Vol. 05, No. 3 (2023) 545-552, doi: 10.24874/PES05.03.017

Proceedings on Engineering Sciences

www.pesjournal.net

DETECTION AND RECOGNITION OF ROAD SIGNS USING YOLOv5

Haitam Ettazi[1]
Najat Rafalia
Jaafar Abouchabaka

Received 02.02.2023.
Accepted 06.06.2023.
UDC – 004.032.26

Keywords:

ADAS, CNN, traffic sign detection, YOLOV5

A B S T R A C T

In the field of deep learning, a convolutional neural network is a class of artificial neural networks that became dominant in various computer vision tasks, which is widely used to solve complex problems in various areas, including driver assistance systems in the auto- motive field. Convolutional neural networks overcome the limitations of others conventional machine learning approaches since they are designed to automatically and adaptively learn the spatial characteristics of features in an image. In this paper, we are going to evaluate the inference and accuracy of YOLOv5s, for effective traffic sign detection in various environments. The results generated upon five classes gives satisfaction by 63.7% for the mean average precision, and over 80% in accordance to 5 categories set in this study. This article compared to YOLOV4 based CSP-DarkNet53 using Indonesia Traffic Signs generate better precision.

1. INTRODUCTION

These Road signs are as old as the roads, traffic signs are a crucial part of our road infrastructure, they provide essential information to road users without these useful signs, and we would encounter a higher rate of accidents.

In the race to develop autonomous vehicles, without the information of what is around these vehicles and their precise location, such a car cannot operate without risk. It is for this reason that the detection of traffic signs plays an essential role in autonomous vehicle systems, which are required to recognize and understand these traffic signs to ensure they follow road regulations.

In recent years, significant progress has been made in the field of computer vision thanks to the development of deep learning techniques such as convolutional neural networks (CNN) that have emerged from the study of the human and animal brain. In recent years, most of the state- of-the-art object-detection algorithms have used convolutional neural networks (CNNs) and have achieved fruitful results in target detection tasks, such as EfficientDet in Tan et al. (2020) and YOLO algorithm.

In Redmon et al. (2016) YOLO is an acronym for the term "You Only Look Once"; an algorithm that identifies and detects the different objects in a real-time image. This model divides the image into several cells, if the center of an object is in a certain cell; the latter is responsible for detecting that object.

[1] Corresponding author: Haitam Ettazi
Email: Haitam.ettazi@uit.ac.ma

In Jocher et al. (2022) YOLOv5 is the last version of the YOLO family, which is a model that consists of a single CNN which makes it very fast compared to other detection methods like R-CNN and Faster R-CNN as presented in Ren et al. (2017).

One of the main advantages of YOLOv5 for traffic sign recognition is its speed and efficiency. YOLOv5 is based on a "Darknet" architecture that is designed to be fast and efficient, making it ideal for real-time applications such as traffic sign recognition. This allows YOLOv5 to process images and videos quickly, even on low-power devices, making it a good choice for use in autonomous vehicles and other on-the-go applications. This paper contains:

- Briefing You Only Look Once (YOLO) architecture.
- Identify the most effective training parameters on YOLOv5s and evaluating its performance.

2. LITTERATURE

2.1 Single-Stage and Two-Stage Algorithms

Modern object detection models can be divided into two categories: two-stage detectors and single-stage detectors (figure 1). Two-step detection models like R-CNN use two networks to carry out the task of proposing the regions and the task of classification, while one-stage models perform both of these tasks by a single network. The proposal of the regions is carried out by what is called RPN (Region Proposal Network). RPN is used to find areas where search for the objects in order to reduce the computation time, this is done by generating bounding box proposals each with the probability that an object exists in this region. After generating a list of locations possible objects, a convolutional feature extraction is performed on each candidate region. In the second step, the content of each framing box is classified, to decide which ones to keep and which to eliminate.

In general, two-stage architectures achieve better accuracy, but they are slower than one-stage architectures. The biggest drawback of two-stage architectures is that they do not excel at performing real-time detection and require multiple GPUs to train the model as shown by Carranza-García et al. (2020), thus, the YOLO model has been proposed to address these limitations.

