AUTOMATIC SHARED E-MOPED MIRROR PRESENCE DETECTION USING DEEP OBJECT DETECTORS

LIZET PAOLA HERNANDEZ VARGAS

THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN DATA SCIENCE & SOCIETY
AT THE SCHOOL OF HUMANITIES AND DIGITAL SCIENCES
OF TILBURG UNIVERSITY

TOTAL NUMBER OF WORDS: 7933

STUDENT NUMBER

2061326 / u370066

COMMITTEE

dr. I. Önal
dr. R. G. Alhama

LOCATION

Tilburg University
School of Humanities and Digital Sciences
Department of Cognitive Science &
Artificial Intelligence
Tilburg, The Netherlands

DATE

January 13, 2022

DISCLAIMER

This thesis does not contain any studies with human participants or animals performed by the author. Data used in this study were previously collected. The original owner of the data retains ownership of the data during and after the completion of this thesis.

ACKNOWLEDGMENTS

I would like to firstly thank felyx sharing B.V. for giving me the opportunity to work on this fun and exciting project. I especially thank Annanina Koster for her always friendly guidance and patience, Hein Harlaar for his accurate advice, and Andi Aliko for his support and inspiration. Furthermore, I want to express gratitude to my thesis supervisor, dr—Itir Önal, for her help and direction through the entire process of this project. Lastly, I would like to thank my parents Fredy and Oliva, my brother Deyvit, my partner Jack, the Huijgen family, and my friends for always motivating me in these strange times.

Mil gracias,
Lizet Paola Hernandez Vargas

AUTOMATIC SHARED E-MOPED MIRROR PRESENCE DETECTION USING DEEP OBJECT DETECTORS

LIZET PAOLA HERNANDEZ VARGAS

Abstract

Safety issues occupy a significant position on the agenda when discussing the threats to society of last-mile transportation means, such as e-mopeds. Therefore, this project aims towards the automatic visual detection of e-moped damages that could endanger drivers. More specifically, it focuses on assessing the efficiency of deep neural network-based object detectors when assessing the presence of rear-view mirrors on e-mopeds from customer-taken images, since it is a mandatory safety regulation to travel. Little research has been done in this area; however, generic object detectors have proven good results when used on scooter/moped parts recognition and localization problems. This study aimed to establish a foundation for future exploration on this topic. The Faster R-CNN and YOLOv5 models were used, comparing both when trained under fully-supervised and semi-supervised settings. Furthermore, the performance of fine-tuned Faster R-CNN and fine-tuned YOLOv5 was compared when trained with different amounts of manually-labeled data. It was found that the best-performing model under fully-supervised consisted of fine-tuned YOLOv5; however, the performance gap between models was not extensive. Lastly, the semi-supervised self-training technique did not show further favorable results for the e-moped rear-view mirror detection problem.

1 INTRODUCTION

Detecting damages on the shared E-mopeds is a process that is currently done through damage report submissions made by either Scooter Support Agents (SSAs) or customers. Given the nature of the process, it is highly time-consuming and prone to error due to fatigue. Additionally, customers tend to wrongly report or not report at all the damages on the mopeds, leading to unnecessary actions over them (e.g., putting the e-mopeds out of service) or even risking other riders' safety. This thesis project

aims towards the automatic visual detection of e-moped damages through images. More specifically, it focuses on assessing the efficiency of deep neural networks when detecting e-moped damages. For the particular case of this study the damage of interest is the presence/absence of rear-view mirrors. For this purpose, a dataset of 86,400 non-labeled images of parked e-mopeds provided by Felyx Sharing B.V. was pre-processed and divided into train, validation, and test sets that later was used as the input of two different models: (1) one-stage object detector: YOLOv5, (2) two-stage object detector: Faster R-CNN. Their performance results under supervised and semi-supervised conditions were compared based on common evaluation metrics.

Urban transportation currently faces numerous challenges, such as congestion and air pollution, mainly produced by the plethora of private cars on the streets. Therefore, the introduction or transition to other means of transportation that contribute to ameliorating the problems mentioned above has become a priority. Micromobility facilities, also known as first/last-mile transportation systems (Degele et al., 2018), in combination with new sources of energy, provide a partial answer to those issues (Christoforou, Gioldasis, de Bortoli, & Seidowsky, 2021). Shared electronic micromobility facilities such as electronic mopeds (e-mopeds) are among the most promising options introduced to multiple cities. This shared mode of transport does not represent a source of greenhouse gas emissions and does respond to the public's appetite for a cheap, convenient, and flexible way to quickly get around increasingly congested cities (Schellong, Sadek, Schaetzberger, & Barrack, 2021). Yet, with the introduction of this means of transportation, use of public space, vandalism, and safety concerns have appeared. Although the use of public space has caused the biggest debates after the introduction of shared e-mopeds in the cities, safety issues occupy a significant position on the agenda when discussing the threats to society of this transportation mode (Gössling, 2020).

While the regularization of the mandatory use of helmets to ensure riders safety has been in the spotlight since the introduction of the vehicles (Gauquelin, 2021) (*Helm wordt vanaf juli volgend jaar verplicht voor snorfietsers | RTL Nieuws*, n.d.), the precedent of mandatory existence of rear-view mirrors as a safety measure, explicitly the left-side rear-view mirror, is already required by law (Gössling, 2020) (*wetten.nl - Regeling - Regeling voertuigen - BWBR0025798*, n.d.). According to felyx internal data, numerous reports related to missing or damaged rear-view mirrors are made daily. However, these damages are not reported in their totality since it is not a mandatory step to ride the e-mopeds. To tackle this issue, the automatic detection of the rear-view mirrors would help to reinforce shared e-mopeds riders' safety by supporting the service providers' reparation operations.

From a business perspective, the automation of the damages detection, as missing rear-view mirrors, reduces operations time, optimizes the mopeds reparation process, and helps to enhance the image of the product and the company itself. For the particular case of felyx, this automatic detection can be carried utilizing Deep Neural Networks together with the obligatory pictures submitted by customers when parking the vehicles.

Visual vehicle's part detection is encompassed by the object detection problem, and as a fundamental computer vision task, it has been widely researched (Wu, Sahoo, & Hoi, 2020). Among the multiple real-world applications of object detection, vehicles' visual part detection has only been applied to cars and helmets (Prajwal, Tejas, Varshad, Murgod, & Shashidhar, 2019). For the vehicle detection and vehicles' parts detection the most commonly used and compared methods are the two-stage and one-stage detectors Faster R-CNN and the YOLO family, respectively Meng, Bao, and Ma (2020) Zaman et al. (2021). This given that each of the detection paradigms presents its own performance advantage. On one hand, two-stage detection algorithms had better detection results in terms of average precision (AP), and on the other hand one-stage detectors are faster and can achieve real-time performance (Wu et al., 2020). Nonetheless, to the best of our knowledge, literature that encompasses the object detection problem through deep neural networks on data related to e-mopeds parts has not been yet researched.

As the nature of the provided data is completely unlabeled and the manual-labeling process is a highly taxing task in terms of time and resources, assessing the amount of labeled data needed to obtain the best possible performance while using the minimum amount of manually-labeled data represents a way to optimize future automatic detection processes of damages with similar characteristics as the one of this study. In the view of the fact that a first appropriate amount of images can be annotated with certainty. In a similar manner, applying semi-supervised learning to problems that have limited annotated data has proved to be a satisfactory approach to foster the data scarcity problem(Van Engelen & Hoos, 2020).

Consequently, this project provides an understanding of domain-specific, e-mopeds rear-view mirrors' presence detection, real-world application of fine-tuned state-of-the-art deep neural network architectures, extensive empirical performance evaluation, and a comparison between the architectures' efficiency. Consequently, the following main research question is defined:

> *How efficient are fine-tuned generic state-of-the-art deep object detectors when identifying e-moped rear-view mirrors presence in images?*

The study addresses the following sub-questions:

RQ1 *To what extend fine-tuned Faster R-CNN and fine-tuned YOLOv5 can detect the presence of e-moped rear-view mirrors?*

RQ2 *How does the amount of labeled data affect the performance of visual object detectors when recognizing e-moped rear-view mirrors in images?*

RQ3 *How does the use of semi-supervised learning affect the performance of fine-tuned Faster RCNN and YOLOv5 object detectors?*

The organization of this document is as follows: First, the related work is explained in Section 2. Section 3 provides further details about the models and algorithms used, followed by the experimental setup of this study in Section 4. The results of the experiments are described in Section 5 and discussed in detail in Section 6. Finally, some conclusions are drawn in Section 7.

