Advances in Artificial Intelligence and Machine Learning; Research 2 (1) 315-337 Received 10-03-22; Accepted 25-03-22; Published 31-03-22

Improving the Pedestrian Detection Performance in the Absence of Rich Training Datasets: A UK Case Study

Juliana Negrini de Araujo negrinij@coventry.ac.uk
Coventry University, Coventry, UK.
Vasile Palade, ab5839@coventry.ac.uk
Centre for Computational Science and Mathematical Modelling, Coventry University, Coventry, UK
Tabassom Sedighi, tabassom.sedighi@aru.ac.uk
Anglia Ruskin University, Cambridge, UK
Alireza Daneshkhah, ac5916@coventry.ac.uk
Centre for Computational Science and Mathematical Modelling, Coventry University, Coventry, UK

Corresponding Author: Vasile Palade.

Abstract

The World Health Organization estimates that well in excess of one million of lives are lost each year due to road traffic accidents. Since the human factor is the preeminent cause behind the traffic accidents, the development of reliable Advanced Driver Assistance Systems (ADASs) and Autonomous Vehicles (AVs) is seen by many as a possible solution to improve road safety. ADASs rely on the car perception system input that consists of camera(s), LIDAR and/or radar to detect pedestrians and other objects on the road. Hardware improvements as well as advances done in employing Deep Learning techniques for object detection popularized the Convolutional Neural Networks in the area of autonomous driving research and applications. However, the availability of quality and large datasets continues to be a most important issue that influences the Deep Learning based model's performance. With this in mind, this work analyses how a YOLO-based object detection architecture responded to limited data available for training and containing low-quality images. The work focused on pedestrian detection, since vulnerable road user's safety is a major concern within AV and ADAS research communities. The proposed model was trained and tested on data gathered from Coventry, United Kingdom, city streets. The results show that the original YOLOv3 implementation reaches a 42.18 % average precision (AP), and the main challenge was in detecting small objects. Network modifications were made and our final model, based on the original YOLOv3 implementation, achieved 51.6 % AP. It is also demonstrated that the employed data augmentation approach is responsible for doubling the average precision of the final model.

Keywords: Keywords: Pedestrian Detection, YOLOv3, Convolutional Neural Networks

Citation: Juliana Negrini de Araujo, et al. Improving the Pedestrian Detection Performance in the Absence of Rich Training Datasets: A UK Case Study. Advances in Artificial Intelligence and Machine Learning. 2022;2(1):21.

1. INTRODUCTION

According to a recent World Health Organization report, it is estimated that 2.1 billion vehicles navigate our roads, and the rate of traffic death per 100.000 habitants has remained almost stable despite the continuous rise of the overall number of vehicles and number of traffic deaths worldwide [1]. Even though this indicates that the situation is not getting more critical, it is necessary to consider that a huge number of people lose their lives due to road accidents. Vulnerable road users, such as cyclists and pedestrians, represent around 26 % of the total victims of road casualties worldwide [2, 3]. Investments in road infrastructure and legislation enforcement have shown to be successful strategies, but the leading causes of accidents are human-related, such as excessive speed, driving under the effect of illegal substances, driver drowsiness and driver distraction [4].

Advanced Driver Assistance Systems (ADASs) and Autonomous Vehicles (AVs) technologies are seen by many as one way to improve road safety. [3] estimate that 29 % of road fatalities that occurred in the United States could be avoided by the usage of ADASs. These systems rely on the car perception features that are usually composed by camera(s), LIDAR and/or radar sensors. Convolutional Neural Networks (CNNs) and other Deep Learning techniques have become popular tools in Autonomous Driving applications [5, 6]. In this work, the performance of YOLO-V3 Deep Neural Network model [7] for real-time pedestrian detection is studied and improved. The main objective is to evaluate how this popular baseline network behaves when it is subject to limited and lower quality training data, i.e. containing glare, shadow areas, sunlight reflections, etc. In addition, a list of modifications and their respective effect on the detection accuracy and speed is presented. After the performed tests, a modified YOLOv3 model is created to achieve higher average precision results. The work focuses on pedestrian detection, since vulnerable road user's safety is a major concern within AV and ADAS research communities.

This work is organised as follows: Section II presents a brief literature review on object detection with focus on pedestrian detection; The dataset, the developed model and further implementations details are described in Section III; The results are shown in Section IV, followed by the Discussion and Conclusion sections.

2. LITERATURE REVIEW

2.1 Real-time Object Detection with Deep Learning

The object detection task is mainly composed of two problems. The first is to estimate where the object is located on the image with a bounding box, and the second is to recognise this object as belonging to a specific class [8], as illustrated in FIGURE 1. The major challenge is that it not known a priori how many objects there will be on the image to be detected. In the computer vision research community, object detection is one of the most essential problems since it serves as a fundamental tool for more specific tasks, such as object tracking, image segmentation and captioning [9], etc.

On their extensive review of twenty years of object detection, Zou et al. [7] describe the main object detection algorithms, including the deep learning methods for object detection. These object detection methods can be divided into one-stage and two-stage detectors [6]. A two-stage detector

Figure 1: Example of pedestrian detection performed on this project. Each detected object is enclosed by a bounding box with an assigned class and probability.

uses a complex pipeline that involves two or more models to perform the extraction of the regions of interest (RoI), and detection and classification of objects. Consequently, they require more time to train and do not usually achieve real-time performance. Currently, CNNs represent the basic framework behind most of the top-performing models for general object detection [5, 6]. The first two-stage object detector was based on a CNN architecture, known as R-CNN and developed in [10]. This work set the momentum for an unprecedented speed of evolution on object detection research. The R-CNN model uses a Selective Search algorithm to propose approximately 2.000 possible Regions of Interest (RoI). The individual RoI is warped to a size of 227x227 and served as input to a CNN with five convolutional layers and two fully connected layers. The network predicts the bounding box offsets and an SVM classifier predicts the object category. R-CNN requires 84 hours to train on the ImageNet dataset and processes one image in 47 seconds (see [10] for further details). Since then, other works have followed to improve the accuracy and reduce the inference time including the Feature Pyramid Networks (FPN; another two-stage model [7, 11]), or the multiscale one-stage object detectors, known as You Only Look Once (YOLO). Newer versions of YOLO and other networks such as RetinaNet, RefineDet and Deconvolutional Single Shot Detector (DSSD) have gained popularity due to the attractive trade-off between real-time performance and accuracy. The YOLO methodology is to apply a single neural network to the input image, divide the image into a grid and perform the predictions for bounding box and classification at the same time. The YOLO framework was updated in [12] by adopting best-practices from other models, including residual blocks, anchor boxes from Fast R-CNN and multi-scale detection from FPN, to create YOLOv3.

Agility in processing information is crucial for many applications. However, for autonomous vehicles or any ADAS system, a fast inference time ensures the vehicle can interpret and react to the environment in a timely-safe manner [13, 10]. FIGURE 2 compares popular object detection models mean average precision (mAP) on the COCO dataset and their inference time in milliseconds measured on a NVidia Titan X GPU. The green area on the graph represents the real-time requirement, considered as 33 *ms* or 30 FPS. The dependency between higher inference time and accuracy is quite clear from the graph, as the models on the far right obtain higher AP. To the best of knowledge, YOLOv3 is the model that achieves higher AP while maintaining the inference speed

below 33 ms. Latest versions of YOLO (i.e., YOLOv4 and YOLOv5) were not included in this analysis, as at the time of running these experiments they were not available.