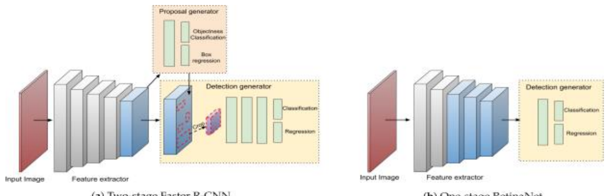


Figure 1. Example of single-stage algorithm and a two-stage algorithm

2.2 YOLO

YOLO is an acronym for the term "You Only Look Once". This is an algorithm that identifies and detects the different objects in a real-time picture. This model divides the image into several cells, if the center of an object is in a certain cell, the latter is responsible for the detection of this object and it consists of a single CNN which makes it very fast compared to other detection methods like R-CNN and Faster R-CNN. In all object detection architectures, the first step is extracting features from the input image through a convolutional network, the depth and type of convolutional backbone used in these architectures affects the speed, accuracy, and memory usage of the model. Recent object detection models have a "Backbone" which is a convolutional neural network used to extract features keys of an input image, the depth and the type of the convolutional "Backbone" used in the model affect the speed, accuracy, and memory usage of the model. The second component is the "Neck" which creates feature pyramids; these pyramids help models successfully generalize objects. It helps in the identification of the same object in different scales and sizes, and performs an aggregation on the characteristics and transmits it to the third component called "Head". The "Head" part of the model is mainly used for the last stage of detection. It applies the bounding boxes to the objects and calculates final output vectors with predictions of the classes. YOLO's model can be applied in many fields which depend on fast object detection, and although the prediction speed of the model was very high, the performance was still not comparable to Faster-RCNN when it was introduced for the first time.

YOLOv2 and YOLO 9000 have been released in 2016; YOLOv2 achieved a mAP of 76.8% mAP at 67 FPS and 78.6% at 67 FPS. YOLO 9000 according to Redmon et al. (2017) uses the architecture of YOLO v2 but it is able to detect more than 9000 classes. However, the mAP value of YOLO 9000 is only 19.7The previous YOLO architecture had many problems. She made a lot of location errors and had a bad "recall". So, the objective of the new article was to improve these defects of YOLO, while maintaining the speed of architecture. YOLOv2 uses a convolutional neural network called DarkNet as feature extractor As a result of these improvements; YOLOv2 offers a good accuracy and detection speed. At 67 FPS, YOLOv2 can give a 76.8% mAP, at 40 FPS the detector gives an accuracy of 78.6% mAP, a better accuracy than other state-of-the-art models such as Faster R-CNN and SSD while running much faster than these models. The 3rd version of YOLO, called YOLOv3 was released in 2018 by the same authors of YOLO and YOLOv2. There are major differences between the architecture of YOLOv2 YOLOv3 and older versions in terms of speed and accuracy. YOLOv2 and YOLOv3 are worlds apart in terms of accuracy, speed, and architecture. YOLOv2 uses Darknet- 19 as a feature extractor base, while

YOLOv3 now uses Darknet-53. Darknet-53 is a "Backbone" directed by YOLO creators Joseph Redmon and Ali Farhadi. According to their articles, Darknet- 53 is 1.5 times faster than ResNet101. This accuracy means no compromise between accuracy and speed for Darknet backbones, because Darknet-53 is still as accurate as ResNet-152 but twice as fast. In addition, you can easily switch between speed and precision by simply changing the size of the model, without the need for re-train the whole model. YOLOv3 also increased mAP for small objects by 13.3%, which is a huge improvement over YOLOv2.

2.3 YOLOv5

YOLOv5 uses CSPNet (Cross Stage Partial Network) as a Backbone, which showed significant improvements in the time of processing using deeper networks. The problem with previous networks is that they required very heavy inference calculations. Cross Stage Partial Network (CSPNet) was created with the aim of solving this problem, by allowing networks more flexibility by including feature maps at the beginning and end of each network step. This avoids having duplicate gradients in the network, because the information is similar between the beginning and the end of each step.