2 BACKGROUND

Since the late 1960's numerous research and applications in the computer vision field have been developed. Applications such as autonomous vehicle navigation, robotic vision, simultaneous visual localization, instance segmentation, and object detection have revolutionized industries throughout the whole spectrum (Kakani, Nguyen, Kumar, Kim, & Pasupuleti, 2020). The damage identification process has also taken advantage of the remarkable development of this field by utilizing object detection and instance segmentation solutions to visually identify the presence of flaws. In this section, the evolution of methods used for object detection and its application on automobile parts detection will be presented, followed by the introduction of existing semi-supervised learning approaches used to enhance object detectors performance.

Object detection aims to identify the location of objects and subsequently differentiate them into a specific class using computational models. Depending on their application, object detectors can be divided into general and domain-specific. The former application focuses on simulating the human vision while the latter target the detection of instances encompass in a particular domain(Zou, Shi, Guo, & Ye, 2019).

The advancement of this technique is recognized to have two main periods: "traditional object detection" and "deep-learning based object detection." The distinction between the two is mainly based on the way the features to detect the objects are extracted, where the "traditional object detectors" relied completely upon handcrafted features and the

"deep-learning based object detectors," as their name implies, utilize deep convolutional neural network (CNN) feature representations (Wu et al., 2020). As for the deep-learning based object detectors, the literature separates them into two paradigms: "two-stage" and "one-stage" object detectors.

In 2014, the first two-stage object detector was proposed by R. Girschik et al. It was an object detector based on region-based convolutional neural networks (RCNN), from there its name. The RCNN model works by starting with a selective search to create a set of object candidate boxes. Those proposed boxes are re-scaled to a fixed size image and used as the input of a pre-trained CNN model producing extracted features as output. The extracted features are fed into linear SVM classifiers to predict the presence of an object in a certain region and to finally categorize the object(Ren, He, Girshick, & Sun, 2016). Even though the accuracy of the RCNN was incomparable at that time, the detection speed was extremely slow given the overlap between the candidate boxes. As solutions to this issue, the Spatial Pyramid Pooling Network (SPPNet), Fast RCNN, and Faster RCNN models were later proposed. SPPNet fixed the necessity of a specific size for the input images (224x224 for AlexNet) by introducing a Spatial Pyramid Pooling layer permitting the generation of a fixed-length representation without re-scaling the images (He, Zhang, Ren, & Sun, 2015). Fast RCNN mended the multi-stage training pipeline issue by enabling the simultaneous training of the detector and bounding box regressor under the same network configurations (Girshick, 2015). Faster RCNN overcame the bottleneck generated by the region proposal computation by introducing a Region Proposal Network (RPN) (Ren et al., 2016).

In 2015, the first one-stage detector, You Only Look Once (YOLO), was presented by R. Joseph et al. Abandoning the paradigm of first proposal detection and next verification, YOLO utilized a neural network to divide the image into regions and simultaneously predict bounding boxes and probabilities for each of those regions(Redmon, Divvala, Girshick, & Farhadi, 2016). The detection speed of the algorithm was incomparable; however, the localization accuracy, especially for small objects, was not high enough to surpass one of the two-stage object detectors. Later versions of this algorithm family tried to overcome this problem(Alex, 2020).

An extensive amount of literature shows the diverse domain-specific and generic applications of Computer Vision's one-stage and two-stage object detectors (Jaffari, Hashmani, Reyes-Aldasoro, Aziz, & Rizvi, 2021)(Wu et al., 2020)(Zou et al., 2019). For the specific case of vehicle detection and vehicles' parts detection, multiple methods have been proposed. X. Zhu, Liu, Zhang, and Duan (2019) proposed a model to identify 31 different car parts based on image segmentation and a Mask R-CNN algorithm (Mask

R-CNN adds an extra branch in to Faster R-CNN, which also predicts segmentation masks for each instance), simultaneously they proposed a damage type and degree framework based on the Inception-v3 architecture integrated with Batch normalization and Convolutional Factorization. Additionally, they used RetinaNet, a one-stage object detector, to identify the presence and position of the vehicles' damages. However, they concluded that algorithms such as YOLO or Faster R-CNN are more suitable options to accomplish this task, given their smaller backbones and the detection complexity. Mhalla, Chateau, Gazzah, and Amara (2018) introduced an embedded system for traffic surveillance. Their system detected and categorized traffic objects under different conditions (day and night conditions) using a modified Faster R-CNN. Given the modifications, replacing the Faster R-CNN four MAX-Pooling layers with Stochastic Pooling layers, the proposed system outperformed generic object detectors with a median improvement of 57%; such improvement is given the capacity of the Stochastic pooling layers to preserve more information.

Meng et al. (2020) provided a comprehensive review of the performance of state-of-the-art generic object detectors together with transfer learning methods when assessing the existence of cars in an image. The visual detectors compared were Faster R-CNN, R-FCN, SSD, YOLOv3, and RetinaNet. Two-stage detection algorithms had better detection results in terms of average precision (AP); nevertheless, their inference time could not achieve real-time performance. From their experiments on the public dataset KITTI Faster R-CNN proved to be the fastest (0.68 FPS), however not the most precise with a 48.37% AP (R-FCN obtained a 66.01% AP) of the two-stage detectors when test on hard examples. Concerning the one-stage detectors, while YOLOv3 and SSD had similar average precision (38.23% AP), RetinaNet was the most precise (68.73% AP) when tested on hard examples, SDD is the fastest among the three of them (14.15 FPS).

As for studies related to motorcycles, scooters, or mopeds parts detection, few efforts have been made, leaving aside the helmet-used visual detection case, which has been broadly evaluated (Lin, Deng, Albers, & Siebert, 2020) (Prajwal et al., 2019) (e Silva, Aires, & de MS Veras, 2018) (Sanjana, Shriya, Vaishnavi, & Ashwini, 2021). It is relevant to mention that all of these methodologies approached the helmet-used detection by first detecting the motorcycle and later the presence of the helmet itself. The methods implemented for the motorcycle detection were mainly YOLOv2, YOLOv3, random forest, multilayer perceptron (MLP), Faster R-CNN, and support vector machine (SVM). Concerning motorcycle visual detection, numerous pieces of literature show different methods to address this process. Zaman et al. (2021) compared the one-stage detector YOLO to Aggregate Channel Features (ACF) and the two-stage detector Faster

R-CNN when detecting motorcycles, and all architectures used ResNet50 as the backbone. Their research showed that YOLO outperforms the other algorithms in precision and inference time when trained on ten epochs and small batches of images. Espinosa, Velastin, and Branch (2018) introduced a deep detector based on Faster-RCNN for motorcycle detection and classification on occluded scenarios. The network achieved results of 75% in average precision (AP) for high occluded scenarios and results of 92% in AP for low occluded ones.

Concerning the issue of limited labeled data and its impact on performance for classification problems, varied literature has shown that semi-supervised learning (SSL) methods tackle this problem (Van Engelen & Hoos, 2020). Semi-supervised learning is a branch of machine learning that combines supervised and unsupervised learning principles to aid the learning process of the algorithm and consequently its performance by utilizing a combination of labeled and unlabeled data during training(X. J. Zhu, 2005).

N. Li and Xia (2018) proved improvement on the affective images classification accuracy by implementing SSL to train a hierarchical classifier. Given the difficulty of obtaining a labeled dataset big enough to obtain good results under a supervised learning training approach, this training method was chosen. Chun and Ryu (2019) proposed a road surface damage detection method based on the SSL self-learning (proxy-labeling) approach to ease the collection of labeled data. The resulted precision of the model train under SSL settings improved compared to that of supervised learning settings. However, the recall experienced a minor decrease. Accomplishing a higher overall F_1-score under SSL settings.

Burton II, Myers, and Rullkoetter (2020) tackled the manual labeling of medical images bottleneck by training 2D and 3D CNNs using a semi-supervised learning framework. Their CNNs trained with SSL settings outperformed those trained using a fully supervised technique and proved competing results compared it to similar literature even though it implemented less labeled data. Weinstein, Marconi, Bohlman, Zare, and White (2019) advanced a method to detect tree-crowns from images by adopting an SSL approach while training a deep learning detection network. Their implementation was compared against an existing Light Detection and Ranging (LIDAR)-based unsupervised technique showing that the model train with the combination of labeled and self-generated labeled data yielded more accurate predictions.

