Deep Learning and Soft Tasking - Towards Respbots (Responsive Collaboration Robotics)

João Pedro Neves da Silva
joaopnsilva@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal

January 2021

Abstract

Despite the recent success of state-of-the-art deep learning algorithms in object detection, the practical applications of these methods and the availability of products which make use of this technology is still very limited. In this work it is evaluated the possibility of applying this technology to a table-top pick and place robot, which can be marketed at a low price to the general public. To that end, two different scenarios are used, one referent to an household environment and another dedicated to simulate an electronics laboratory. It is also studied two different philosophies for the training data. In the first case a purposely curated dataset is used to collect the data, while on the second the data is collected from standard internet searches and processed by the author. Furthermore, several object detection methods (Faster R-CNN, YOLO, SSD) are tested in order to evaluate the practical feasibility of this technology but also to evaluate which method would be best in order to consummate this objective. In light of that, this work can be seen as a proof of concept of using deep learning based object detection algorithms in a pick and place robot of small dimensions.
Keywords: Object Detection, Object Recognition, Deep Learning in Computer Vision, Pick and Place Robot.

1. Introduction

For the last century, robots have changed from science fiction to a reality in certain areas of our life. Since the coining of the term by the brothers Čapek [6], in the late 1910's, robots have greatly developed and are now present in all kinds of fields, such as industry, transportation or medicine. Their depiction in literature and cinematic fiction is extremely diverse. These depictions, and countless other popular culture references, show a widespread desire of integrating robots in menial and everyday tasks.

This desire has given origin to several wide fields of research, amongst which we can count Human-robot Collaboration and Cobots. In fact, both these fields have real applications, for instance, in healthcare [5], military applications [3], rescue [28], industrial production [34], domestic environment [10] and many more.

Companies and markets have also taken a keen interest in such devices, as they expect these to become a big profitable market. A 2018 report published by *MarketsandMarkets* [27] estimated that in 2025 the collaborative robot market will be worth more than $12.000 Million. This expectation led many companies to pursue the Cobots market and many major players have presented their solutions.

The interest from such companies is justified by the interest of end users and the population in general. Several surveys [19, 38] demonstrates that, despite the reality of this technology being far away from day-to-day use, there is an acceptability and, in fact, a demonstrated necessity in the general population for such kind of technology.

From this necessity, allied with the drop in price of most of the technologies needed, one can easily imagine the development in the near future of several devices that would complement humans into tasks more general, more abstract and less structured than the ones Cobots perform today. Independent of which task we are previewing, in order to achieve good performance, there is a set of operations of paramount importance: such a device must be capable of accurately depict the outside world. That is, any robot talked above must be capable of extracting the location of objects from an input sensory system, for instance a video feed, and be able to classify its nature, so that the object can be moved to a designated box, for example. This set of operations is the main focus of this work.

Provided the circumstances, namely that there are several generic objects in an image and the main

goals are to extract information related to its location as well as categorize such objects into classes, this clearly takes us to the computer vision task of object recognition. Andreopoulos and Tsotsos [2] state that the recognition problem denotes the general problem of identifying all the objects present in the image and providing accurate location information of the respective objects. Russakovsky at al. [32] use the term object recognition broadly to encompass both image classification, a task requiring an algorithm to determine what object classes are present in the image, as well as object detection, a task requiring an algorithm to localize, as in restrict to a particular place, all objects present in the image. Szeliski [36] defines object detection as categorizing not just whole images but delineate (with bounding boxes) where various objects are located. This shows a great similarity with the previous definitions of object recognition, reason for which the terms are used interchangeably.

The tasks of identifying, locating and categorizing objects are among the earliest in the field of Computer Vision. In fact, one of the initial drives of the field was to artificially recreate this process, which comes effortlessly to virtually any animal and certainly to intelligent animals. However, the biggest revolution for the task of object recognition appeared in the 2012 edition of the ImageNet challenge [32]. A team from the University of Toronto enter a convolutional neural network (CNN) model [20], and won the competition with a major error-rate drop from previous years. Since 2012, all the winners have used CNN models and error-rates have steadily dropped to a very small percentage, being on par or even surpassing human capacities.

Hence, from 2012 onward, it has been common practice and, in fact, the leading strategy to tackle problems such as Image Classification and Object Detection to use some form of algorithm with a CNN backbone. For that reason, this will be the strategy used for this work.

