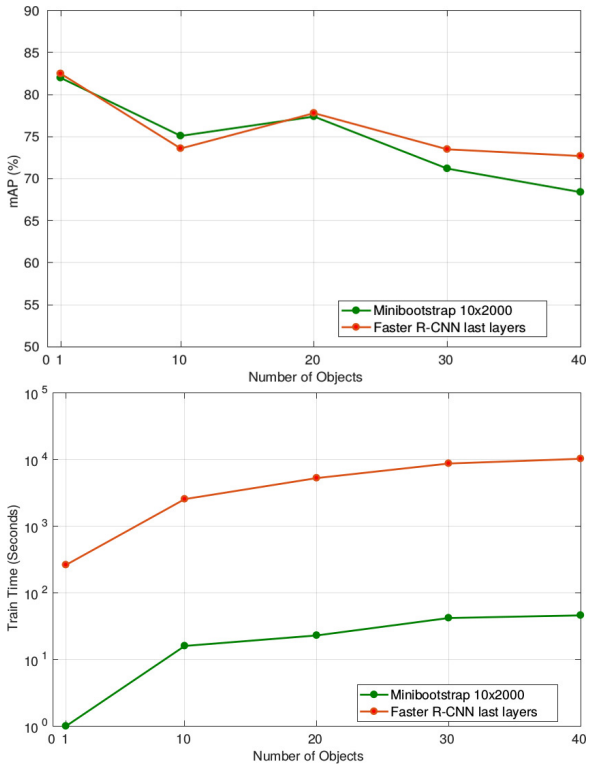

Fig. 4: We compare performance of FALKON + MINIBOOTSTRAP 10x2000 trained on different TARGET-TASKs, varying the number of objects in a range from 1 to 40. We show mAP (**Top**) and Train Time (**Bottom**). We compare results with fine-tuning Faster R-CNN last layers.

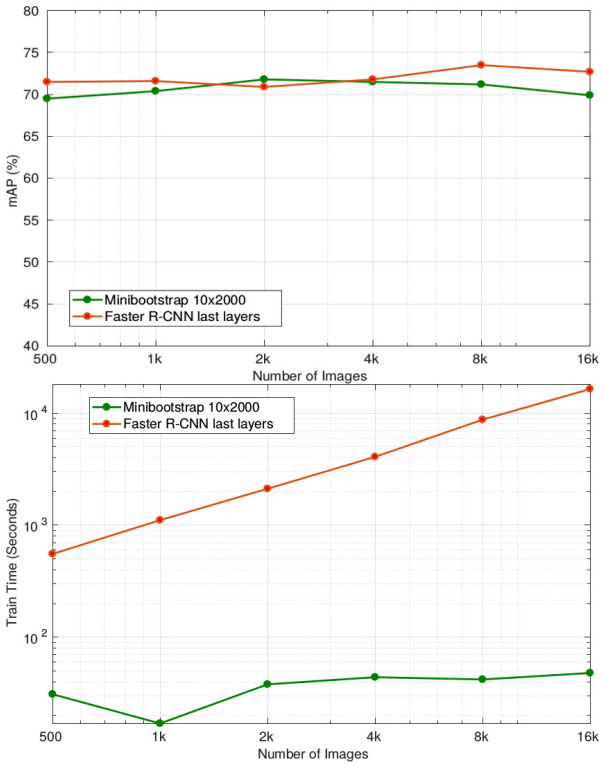

Fig. 5: We compare performance of FALKON + MINIBOOTSTRAP 10x2000 trained on a 30 objects identification TARGET-TASK considering different numbers of samples for each class. We show mAP (**Top**) and Train Time (**Bottom**). We compare results with fine-tuning Faster R-CNN last layers.

the cases of bigger datasets, thus a poorer representation of the negatives. As we showed in Fig. 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 ∼80 seconds and 72.9% in ∼3 minutes.