For this study, the state-of-the-art YOLO version 5 and Faster R-CNN algorithms are used given their proven performance and speed when targeting detection of objects encompassed in a similar context as the ones of this research. As for the SSL method, self-training will be implemented

following an incremental approach considering the amount of unlabeled data present in the dataset.

Based on this preliminary literature review, the contribution of this research is to provide an understanding of a fine-tuned state-of-the-art deep object detector architectures domain-specific application, e-mopeds rear-view mirrors' presence detection, along with their empirical performance evaluation under supervised and semi-supervised settings to explore performance enhancement from the unlabeled data, furthermore a comparison between the architectures' efficiency is presented.

3 METHODS

The current section provides a general explanation of the object detector architectures and the semi-supervised learning approach used in the making of this research.

3.1 *Faster R-CNN*

Faster R-CNN, one of the latest improvements of the R-CNN (see section 2.) algorithm proposed by Ren et al. (2016), is composed of two modules, a deep fully convolutional network that proposes regions (RPN) and the Fast R-CNN detector that uses the proposed regions to perform classification of objects. The RPN aims to help the Fast R-CNN detector to know where to direct its attention by ranking region boxes (anchors) and proposing the ones most likely containing objects.

The detection process starts with an image represented as $HeightxWidth\ xDepth$ tensors (multidimensional arrays) that is input to a pre-trained for the classification task CNN (current implementations use ResNet) to obtain a convolutional feature map by using the output of an intermediate layer. The convolutional feature map is the result of the abstractions (edges, patterns in edges, etc.) created by each of the convolutional layers that conform the network, and it has spatial dimensions much smaller than the original image and greater depth. This dimensional reduction has place given the pooling applied in-between convolutional layers, and the greater depth the result of the number of filters that each of the convolutional layers learns.

Afterward, the RPN and the convolutional feature map are used to identify regions containing objects. For this purpose, reference boxes, anchors as Ren et al. (2016) called them, are used to help the network learning to predict offsets from those predefined anchors. These anchors are pre-established bounding boxes, defined as $x_{center}, y_{center}, width, height$, of different sizes and ratios arranged all through the original image. Even

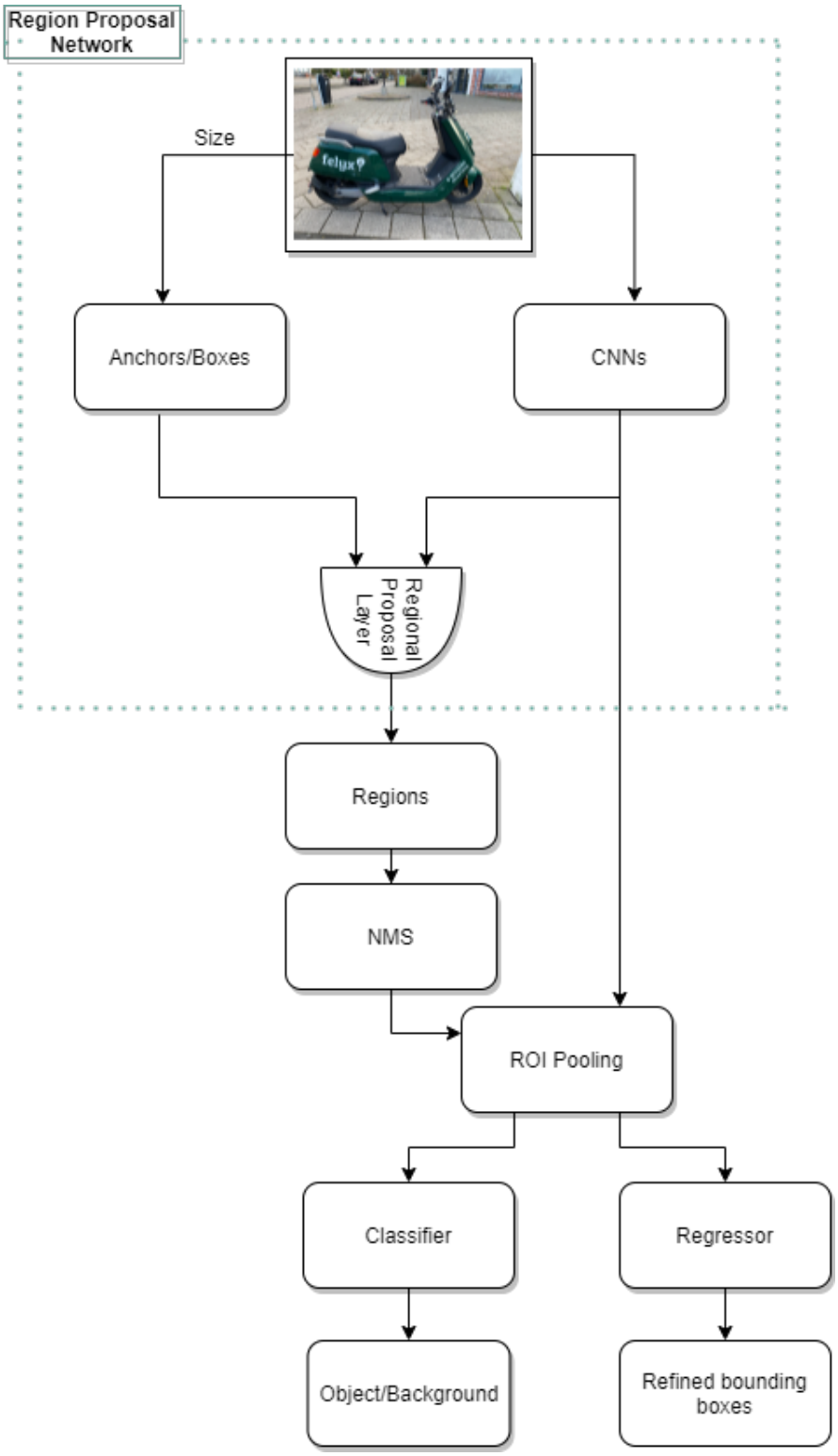


Figure 1: Faster R-CNN architecture.

though the final anchors refer to the original image, these are defined based on the convolutional feature map; therefore, there is a set of anchors for each of the points that compose it. Utilizing these anchors and the RPN, a set of good region proposals for objects is generated by obtaining two different outputs for each of the anchors. The first output is the probability that an anchor does contain an object, called by Ren et al. (2016) objectness score. This score is understood as the likelihood that the anchor contemplates "something" other than the background of the image, i.e., foreground. The second output is the bounding box regression that will improve the fit of the anchors to the object being predicted.

The authors implemented the RPN in a fully convolutional manner, using the convolutional feature map as the input to a convolutional layer with 512 channels and 3x3 kernel size that bifurcates into two parallel convolutional layers using a 1x1 kernel and number of channels depending on the number of anchors per point. The layer that outputs the objectness score requires 2000 channels, and the one that outputs the bounding box adjustments requires 4000 output channels. Post-processing of the proposed regions is necessary after the RPN generates them since proposals end up overlapping over the same object given the overlap of the reference anchors used to generate them.

The method used to solve the duplication of proposals is called Non-Maximum Suppression (NMS), and it works by iterating over a list of proposals sorted based on score and discarding those proposals that have an Intersection over Union (IoU) lower than a predefined threshold but keeping the one that has the highest score. With the not duplicated object proposals generated by the RPN, the location of the objects is known, but their classes still need to be assigned. However, given that classifying each of the 2000 proposals is highly inefficient and slow, Ren et al. (2016) fixed this problem by reusing the previously generated feature map and applying region of interest pooling to it. The purpose of this technique is to perform max pooling on inputs of nonuniform sizes, in this case, the RPN-generated convolutional feature map, to obtain fixed-size feature maps for each object proposal. Such a process is needed since the R-CNN used to classify the objects inside the proposal needs a fixed-size input to output a fixed number of classes.

As previously mentioned, Fast R-CNN is the second and final step of object detection, and it aims to classify proposals as one of the object classes or as background and to adjust in a better way the bounding boxes for the proposals according to the predicted class. The R-CNN flattens the pre-extracted proposals' convolutional feature maps and injects them into two different dense layers to reach those goals. Where one of the dense layers has $C + 1$ units ($C + 1$ being the total number of classes plus

background), and the other has 4C units (4 given the four parameters that defined the bounding boxes)(Ren et al., 2016).