2. Background

Convolutional Networks are defined by Goodfellow at al. [14] as a specialized kind of neural network for processing data that has a known, grid-like, topology and employ a mathematical operation called *convolution*, which is a specialized kind of linear operation. In other words, convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers.

As was referred previously, they have been one of the vital tools in Object Detection. Several architectures - Alexnet [20], VGG [33], GoogLeNet [35], ResNet [15], Darknet [29], among others - have proven to be fundamental feature extractors for the task.

2.1. Object Detection Methods

Having these networks as reference, there is a pretty elementar solution to the task of object recognition. Called the sliding window object localization, it consists of running a classifier function (CNN or not) over many rectangular subregions of an image and taking its maximum as the object's location. However, this is evidently an inefficient method, because objects can appear in several locations, sizes or aspect ratios, which leads to an enormous quantity of regions needed to be evaluated.

Consequently, several methods emerged to tackle this issue. These methods can generally be categorized into two types [39]:

- One follows a more traditional object detection pipeline: generating candidates, region proposals, at first and then proceeding to classify each proposal. The region proposal stage usually makes use of one of several different possible algorithms to generate regions of interest (RoIs) to be analyzed, like selective search [37], objectness [1] or fully connected networks. It regularly is the bottleneck of the method. The region proposal based methods may include R-CNN [13], Fast R-CNN [12], Faster R-CNN [31], R-FCN [9] and FPN [24].
- The second adopts a unified framework to achieve classification and localization directly, regarding object detection as a regression and classification problem. Examples of this approach include MultiBox [11], YOLO [29] and SSD [26].

Table 1 shows a comparison of the methods, from the results obtained by Zhao et al. [39].

Method	mAP(%)	Rate (FPS)
R-CNN	66.0	0.03
Fast R-CNN	66.9	0.6
Faster R-CNN (VGG16)	73.2	9.1
Faster R-CNN (ResNet101)	**83.8**	0.4
YOLO	63.4	**46**
SSD300	74.3	**45**
SSD512	76.8	19
YOLOv2	78.6	40

Table 1: Object Detection methods: comparison of performance on VOC07 test set (Zhao et al. 2018).

From the analysis of Table 1 it becomes pretty clear the general advantages and disadvantages of

both types of methods. The methods that have a candidate generation stage are generally more accurate than the single shot methods, however that accuracy comes at the expense of speed. When considering an application of object detection on video feeds, both these parameters are determinant to the success of the application and their relative importance should be evaluated in a case by case basis.

2.2. Transfer Learning

In practice, very few people train an entire Convolutional Network from scratch as it requires large quantities of data to perform well and a training process that takes an incredibly large amount of time. Instead, it is common to pretrain a CNN on a very large dataset, such as ImageNet or MSCOCO, and then use it either as an initialization or a fixed feature extractor for the task to be implemented. In fact, all of the methods described above make use of this pretrained networks to implement the task of object detection. This process is known as Transfer Learning. Goodfellow et al. [14] define Transfer Learning as the situation where what has been learned in one setting is exploited to improve generalization in another setting. The same authors give the following example: "we may learn about one set of visual categories, such as cats and dogs, in the first setting, then learn about a different set of visual categories, such as ants and wasps, in the second setting. If there is significantly more data in the first setting, then that may help to learn representations that are useful to quickly generalize from only very few examples drawn from the second setting". When training an object detector, it is extremely common to use Transfer Learning. Several libraries and frameworks provide *Model Zoos* with several example models, which can be fine-tuned to perform a specific task. This will be the procedure adopted for this work.

2.3. Faster R-CNN

Faster R-CNN was proposed in 2016 by Ren et al. [31]. It eliminates the selective search algorithm of previous algorithms from the RCNN family and lets the network learn the region proposals. Similar to Fast R-CNN, the image is provided as an input to a convolutional network which generates a convolutional feature map. Instead of using selective search algorithm on the feature map to identify the region proposals, a separate network is used to predict the region proposals - the Region Proposal Network (RPN). This lowers the region proposal time from 2 seconds per image, to about 10 ms, while it also allows the region proposal stage to share layers with the following detection stages, causing an overall improvement in feature representation. The Region Proposal Network also makes the use of a set of Anchors. For every point in the output feature map, the network has to learn whether an object is present in the input image at its corresponding location and estimate its size. This is done by placing a set of Anchors on the input image for each location on the output feature map of the backbone network. These anchors are a set of rectangular shapes in various sizes and aspect ratios at this location. The RPN is divided in two branches: the regression branch and the classification branch. The former outputs the 4 regression coefficients used to improve the coordinates of the anchors boxes that contain objects as done in Fast R-CNN. The latter gives the probabilities of whether or not each point of the feature map contains an object within all 9 of the anchors at that point. Overall, Faster R-CNN can be thought as the RPN as a region proposal algorithm and Fast R-CNN as a detector network.