In YOLOv5, PANet (Path Aggregation Network) plays the role of the Neck of the model, it consists of a series of layers to combine and aggregate the image features to pass them to the prediction layers.

The Head of YOLOv5 remains the same as the YOLOv3 by Redmon et al. (2018) and YOLOv4 versions by Bochkovskiy et al. (2020). YOLOv5 has several varieties of pre-trained models. The difference between them is a trade-off between model size and inference time, The YOLOv5s version is small in size but not the most accurate, while the YOLOv5x version is the biggest in size but it is the best of the YOLOv5 family in terms of accuracy.

2.4 Related work

Current research in the field of computer vision involves building a better image detection system for comprehensive machine learning use. For several decades new traffic signs have been introduced. The first stage observes traffic signs determining the proper placement of traffic signs in relation to the size and location of each sign. The second stage of the project is the process of displaying an image and understanding the different signs based on colors and shapes associated with them. Traffic sign detection is commonly used in ADAS. Neural Network Classifiers like CNN use classifiers to categorize data in general, issues with object recognition persist in certain models with unfair highlights. Using the word CNN is more appropriate when referring to the subject.

In the last years, many researchers have worked on the detection and recognition of traffic signs using different object detection models.

Mulyanto et al. (2021) used YOLOv4 to perform the detection on a dataset of traffic signs from Indonesia, since the research that was done the model showed a main average accuracy of (mAP@0.5) of 74.91% for 26 classes of traffic signs. However, the weak point of their model was the recognition of images of panels that are strongly identical, small differences between some Panel categories seem to pose challenges for deep learning and show the limitations of YOLOv4. Despite this, the model satisfies the requirements of ADAS (Advanced driver assistance system) to provide reliable information to drivers related to the presence of traffic signs on the road.

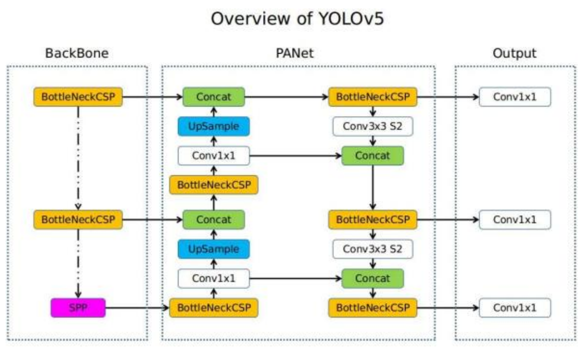


Figure 2. Architecture of YOLOv5

Dewi et al. (2021) proposes a study which analyzes and compares CNN models and extractor’s features, in particular the two object detection models; YOLOv3 and YOLOv4. This comparison uses images that contain 60 traffic signs in different conditions and environments.

Their work also presents an approach that combines synthetic images with real images in order to improve the diversity of datasets and check the effectiveness of dataset synthetics. They concluded that YOLOv4 is more accurate than YOLOv3 using a dataset that combines original images with synthetic images, obtaining an accuracy of 84.9% for YOLOv3 and 89.33% for YOLOv4. This study also shows that training a model with a combination of original images and generated images, improves performance when compared to the mere use of the original images.

Sang et al. (2018) modified the number of convolutional layers In the network based on YOLOv2, proposed an improved single-stage traffic sign detector and using Chinese Traffic Sign Dataset Training in an effort to make it more adapted to Chinese Street View, a novel Perceptual Generative Adversarial Networks Developed to detect small flow signs, which improves detection performance and generate super-resolution maps for small signs.

3. APPROACH

3.1 Dataset

A dataset is a group of annotated images. Annotation means specifiying the position and the class of the object. Each dataset contains a specific number of classes.

According to Neuhold et al. (2017), Mapillary is a crowdsourced and open-source sharing service for images, Mapillary offers different capture modes, including on feet, on a bicycle or on a car. With data from 190 countries, Mapillary Traffic Sign Dataset is one of the largest traffic sign datasets with great variability in conditions, weather, time of the day and camera sensors and viewpoints, which is publicly available for use in machine learning, to detect and recognize traffic signs.