Moreover, Fig. 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 Discussion

In this work we presented a pipeline that allows to naturally train a humanoid robot in only a few seconds to detect novel objects. The system is the result of the integration of (i) the automatic data collection procedure proposed in (Pasquale et al., 2016b) and validated for detection in (Maiettini et al., 2017) with (ii) the fast learning algorithm proposed in (Maiettini et al., 2018).

Specifically, this work presents a complete and extensive empirical evaluation of the method, demonstrating the benefits of its adoption in real world robotic applications. In the scenario considered in our analysis a robot is asked to learn a *Detection module* during a few seconds of interaction with a human, by relying on a previously trained *Feature Extraction module*.

We first validated the pipeline on two different benchmarks considering Pascal VOC and ICWT datasets, to prove the effectiveness of the method in both a general-purpose computer vision scenario and in a robotic setup. Then, we performed a detailed evaluation of the main components of the method. We considered variants for

learning the *Feature Extraction module*. We motivated our choice of FALKON as classification method with respect to other methods proposed in the literature. We showed the influence of the three main hyper-parameters of the proposed method and finally we challenged the method by considering some typical real robotic scenarios.

We believe that this learning pipeline can be used for further improvements of detection systems in robotics since it allows to be quickly adapted while preserving state-of-the-art accuracy. In the following paragraphs, we identify some possible directions for future research.

Although our experiment 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 our scenario, because training sequences may contain insufficient variability (in terms of background, illumination or partial occlusions). This is not a problem in the proposed object detection, but it is however, a challenge that we will have to address in the future. Moreover, detection in cluttered scenes is a particularly challenging task. 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 human work load required for learning a detection model is dramatically reduced with the proposed pipeline, as the manual annotation has been substituted with a more natural teacher-learner interaction. However, the human effort can be further reduced by considering techniques of active learning (Settles, 2012), which would require the human only to reply to direct inquiries of the robot, e.g., by adapting existing approaches designed for computer vision (Wang et al., 2018). Another possibility is to completely remove the human contribution in the acquisition pipeline by considering approaches of autonomous exploration of the objects of interest like in (Pinto et al., 2016).

In Sec. 6.2, we showed that we achieve a noticeable fast training time 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.

8 Conclusions

In this paper we propose an extended and detailed empirical evaluation of a visual robotic system which allows to train a humanoid robot to detect novel objects by naturally interacting with it for few seconds.

Natural interaction and fast learning methods are both fundamental features for humanoid robots that are asked to adapt quickly to their environment. In this sense we believe that the system proposed in this work represents a baseline for further improvements, being a first step toward the implementation of more adaptive robotic systems.

References

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

Browatzki, B., Tikhanoff, V., Metta, G., Blthoff, 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.

Dai, j., Li, Y., He, K., and Sun, J. (2016). 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.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2014). Decaf: A deep convolutional activation feature for generic visual recognition. In Jebara, T. and Xing, E. P., editors, *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 647–655. JMLR Workshop and Conference Proceedings.

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. 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.

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)*.

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*.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia - MM '14*, pages 675–678. ACM Press.

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

Lin, T., Goyal, P., Girshick, R. B., He, K., and Dollár, P. (2017). Focal loss for dense object detection. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2999–3007.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollr, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, Zrich. Oral.

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

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)*.

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.

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 robots 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.

Patten, T., Zillich, M., and Vincze, M. (2018). Action selection for interactive object segmentation in clutter. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6297–6304.

Pinheiro, P. O., Collobert, R., and Dollar, P. (2015). Learning to segment object candidates. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 1990–1998. Curran Associates, Inc.

Pinheiro, P. O., Lin, T.-Y., Collobert, R., and Dollr, 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.

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*.

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)*.

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.

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.

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.

Settles, B. (2012). Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114.

Sharif Razavian, A., Azizpour, H., Sullivan, J., and Carlsson, S. (2014). Cnn features off-the-shelf: An astounding baseline for recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

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.