This work makes use of the Faster R-CNN algorithm, seen as it is the most improved and faster of the R-CNN family. It is also important to note it achieve the best results in terms of accuracy when compared to others methods.

2.4. YOLO: You Only Look Once

You Only Look Once (YOLO) is a general purposed object detection algorithm that frames the task as a unified problem, using the same deep neural network to predict bounding boxes and class probabilities [29]. YOLO was the first of the deep learning based algorithms to achieve real time object detection. YOLO divides the input image into a $S \times S$ grid. Each grid cell is responsible for the prediction of B bounding boxes and the confidence scores for those boxes. From the second version onwards, the use of anchors boxes is also an important feature of the algorithm.

To overcome the disadvantages of YOLOv2, YOLOv3 [30] was published in 2018. There were changes made, which account for a small but important improvement when comparing with YOLOv2,in terms of accuracy. This version can be considered as an incremental improvement to the works published before. The changes of this version are: changes in the loss function computation, more specifically, to the objectness scores of the loss function; changes in class prediction, that is, change from the use of softmax classifiers to independent logistic classifiers, because the former impose the assumption that each box has exactly one class; prediction across 3 scales; new feature extractor network, Darknet-53, constituted by 53 convolutional layers and no pooling layers.

This work makes use of YOLOv3 because it is the last stable version of the YOLO family of algorithms and it has achieve state-of-the-art results in terms of speed, while approaching the state-of-the-art results in terms of accuracy.

2.5. SSD: Single Shot MultiBox Detector

Introduced in 2016 by Liu et al. [26], the Single Shot Detector (SSD) is a single stage method for object detection, just like YOLO, reportedly bringing a great improvement in accuracy for such class of methods. This is fairly important because the greater speed of single shot methods is many times obtained at the expense of accuracy. The algorithm uses a pretty simple and straightforward framework. It discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. The use of this default bounding boxes is similar to the uses of anchors in Faster R-CNN, however they are applied on several feature maps of different resolution, that is, on different stages of the CNN. Instead of the fully-connected networks, convolutional layers are added, allowing prediction of detections at multiple scales. Each added feature layer can produce a fixed set of detection predictions using a set of convolutional filters. For a feature layer of size $m \times n$ with p channels, the basic element for predicting parameters of a potential detection is a $3 \times 3 \times p$ kernel that produces either a score for a category, or a shape offset relative to the default box coordinates. At each feature map cell, the model predicts the offsets relative to the default box shapes in the cell, as well as the per-class scores that indicate the presence of a class instance in each of those boxes. Specifically, for each box out of k at a given location, it computes c class scores and the 4 offsets relative to the original default box shape. This results in a total of $(c + 4)k$ filters that are applied around each location in the feature map, yielding $(c + 4)kmn$ outputs for a $m \times n$ feature map.

This works makes use of this algorithm because it achieved results very closely matched with theones obtained by the YOLO algorithm, to an extend that is difficult to separate them in terms of performance

3. Implementation

The implementation can be separated into two distinct phases corresponding to two different environments and groups of objects to be detected.The first phase intends to mirror an household or office environment, more specifically a desk with everyday items that are used in these environments, such as pens, cellphones, mugs, bottles, among others. In this stage, the images and annotations used for training were downloaded from an online dataset, in order to evaluate if the quantity and quality of such open sourced data is enough to obtain a working model. The second phase corresponds to a simulated electronics laboratory or workshop, where the detection is intended for objects such as capacitors, resistors or potentiometers. The images used at this point, were downloaded from standard searches on the internet and manually annotated by the author. This was done in order to determined if an untrained subject could provide quality data to train the models.

All the models were implemented in the Python programming language. All the discussed models were trained, except otherwise stated, until the loss stagnated. Training stoppage was forced manually by the author, that is, no automatic stoppage criteria was programmed. Furthermore, all the hyperparameters of training were the default parameters of the frameworks used.