The Mapillary traffic sign dataset of fully annotated images consists of 52,000 images that reach a total size of 41.5 GB, distributed over more than 300 different classes. Because of ram and storage limitations, it was necessary to choose a few classes among these 300 classes to work on them. The following classes were retained with a total of 5772 instances:

- work regulatory–keep-right–g1: 1242 instances.
- Regulatory–no–entry–g1: 2048 instances.
- Regulatory–yield–g1: 2775 instances.
- Warning–pedestrians-crossing–g4: 1124 instances.
- Regulatory–stop–g1: 1386 instances.

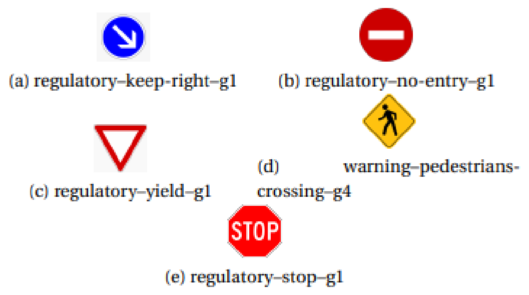

(a) regulatory–keep-right–g1
(b) regulatory–no-entry–g1
(c) regulatory–yield–g1
(d) warning–pedestrians-crossing–g4
STOP
(e) regulatory–stop–g1

Figure 3. Retained classes for training

3.2 Training the model

The models are trained on 5772 instances from the Mapillary road traffic signs dataset in five classes: keep right, no entry, yield, warning pedestrians crossing and stop. The model is trained for 200 epochs, with batch sizes ranging from 32 up to 80 using Google Colab's Tesla T4 GPU, and 16GB of VRAM. Google Colab is a hosted service of Jupyter notebook that requires no configuration and allows access without charge to computing resources, including GPUs. Google Colab is a complete tool for training and testing quickly machine learning models without having a hardware limitation.

Before the training process, it is necessary to divide it into three parts: training, testing, and validation.

The training data will be used to train the model during the machine learning process. The validation data will be used to tune the hyper-parameters to achieve the best possible configuration. The model is trained on the training set and simultaneously evaluated on the validation set after each epoch to optimize the model performance.

The test data will be used to verify the performance of the trained model, in order to avoid errors or anomalies, and to achieve performance characteristics such as accuracy and recall.

There are no strict rules on partitioning, but if there are multiple hyper-parameters to tune, the machine learning model requires a larger validation set than in less complex cases, typically putting 70% of the data in the training set, 20% in the validation set, and 10% in test set. To start the training, it is necessary to specify the dataset, the batch size (batch-size), image size (image-size), and either weights pre-trained or randomly generated weights. Before changing any parameter, it is recommended to train the model first with the default parameters to establish a performance base on which to improve.

Epoch: At each epoch, an entire data set is transmitted to the neural network once. As the number of epoch increases, the weights of the neural network are modified more times and the model goes from underfitting to an equilibrium point and then to overtraining.

Ultralytics suggests starting with 200 epochs. If this produces an overfitting we reduce the epochs. If overfitting does not occur after 200 epochs, one trains the model longer, i.e. 300, 600, 1200 epochs, etc.

Batch-size: Since an epoch is too large to transmit it to the neural network at a time, it must be divided into several batches (batch) smaller. The batch size or batch-size is the number of samples (in our case, the pictures) in each batch. In the case of YOLOv5, it is better to use the largest batch-size allowed by our hardware, since a small value of batch-size produces poor batch standard statistics and should be avoided.

Image size: The larger the image size, the results are generally better, but the model takes longer to train, the default image size value for YOLOv5s is 640, in most cases of good results can be obtained without modification on the default value of image-size, so the value 640 is retained for our training.

The training batch size has a big impact on GPU memory required for training a neural network, the larger the batch size, the more images are transmitted through the neural network at once, this will directly cause the required GPU memory to increase. We start with a large batch size value of 128; this causes the training process to stop prematurely since the required GPU memory is greater than the one available. We reduce the lot size value and we run the training again and repeat this process until we find the highest value possible with our GPU memory for the batch size , the value we end up with for the batch-size after this process is 80.