3.2 *YOLO*

The first algorithm of the YOLO family was proposed, as mentioned in section 2., by Redmon et al. (2016) This architecture unified the bounding box and classification of objects tasks by means of a single neural network that considers the entirety of the image and all the objects in it. This results in the possibility of end-to-end training and real-time speed detection.

The detection process starts by dividing the input image into a grid of SxS size. Where if the center of an object is located at a certain grid cell, such cell is the one that will detect that object. Each of the grid cells has as a task to predict a set of B bounding boxes and confidence scores for those boxes. The level of confidence is defined by Redmon et al. (2016) as $P_r * IoU$. Given this definition, if there is no object present in a cell, the confidence score is zero. In order to consider a prediction confident enough, the authors set a threshold of at least the same value as the IoU between the predicted box and the ground truth. The predicted bounding boxes are defined as a set of x_{center}, y_{center} coordinates (parametrized to be offsets of a particular grid cell location, $0 < x_{center} < 1$ and $0 < y_{center} < 1$), a $width, height$ measures relative to the whole image ($0 < w < 1$ and $0 < h < 1$), and a confidence score. As for the P_r these are predicted for each of the cells as $P_r(Class_i|Object)$ and a single grid could predict C conditional class probabilities. During the test phase, the probability of a class appearing in a certain box and how well the predicted box fits the object are encoded by the class-specific confidence scores (Equation 1).

$$P_r(Class_i|Object) * P_r(Object) * IoU = P_r(Class_i) * IoU \tag{1}$$

Inspired by the GoogLeNet model for image classification, the YOLO architecture has 24 convolutional layers followed by two fully connected layers. However, the inception modules were replaced by $1X1$ reduction layers followed by $3X3$ convolutional layers. In where the first convolutional layers extract features from the image, and the fully connected layers are in charge of the probabilities and coordinates. The training of the network is described by Redmon et al. (2016) as first pretraining the 20 initial convolutional layers with the ImageNet 1000-class competition dataset, using an input size of $224x224$, followed by an increase of the input resolution to $448x448$ and the conversion of the network from classification to detection by adding four convolutional layers and two fully connected layers with randomly initialized weights, where the final layer outputs both class probabilities and bounding box coordinates. As for the activation

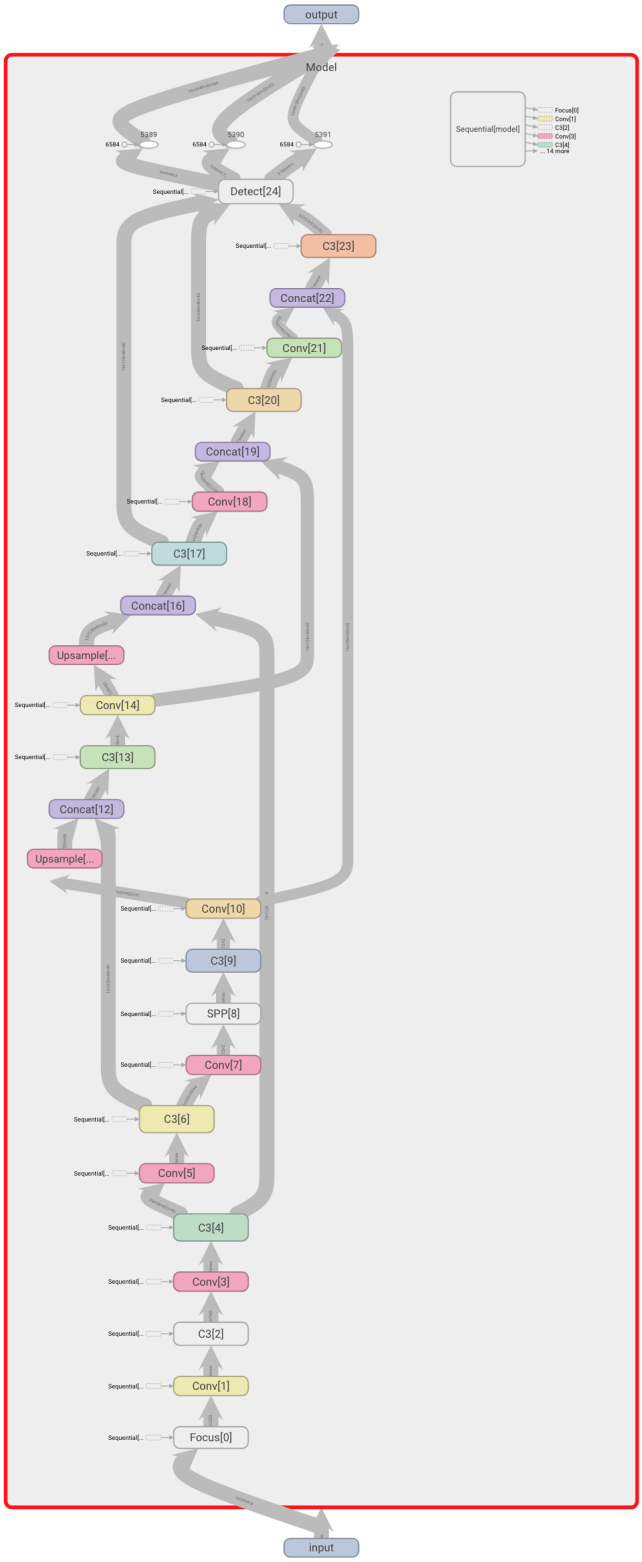


Figure 2: YOLO architecture. Source: YOLOv5 Repository

functions used, all the layers except the final one utilize a leaky rectified linear activation function (Leaky ReLu), and the final layer uses a linear activation function. Finally, the network is trained on the Pascal VOC 2007-2012 datasets.

The loss function (Equation 2) used is a modified squared sum where $\lambda_{coord} = 5$ and $\lambda_{noobj} = 0.5$ are introduce in order to avoid having the same weight for localization error, the difference between predicted and true bounding box coordinates, and class prediction error. Additionally, the width and height differences are square-rooted to prevent bias prediction of only small bounding boxes.

$$\begin{aligned}
&\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \\
+\lambda_{coord} &\sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} (C_i - \hat{C}_i)^2 \\
&+\lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} (C_i - \hat{C}_i)^2 \\
&+ \sum_{i=0}^{S^2} 1_{i}^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2
\end{aligned} \tag{2}$$

The limitations of the model encompassed difficulty to detect and segregate small objects that appear in groups, low accuracy when compared with two-stage object detectors (Faster R-CNN). YOLO version 2 was proposed to fix the detection of small objects in groups problem. This proposal introduced the YOLO architecture anchor boxes allowing the prediction of multiple bounding boxes from a single cell. Later, YOLO version 3 was proposed to improve the detection accuracy of its predecessors by utilizing a much more complex backbone, DarkNet-53. YOLOv4 proposed by improved, even more, the detector performance by adding Weighted Residual Connections, Cross Mini Batch Normalization, Cross Stage Partial Connections, Self-Adversarial Training, and Mish Activation as methodological changes amongst modern methods of regularization and data augmentation. Finally, YOLOv5, a PyTorch implementation of this one-stage object detector by Ultralytics, reduced the training and prediction times of YOLOv4 with similar performance.

3.3 *Proxy-label Semi-supervised Learning method*

Based on the taxonomy proposed by Van Engelen and Hoos (2020) the process of first training a classifier on labeled data and then using the predictions of the resulting classifier to generate additional labeled data is called a wrapper procedure, and it belongs to the many existent semi-supervised techniques. According to X. J. Zhu (2005) wrapper methods are among the oldest and most widely known algorithms for semi-supervised learning. Among wrapper methods, self-training techniques are the most basic of the approaches.

In words of Triguero, García, and Herrera (2015), "they consist of a single supervised classifier that is iteratively trained on both labeled data and data that has been pseudo-labeled in previous iterations of the algorithm. At the beginning of the self-training procedure, a supervised classifier is trained on only the labeled data. The resulting classifier is used to obtain predictions for the unlabeled data points. Then, the most confident of these predictions are added to the labeled data set, and the supervised classifier is re-trained on both the original labeled data and the newly obtained pseudo-labeled data. This procedure is typically iterated until no more unlabeled data remain."

The self-learning algorithm works as follows:

1. Train the classifier with the existing labeled dataset.
2. Predict a portion of samples using the trained classifier.
3. Add the predicted data with a high confidence score into the training set.
4. Repeat all steps above until meeting the stopping criteria.