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.

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.

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.

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.

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.

Viola, P., Jones, M., et al. (2001). Rapid object detection using a boosted cascade of simple features. *CVPR (1)*, 1(511-518):3.

Wang, K., Yan, X., Zhang, D., Zhang, L., and Lin, L. (2018). Towards human-machine cooperation: Self-supervised sample mining for object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

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.

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.

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.

Zitnick, C. L. and Dollár, P. (2014). *Edge Boxes: Locating Object Proposals from Edges*, pages 391–405. Springer International Publishing, Cham.

A Complete Minibootstrap procedure

This section reports the complete pseudo-code (Alg. 2) for the Minibootstrap procedure described in Sec. 3.3.

Algorithm 2 Minibootstrap Complete pseudo-code for the Minibootstrap procedure. See Sec 5.2 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*
$[N_1, ..., N_{n_B}] \leftarrow \text{CreateRandNegativesBatches}(N, BS, n_B)$
Stage 2: *Train classifier using first batch*
$D_1 \leftarrow P \bigcup N_1$;
$M_1 \leftarrow \text{TrainClassifier}(D_1)$;
$N_{chosen\_1} \leftarrow N_1$
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:
$N_i^H \leftarrow \text{SelectHard}(M_{i-1}, N_i)$
$D_i \leftarrow P \bigcup N_{chosen\_i-1} \bigcup N_i^H$
2) Train classifier with the new dataset:
$M_i \leftarrow \text{TrainClassifier}(D_i)$
3) Prune easy negatives from D_i using M_i:
$N_{chosen\_i} \leftarrow \text{PruneEasy}(M_i, N_{chosen\_i-1} \bigcup N_i^H)$
end for
Stage 4: *Train final model with selected dataset*
$D_{final} \leftarrow P \bigcup N_{chosen_{N_{n_B}}}$
$M_{final} \leftarrow \text{TrainClassifier}(D_{final})$

B The iCubWorld Transformations Dataset

For validating the proposed pipeline in a robotic scenario, we considered the iCubWorld Transformations Dataset (iCWT). This dataset is part of a robotic project, called *iCubWorld*[8] 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. We refer to (Pasquale et al., 2019) for details about the acquisition setup.

B.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, 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, lately we also provided object detection annotations in Imagenet-like format. Moreover we manually annotated a subset of images, that could be used to validate object detection methods trained with automatically collected data, as we did in (Maiettini et al., 2017). Specifically, for this work we released a new and larger set of manually annotated images[9].

For the experiments of this work we consider as train set, for the objects of the considered task, a subset of the union set of 2D ROT, 3D ROT, BKG and SCALE, while as test set we 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[10]. We 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).

For defining the FEATURE-TASK presented in 4.3, we consider all the 10 instances of the categories: *'cellphone', 'mouse', 'perfume', 'remote', 'soapdispenser', 'sunglasses', 'glass', 'hairbrush', 'ovenglove', 'squeezer'*, while we define the TARGET-TASKs presented in Sec. 4.3 and in Sec. 6 by considering the remaining 10 categories by choosing the instances as following:

- **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'*
- **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'*

C Examples of detected images

We report in Fig. 6 and Fig. 8 examples of detections predicted by the FALKON + Minibootstrap on random sampled images from respectively the test set of Pascal VOC (Everingham et al., 2010) and of iCWT (Pasquale et al., 2019).

D Stopping Criterion for Faster R-CNN Fine-tuning

In this section we report on the cross validation carried out to study the convergence of Faster R-CNN and to choose when to stop the learning for the tasks of this work.