3.1. Tensorflow API and Ultralytics repository

TensorFlow Object Detection Application Programming Interface (Tensorflow API) [16] is an open source framework built on top of TensorFlow that aims to make it easy to construct, train and deploy object detection models. Together with the TensorFlow Model Zoo, TensorFlow Object Detection API provides the user with multiple pre-trained object detection models with instructions and example codes for fine-tuning and transfer learning.

In this work, a Faster R-CNN model and a SSD model, both with a backbone convolutional architecture using the Inception module and both pretrained on MSCOCO [25], were selected from the Model Zoo and trained using the Tensorflow API.

While the SSD and Faster R-CNN models were possible to be trained using the Tensorflow API, the same cannot be said for the YOLO models. When YOLO was proposed, the author also released a specific open-sourced framework written in C, *Darknet*, in order to facilitate the training of such models. However, seen as C is not as user-friendly of a programming language as Python, several translators were built, generally with compatibility with Tensorflow or PyTorch. These libraries are very useful, seen as, they enable Transfer Learning from the pretrained models in huge datasets. This work makes use of one of those libraries [18], with compabilbity with PyTorch.

3.2. The Open Images Dataset

The Open Images Dataset V4 [23] was launched in 2018, by a team working in Google AI. For object detection, it contains 15.4 Million bounding boxes for 600 object classes, annotated on 1.9 Million images, which, the authors state, is 15 times more bounding boxes than the next largest datasets.

One of the major advantages of this dataset has to do with the way it was created. The images it contains were collected from *Flickr*, a popular images and video hosting site, without any predefined list of classes or tags. The images that appeared somewhere else on the internet were then removed, in order to reduce the bias towards web imagery. This is greatly beneficial when training

an object detection algorithm because the images are presented in a natural ambient, often in cluttered environments, easily verified by its average of 8 annotated objects per image.

When analysing the quality of the dataset the authors report quite good results for the dataset.The precision and recall for the analysed images are respectively 97.7% and 98.2%, both very high values. Furthermore, the authors also analysed the quality of the bounding boxes, reporting an IoU of 0,87 when evaluating the geometric agreement of the boxes, that is, the IoU of two boxes drawn of the same object by two different annotators. In light of this, the Open Images Dataset was chosen and OIDv4 Toolkit [76] was used when downloading the images

3.3. Models trained on 1 class

The order to test the feasibility of training an object detector and get a feeling for some of the conditions needed for the train process, for example the number of images and the time needed for the process, the first practical application tested was training a model on just one class. Such a class was chosen considering it is an object of everyday, intensive and widespread use. It was also needed to take into consideration both the scenarios and the environments that serve as reference for this report. Several classes of the Open Images Dataset were considered, finally settling on the *Pen* class. In the used dataset, 198 images were available for such a class, annotated with 293 bounding boxes, which were divided into training and testing partitions.

3.4. Models trained on 4 classes

To further the knowledge on training and to analyze the impact of having multiple classes on a trained model, a second set of models were trained to detect four classes. Besides the class already chosen before, the new classes chosen were: *Glasses* (as in vision-aiding glasses), *Mobile phone* and *Mug*. These classes were chosen because they are commonly found items on a office or home tables and, combined with that, they have a comprehensive set of images in the Open Images Dataset. Hence, the training partition entailed 1375 images and the testing partition got 381 images, with 1805 and 475 bounding boxes, respectively.

3.5. Models trained on 10 classes

The final set of models for the first phase were trained on ten different classes. Following the criteria attended previously, the six classes added were: *Book*, *Bottle*, *Coin*, *Headphones*, *Human Hand* and *Tin Can*. Despite not being an object found frequently on a table, the *Human Hand* class was also chosen because of the possible real-time in-frame movement of objects by an human subject.

In order to investigate the impact of class imbalance, the decision was made to use three different datasets. The first was composed by all the images and bounding boxes available for the classes in the Open Images Dataset. On the second, images were randomly selected so that their number for each class in the train and test partition were about the same. On the third, mages were randomly selected so that the number of bounding boxes for each class in the train and test partition was about the same. The number of images and bounding boxes in the training and testing partitions for each dataset can be seen in Table 2.

Dataset	Images		BB	
	Train	Test	Train	Test
(1)	3785	946	9917	2514
(2)	1497	472	3389	1188
(3)	1284	326	2282	556

Table 2: Number of images and bounding boxes on each set used to train the 10 classes models.