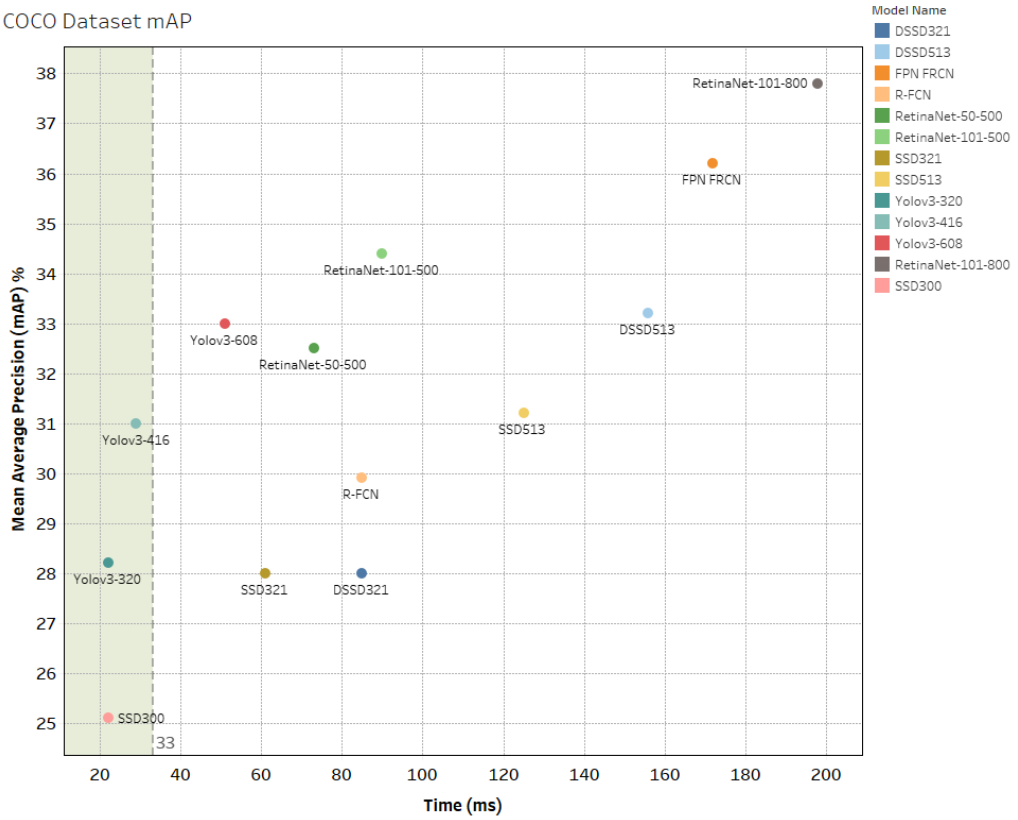
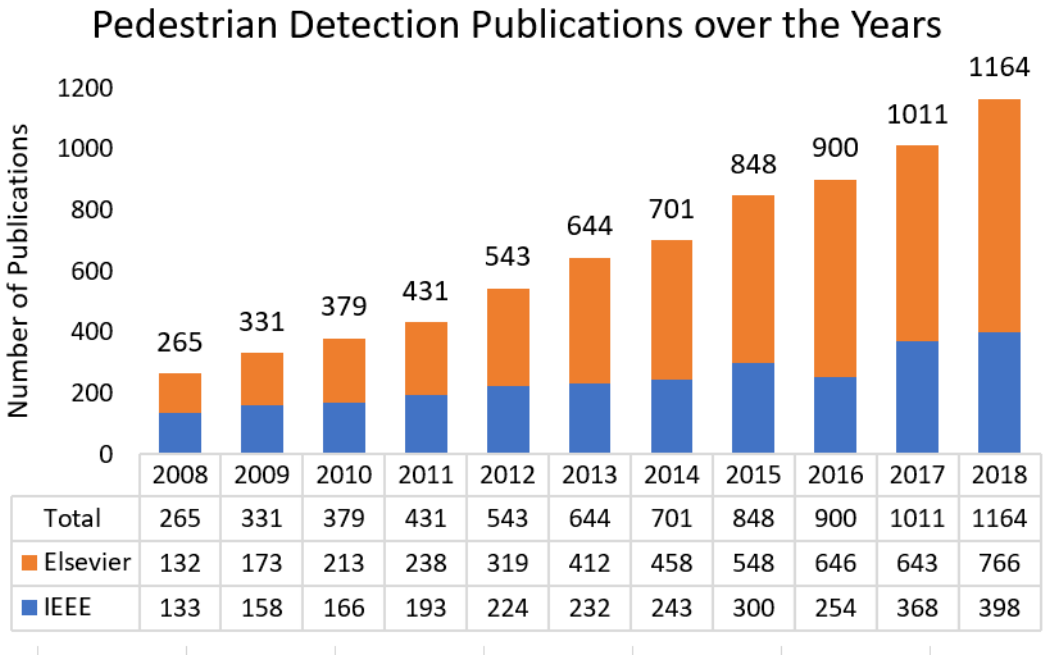


Figure 2: Comparison of popular object detection models on mean Average Precision and inference times on the COCO dataset. If present, the last three digits after the model name represent input size in pixels. Data extracted from [5, 14, 15].

2.2 Pedestrian Detection

Pedestrian detection is a popular research topic and under continuous growth, as demonstrated in FIGURE 3. The graph displays the number of articles published by IEEE and Elsevier regarding this subject each year. Compared to 2008, the number of publications has grown over 400 %.

To this day, it remains a challenging task to adapt the model to the various situations a vehicle may encounter a pedestrian on the road [16]. State that nocturnal scenes, shadow areas, sunlight on the camera and low-resolution images can impact the detection accuracy. Accessories, bags or any other kind of unusual occlusions make difficult to the model to recognise the pedestrian if it has not been prepared to deal with these situations [17]. It is also challenging to define the individual bounding boxes in a crowded scene, as pedestrian features are partially occluded or overlapped [16, 18]. [19, 20], have also highlighted concerns regarding False Positive readings as it can cause the



	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018
Total	265	331	379	431	543	644	701	848	900	1011	1164
Elsevier	132	173	213	238	319	412	458	548	646	643	766
IEEE	133	158	166	193	224	232	243	300	254	368	398

Figure 3: Total number of articles related to ”Pedestrian Detection” published by year, considering IEEE and Elsevier publishers.

vehicle to execute a drastic manoeuvre, and, for this reason, it is crucial to focus on increasing the True Positives as well as reducing the False Positives (see also [21] for a general scenario-based testing for safety validation of AV systems).

In order to cope with the specific challenges of pedestrian detection, researchers have developed modifications to the original object detection frameworks. [22] modified Faster R-CNN by adding an attention network that responds to different occlusion patterns and re-weights the CNN network accordingly. At the time it was published, this approach achieved state-of-the-art results on the Caltech dataset. A novel occlusion-aware Faster R-CNN was proposed by [23], and achieved top results on CityPersons and ETH benchmark datasets. To improve the detection of occluded pedestrians, they formulated a new loss function that improved the region proposal module. Additionally, they replaced the ROIpooling with a pooling unit that contained a separate CNN, which predicted the pedestrian visibility score.

Lan et al. [22], proposed an enhancement of the YOLOv2 framework for real-time pedestrian detection. From the original YOLOv2 model, they moved the pass-through layer to an earlier position in the network and also added three more pass-through layers to improve the network miss rate results on the INRIA dataset by 1.3 %.Zhang S, et al. [24], proposed another modification in the YOLOv2 framework. They added a dense block of connected convolutional layers to improve the network capability for feature extraction and have also implemented spatial pyramid pooling. Compared to YOLOv2, the mAP of DC-SPP-YOLO is 1.6 % more accurate on the PASCAL VOC 2007 dataset. A comparative study between Faster RCNN and SSD was performed by [25] focusing on pedestrian detection datasets. SSD outperforms Faster RCNN in four out of five datasets. Shao

et al. [26], presented an FPN network pre-trained on their CrowdHuman dataset and currently hold the state-of-the-art for Caltech and CityPersons dataset. Their work serves as an example of how valuable pre-training with quality data can make a significant contribution to achieving better results, even with a standard object detection framework. By examining these recent works, we can conclude that R-CNN and FPN based networks achieve higher scores on pedestrian detection benchmarks, and where speed is not a constraint. For real-time pedestrian applications, SSD and YOLO seem to be the most viable solutions. No study was found that directly compares SSD and YOLOv3 for pedestrian detection applications at the time of this paper being submitted.

2.3 You Only Look Once – YOLOv3

The YOLO model is a neural network capable of predicting the bounding boxes around the objects and their class in a single pass. The first version of YOLO did not reach the same accuracy of R-CNN or other two-stage methods [27]. Newer versions of the algorithm, YOLOv2 [28] and YOLOv3 [5], addressed the accuracy issue at the cost of inference speed. While the first version achieved 45 FPS for a 448x448 input, YOLOv3 achieves 34 FPS. A brief overview of the YOLOv3 architecture is presented below.

The YOLOv3 model is based on Darknet-53. Convolution operations with residual blocks are performed until the first scale detection occurs on layer 82 (stride of 32). For the second detection, the network routes back to layer 79, goes through a 1x1 convolution layer and the resulting feature map is upsampled by 2. Layer 61 and the upsampled feature map, layer 85, are depth concatenated together, which means merging the feature maps of both layers to create the deeper layer 86. The advantage of depth concatenation is the fact that a previous layer may hold information that could have been lost along with the network [29]. The output of the second detection occurs on layer 94. This layer has a stride of 16. The resulting feature map size is double the size of first scale. The process repeats for the third scale detection. The network routes back to layer 91, upsamples the feature map by 2x and depth concatenates it with layer 36. With this method, the third detection benefits from the fine-grained features that are detected earlier in the network as well as all the previous computations [5]. The simplified diagram of YOLOv3 is illustrated in FIGURE 4. Further information regarding the YOLOv3 framework can be found in the original release technical report [5].

3. METHODOLOGY

3.1 Dataset

The dataset is composed of traffic scenes collected from driving around the Coventry city, UK, and it consists of of ten video clips of one-minute drives each. The data was collected and provided by Coventry University. Overall, the dataset is challenging for pedestrian detection due to its size, sunlight reflections in the camera, shadow areas, crowded scenes and a mixture of small and medium size pedestrians to be detected.