4. NUMERICAL EVALUATION

4.1 Evaluation metrics

Precision, this measures the accuracy of our predictions which can be described as the percentage of predictions that is correct.

$$\textit{Precision} = \frac{TP}{TP + FP}$$

Recall is the percentage of true positives, among the total number of real objects. For example, if a model correctly detects 90 people in an image or there are 100 people, the recall is 90%.

$$\textit{Recall} = \frac{TP}{TP + FN}$$

- TP (True Positive): The case where the prediction is positive and the actual value is actually positive. Example: The model detects the presence of a traffic sign, and the sign actually exists in the image.
- FP (False Positive): cases where the prediction is positive, but the actual value is negative. Example: The model detects the presence of a traffic sign, but the sign does not actually exist in the image.
- TN (True Negative): Cases where the prediction is negative and the actual value is actually negative. Example: The model does not detect any traffic signs, and indeed no signs exist in the image.
- FN (False Negative): cases where the prediction is negative, but the actual value is positive. Example: The model does not detect any traffic signs, but the image does indeed contain a sign.

The mAP value (mean average precision) is a popular evaluation metric used for object detection, many object detection algorithms, such as SSD, Faster R-CNN and YOLO, use the mAP value to evaluate their models for research publication.

To calculate the mAP value, we use the intersection on union for that we need:

- The actual bounding boxes, i.e. the bounding boxes that specify where our object is in the image labeled by hand in the test set.
- The bounding boxes predicted by our model.

4.2 Results

Once a model is trained, it is necessary to be able to evaluate its performance. The model changes its weights through the training set; it happens by reducing the result of the cost function for the training set.

A good way to verify the effectiveness of the model is to observe the curve of the loss function during training, if the predictions deviate too much from the actual results; the loss function would take a very large number.

The loss function is a method for evaluating how much an algorithm models the training data, using an optimization function, the loss function learns to reduce the prediction error over time.

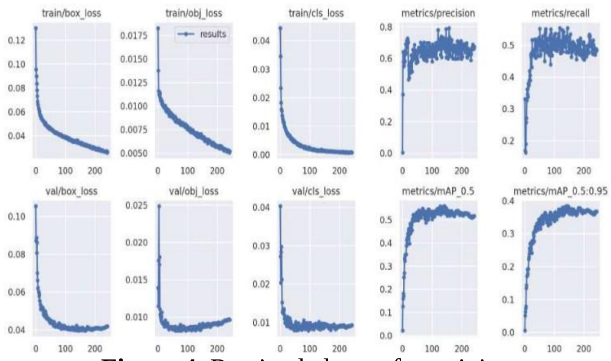


Figure 4. Retained classes for training

According to the evolution curve of the loss function that we obtained, the model converges, we also observe a slight over-fitting, because the cost function on the training set is smaller than on the validation set, so it is not possible to continue training to improve performance after that point; the most optimal results observed were during the 76th epoch.

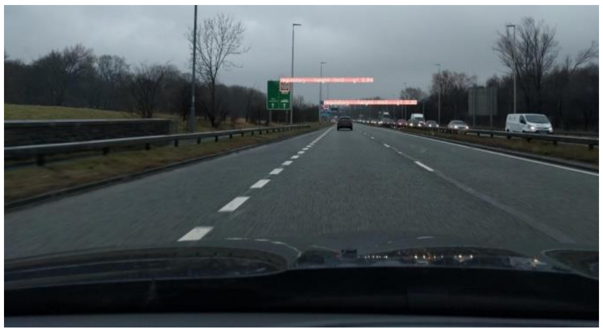
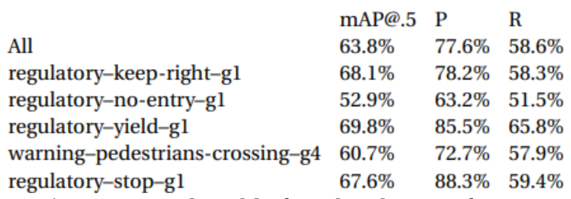