3.6. Models trained with manually annotated images

There were several purposes in training models with manually annotated images. The first objective is to determine if it is possible to obtain a set of images and annotations, from a source other than a purposely curated database, good enough to train a workable model, seen as such databases are very rare and possess a relative small number of classes. The second purpose, related to the main objective of this work, is to reflect on and check if it would be possible for a program to be coded, that would allow training "at home", that is, the training of a model by an untrained end user of a possible product.

The choice of classes and general environment to be simulated were also object of consideration. The chosen classes were: *capacitor*, *potentiometer* and *resistor*. These point out to an electronics related environment such as an electronics workshop or an electronics laboratory. This invalidates the workflow of the previous phase. The workflow of this phase can be encapsulated into three stages: obtaining and selecting images, annotating the images and training the models.

Obtaining and selecting images: The images used in this phase were obtained through several query searches on *Google Images*. The images were then downloaded in bundle, that is, all the resulting images were downloaded, resulting in a total of 1175 images for the *capacitor* class, 1061 images for the *potentiometer* class and 1333 images for the *resistor* class. Next, the images had to be filtered, that is, a selection of the useful images had to

be made. Images such as drawings, animations, schematics, graphics, pages from technical documentation, table duplicate images were removed. Also removed, were images too strenuous to annotate, images where the objects were not visible and images that didn't correspond to the intended search. The resulting dataset from this process was comprised of 320 images for the *capacitor* class, 273 images for *potentiometer* and 313 for *resistor*.

Annotating the images: The resulting dataset had to be annotated, that is, indicate the location of the objects in each image by means of bounding boxes. A program was produced, where the annotators indicated the top-left and bottom-right corners of the relevant objects in the image, using mouse clicks, and the classes of such objects, by the ways of keyboard selection. After a final review of the annotated objects and acceptance of the results, the program produced two annotations files accordingly to the specifications of the frameworks later used for training - a *.txt* file for YOLO training and a *.csv* file for the Tensorflow API. It is important to denote that the annotation process is quite strenuous. It took two annotators a combined time of about 25 hours to produce the annotations for the 906 images of the dataset.

Training the models: In order to further investigate the impact of class imbalance, the decision was made to use two different datasets. The first one was comprised of all the collected and selected images. The second one was a balanced dataset, where images were randomly selected so that the number of bounding boxes for each class was the same.

4. Results

Taking into consideration that the main goal of this work is to be able to perform real-time or near real-time object detection on video feeds, several sample videos were taken, in order to evaluate the resulting models. The videos were capture with a simple webcam of model SelecLine PPW-10, with definition 640×480. The purpose of using such a device is to evaluate the possibility of adapting the trained models to a household item, readily available at a low price. These videos were implicitly different in factors such as: illumination, distance to the objects intended to be identified, background color, the pose of the objects to be detected, the presence or absence of movement, the area of the frame where objects were present or occlusion of the objects to be detected. From these videos, several frames were then randomly selected. These were then manually annotated in order to obtain the ground-truth bounding boxes. The comparison of such bounding boxes with the ones predicted by each model enables the computation of the mAP. FPS was simply measured as the inverse of the time between predictions on the video feed of the webcam. It is important to denote that the best result possible for this metric is 30 FPS, seen as this is the rate of capture for the webcam feed.

4.1. Models trained on 1 class

The verification set obtained for these models, using the process described above, was composed by 89 images with 141 ground-truth boxes for the *Pen* class. The obtained results for the models trained in this section can be seen on Table 3.

Model	mAP(%)	Rate (FPS)
Faster R-CNN	**72.20**	6
SSD	12.58	**23**
YOLO	58.14	**23**

Table 3: Results for the models trained on 1 class.

We can see the Faster R-CNN model performs the best in terms of accuracy while being significantly slower when predicting the results. The SSD model matches the YOLO model has the fastest model but performs very poorly in terms of accuracy. Furthermore, the YOLO model performs very well in terms of speed but lags behind the R-CNN model when accuracy is concerned.

Despite good performance, the Faster R-CNN model seems to be sometimes affect by illumination and distance to the object to be detected. Hence, despite some drawbacks, the R-CNN model seems to produce good results, close to the state-of-the-art results.

4.2. Models trained on 4 classes

The composition of the verification set can be seen in Table 4. The obtained results for the models of this section can be seen in Table 5.