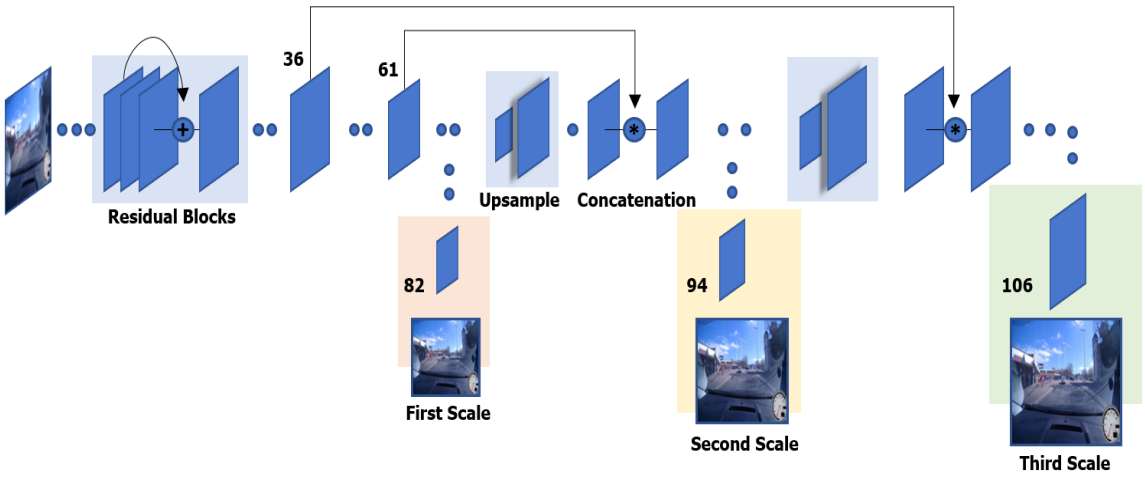


Figure 4: YOLOv3 model diagram representing the residual blocks, different scales of detection, upsampling and concatenation operations.

The first step was to convert the videos into images. Each video second contains 60 frames, or images, that can be extracted. It was decided to obtain one image for every ten frames to achieve a reasonable number of images that were slightly different from each other. For every video, 362 frames were created, resulting in a total of 3.620 images.

With the images ready as described, it was possible to analyse the data. The first conclusion was that it would not be possible to use all the images, as some images had a very high shine or shadow areas that rendered difficult to accurately label pedestrians. To generate the bounding-boxes for these situations would eventually result in inaccurate or inconsistent labels, and for this reason, these images were ignored. As stated by [12], data with inaccurate labelling can limit model performance. FIGURE 5 are examples of images that were removed as it was not possible to accurately label the pedestrians, red arrows indicating were pedestrians are located. The final dataset comprises 2.007 images with 7.704 detections and 99 images where no pedestrian is present. From this point forward the dataset created on this project is referred to as CV-01 dataset.

3.2 Training and Testing

The proportion of data reserved for training, validation and testing was inspired by the methodology used in the KITTI and Caltech datasets [30, 31]. The KITTI dataset contains 7.481 training and 7.518 for testing images, allowing the researcher to define the validation set size. The Caltech dataset also uses a similar ratio, with 60 % training and 40 % testing.

For both datasets, Caltech and KITTI, the training set does not contain images from videos that are used in the test set, and vice-versa [32]. The data was divided according to TABLE 1 to be in accordance to AVs datasets practice. Frames from Video 2 were reserved to the validation set, and frames from Video 1 were used in the test set. The remaining images were used to train the model. The final data ratio is approximately 65 % training, 15 % validation and 20 % test set.

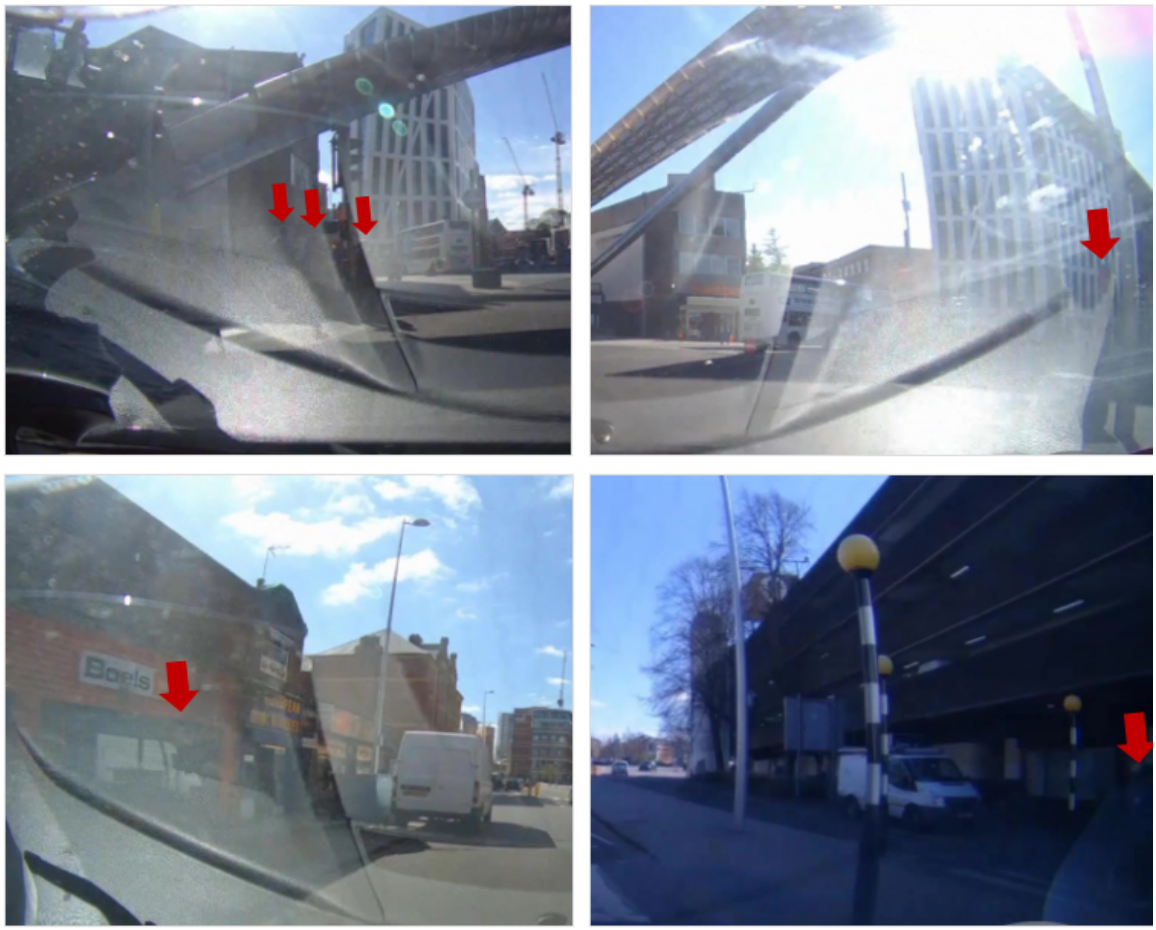


Figure 5: Examples of images were it was not possible to fully define the pedestrian location and which were not added to the dataset.

Table 1: Training, Validation AND Test Set Split.

Set	***Videos Used***	***Number of Images***
Training	Videos 3 through 10	1,313
Validation	Video 2	332
Test	Video 1	362

3.3 YOLO-V3 Model Variations

As a preliminary test, the original YOLOv3 baseline was far from the necessary requirements in terms of accuracy. The next step of the process was to optimise the model according to the application. The possible improvements to the baseline YOLOv3 model were identified by analyzing previous works, such as [22, 24], where the YOLO architecture was modified to improve the

accuracy. In addition, due to the small size of the dataset, data augmentation could also potentially provide improvements. From [6], larger input dimensions have indicated to improve accuracy to the cost of performance (see also [33]). In conclusion the main identified tasks that can lead to improvements were:

1. Increase input size;
2. Customize anchors;
3. Customize the number of detections layers;
4. Modify feature maps output size;
5. Data augmentation and hyperparameter tuning.

The modifications were tested separately to allow a better understanding of how much they contributed to the accuracy and speed of the baseline network. In total, eleven different models were created, and their individual outcomes are shown in Section 4. TABLE 2 provides a brief description of the developed models.