The most confidence predictions are selected based on a predefined confidence measure, and its selection is of great importance given that an inaccurate confidence measure leads to adding mislabeled examples to the training set, bringing with it a classification performance degradation (Triguero et al., 2015). As for the addition mechanism, there are a variety of schemes in which the training set can be incremented (e.g., incremental, amending, or batch). Incremental addition consists of the instance by instance enlargement of the training set after meeting the confidence measure threshold. The batch approach involves first assessing each of the instances' confidence measures before adding any of them to the training set. This followed by adding to the training data a batch of all of those that do meet the confidence criteria. Amending addition, contrary to the other

two methods, allows to iteratively add or remove any instance that does or does not fulfill the confidence criteria. In that way, this addition mechanism provides a rectification of previous self-labeled instances, avoiding the introduction of noise.

The stopping criteria is the mechanism used to stop the self-labeling process and according to Triguero et al. (2015) there exist three main approaches. The first technique utilizes during the self-labeling process all the unlabeled data available. The second approach involves choosing instances from a smaller pool instead of the whole unlabeled data to form the new training set by establishing a limited number of iterations. The third method is based on the unchanging status of the classifier learned hypothesis. When the learned hypothesis remains the same for a couple of iterations, the process ends.

3.4 *Evaluation metrics*

The evaluation and comparison of the model's performance are made using the standard metrics used for similar object detection tasks, namely, Mean Average Precision (mAP) at some specific IoU threshold (e.g., mAP@0.5), and average precision(AP).

The Confidence Score is the probability that an anchor box contains an object. Intersection over Union (IoU) (Equation 3) is defined as the area of the intersection divided by the area of the union of a predicted bounding box (B_p) and a ground truth box(B_{gt}). IoU metric ranges from 0 and 1, with 0 representing no overlap and 1 meaning perfect overlap between B_p and B_{gt}. With the IoU, metric a threshold is necessary to distinguish a valid detection from an invalid one. A detection is considered a true positive (TP) only if: (1)$ConfidenceScore > threshold$; (2) the predicted class matches the class of a ground truth; and (3) the predicted bounding box has an IoU greater than a threshold with the ground truth. Violation of either of the latter two conditions produces a false positive (FP). Recall (Equation 4) measures the ability of the model to find all relevant cases (all ground truths). It is defined as the proportion of TPs detected among all ground truths. Precision (Equation 5) is the ability of a classifier to identify relevant objects only. It is evaluated as the proportion of TPs detections among all detections. A good model is a model that can identify most ground-truth objects (high recall) while only finding the relevant objects (high precision) often. AP (Equation 6) is calculated based on the Precision-Recall curve at a certain IoU threshold for each of the classes, and it condenses the

weighted mean of precisions for each threshold with the increase in recall. mAP (Equation 7) is the average of AP values over all classes.

$$IoU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \tag{3}$$

$$R = \frac{TP}{TP + FN} \tag{4}$$

$$P = \frac{TP}{TP + FP} \tag{5}$$

$$AP@\alpha = \int_0^1 p(r)dr \tag{6}$$

$$mAP@\alpha = \frac{1}{n} \sum AP_i, fornclasses \tag{7}$$

4 EXPERIMENTAL SETUP

The present chapter describes the specifics of the dataset and experimental procedure undertaken.

4.1 *Software*

Python 3.7 was used to implement the experiments of this study. The experiments were executed on Jupyter notebooks hosted on a Google Cloud Platform Server Virtual Machine with a Debian kernel, 4vCPUs, 15GB RAM, an NVIDIA Tesla T4 GPU, and CUDA 11.0. A Pytorch based implementation of Faster R-CNN by Open MMLab (Chen et al., 2019) was used for this study, and YOLOv5 was adapted from the Ultralytics' implementation (Jocher et al., 2021). The different packages and modules used to complement the previously mentioned algorithms were: Numpy, Pandas, Matplotlib, Seaborn, Torch & Torchvision.

4.2 *Data*

The dataset was provided by felyx Sharing B.V., and it contains 86,400 pictures in jpg format with different resolutions. On each of the images, it is possible to completely or partially observe parked e-moped(s) from diverse angles and with different backgrounds. The images were taken by felyx Sharing B.V. customers with various cellphone models. All the pictures have three channels (RGB) and come in different dimensions, see Appendix A (page 32). The images required labeling and re-scaling to

proceed with the presence mirror detection through supervised learning. This process was conducted using Roboflow's bounding boxes labeling feature (Roboflow, 2022). Only a limited amount of pictures were labeled, approximately 3000. As for the remaining unlabeled data, an amount (defined by the SSL method) of them were used for semi-supervised learning purposes.

4.3 *Preprocessing*

Images can be annotated in different fashions depending on the use case (e.g., semantic segmentation, bounding boxes, polygons, and others) under supervised learning settings. For the purpose of this research, a bounding box approach was undertaken. A total of 2955 randomly selected images were manually-annotated utilizing Roboflow's bounding boxes labeling feature. The labels assigned to each of the images, when possible, were (0) "left mirror" and (1) "right mirror." Finally, 215 null-images (images that do not contain mirrors) were added to the annotated dataset. An example of the annotations can be found in Figure 3. The dataset containing the manually-annotated 3170 images was divided in a stratified manner into train ($n = 2220$), validation ($n = 633$), and test ($n = 317$) set, each of them containing an approximately 6% of null-images ($m = 158, 38, 19$, respectively). The annotations of each of the splits has a fairly balanced distribution of labels, for the train set from a total of 4218 annotations 2116 (50.166%) belong to the 'left mirror' class, for the validation set from a total of 1994 annotations 607 (30.441%) belong to the 'left mirror' class, and for the test set from a total of 628 annotations 322 (51.274%) belong to the 'left mirror' class (Table 1.) The average number of mirrors (annotations) per image for the annotated data set is 2.053, and the total number of images with only one mirror (annotation) is 232, 131 images with only one left mirror and 101 images with only one right mirror.

Table 1: Annotations details.

Split	**No. images**		**No. Annotations**	
	Labeled	**Null**	**Left**	**Right**
Train	2062	158	2116	2102
Validation	595	38	607	1387
Test	298	19	322	306

The images on each set were re-seized and transformed to the appropriated ingestion format for each algorithm. For YOLOv5, an input size of $640X640$ and a format that implies a collection of images where each

of the images has a unique name together with a TXT file, with the same name as its corresponding image, containing x and y coordinates of the bounding box(es) and the assigned label to the same(s). These files were stored in different folders, for each set, containing two separate folders for the images and the TXT files, respectively. For Faster R-CNN, the input size is 1333*X*800, and different ingestion formats are supported; however, the COCO format was selected for this study. The COCO format consists of raw image data and a single JSON file that contains all of the annotations, metadata of the images, categories, and other information about the dataset. The Faster R-CNN files were stored based on train, validation, and test set in different folders that store all the images of the corresponding set and their JSON file.

Figure 3: Manually annotated images example

4.4 *Experimental procedure*

The process of identifying the presence of a rear-view mirror through images is an object detection problem that could be carried out in a supervised or semi-supervised manner, with the aim of assessing the performance of fine-tuned generic state-of-the-art deep object detectors both approaches were adopted. This sub-section presents the details of the experimental procedure.

4.4.1 *YOLOv5 training and fine-tuning*

YOLOv5 is available in five different base models, all of which are pre-trained on the standard COCO dataset and differ in size. One of the most miniature versions, YOLOv5s, was chosen for this study, given its detection speed and size-performance trade-off, to see other models' characteristics go to Appendix B (page 33).

Yolov5s architecture was trained to detect the 80 different classes present on the COCO dataset, and therefore the number of classes to

be identified by the architecture's head detection layer had to be fine-tuned, from 80 to two classes for this case, "left mirror" and "right mirror," labeled as 0 and 1 respectively. As for the rest of the architecture, it remained the same. The source paths of the dataset must be previously established in a configuration file of YAML type. Such file comprehends the train, and validation directories paths, number of classes to be detected, and their names. As mentioned, the images are required to be re-sized (640X640) before starting with the training. For the fine-tuning of the model, the weights obtained after training the YOLOv5s on the COCO dataset were preloaded, and the layers comprehending the backbone (0-9) were frozen to speed up the training process. The baseline model was considered as the previously described configuration of the model trained without augmentation and with the default hyperparameters.