Class	Pen	Mug	Phone	Glasses
BB	60	56	45	37

Table 4: Verification set for the 4 classes models.

Model	mAP(%)	Rate (FPS)
Faster R-CNN	**59.36**	6
SSD	16.79	**20**
YOLO	45.99	**20**

Table 5: Results for the models trained on 4 classes.

The accuracy values tend to decrease from the models trained on 1 class, indicating the models perform worse when detecting the classes that aren't *Pen*. For the R-CNN model the speed doesn't seem to be affected when the total number of classes the model can detect is increased, while for the

other models the speed is slightly lower than before. The Faster R-CNN model continues to be the more accurate, while the others continue to lead on speed.

Delving deeper into the computation of mAP for each model, it is possible to see that the presence of the *Mug* class greatly decreases the value of mAP for the models. In fact, if the class was to be disregarded, the value of mAP for the R-CNN model would be equal to 73.19%, higher than the one obtained on Section 4.1. The same happens with the other models. If the 2 worst performing classes were to be disregarded for the YOLO model, its mAP would be 67.32%.

Taking the previous information into consideration, it is visible the R-CNN model is again the best performing model and produces good results, specially if we exclude the worst performing class

4.3. Models trained on 10 classes

The composition of the verification set used for this phase can be seen in Tabel 6. Equally, the obtained results for the models can be seen in Table 7. The references (1), (2) and (3) correspond, respectively, to the models trained with an imbalanced dataset, a dataset balanced in the number of images and a dataset balanced in the number of bounding boxes.

Class	Pen	Mug	Phone	Glasses
BB	55	43	47	47
Class	Coin	Headphones	Bottle	Book
BB	58	49	45	44
Class	Tin Can		Human Hand	
BB	47		43	

Table 6: Verification set for the 10 classes models.

Model	mAP(%)			Rate (FPS)
	(1)	(2)	(3)	
R-CNN	**38.44**	33.66	20.99	6
SSD	11.00	6.83	4.88	19
YOLO	26.68	18.50	22.50	**20**

Table 7: Results for the models trained on 10 classes.

The increase in the number of classes to detect seems to have an effect on the accuracy of the models. Analyzing the detections, the problem looks to be located on images with cluttered environments, that is, with several objects in an image. In this kind of image, the models seem to correctly predict one or two classes, but fail to make predictions on the remaining objects of the images. This appears to be confirmed by the accuracy results when the verification set is reduced to images with 1 or 2 objects. These results are shown in Table 8. The table clearly shows the models perform much better with less objects to predict per image.

Model	mAP(%)		
	(1)	(2)	(3)
Faster R-CNN	**58.54**	47.25	30.50
SSD	18.03	4.37	7.37
YOLO	44.32	19.34	25.81

Table 8: Results for the models trained on 10 classes using images with 2 objects or less.

A consideration that is possible to take for the results relates to the balancing of the training dataset. It is quite clearly that the efforts made in that area didn't produce good results. However,the effect of the balancing isn't straightforward. Although, this effect isn't equal for all the classes, the classes with less examples in dataset (1) seem to be benefited by balancing while the ones with the most examples seem to be hindered.

We'll explore the best performing model, the Faster R-CNN model trained with dataset (1). As is the case in Section 4.2 if we exclude the worst performing class there is an increase in accuracy. The obtained mAP would be 42.07% for the entire verification dataset and 65.05% for the smaller verification dataset. If we go further and eliminate also the influence of the second worst performing class, the value for the second verification set would be 69.61%. Although not state-of-the-art, this value approaches those reported.

4.4. Models trained with manually annotated images

As stated in Section 3.6 two dataset - balanced and imbalanced - were produced for training models in this phase of the work. The general results obtained for the consequently 6 models can be seen in Table 10. Furthermore the composition of the verification set by class can be seen in Table 9.

Class	Resistor	Capacitor	Potentiometer
BB	113	102	82

Table 9: Verification set for the manually annotated models.