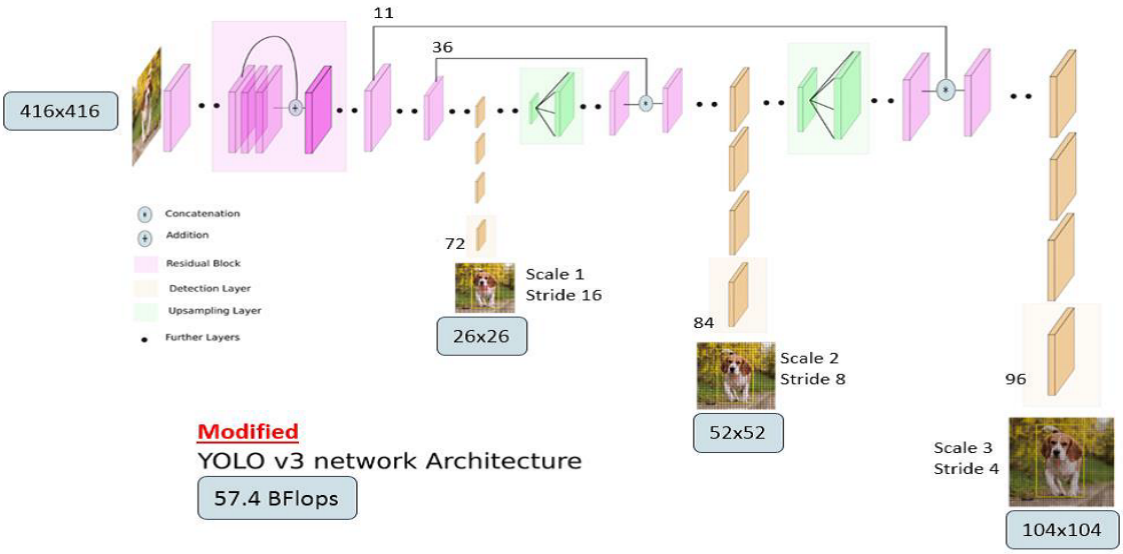


Figure 6: Modified YOLOv3 architecture for this study

FIGURE 6 illustrates the modified YOLOv3 for this study. From the test results of Model-10, it was possible to conclude that the first detection layers (First scale) were not contributing to the final model AP. The removal of the first scale detection layers reduced the computational cost and improved the model accuracy. This inspired us to discard the original first detection layer and add a new third detection layer. As a result, Model-13 was built and, if compared to the original model, the modified network contains larger feature maps as outputs to improve the detection of smaller objects while maintaining a similar computational cost. TABLE 2 shows the structure of the original YOLOv3 network, and the modified model proposed in this study for a 416x416 input size. The

final models were originated from Model-13. Model-15 and Model-16 have the architecture shown in TABLE 2, however, they differ in the number of anchors, i.e. 9 and 12, respectively.

The anchors' dimensions were customised to the proportions of the pedestrian bounding boxes of the CV-01 training set using the k-means algorithm, as done in the original YOLOv3 implementation. TABLE 4 shows the anchor dimensions. Twelve anchors provided a better result than the default nine anchors used by YOLOv3. All models used pre-trained weights from the COCO dataset, available at [34]. A preliminary test was performed without the pre-trained weights, and after four hours (1.000 iterations) a 0 % average precision was obtained. Speeding up training and improving the accuracy were the main reasons to use the pre-trained weights.

Table 2: Description of Trained Models.

Model Name	***Description***
Model-05	Original YOLOv3 architecture, with 416x416 input and COCO default 9 anchors dimension.
Model-06	Original YOLOv3 architecture, with 608x608 input and COCO default 9 anchors dimension.
Model-02	Original YOLOv3 architecture, with 832x832 input and COCO default 9 anchors dimension.
Model-10	Modified YOLOv3 architecture with only Second and Third Detection Scales (removed First Detection Scale), with 416x416 input and COCO default anchors dimension.
Model-08	Modified YOLOv3 architecture with additional Fourth Detection Scale (with custom anchors), with 416x416 input. First three scales with COCO default 9 anchors dimension.
Model-09	Modified YOLOv3 architecture with additional Fourth and Fifth Detection Scales (with custom anchors), with 416x416 input. First three scales with COCO default 9 anchors dimension.
Model-12	Original YOLOv3 architecture, with416x416 input and 9 customised anchors dimension.
Model-11	Original YOLOv3 architecture, with416x416 input and 12 customised anchors dimension.
Model-13	Modified YOLOv3 Architecture, see Table 3.
Model-15	Model-13, with step Learning Rate and hyperparameters tuning with 9 customised anchors
Model-16	Model-13, with step Learning Rate and hyperparameters tuning with 12 customised anchors

Initially, the models were trained for 10.000 iterations (500 epochs). No improvements were noticed after 4.000 iterations, or 300 epochs, and this was defined as the maximum iteration number. During

Table 3: Final Model - Modified YOLO Architecure.

Original YOLOv3								Modified YOLOv3 for this Project							
Layer	Layer Type	No. filters	Size / Stride	input	output	Flops		Layer	Layer Type	No. filters	Size / Stride	input	output	Flops	
0	conv	32	3 x 3/ 1	416 x 416 x 3	416 x 416 x 32	0.299 BF		0	conv	32	3 x 3/ 1	416 x 416 x 3	416 x 416 x 32	0.299 BF	
1	conv	64	3 x 3/ 2	416 x 416 x 32	208 x 208 x 64	1.595 BF		1	conv	64	3 x 3/ 2	416 x 416 x 32	208 x 208 x 64	1.595 BF	
2	conv	32	1 x 1/ 1	208 x 208 x 64	208 x 208 x 32	0.177 BF		2	conv	32	1 x 1/ 1	208 x 208 x 64	208 x 208 x 32	0.177 BF	
3	conv	64	3 x 3/ 1	208 x 208 x 32	208 x 208 x 64	1.595 BF		3	conv	64	3 x 3/ 1	208 x 208 x 32	208 x 208 x 64	1.595 BF	
4	Shortcut Layer	1						4	Shortcut Layer	1					
5	conv	128	3 x 3/ 2	208 x 208 x 64	104 x 104 x 128	1.595 BF		5	conv	128	3 x 3/ 2	208 x 208 x 64	104 x 104 x 128	1.595 BF	
6	conv	64	1 x 1/ 1	104 x 104 x 128	104 x 104 x 64	0.177 BF		6	conv	64	1 x 1/ 1	104 x 104 x 128	104 x 104 x 64	0.177 BF	
7	conv	128	3 x 3/ 1	104 x 104 x 64	104 x 104 x 128	1.595 BF		7	conv	128	3 x 3/ 1	104 x 104 x 64	104 x 104 x 128	1.595 BF	
8	Shortcut Layer	5						8	Shortcut Layer	5	5				
9	conv	64	1 x 1/ 1	104 x 104 x 128	104 x 104 x 64	0.177 BF		9	conv	64	1 x 1/ 1	104 x 104 x 128	104 x 104 x 64	0.177 BF	
10	conv	128	3 x 3/ 1	104 x 104 x 64	104 x 104 x 128	1.595 BF		10	conv	128	3 x 3/ 1	104 x 104 x 64	104 x 104 x 128	1.595 BF	
11	Shortcut Layer	8						11	Shortcut Layer	8					
12	conv	256	3 x 3/ 2	104 x 104 x 128	52 x 52 x 256	1.595 BF		12	conv	256	3 x 3/ 2	104 x 104 x 128	52 x 52 x 256	1.595 BF	
13	conv	128	1 x 1/ 1	52 x 52 x 256	52 x 52 x 128	0.177 BF		13	conv	128	1 x 1/ 1	52 x 52 x 256	52 x 52 x 128	0.177 BF	
14	conv	256	3 x 3/ 1	52 x 52 x 128	52 x 52 x 256	1.595 BF		14	conv	256	3 x 3/ 1	52 x 52 x 128	52 x 52 x 256	1.595 BF	
15	Shortcut Layer	12						15	Shortcut Layer	12					
16	conv	128	1 x 1/ 1	52 x 52 x 256	52 x 52 x 128	0.177 BF	7x	16	conv	128	1 x 1/ 1	52 x 52 x 256	52 x 52 x 128	0.177 BF	7x
17	conv	256	3 x 3/ 1	52 x 52 x 128	52 x 52 x 256	1.595 BF		17	conv	256	3 x 3/ 1	52 x 52 x 128	52 x 52 x 256	1.595 BF	
36	Shortcut Layer	33						36	Shortcut Layer	33					
37	conv	512	3 x 3/ 2	52 x 52 x 256	26 x 26 x 512	1.595 BF		37	conv	512	3 x 3/ 2	52 x 52 x 256	26 x 26 x 512	1.595 BF	
38	conv	256	1 x 1/ 1	26 x 26 x 512	26 x 26 x 256	0.177 BF		38	conv	256	1 x 1/ 1	26 x 26 x 512	26 x 26 x 256	0.177 BF	
39	conv	512	3 x 3/ 1	26 x 26 x 256	26 x 26 x 512	1.595 BF		39	conv	512	3 x 3/ 1	26 x 26 x 256	26 x 26 x 512	1.595 BF	
40	Shortcut Layer	37						40	Shortcut Layer	37					
41	conv	256	1 x 1/ 1	26 x 26 x 512	26 x 26 x 256	0.177 BF	7x	41	conv	256	1 x 1/ 1	26 x 26 x 512	26 x 26 x 256	0.177 BF	7x
42	conv	512	3 x 3/ 1	26 x 26 x 256	26 x 26 x 512	1.595 BF		42	conv	512	3 x 3/ 1	26 x 26 x 256	26 x 26 x 512	1.595 BF	
61	Shortcut Layer	58						61	Shortcut Layer	58					
62	conv	1024	3 x 3/ 2	26 x 26 x 512	13 x 13 x1024	1.595 BF		62	conv	512	1 x 1/ 1	26 x 26 x 512	26 x 26 x 512	0.354 BF	
63	conv	512	1 x 1/ 1	13 x 13 x1024	13 x 13 x 512	0.177 BF		63	conv	256	1 x 1/ 1	26 x 26 x 512	26 x 26 x 256	0.177 BF	
64	conv	1024	3 x 3/ 1	13 x 13 x 512	13 x 13 x1024	1.595 BF		64	route	63 61					
65	Shortcut Layer	62						69	conv	256	1 x 1/ 1	26 x 26 x 768	26 x 26 x 256	0.266 BF	3x
66	conv	512	1 x 1/ 1	13 x 13 x1024	13 x 13 x 512	0.177 BF	3x	70	conv	512	3 x 3/ 1	26 x 26 x 256	26 x 26 x 512	1.595 BF	
67	conv	1024	3 x 3/ 1	13 x 13 x 512	13 x 13 x1024	1.595 BF		71	conv	24	1 x 1/ 1	26 x 26 x 512	26 x 26 x 24	0.017 BF	
74	Shortcut Layer	71						72	yolo						
79	conv	512	1 x 1/ 1	13 x 13 x1024	13 x 13 x 512	0.177 BF	3x	73	route	69					
80	conv	1024	3 x 3/ 1	13 x 13 x 512	13 x 13 x1024	1.595 BF		74	conv	128	1 x 1/ 1	26 x 26 x 256	26 x 26 x 128	0.044 BF	
81	conv	18	1 x 1/ 1	13 x 13 x1024	13 x 13 x 18	0.004 BF		75	upsample	2x		26 x 26 x 128	52 x 52 x 128		
82	yolo							76	route	75 36					
83	route	79						81	conv	128	1 x 1/ 1	52 x 52 x 256	52 x 52 x 128	0.177 BF	3x
84	conv	256	1 x 1/ 1	13 x 13 x 512	13 x 13 x 256	0.044 BF		82	conv	256	3 x 3/ 1	52 x 52 x 128	52 x 52 x 256	1.595 BF	
85	upsample	2x		13 x 13 x 256	26 x 26 x 256			83	conv	24	1 x 1/ 1	52 x 52 x 256	52 x 52 x 24	0.033 BF	
86	route	85 61						84	yolo						
91	conv	256	1 x 1/ 1	26 x 26 x 512	26 x 26 x 256	0.177 BF	3x	85	route	81					
92	conv	512	3 x 3/ 1	26 x 26 x 256	26 x 26 x 512	1.595 BF		86	conv	128	1 x 1/ 1	52 x 52 x 128	52 x 52 x 128	0.089 BF	
93	conv	18	1 x 1/ 1	26 x 26 x 512	26 x 26 x 18	0.012 BF		87	upsample	2x		52x 52 x 128	104 x 104 x 128		
94	yolo							88	route	87 11					
95	route	91						93	conv	64	1 x 1/ 1	104 x 104 x 128	104 x 104 x 64	0.177 BF	3x
96	conv	128	1 x 1/ 1	26 x 26 x 256	26 x 26 x 128	0.044 BF		94	conv	128	3 x 3/ 1	104 x 104 x 64	104 x 104 x 128	1.595 BF	
97	upsample	2x		26 x 26 x 128	52 x 52 x 128			95	conv	24	1 x 1/ 1	104 x 104 x 128	104 x 104 x 24	0.066 BF	
98	route	97 36						96	yolo						
103	conv	128	1 x 1/ 1	52 x 52 x 256	52 x 52 x 128	0.177 BF	3x								
104	conv	256	3 x 3/ 1	52 x 52 x 128	52 x 52 x 256	1.595 BF									
105	conv	18	1 x 1/ 1	52 x 52 x 256	52 x 52 x 18	0.033 BF									
106	yolo														
Total	BFLOPS	65.296						Total	BFLOPS	57.432					