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

[9] https://robotology.github.io/iCubWorld/#icubworld-transformations-modal/

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

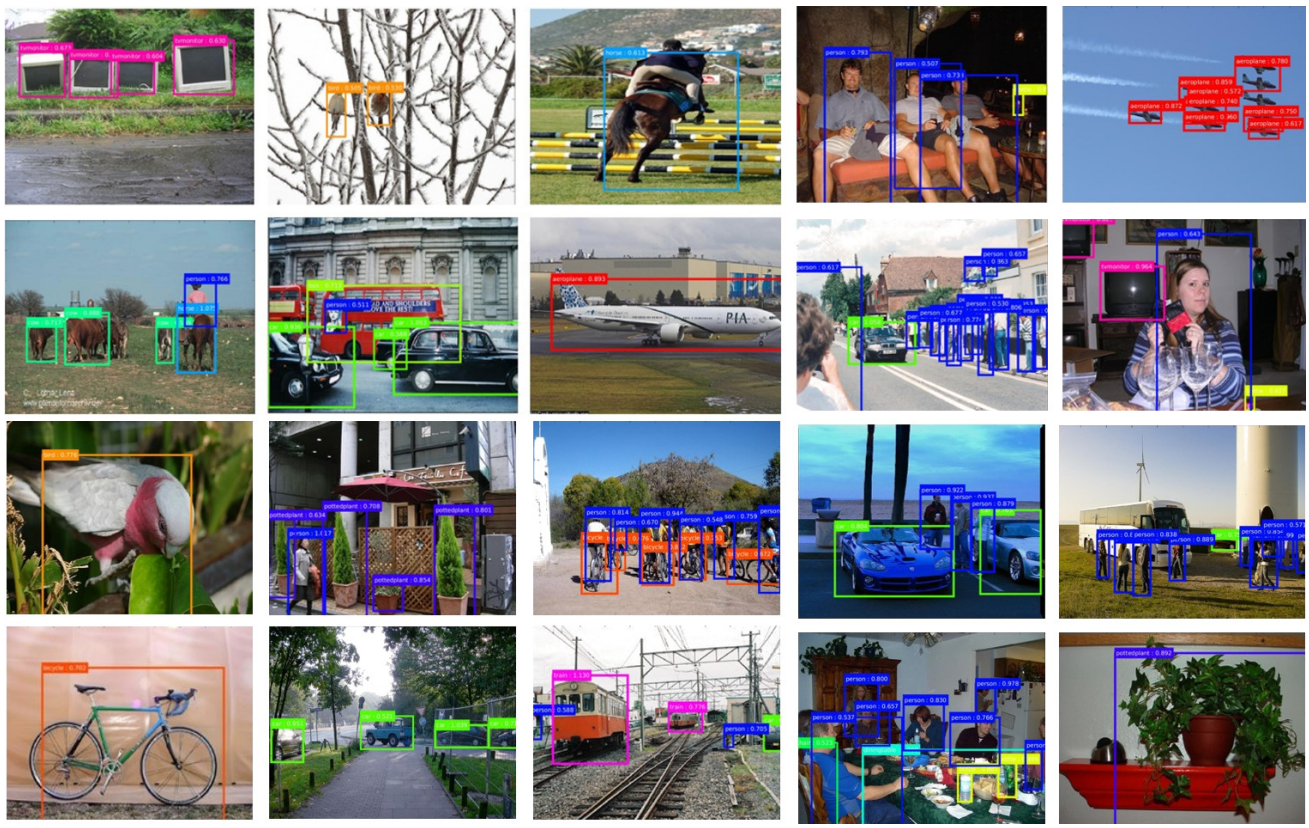

Fig. 6: Randomly sampled examples of detections on the PASCAL VOC 2007 test set, obtained using the proposed learning pipeline. CNN backbone for Faster R-CNN for feature extraction is Resnet101, train data is the set of images *voc07++12* and the configuration used is FALKON + MINIBOOTSTRAP 10x2000 (1 minute and 40 seconds of **Train Time** and 70.4% of **mAP**).