To yield better model performance, a model hyperparameter tuning was carried by means of manual search, given that an exhaustive grid search represented a high computational cost. The summary of the specific hyperparameter settings explored during training is presented in Table 3. In order to find the best combination of hyperparameters, each of the parameters was changed one at a time and asses according to the validation set mAP@0.5-0.9, its classification, and bounding boxes loss. An SGD optimizer with different learning rates and small batch sizes together with momentum was utilized (Xing, Arpit, Tsirigotis, & Bengio, 2018). The model was trained for 50, 150, and 300 epochs to observe what was the most suitable value to prevent overfitting or underfitting the train set data. According to the results obtained from the manual search, the best-founded configuration for this model had a learning rate of 0.01, a momentum of 0.937, and a batch size set to 32. To enforce light condition changes robustness of the model, photometric augmentation (see Table 2) was implemented on the training data (Tellez et al., 2019).

4.4.2 *Faster R-CNN training and fine-tuning*

The Faster R-CNN model has several implementations available; for this project, the implementation of Open MMLab was used for the mirrors detection problem, given its high efficiency and training speed in comparison to other implementations such as Detectron2 or SimpleDet (*open-mmlab/mmdetection: OpenMMLab Detection Toolbox and Benchmark*, 2021). The selected backbone for the Faster R-CNN model implemented on PyTorch style was ResNet50, and the network was initialized using the pre-trained weights obtained with the COCO dataset. For the model fine-tuning, the head of the architecture's second stage, classification of objects stage, had to be modified by adjusting the number of classes from 80 to 2 classes, "left mirror" and "right mirror," labeled as 0 and 1, respectively. Additionally,

Table 2: Data Augmentation for both models.

Data Augmentation
Image HSV-Hue = 0.015 (fraction) Image HSV-Saturation = 0.7 (fraction) Image HSV-Lightness = 0.4 (fraction) Image HSV-Contrast = 0.5 (fraction)
→ Note: Image flip left-right must be omitted by all means.

all the weights of the model's first stage, region proposal generation stage, were frozen. As mentioned, the selected format of the data for ingestion was COCO format, and the images required non-destructive re-sizing (1333X800) to start with the training of the model. The baseline model was considered as the previously described configuration of the model trained without augmentation and with the default hyperparameters.

Manual hyperparameter tuning was undertaken to achieve good performance and generalization ability of the model. Based on the validation set mAP@0.5-0.9, its classification, and bounding boxes loss, the best-founded parameters for this model were learning rate set to 0.01, batch size of 4, and a momentum of 0.9. The explored hyperparameters are described in Table 4. Only one parameter at a time was changed. Data photometric augmentation was also done for this model (see Table 2.) The optimizer used for this model was SGD since the authors of the toolbox do not recommend using ADAM given that the performance could drop considerably. The number of epochs used for training was 6, and 12. Obtaining an appropriate fitting with 12 epochs.

4.4.3 *Amount of labeled data*

After obtaining the best configuration for both models, the training data set was used to randomly select bundles of labeled images of different sizes, namely, 500, 1000, 1500, and 2000 images. Those bundles of data were then used to re-trained the models so to see how the amount of manually-labeled data affected the performance of the models when detecting e-moped rear-view mirrors. The results can be observed in Table 6 in the Results section.

4.4.4 *Semi-supervised learning*

As explained in section 3., semi-supervised learning requires a proxy-labeled data addition mechanism, a confidence measure for addition, and a stopping criteria mechanism. The addition mechanism selected for both models was the batch addition mechanism given that this technique, in contrast to the incremental mechanism, do not alter the learned hypothesis

during training phase allowing to prioritize the most confident self-labeled instances when injecting them for re-training after filtration. The stopping criteria the so-called by Triguero et al. (2015) unchanging status stopping criteria reinforced with help of an early stopping mechanism, based on $mAP@0.5$ value evaluated over the validation set, with a patience of 50 for YOLOv5 and of 4 for Faster R-CNN; such stopping mechanism was chosen as it limits the number of iterations without having to unnecessarily over or underused all the available unlabeled data, opposite to the other to previously mentioned methods. The confidence measure for addition the probability that the anchor box contains an object (detection confidence score.) The confidence score threshold was defined to 0.7 following the criteria selected by Y. Li, Huang, Qin, Wang, and Gong (2020) to obtain their empirical results. The final size of the train data set and performance results for each of the models are described in section 5.

5 RESULTS

This section is divided as follows: firstly, an overall comparison of the fine-tuned models trained on manually-labeled data will be made. Subsequently, the impact of the amount of labeled data while training on the model's performance will be discussed. Finally, the detection results of the supervised and semi-supervised trained models will be compared.

5.1 *Fine-tuned models comparison*

To help answer the first sub-question, a comparison was made between two state-of-the-art object detectors: Faster R-CNN and YOLOv5. Table 5 shows the difference in performance (measured based on $mAP@0.5$, $mAP@0.5:0.95$, and $AP@0.5$) between the two approaches, which was evaluated on the test partition of the dataset. As shown in the table, the effect of fine-tuning on the performance of YOLOv5 and Faster R-CNN can be observed. Fine-tuning the YOLOv5 and Faster R-CNN models on domain-specific knowledge substantially increases the performance of the models. The increase in performance for the YOLOv5 model was on average 76.9% for all measures. For example, compared to the highest $mAP@0.5$ reported for the baseline models, fine-tuning increased the $mAP@0.5$ of the fine-tuned YOLOv5 model from 0.587 to 0.965. Similarly, the Faster R-CNN model performance increased on average 17.3% for all measures. As for the contrast between the fine-tuned models, from the values evaluated, it is possible to see that both models have similar performance; however, YOLOv5 does a slightly better job when detecting

mirrors at high confidence scores, see $mAP@0.5:0.95$. That is to say, that the localization of the objects is more precise for this model.

Table 3: Hyperparameter search for YOLOv5. The values presented in boldface are the ones that gave the best results for this model.

Hyperparameter	Values		
Lr	0.001	**0.01**	0.05
Momentum	0.537	0.737	**0.937**
Batch size	16	**32**	64
$\star$ Optimizer	SGD		

Table 4: Hyperparameter search for Faster R-CNN. The values presented in boldface are the ones that gave the best results for this model.

Hyperparameter	Values		
Lr	0.001	**0.01**	0.02
Momentum	0.5	0.7	**0.9**
Batch size	2	**4**	8
$\star$ Optimizer	SGD		

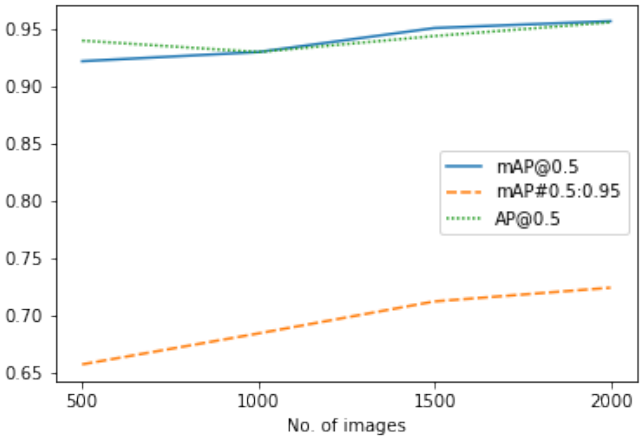
Table 5: Performance of the models in rear-view mirrors detection. The results of all models were obtained using the test set.

Models		mAP@0.5	mAP@0.5:0.95	AP@0.5
YOLOv5	Baseline	0.587	0.362	0.577
	Fine-tuned	0.965	0.728	0.96
Faster R-CNN	Baseline	0.817	0.592	0.817
	Fine-tuned	0.955	0.7	0.955

5.2 *Impact of amount of labeled data*

Besides the performance of the fine-tuned models when detecting the presence of e-moped rear-view mirrors, the performance of the same when trained with different amounts of manually-labeled data has been assessed and can be seen in Table 6. As it can be appreciated, when increasing the number of data the performance of both models improves. Especially

Figure 4: Performance of the fine-tuned models depending on the amount of data used for training



for the mAP across detections with values of IoU between 0.5 and 0.95, observing an increase of 10.19% between extreme values (500 and 2000 labeled images) for YOLOv5, and for Faster R-CNN of 3.9%. Nevertheless, as it is perceive for both models a plateau for the mAP with IoU of 0.5 took place after using more than 1500 labeled images.