Table 4: Anchor boxes dimensions and the respective detection scales.

Anchor	***1st Scale***	***2nd Scale***	***3rd Scale***
1	30x145	13x83	4x18
2	35x175	15x61	5x28
3	40x200	19x96	7x40
4	59x300	28x129	10x57

the training, the accuracy sharply oscillated at initial training iterations, especially around 1.000 and 2.000 iterations. For this reason, the learning rate was reduced to 1.000, 2.000 and 3.000 epochs by a factor of 0.1, 0.075 and 0.05, respectively. A larger learning rate is used during the first epochs, as recommended by [35, 36]. The final model parameters are presented in TABLE 5.

Table 5: Hyperparameters of the final model.

Parameter	*Value*
Initial Learning Rate	0.01
Weight Decay	0.0005
Momentum	0.9
Optimizer	Stochastic Gradient Descent (SGD)
Batch Size	64
Training Iterations	4.000

3.4 Hardware and Implementations

Nvidia Titan X GPU is commonly used to measure the speed of recent real-time object detection algorithms [5, 19, 37]. YOLOv3 authors state that the model is capable of achieving 34.4 FPS (29 ms) on a Titan X GPU for a 416x416 pixels input. The same YOLOv3 model achieved 16.1 FPS (62 ms) on a Virtual Machine with Tesla K80 GPU, which is 2.13 times slower. The speed results reported on this project are multiplied by a factor of 2.13 to simulate the results on a Titan X GPU, the same hardware as the original authors.

The original YOLOv3 source code is available to download from the author's website [34]. It is originally built to run on a Linux machine and written in C++ language. The author also provides the pre-trained weights from the COCO dataset.

4. RESULTS

This section presents the results of the developed pedestrian detection model on the CV-01 dataset. As already mentioned, several characteristics of the YOLOv3 architecture were analysed separately to compose the final model. First, we present a brief overview of the metrics used to measure the model performance. Next, the test set results regarding the input size variations are presented, followed by the number of anchors, number of layers, data augmentation/hyperparameters and lastly the final model results.

Similar to [22, 24, 5], our results are measured using Average Precision (AP). The Average Precision metric was first introduced by the PASCAL Visual Object Classes (VOC) challenge in 2007 [4]. The AP metric is defined as the area under the Precision-Recall curve. The definition of True Positives (TP) or False Negatives (FN) in the context of object detection is according to the Intersection over Union (IoU) threshold. The IoU measures the overlap between the predicted and ground-truth bounding box areas. Usually, a detection is considered as a true positive (TP) if the area of the predicted bounding box (B_p) and ground-truth bounding box (B_t) overlap by a value greater than

0.5 [38]. The IOU is calculated according to Eq. (1).

$$IoU = \frac{area\ B_p \cap B_t}{area\ B_p \cup B_t} \quad (1)$$

To plot the Precision-Recall curve, the samples are ranked by the classification confidence in decreasing order [38]. The IoU result dictates if the prediction is accurate enough to be considered as correct (TP). Precision, Recall and IoU are calculated for each sample. Currently, the AP can be calculated according to Pascal VOC or the Microsoft's Common Objects in Context (COCO) dataset [11]. Both methods use the Precision-Recall curve as reference, however Pascal VOC computes the area under the Precision-Recall curve using a fixed IoU of 0.5 [38]. On this work we have also applied the Pascal VOC method, as it is also used by the original YOLOv3 authors.

TABLE 6 shows the results obtained with the original YOLOv3 model under different input sizes. For this test case, the 832x832 input dimension achieves higher Average Precision, but it only runs at 10 FPS.

Table 6: YOLOv3 AP and inference performance with three different input sizes.

Model Name	***Description***	***Test AP***	***FPS***
Model-05	YOLOv3 with 416x416 input	42.18 %	32
Model-06	YOLOv3 with 608x608 input	50.63 %	17.2
Model-02	YOLOv3 with 832x832 input	67.42 %	10.2

The results that follow use the 416x416 model, i.e. Model-05, as a baseline, since it was the only one capable of achieving the speed requirement of 30 FPS. By default, the YOLOv3 network uses nine anchors with dimensions to fit the bounding boxes of the COCO dataset. The results in TABLE 7 compares the model with default anchors, Model-05, with Model-12 and Model-11, where nine and twelve customised anchor dimensions were used, respectively. Customizing the anchors to the CV-01 dataset improves the accuracy with minor impact on inference speed.

Table 7: YOLOv3 AP and inference performance with nine and twelve customized anchors.

Model Name	***Description***	***Test AP***	***FPS***
Model-05	YOLOv3 baseline	42.18 %	32
Model-12	YOLOv3 with 9 custom. anchors	49.49 %	31.2
Model-11	YOLOv3 with 12 custom. anchors	43.62 %	31.4

TABLE 8 presents three different models together with the baseline, Model-05. With only two layers, Model-10 obtained higher accuracy. Models with more than three layers did not achieve higher accuracy and achieved longer inference times.

Table 8: YOLOv3 AP and inference performance with two, four and five detection layers.