The Faster R-CNN model continues to be the best performing model in terms of accuracy, while the others have an edge when speed is concerned. The balancing of the dataset seems to produce either virtually no difference or a negative one, when talking about accuracy. The performance of the models, in terms of accuracy, greatly decreases from

Model	mAP(%)		Rate (FPS)
	Balanced	Imbalanced	
R-CNN	36.83	**37.50**	6
SSD	11.01	4.68	**20**
YOLO	19.52	24.69	**20**

Table 10: Results for the models trained with manually annotated images.

the models of previous sections and from the results reported for each algorithms in the literature. The dip in performance may be due to several factors. First, it was expected that the images used in this phase were not as appropriate for the task of object detection as images from a curated dataset. Second, the size of the objects in the verification set is quite small, which leads to a a big difference between the images on the dataset used for training and the images on the verification set.

5. Conclusions

Regarding the main general objective, the study of the input sensory system of a tabletop pick and place robot, we can see from the work developed that it is possible to devise a general system capable of recognizing to a good extend everyday objects and possible to be applied to commercial use. Comparing with other solutions published [21, 22], the accuracy metric on those solutions is better. However, such solutions, and by consequence their verification sets, are designed for very limited scenarios. In this work, the purpose of the product is more general and, despite lower accuracy, the solution found can be satisfactory. However, it is fairly clear that such a product has several limitation and should be design with those limitations in mind.

The first of the constraints to be considered is the fact that it is pretty explicit that some classes of objects are more suitable to this purpose than others. For instance, small objects seem to be harder to identify and locate than others of superior dimensions. It is clear that the training examples for the image recognition algorithms can't be indiscriminate and dissociated from the objective of the system to be projected.

The second constraint relates to the detection of objects in cluttered environments. In such environments, the algorithms suffer a dip in accuracy performance, which means that any device using these algorithms should be design having that in consideration. That is, it should avert situations where the objects are closely together. Alternatively, it could also take advantage from the fact that, despite having a dip in performance, the detector seems to still predict correctly some of the object in the image.

The third consideration necessary is the fact that the best performing algorithms are affected by some hindering factors. Low illumination can sometimes decrease the performance of the algorithms. Further study is necessary to determine how much it contributes to such decrease. The distance to the objects to be identified also can be a problem, but seems to only affect small objects as previously discussed. Background color also appears to be a factor with importance but its effects were only seen in limited cases.

Relating to the specific situations studied in this work, it can be concluded that the choice of algorithm to perform object detection should fall onto Faster R-CNN when accuracy is the top requirement for a situation. On the other hand, if speed is preferred, the YOLO algorithm should be favored. Regarding the SSD algorithm, the results obtained are very distant, in terms of accuracy, from the ones reported by the authors and by other sources. This could mean that a series of blunders may have affected the training of such models, affecting the results obtained.

Another conclusion regarding the implementation of this work concerns the balancing of the training datasets. It is possible to comprehend that the balancing of the datasets in this work wasn't successful and, in fact, lowered the performance of the models. The most probable cause of such a situation was the fact that the method used for the balancing, subsampling, may not be the most appropriate for this task. Other methods such as oversampling, data augmentation or SMOTE [8] can be used to test the effects of data unbalancing and understand if it is possible to improve performance.

Regarding the possible annotation of images by an end user, it may be fair to say that such a process would be very difficult to implement. That is justified by two arguments. The biggest hurdle to such a process would be the difficulties that would be presented to an untrained annotator. As is referred, the annotating process is very time consuming and tedious. Furthermore, it is abundantly explicit throughout this work that some images are more appropriate to train a model than other. As an end user wouldn't be alert to this fact, there is no guarantee that he images chosen for training would be suitable.

Finally, it is important to mention that the models studied in this work can also be applied to many other situations. This unspecialization and possibility for general use is one the best feature of the object detection algorithms studied.

5.1. Future Work

In the future, it is essential to increase the capacity of the implemented models by expanding the database used for training and the number of classes

they are able to recognize. It is also important, the study of future or recently published algorithms for object detection such as YOLOv4 [4], YOLOv5 or End-to-End Object Detection with Transformers [7]. Equally, it should be important to accompany developments in the databases with object detections tasks. Regarding the applications related to pick and place robots, it would also be important to add information on the depth aspect of images. Using cameras with such capabilities, it may be important for this task to perform object detection on RGB-D images. Regarding the device suggested in this work, it would be beneficial to evaluate the possibility of implementing the studied algorithms in an embedded system or a single board computer such as a Raspberry Pi or the more specialized NVIDIA Jetson, Google Coral or Khadas VIM series. It is also possible to imagine a semi-automatic process of training for unknown objects in a scene, where a model would provide the location information, maybe following recent works in class agnostic object detection [17], and the user would provide the name for the class.