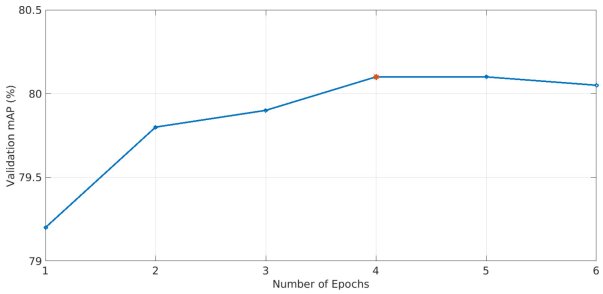

Fig. 7: We show the validation accuracy trend with respect to the number of epochs for the Pascal VOC dataset (**blue line**) and we highlight (**red star**) the number of epochs chosen to train the Faster R-CNN baseline, reported in Tab. 1).

In Fig. 7 we 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, we 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.

Similarly, in Fig. 9 we report the validation accuracy trend with respect to the number of epochs for the ICWT dataset. In this case, we considered as train set the same ∼8k images used for the TARGET-TASK in Sec. 4, while we 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. 10 we 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 we used the 4-Steps alternating training procedure as in (Ren et al., 2015), we 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. We consider the same training and validation splitting as in Fig, 9.

Note that, we used these results for our stopping criterion, that consists in choosing the model at the epoch achieving the highest mAP on the validation set (we stopped when no mAP gain was observed in the three plots). We highlighted in red in the three plots, the configurations chosen to train the baselines on the tasks at hand, in Table 1, 2 and 3 and in Fig. 3.