Model Name	***Description***	***Test AP***	***FPS***
Model-05	YOLOv3 baseline	42.18 %	32
Model-10	YOLOv3 with 2 Detection Layers	43.57 %	32.2
Model-08	YOLOv3 with 4 Detection Layers	41.92 %	27
Model-09	YOLOv3 with 5 Detection Layers	43.52 %	17.8

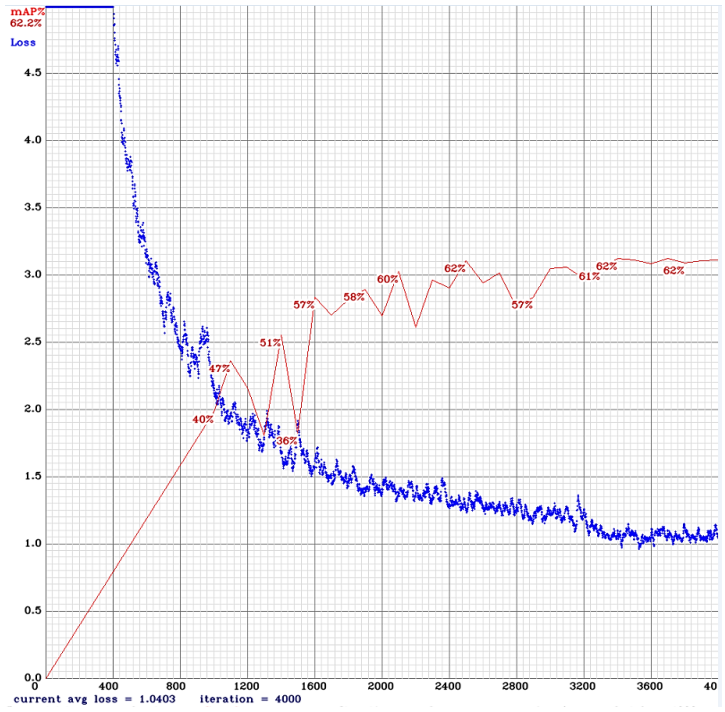
The lessons learned from the results presented so far inspired the architecture described in TABLE 2 Model-13 shows significant improvements over the original YOLOv3 framework, as demonstrated in TABLE 9. Further enhancements were made on Model-15 and Model-16 by applying the step learning rate and fine-tuning the hyperparameters. Model-16 outperforms Model-15 by 1.44 %, and Model-13 by 2.09 % on the test set.

Table 9: YOLOv3 AP and inference performance with the modified architecture and step Learning Rate.

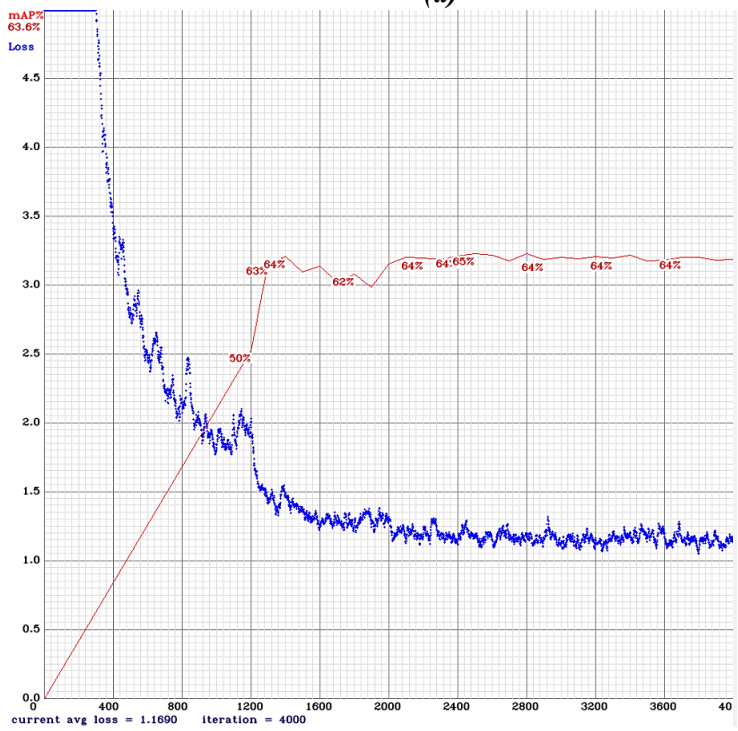
Model Name	***Description***	***Test AP***	***FPS***
Model-05	YOLOv3 baseline	42.18 %	32
Model-13	Modified YOLOv3 Architecture	49.50 %	32.2
Model-15	Model-13 + LR + 9 custom anchors	50.15 %	31.6
Model-16	Model-13 + LR + 12 custom anchors	51.59 %	30.4

TABLE 10 shows the beneficial effect of data augmentation. Model-17 is the same as Model-16 but without any data augmentation settings. The data augmentation doubles model accuracy on the validation and test sets.

FIGURE 7 illustrates the comparison of the training set loss and validation set average precision during training for Model-13 and Model-16. The main difference between the two models is the application of the step learning rate. On the left, Model-13 shows in red a very sharp and unstable accuracy curve during the learning process. A controlled learning rate promoted a smoother accuracy curve, faster convergence and improved accuracy, as shown by Model-16 in Figure 7 (b).



(a)



(b)

Figure 7: Validation set accuracy and training loss graphs for Model-13 (a) and Model-16 (b).

Table 10: YOLOv3 AP and inference performance with and without data augmentation.

Model Name	***Description***	***Validation AP***	***Test AP***
Model-16	Model-13+12 anc.	65.00 %	51.59 %
Model-17	Model-16 without data augmentation	28.74 %	22.79 %

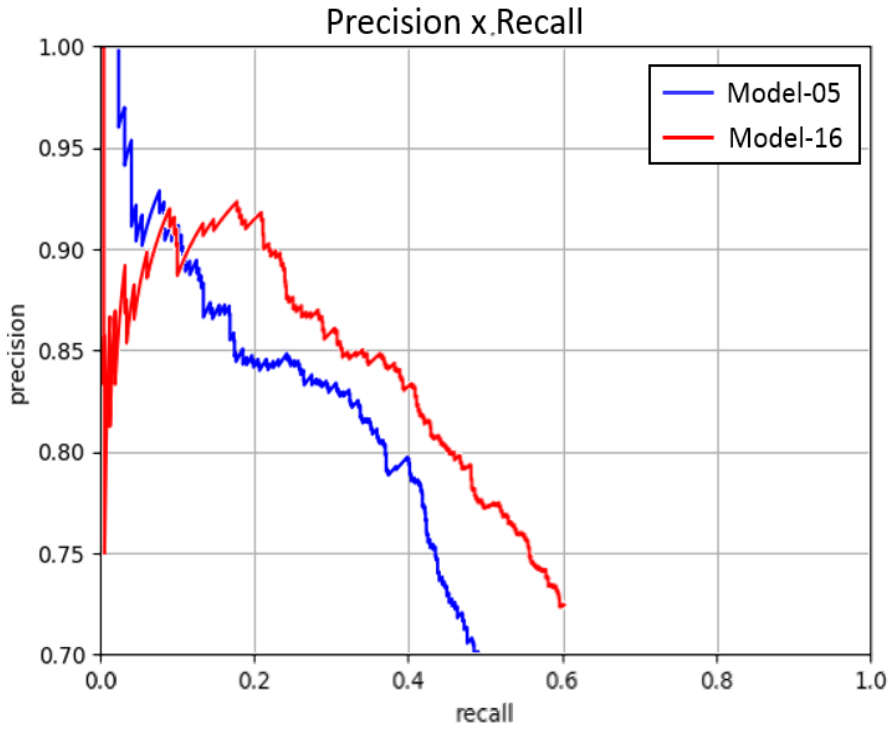


Figure 8: Precision and Recall curve for Model-05 and Model-16. The Average Precision metric is the area under the curve for each model.

The Precision and Recall curve from Model-05 and Model-16 are presented in FIGURE 8. As expected, Model-16 obtained a higher AP as the red Precision-Recall curve covers a larger area than Model-05.

TABLE 11 lists the overall results over different metrics for all models. The top-three results for each metric are highlighted in bold to facilitate discussion and visualisation. All metrics refer to the validation set results but the Test AP column. Model-02 outperformed all others with regards to accuracy-related measures, but it is not capable of reaching real-time performance. Model-16 presented the best trade-off between accuracy across validation/test sets and inference speed.

FIGURE 9 through FIGURE 11 demonstrates the detections examples extracted from the test and validation sets. Ground-truth boxes are shown in yellow, and detections from Model-16 are displayed in red. It is possible to notice that due to the shadow area near the bottom of FIGURE 9, it causes the model to predict a larger box than required. Most of the pedestrians were detected in FIGURE 10, however overall lower IoU and two false positives are noticed. The high glare and shadow of FIGURE 11 caused the model to miss three out of the nine pedestrians of the frame. Additionally, two false positives are present on the right side of the image.

5. DISCUSSION

From the test results obtained, it was possible to analyse several characteristics of the YOLOv3 model. The first was the high correlation between accuracy and input size. Model-02, with an input size of 832x832 and no further modifications to the original framework, almost achieved the accuracy requirement of 70 %. This is an indication that the limitation is not on the algorithm, but on the hardware, and will probably be overcome in the following years as the technology rapidly advances.