Acknowledgements
The author would like to thank Prof. Mário António Ramalho, Miguel Martins, Gabriel Nunes, all his friends and family and, specially, his parents.

References

[1] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2189–2202, Nov 2012.

[2] A. Andreopoulos and J. K. Tsotsos. 50 years of object recognition, directions forward. *Computer Vision and Image Understanding*, 117(8):827–891, August 2013.

[3] C. A. Avizzano. *Human-Robot Interactions in Future Military Operations*. Routledge, 2016.

[4] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao. Yolov4: Optimal speed and accuracy of object detection, 2020.

[5] J. Broekens, M. Heerink, and H. Rosendal. Assistive social robots in elderly care: a review. *Gerontechnology*, pages 94–103, 2009.

[6] K. Čapek. *R.U.R. - Rossum's Universal Robots*. Aventinum, 1920.

[7] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers, 2020.

[8] N. Chawla, K. Bowyer, L. Hall, and W. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *J. Artif. Intell. Res. (JAIR)*, 16:321–357, 06 2002.

[9] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. In *NIPS*, 2016.

[10] A. de Santis, B. Siciliano, A. de Luca, and A. Bicchi. An atlas of physical human–robot interaction. *Mechanism and Machine Theory*, 43(3):253–270, March 2008.

[11] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2155–2162, 2014.

[12] R. B. Girshick. Fast r-cnn. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 04 2015.

[13] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[14] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016.

[16] J. Huang, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. 11 2016.

[17] A. Jaiswal, Y. Wu, P. Natarajan, and P. Natarajan. Class-agnostic object detection, 2020.

[18] G. Jocher, Y. Kwon, guigarfr, perry0418, J. Veitch-Michaelis, Ttayu, D. Suess, F. Baltacı, G. Bianconi, IlyaOvodov, Marc, C. Lee, D. Kendall, Falak, F. Reveriano, FuLin, GoogleWiki, J. Nataprawira, J. Hu, LinCoce, LukeAI, N. Zarrabi, O. Reda, P. Skalski, S. Song, T. Havlik, T. M. Shead, and W. Xinyu. ultralytics/yolov3: v8 - Final Darknet Compatible Release, Nov. 2020.

[19] Z. A. Khan. Attitudes towards intelligent service robots. S-100 44 STOCKHOLM, Sweden, Aug 1998. Royal Institure of Technology (KTH).

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012.

[21] R. Kumar, S. Lal, S. Kumar, and P. Chand. Object detection and recognition for a pick and place robot. In *Asia-Pacific World Congress on Computer Science and Engineering*, pages 1–7, 2014.

[22] S. Kumar, A. Majumder, S. Dutta, R. Raja, S. Jotawar, A. Kumar, M. Soni, V. Raju, O. Kundu, E. Behera, K. Venkatesh, and R. Sinha. Design and development of an automated robotic pick & stow system for an e-commerce warehouse. 03 2017.

[23] A. Kuznetsova, H. Rom, N. Alldrin, J. R. R. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, T. Duerig, and V. Ferrari. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *ArXiv*, abs/1811.00982, 2018.

[24] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature pyramid networks for object detection. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, 2017.

[25] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. volume 8693, 04 2014.

[26] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016.

[27] MarketsandMarkets. Collaborative robot market size, growth, trend and forecast to 2025, Oct 2018.

[28] R. R. Murphy. Human-robot interaction in rescue robotics. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 34(2):138–154, May 2004.

[29] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[30] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *ArXiv*, abs/1804.02767, 2018.

[31] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:1137–1149, 06 2015.

[32] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[33] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, arXiv 1409.1556, 09 2014.

[34] N. Syll, V. Bonnet, F. Colledani, and P. Fraisse. Ergonomic contribution of able exoskeleton in automotive industry. *International Journal of Industrial Ergonomics*, 44(4):475–481, July 2004.

[35] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[36] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag, Berlin, Heidelberg, 1st edition, 2010.

[37] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104:154–171, 2013.

[38] M. Wongphati, Y. Matsuda, H. Osawa, and M. Imai. Where do you want to use a robotic arm? and what do you want from the robot? In *Proceedings - IEEE International Workshop on Robot and Human Interactive Communication*, pages 322–327, 09 2012.

[39] Z.-Q. Zhao, P. Zheng, S. tao Xu, and X. Wu. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 2018.