Table 6: Performance of the fine-tuned models depending on the amount of data used for training. The best results are presented in boldface. All results were evaluated on the test set.

Model	labeled data(#)	mAP@0.5	mAP@0.5:0.95	AP@0.5
YOLOv5	500	0.922	0.657	0.94
	1000	0.930	0.684	0.930
	1500	0.951	0.712	0.944
	2000	**0.957**	**0.724**	**0.956**
Faster R-CNN	500	0.910	0.665	0.910
	1000	0.925	0.681	0.925
	1500	0.935	0.686	0.935
	2000	**0.935**	**0.691**	**0.935**

5.3 *Fully supervised vs. Semi-supervised models*

To answer sub-question three, the models were trained under SSL settings. For YOLOv5 not improvement was seen from the addition of self-labeled data, contrarily a slight deterioration of 2.47% for the model's $mAP@0.5$:

0.95 over all classes was appreciated. After injecting 624 images and their proxy-labels to the train set the SS learning training was stopped given no performance improvement, see Tables 7. Additionally, the increment on the AP@0.5 appreciated for the first batch of added self-labeled images represented a diminished AR@0.5. As for Faster R-CNN, similar results were obtained even though the amount of injected self-label images per batch was slightly larger than the one of the one-stage detector. A decline of 6.71% for the model's mAP@0.5 : 0.95 over all classes was obtained, as well as a drop of the same proportion on the AP@0.5. Therefore, the FL trained model represents a more suitable approach for the objective of this work.

Table 7: Comparison between Fully-Supervised (FS) trained models and Semi-Supervised (SS) trained models performance. The performance of the model under semi-supervised learning was register after the addition of each self-labeled data batch. All results were obtained using the test set.

Model	Approach	Train set size	mAP@0.5	mAP@0.5:0.95	AP@0.5
YOLOv5	FS	2220	0.965	0.728	0.96
	SS	2568	0.96	0.719	0.981
		2844	0.959	0.71	0.972
FasterR-CNN	FS	2220	0.955	0.7	0.955
	SS	2842	0.94	0.676	0.94
		3313	0.938	0.653	0.938

6 DISCUSSION

The primary goal of this study was to assess how efficient generic state-of-the-art deep object detectors are when locating and classifying e-moped rear-view mirrors by answering three sub-questions. The main finding was that generic fine-tuned object detectors such as Faster R-CNN and YOLOv5 could efficiently detect e-moped rear-view mirrors from images.

The first sub-question explored how well fine-tuned Faster R-CNN and YOLOv5 work when applied on a domain-specific detection application. Results showed that both models have similar performance when detecting e-moped rear-view mirrors; nonetheless, YOLOv5 has a slightly better performance than Faster R-CNN, especially for the task of mirrors localization inside a picture. Previous studies that carried comparisons between both architectures have shown results in line with the ones obtained in this examination. (Kim, Sung, & Park, 2020) contrasted the efficiency of Faster R-CNN to the one of YOLO version 4 and SSD when detecting vehicles in real-time. Their results show that YOLO version 4 has a better performance

Figure 5: Automatically detected mirrors in test set images using Faster R-CNN. Examples of correct and incorrect detections.

and faster detection than Faster R-CNN. Similarly, Benjdira, Khursheed, Koubaa, Ammar, and Ouni (2019) presented a comparison between YOLO version 3 and Faster R-CNN when detecting vehicles from aerially-taken images. The precision of YOLOv3 was to some extent better than Faster R-CNN's, yet the recall was notably higher for the one-stage object detector. A possible reason for Faster R-CNN to not have as favorable results as the other model could be its two-stage architecture, typically slower, and the fact that it was trained for a relatively small amount of epochs compared to YOLOv5. Another limitation of this experiment can be found in the manual hyperparameter tuning undertaken to find the best-performing version of the models.

Figure 6: Automatically detected mirrors in test set images using YOLOv5. Examples of correct and incorrect detections.

The second sub-question focused on the effect of manually-labeled data amount on the performance of fine-tuned object detectors. The empirical results agreed with the literature (Van Engelen & Hoos, 2020) as the performance of both detectors increases as the size of the training set grows. Notably, the evaluation outcomes exhibited that the detection at higher confidence levels enhanced with a more considerable amount of data to learn from. From this experiment, it is possible to notice that YOLOv5 required fewer data to yield moderately better results than Faster R-CNN. This behavior could be given by the feature of, introduced to the YOLO family with YOLOv3 PyTorch implementation, learning anchor boxes based on the distribution of bounding boxes in the custom dataset with K-means and genetic learning algorithms. This feature helps the

model tackle custom tasks with various distributions of bounding box sizes and locations more effectively than the Faster R-CNN's RPN. Accordingly, limitations for comparing both approaches can be found in the architecture of Faster R-CNN. In future works, a comparison between YOLOv5 and Cascade R-CNN (Cai & Vasconcelos, 2018) could give fairer results.

The third sub-question examined how implementing semi-supervised learning affects the detection of Faster R-CNN and YOLOv5 e-moped rear-view mirrors. This question was raised given the nature of the provided data (mainly unlabeled) and the fact that manual labeling is a highly taxing task. Literature has proven that the bigger the training dataset, the better the results, and that proxy labels help enlarge a training dataset when aiming for better results (X. Zhu & Goldberg, 2009). Nonetheless, after evaluating the SSL trained object detectors, the results obtained were opposed to expectations, attaining a slight drop in performance after adding two batches of proxy-labeled data. However, other studies also recorded an unsatisfactory impact on model efficiency after using SSL training, as it is stated in the work of Y.-F. Li and Zhou (2014). One of the most significant limitations was the selected confidence measure threshold ($cm > 0.7$); this could be set to a more discriminative value that does not allow wrongly labeled images, noise, to be introduced to the training set.

From the explorations and limitations of this study, a groundwork for future work in the area of detection of moped-related damages has been laid out, and from it, few recommendations emerged. Future studies are advised to explore instance segmentation for the purpose of e-moped mirror detection and other scooter and rear-view mirror damages detection. This since the level of detail of the annotations, pixel by pixel, can improve the performance without extensive training datasets (He, Gkioxari, Dollár, & Girshick, 2017). Additionally, exploring the effect of different backbones, pre-trained weights, and a grid search-based hyperparameter tuning for each of the architectures is encouraged since this could lead to possible further improvements of the models. Lastly, applying more complex semi-supervised learning methods (e.g., adversarial SSL learning) or selective self-training methods (Oliver, Odena, Raffel, Cubuk, & Goodfellow, 2019) would be interesting.

7 CONCLUSION

All things considered, this study exhibited that generic fine-tuned object detectors such as Faster R-CNN and YOLOv5 could efficiently carry out automatic visual detection of e-moped rear-view mirrors, and potentially detection of other kind of e-moped damages. In line with previous literature, results displayed a slight difference between the one-stage and the

two-stage object detectors. Therefore, a selection between one or the other depends on the deployment preferred platform restrictions and/or the detection speed requirements. However, it is essential to highlight that YOLOv5 required fewer data to yield moderately better results than Faster R-CNN; hence its use for applications where minimal data is available could be recommended based on this work results. The semi-supervised learning self-training method, used to take advantage of all the provided data, did not prove an enhancement on the performance of neither of the object detectors; however, these results could have been caused by improper confidence measure threshold selection or lack of discrimination of the proxy-labeled images that fulfilled the addition-criteria. Consequently, future works are encouraged to explore further the effect of SSL in detection problems encompassed in the moped-related areas. Additionally, future research should explore the effects of different backbones, pre-trained weights, and hyperparameter tuning strategies over the object detectors postulated before. Finally, this study can help with the development of a tool that aids the reparation process of e-moped damages by reducing missing mirrors detection time, helping to ensure the integrity of the vehicles and, subsequently, the driver's safety.

REFERENCES

Alex, R. (2020). Object detection algorithms: A review.

Benjdira, B., Khursheed, T., Koubaa, A., Ammar, A., & Ouni, K. (2019). Car detection using unmanned aerial vehicles: Comparison between faster r-cnn and yolov3. In *2019 1st international conference on unmanned vehicle systems-oman (uvs)* (pp. 1–6).

Burton II, W., Myers, C., & Rullkoetter, P. (2020). Semi-supervised learning for automatic segmentation of the knee from mri with convolutional neural networks. *Computer methods and programs in biomedicine*, *189*, 105328.

Cai, Z., & Vasconcelos, N. (2018). Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 6154–6162).

Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., ... Lin, D. (2019). MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*.