False positives are a big concern in autonomous driving, as highlighted by [19, 20]. All top-three models highlighted in TABLE 11 presented Precision above 80 %. The model with the highest number of false positives was Model-12, with nine customised anchors. One explanation is that the anchors' proportions were too fitted to the pedestrian dimensions of the training set, and this limited the model capabilities to generalise to other scenarios. This argument aligns with the fact that Model-11, containing twelve anchors, presented half the number of false positives if compared to Model-12. With more anchors, the network could generalise across the training, validation and test set. Model-16 uses the concepts from Model-11 and Model-13, and, consequently, it presented a good balance between the FP, TP and FN metrics. The tests have shown that FP and anchors' selection are related. Model-16 shows a 20 % reduction on false positives if compared to the original YOLOv3, Model-05.

The models with four and five detection layers, Model-08 and Model-09, respectively, have not shown much accuracy improvements on the test set AP. Besides, these models were not capable of reaching real-time speed requirements. Surprisingly, Model-10, where the detection layer for larger objects was removed had a better performance than the baseline Model-05. A plausible reason for these results can be found in the work from [39]. The authors describe those deeper models can present worse accuracy than shallow networks, mainly because complex models are harder to optimise.

TABLE 10 compares the results of the same model with and without data augmentation. Especially for small datasets, data augmentation is a mandatory strategy to avoid overfitting, and increase the number of samples [40]. Data augmentation is responsible for doubling the AP results on the validation and test set, demonstrating the importance of this task. FIGURE 7 presents the training loss and validation set average precision curve for Model-16 and Model-13. In FIGURE 7(a), the fixed learning rate causes the accuracy to oscillate heavily, as an indication that the learning rate is too large and cause divergence [35]. As a consequence, it presents more iterations until stabilisation. With the step LR strategy, shown in FIGURE 7(b), the final model stabilised around 2.000 iterations and achieved a better accuracy on the validation and test set. The results are in accordance with

Table 11: Overall results of all models, showing Validation and Test set results. The Precision, Recall, TP, FP and FN metrics are from the Validation set

Model Name	***Description***	***Precision***	***Recall***	***TP***	***FP***	***FN***	***Validation AP***	***Test AP***	***FPS***
Model-05	YOLOv3 baseline	73.0 %	40.0 %	848	319	1261	44.63 %	42.18 %	32
Model-06	YOLOv3 608x608 input	82.0 %	52.0 %	1106	246	1003	61.26 %	50.63 %	17.2
Model-02	YOLOv3 832x832 input	87.0 %	59.0 %	1248	186	861	69.61 %	67.42 %	10.2
Model-10	YOLOv3 - 2 Detection Layers	75.0 %	36.0 %	763	251	1346	43.08 %	43.57 %	32.2
Model-08	YOLOv3 - 4 Detection Layers	78.0 %	44.0 %	925	265	1184	55.86 %	41.92 %	27
Model-09	YOLOv3 - 5 Detection Layers	79.0 %	37.0 %	775	201	1334	55.08 %	43.52 %	17.8
Model-12	YOLOv3 - 9 custom. anchors	66.0 %	44.0 %	929	486	1180	43.70 %	49.49 %	31.2
Model-11	YOLOv3 - 12 custom. anchors	79.0 %	40.0 %	834	222	1275	44.53 %	43.62 %	31.4
Model-13	Modified YOLOv3 Architecture	78.0 %	57.0 %	1192	336	917	62.15 %	49.50 %	30.5
Model-15	Model-13 + LR + 9 custom anchors	77.0 %	60.0 %	1269	383	840	64.76 %	50.15 %	31.6
Model-16	Model-13 + LR + 12 custom anchors	82.0 %	57.0 %	1197	257	912	65.00 %	51.59 %	30.4

[41], where a learning rate schedule also offered faster convergence. It is possible to conclude that the strategy applied to the Learning Rate has allowed a better balance between exploration and exploitation.

From all the eight metrics analysed in TABLE 11, Model-16 was one of the top-performing on six of them and achieved 30.4 FPS as per the requirement. Model-16 was selected as the final model because of the good overall accuracy on the validation and test sets while maintaining the real-time requirement. With a 416x416 input size, it was possible to achieve better accuracy than Model-06, the original YOLOv3 with 608x608 input. This result highlights the importance of understanding and customising popular frameworks for the specific application. FIGURES 9 to 11 illustrate a selection of the final model detections against the ground-truth. By analysing the outputs, detecting

Figure 9: Test set frame, an example of detection under the shadow area.

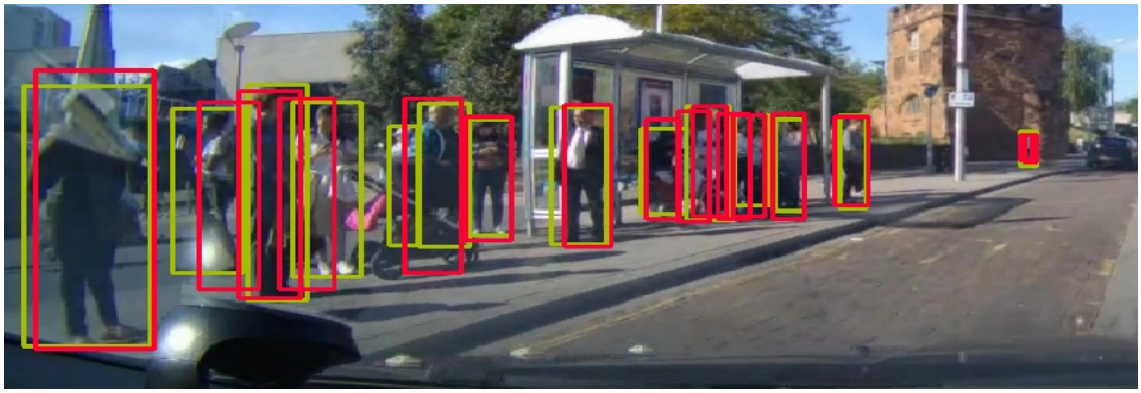


Figure 10: Validation set frame, an example of detection under a crowded scene.

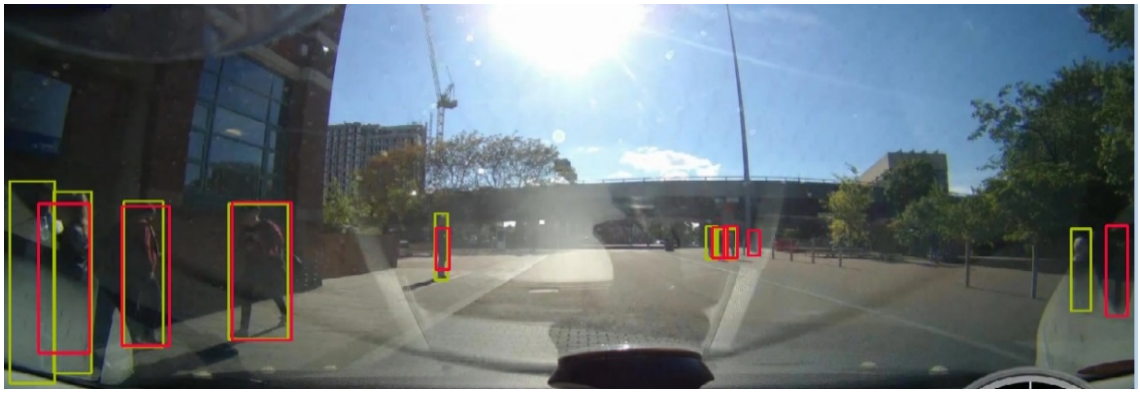


Figure 11: Test set frame, an example of detection of small and large objects on the same scene.

small objects has shown to be a laborious task to the network, as well as the reflections and shadow areas of the traffic scenes. Even with limited samples, the model was capable of learning to detect shapes under challenging situations.

6. CONCLUSION

The main lesson learned is that between the two challenging factors of the dataset, data limitation and low-quality, the first weights the most. Even with low-resolution images, the network is capable of extracting and extrapolating the features that compose a person. The changes to the original framework seemed to reach an accuracy plateau, where only 0.5-1 % improvements started to occur. This indicates that the model reached its potential, and a more significant accuracy leap can only be achieved by drastically changing the methodology or presenting more data. In conclusion, there are two main achievements of the work reported in this paper. The first is the creation of a unique dataset for pedestrian detection from the Coventry city, UK. Secondly, the final model presents a real-time performance of 30.4 FPS and better accuracy than the original YOLOv3 model, by 9.4 % for this application.

7. FUTURE WORK

For future work it is recommended to address the data limitation issue. This can be done by using more advanced data augmentation techniques, where reinforcement learning or other machine learning techniques can be applied for choosing the best data augmentation policy [40]. Another solution is to use Generative Adversarial Networks (GAN's) to produce new samples, as done by [42, 3, 5]. A more in-depth study of how different strategies of updating the learning rate during the training process is also required. Recent works have started to look into more advanced techniques, as described in [36]. Sensor fusion is also an interesting and practical method to improve the accuracy, if radar or LiDAR data is made available for this dataset in the future. Pre-training is extremely important for CNN based object detection models [26, 43]. As a suggestion, pre-train YOLOv3 on a pedestrian dataset instead of using COCO pre-trained weights could be beneficial for the final model [44].