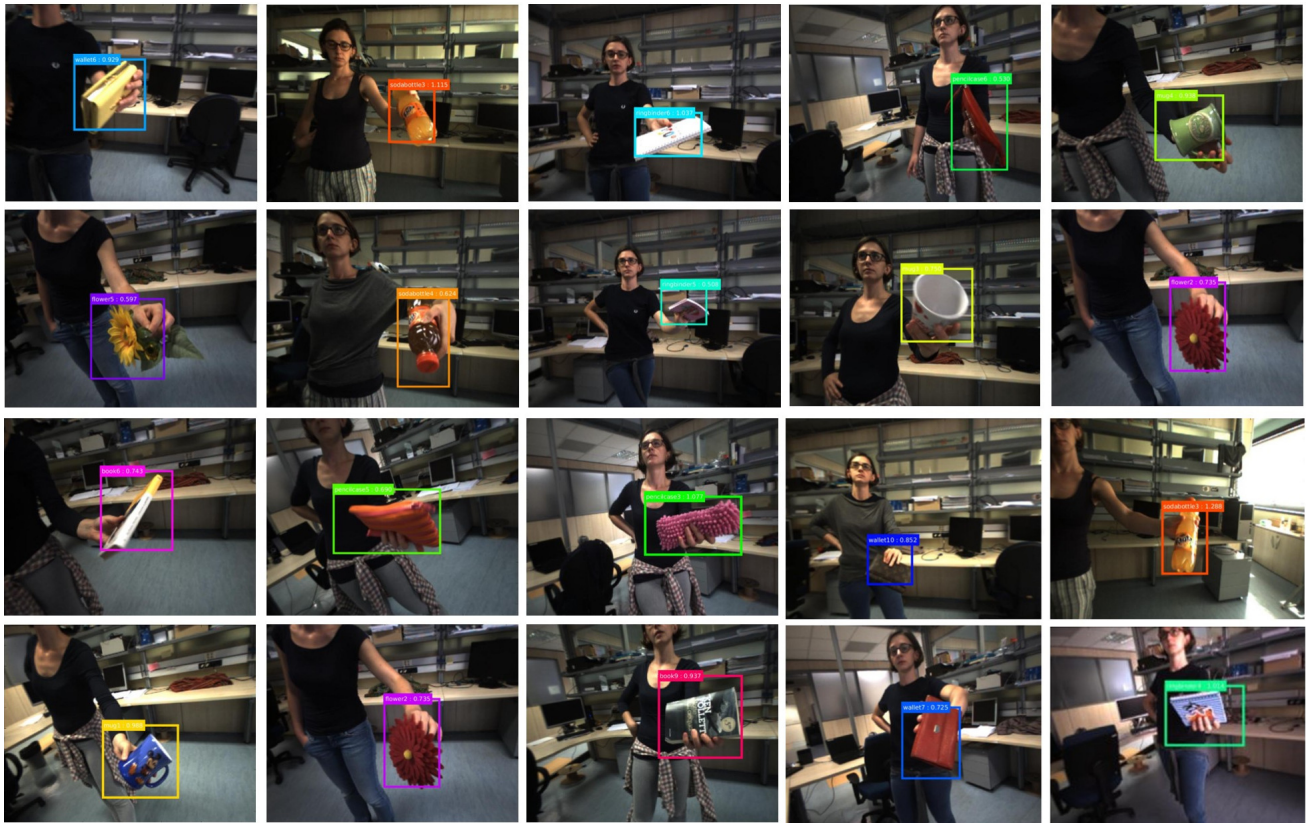

Fig. 8: Randomly sampled examples of detections on ICWT, obtained using the proposed learning pipeline. CNN backbone for Faster R-CNN for feature extraction is Resnet50, FEATURE-TASK and TARGET-TASK are respectively the 100 and 30 objects tasks described in Sec. 4.3 and the configuration used is FALKON + MINIBOOTSTRAP 10x2000 (40 seconds of **Train Time** and 71.2% of **mAP**).

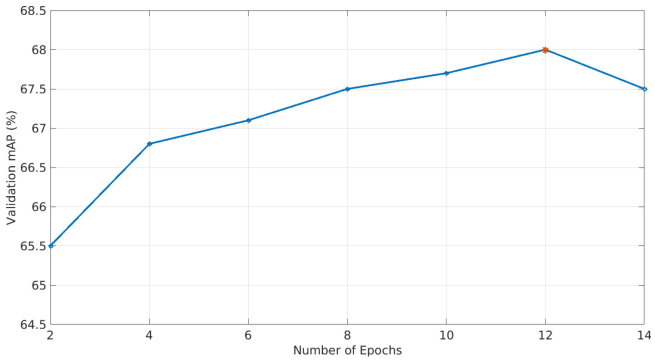

Fig. 9: We show the validation accuracy trend with respect to the number of epochs for the ICWT dataset (**blue line**) and we highlight (**red star**) the number of epochs chosen to train the Faster R-CNN baselines, reported in Tab. 2 and Fig. 3.

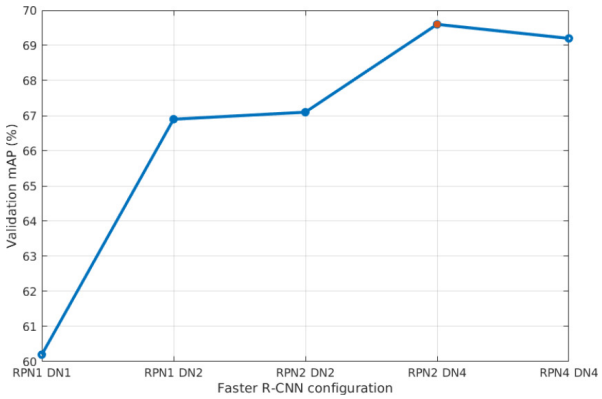

Fig. 10: We show the validation accuracy trend for different configurations of epochs of the 4-Steps alternating training procedure (Ren et al., 2015) for the ICWT dataset (**blue line**). Each tick of the horizontal axis represents a different configuration. The numbers of epochs used for the RPN and for the Detection Network are reported, respectively after the labels *RPN* and *DN*. We highlight (**red star**) the configuration chosen to train the Faster R-CNN baseline, reported in Tab. 3.