Christoforou, Z., Gioldasis, C., de Bortoli, A., & Seidowsky, R. (2021). Who is using e-scooters and how? evidence from paris. *Transportation research part D: transport and environment*, *92*, 102708.

Chun, C., & Ryu, S.-K. (2019). Road surface damage detection using fully convolutional neural networks and semi-supervised learning. *Sensors*,

19(24), 5501.

Degele, J., Gorr, A., Haas, K., Kormann, D., Krauss, S., Lipinski, P., ... Hertweck, D. (2018). Identifying e-scooter sharing customer segments using clustering. In *2018 ieee international conference on engineering, technology and innovation (ice/itmc)* (pp. 1–8).

e Silva, R. R., Aires, K. R., & de MS Veras, R. (2018). Detection of helmets on motorcyclists. *Multimedia Tools and Applications*, *77*(5), 5659–5683.

Espinosa, J. E., Velastin, S. A., & Branch, J. W. (2018). Motorcycle detection and classification in urban scenarios using a model based on faster r-cnn.

Gauquelin, A. (2021, Jun). *Moped-sharing: The dutch connection.* Retrieved from https://shared-micromobility.com/moped-sharing-the-dutch-connection/

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the ieee international conference on computer vision* (pp. 1440–1448).

Gössling, S. (2020). Integrating e-scooters in urban transportation: Problems, policies, and the prospect of system change. *Transportation Research Part D: Transport and Environment*, *79*, 102230.

He, K., Gkioxari, G., Dollár, P., & Girshick, R. B. (2017). Mask R-CNN. *CoRR*, *abs/1703.06870*. Retrieved from http://arxiv.org/abs/1703.06870

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, *37*(9), 1904–1916.

Helm wordt vanaf juli volgend jaar verplicht voor snorfietsers | rtl nieuws. (n.d.). https://www.rtlnieuws.nl/nieuws/politiek/artikel/5233620/helm-wordt-verplicht-voor-snorfietsers-vanaf-juli-volgend-jaar.

Jaffari, R., Hashmani, M. A., Reyes-Aldasoro, C. C., Aziz, N., & Rizvi, S. S. H. (2021). Deep learning object detection techniques for thin objects in computer vision: An experimental investigation. In *2021 7th international conference on control, automation and robotics (iccar)* (pp. 295–302).

Jocher, G., Stoken, A., Chaurasia, A., Borovec, J., NanoCode012, TaoXie, ... wanghaoyang0106 (2021, October). *ultralytics/yolov5: v6.0 - YOLOv5n 'Nano' models, Roboflow integration, TensorFlow export, OpenCV DNN support.* Zenodo. Retrieved from https://doi.org/10.5281/zenodo.5563715 doi: 10.5281/zenodo.5563715

Kakani, V., Nguyen, V. H., Kumar, B. P., Kim, H., & Pasupuleti, V. R. (2020). A critical review on computer vision and artificial intelligence in food industry. *Journal of Agriculture and Food Research*, *2*, 100033.

Kim, J.-a., Sung, J.-Y., & Park, S.-h. (2020). Comparison of faster-rcnn, yolo, and ssd for real-time vehicle type recognition. In *2020 ieee international*

conference on consumer electronics-asia (icce-asia) (pp. 1–4).

Li, N., & Xia, Y. (2018). Affective image classification via semi-supervised learning from web images. *Multimedia Tools and Applications*, *77*(23), 30633–30650.

Li, Y., Huang, D., Qin, D., Wang, L., & Gong, B. (2020). *Improving object detection with selective self-supervised self-training.*

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

Lin, H., Deng, J. D., Albers, D., & Siebert, F. W. (2020). Helmet use detection of tracked motorcycles using cnn-based multi-task learning. *IEEE Access*, *8*, 162073-162084. doi: 10.1109/ACCESS.2020.3021357

Meng, C., Bao, H., & Ma, Y. (2020). Vehicle detection: A review. In *Journal of physics: Conference series* (Vol. 1634, p. 012107).

Mhalla, A., Chateau, T., Gazzah, S., & Amara, N. E. B. (2018). An embedded computer-vision system for multi-object detection in traffic surveillance. *IEEE Transactions on Intelligent Transportation Systems*, *20*(11), 4006–4018.

Oliver, A., Odena, A., Raffel, C., Cubuk, E. D., & Goodfellow, I. J. (2019). *Realistic evaluation of deep semi-supervised learning algorithms.*

open-mmlab/mmdetection: Openmmlab detection toolbox and benchmark. (2021). https://github.com/open-mmlab/mmdetection.

Prajwal, M., Tejas, K., Varshad, V., Murgod, M., & Shashidhar, R. (2019). Detection of non-helmet riders and extraction of license plate number using yolo v2 and ocr method. *Int. J. Innov. Technol. Exploring Eng.(IJITEE)*, *9*(2).

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 779–788).

Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster r-cnn: towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, *39*(6), 1137–1149.

Roboflow, I. (2022). *Roboflow annotation tool.* https://roboflow.com/annotate.

Sanjana, S., Shriya, V., Vaishnavi, G., & Ashwini, K. (2021). A review on various methodologies used for vehicle classification, helmet detection and number plate recognition. *Evolutionary Intelligence*, *14*(2), 979–987.

Schellong, D., Sadek, P., Schaetzberger, C., & Barrack, T. (2021, Jan). *The promise and pitfalls of e-scooter sharing*. BCG Global. Retrieved from https://www.bcg.com/publications/2019/promise-pitfalls-e-scooter-sharing

Tellez, D., Litjens, G., Bándi, P., Bulten, W., Bokhorst, J.-M., Ciompi, F., & van der Laak, J. (2019). Quantifying the effects of data augmentation and stain color normalization in convolutional neural networks for computational pathology. *Medical image analysis*, *58*, 101544.

Triguero, I., García, S., & Herrera, F. (2015). Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Knowledge and Information systems*, *42*(2), 245–284.

Van Engelen, J. E., & Hoos, H. H. (2020). A survey on semi-supervised learning. *Machine Learning*, *109*(2), 373–440.

Weinstein, B. G., Marconi, S., Bohlman, S., Zare, A., & White, E. (2019). Individual tree-crown detection in rgb imagery using semi-supervised deep learning neural networks. *Remote Sensing*, *11*(11), 1309.

wetten.nl - regeling - regeling voertuigen - bwbr0025798. (n.d.). https://wetten.overheid.nl/BWBR0025798/2021-01-21#Hoofdstuk5.

Wu, X., Sahoo, D., & Hoi, S. C. (2020). Recent advances in deep learning for object detection. *Neurocomputing*, *396*, 39–64.

Xing, C., Arpit, D., Tsirigotis, C., & Bengio, Y. (2018). *A walk with sgd.*

Zaman, F. H. K., Abdullah, S. A. C., Razak, N. A., Johari, J., Pasya, I., & Kassim, K. A. A. (2021). Visual-based motorcycle detection using you only look once (yolo) deep network. In *Iop conference series: Materials science and engineering* (Vol. 1051, p. 012004).

Zhu, X., & Goldberg, A. B. (2009). Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, *3*(1), 1–130.

Zhu, X., Liu, S., Zhang, P., & Duan, Y. (2019). A unified framework of intelligent vehicle damage assessment based on computer vision technology. In *2019 ieee 2nd international conference on automation, electronics and electrical engineering (auteee)* (pp. 124–128).

Zhu, X. J. (2005). Semi-supervised learning literature survey.

Zou, Z., Shi, Z., Guo, Y., & Ye, J. (2019). Object detection in 20 years: A survey. *arXiv preprint arXiv:1905.05055*.

APPENDIX A

Figure 7: Examples of images present in the provided dataset

Model	size (pixels)	mAP^{val} 0.5:0.95	mAP^{val} 0.5	Speed CPU b1 (ms)	Speed V100 b1 (ms)	Speed V100 b32 (ms)	params (M)	FLOPs @640 (B)
YOLOv5n	640	28.4	46.0	**45**	**6.3**	**0.6**	**1.9**	**4.5**
YOLOv5s	640	37.2	56.0	98	6.4	0.9	7.2	16.5
YOLOv5m	640	45.2	63.9	224	8.2	1.7	21.2	49.0
YOLOv5l	640	48.8	67.2	430	10.1	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	766	12.1	4.8	86.7	205.7

Figure 8: YOLOv5 available models. Source: YOLOv5 Repository

APPENDIX B