References

[1] https://apps.who.int/iris/bitstream/handle/10665/272722/9789241514187-eng.pdf.

[2] Batsch F, Daneshkhah A, Palade V, Cheah M. Scenario Optimisation and Sensitivity Analysis for Safe Automated Driving Using Gaussian Processes. Applied Sciences. 2021;11:775.

[3] Spooner J, Palade V, Cheah M, Kanarachos S, Daneshkhah A. Generation of Pedestrian Crossing Scenarios Using Ped-Cross Generative Adversarial Network. Applied Sciences. 2021;11:471.

[4] Rolison JJ, Regev S, Moutari S, Feeney A. What Are the Factors That Contribute to Road Accidents? An Assessment of Law Enforcement Views, Ordinary Drivers' Opinions, and Road Accident Records. Accident Analysis & Prevention. 2018;115:11-24.Benson AJ, Tefft BC, Svancara AM, Horrey WJ. Potential reductions in crashes, injuries, and deaths from large-scale deployment of advanced driver assistance systems. Research Brief.2018.

[5] Spooner J, Cheah M, Palade V, Kanarachos S, Daneshkhah A. Generation of Pedestrian Pose Structures Using Generative Adversarial Networks. In 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA).IEEE.2019 :1644-1650.

[6] https://arxiv.org/pdf/1804.02767.pdf

[7] Zou Z, Shi Z, Guo Y, Ye J.'Object Detection in 20 Years: A Survey'. IEEE Transactions on Pattern Analysis and Machine Intelligence.2019:1-38.

[8] Sukanya C, Roopa G, Vince PA. Survey on Object Recognition Methods. International Journal of Computer Science Engineering and Technology. 2016;6:48-52.

[9] Zhao ZQ, Zheng P, Xu St. Wu X. Object Detection With Deep Learning: A Review'. IEEE Transactions on Neural Networks and Learning Systems.2018:1-21.

[10] Girshick RB, Donahue J, Darrell T, Malik J. 'Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation'.2014. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2014, 'CVPR 2014'. held 23-28 June 2014. Providence: IEEE, 580-587.

[11] Lin TY, Maire M, Belongie S, Hays J, Perona P, et al. Microsoft Coco: Common Objects in Context. In European conference on computer vision. Springer, Cham.2014:740-755.

[12] Redmon J, Farhadi A. YOLO9000: Better, Faster, Stronger'. in DiCarlo, J. J, Shum, H. and Jurafsky, D. (eds.) Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2017, 'CVPR 2017'. 2017. Honolulu: IEEE.2017:6517-6525.

[13] https://arxiv.org/abs/1804.02767

[14] Maurer M, Gerdes JC, Lenz B. and Winner, H. (2016). Autonomous Driving: Technical, legal and Social Aspects. 1st ed. Ladenburg: Springer Open.

[15] Zhang X, Hu HM, Jiang F, Li B. 'Pedestrian Detection Based on Hierarchical Co-occurrence Model'. Neurocomputing.2015;168, 861-870.

[16] Brummelen J V, O'Brien M, Gruyer D, Najjaran H. 'Autonomous Vehicle Perception: The Technology of Today and Tomorrow'. Transportation Research Part C.2018;89:384-406.

[17] Liu W, Anguelov D, Erhan , Szegedy C, Reed S, et al. 'SSD: Single Shot MultiBox Detector'. Lecture Notes in Computer Science, Computer Vision – ECCV. 2016;9905:21-37.

[18] Zhang S,Longyin Wen, Xiao Bian, Zhen Lei, Stan Z Li, et al. (2018a) 'Single-Shot Refinement Neural Network for Object Detection'. in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 'CVF 2018'. 18-23 June 2018. Salt Lake City: IEEE.2018; 4203-4212.

[19] Bali S, Tyagi SS. A Review of Vision-Based Pedestrian Detection Techniques'. International Journal of Advanced Studies of Scientific Research. 2018;3:100-104.

[20] https://arxiv.org/abs/1904.03629

[21] Batsch F, Daneshkhah A, Cheah M, Kanarachos S, Baxendale A. Performance Boundary Identification for the Evaluation of Automated Vehicles Using Gaussian Process Classification. In 2019 IEEE Intelligent Transportation Systems Conference (ITSC).IEEE.2019:419-424.

[22] Yang J, Zhang S, Schiele B. Occluded Pedestrian Detection Through Guided Attention in CNNs'. in Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 'CVPR 2018'. held 18-22 June 2018. Salt Lake City: IEEE. 2018;6995-7003.

[23] https://www.wired.com/story/self-driving-cars-uber-crash-false-positive-negative/

[24] Zhang S, Wen L, Bian X, Lei Z, Li SZ, et al. 'Occlusion-Aware R-CNN: Detecting Pedestrians in a Crowd'. in Ferrari V., Hebert M., Sminchisescu C., Weiss Y. (eds.) Proceedings of the European Conference on Computer Vision 2018, 'ECCV 2018'. held 8–14 September 2018. Munich: Springer, Cham.2018:657-674.

[25] Lan, W., Dang, J., Wang, Y. and Wang, S. 'Pedestrian Detection Based on YOLO Network Model'. in Proceedings of the IEEE International Conference on Mechatronics and Automation, 'ICMA 2018'. 2018. Changchun: IEEE. 2018;1547-1551.

[26] https://arxiv.org/ftp/arxiv/papers/1903/1903.08589.pdf

[27] Baabou S.Fradj AB; Farah MA;Abubakr AG, François B, et al. A Comparative Study and State-Of-The-Art Evaluation for Pedestrian Detection'. in Proceedings of the 19th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering.'ISA'. March 2019. Sousse: IEEE. 2019:485-490.

[28] https://arxiv.org/pdf/1805.00123.pdf

[29] Redmon J, Divvala S, Girshick R, Farhadi A. You Only Look Once: Unified, Real-time Object Detection'. in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2016, 'CVPR 2016'. 2016. Las Vegas: IEEE. 2016:779-788.

[30] Szegedy C, Liu W, Jia Y, Sermanet P, Reed Scott, et al. Going Deeper with Convolutions'. in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2015, 'CVPR 2015'.2015. Boston: IEEE.2015:1-9.

[31] Taiana M, Nascimento J, Bernardino A. An Improved Labelling for the INRIA Person Data Set for Pedestrian Detection'. Lecture Notes in Computer Science.2013;7887:286-295.

[32] Geiger A, Lenz P, Urtasun R. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite'. in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2012, 'CVPR 2012'. 2012. Providence: IEEE.2012:3354 -3361.

[33] de Araujo JN. Pedestrian Detection from Camera Images for Automotive Applications. MSc dissertation, Coventry University, 2019.

[34] Dollár P, Wojek C, Schiele B, Perona P.'Pedestrian Detection: An Evaluation of the State of the Art'. IEEE Transactions on Pattern Analysis and Machine Intelligence.2012;34:743-761.

[35] Cordts, M. et al. (2016) 'The Cityscapes Dataset for Semantic Urban Scene Understanding'. in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2016, 'CVPR 2016'. held 26 June - 1 July 2016. Las Vegas: IEEE, 3213 – 3223.

[36] https://arxiv.org/abs/1506.02640

[37] Bengio Y. Practical Recommendations for Gradient-Based Training of Deep Architectures'. in Neural networks: Tricks of the trade. ed. Montavon G, Orr GB and Müller KR. London: Springer-Verlag Berlin Heidelberg.2012: 437-478.

[38] Shrivastava A, Pfister T, Tuzel O, Susskind J, Wang W, et al. Learning from Simulated and Unsupervised Images through Adversarial Training'. in DiCarlo, J. J, Shum, H. and Jurafsky, Dan (eds.) Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2017, 'CVPR 2017'. held 21-26 July 2017. Honolulu: IEEE.2017:2242-2251.

[39] Seong S, Lee Y, Kee Y, Han D, Kim J. Towards Flatter Loss Surface Via Nonmonotonic Learning Rate Scheduling'. in Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence 2018, 'AUAI 2018'. 2018. Monterey: AUAI Press. 2018:1020-1030.

[40] Huang J, Rathod V,Sun C, Zhu M, Korattikara A, et al. Speed/Accuracy Trade-offs for Modern Convolutional Object Detectors'. in DiCarlo, J. J, Shum, H. and Jurafsky, D. (eds.) Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2017, 'CVPR 2017'. 2017. Honolulu: IEEE.2017;3296-3297.

[41] https://arxiv.org/abs/1506.02640

[42] He K, Zhang X, Ren S, Sun J. Deep Residual Learning for Image Recognition'. in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2015, 'CVPR 2015'. 2015. Boston: IEEE.2015;770-778.

[43] Cubuk ED, Zoph B, Mane D, Vasudevan V, Le QV, et al. 'AutoAugment: Learning Augmentation Policies from Data'. in Proceedings of the IEEE Conference on ComputerVision and Pattern Recognition 2018, 'CVPR 2018'.2018. Salt Lake City: IEEE.2018:113-123.

[44] https://arxiv.org/abs/1904.04620