Figure 5. Correctly classified image

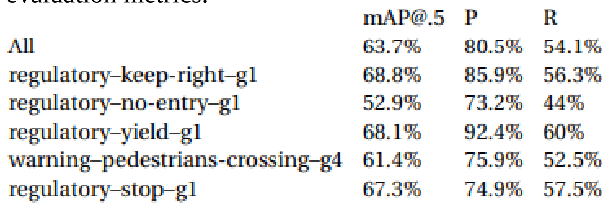
There are three parameters used to evaluate the YOLOv5 model from the training results, namely the Precision, Recall and mean Average Precision (mAp).



	mAP@.5	P	R
All	63.8%	77.6%	58.6%
regulatory–keep-right–g1	68.1%	78.2%	58.3%
regulatory–no-entry–g1	52.9%	63.2%	51.5%
regulatory–yield–g1	69.8%	85.5%	65.8%
warning–pedestrians-crossing–g4	60.7%	72.7%	57.9%
regulatory–stop–g1	67.6%	88.3%	59.4%

Figure6. Results table for a batch-size of 80

According to Fig. 5, the experimental result shows that the system returns 78.2%, 63.2%, 85.5%, 72.7%, 88.3% precision rate for 1st, 2nd, 3rd, 4th, and 5th sign, respectively, this gives us a mean precision of 77.6%. This table describes the performance of the proposed on Mapillary traffic sign recognition using standard evaluation metrics.



	mAP@.5	P	R
All	63.7%	80.5%	54.1%
regulatory–keep-right–g1	68.8%	85.9%	56.3%
regulatory–no-entry–g1	52.9%	73.2%	44%
regulatory–yield–g1	68.1%	92.4%	60%
warning–pedestrians-crossing–g4	61.4%	75.9%	52.5%
regulatory–stop–g1	67.3%	74.9%	57.5%

Figure7. Results table for a batch-size of 64

These tables display that the five classes are recognized with a mean Average Precision (mAP) of 63,7%. For the five categories, YOLOv5 achieved a precision of above 80% using a batch-size of 64. However, when using a batch-size of 80 the precision value is less than 80%, this is due to the amount of training data and time to train the model and some images may be identical.

This shows that even if the documentation recommends using the highest possible batch-size value for better performance, the model may benefit from using a smaller value such as 64 in our case, however using an even smaller value ie: 32; did not produce better results. When comparing these results with a paper that implemented YOLOv4- based CSP-DarkNet53 deep learning model using the Indonesia Traffic Signs (ITS) dataset, that obtained a performance with the main average Precision (mAP@0.5) of 74.91% for six categories but a precision of 74%.

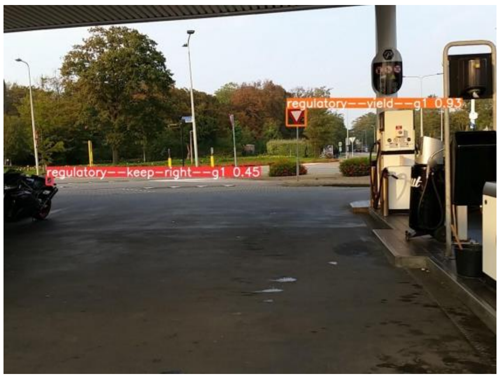


Figure 8. Incorrectly classified image

Learning Object detectors such as YOLOv5 are easy to be deceived by objects similar to road signs. One of the main issues is that the ability of the object detector is limited in certain cases, for example, a reflection of the traffic sign, or an object that is similar in shape, which is just a local pattern of the whole image, but ignores the other information like background.

5. CONCLUSION

One of the main advantages of YOLOv5 is its ability to detect objects in real-time. This is particularly important in the context of traffic sign recognition, where the ability to quickly and accurately identify traffic signs is crucial for ensuring the safety of drivers and pedestrians.

An additional strength of YOLOv5 is its ability to accurately detect and classify a wide range of objects, including traffic signs. This is achieved through the use of convolutional neural networks (CNNs), which are able to learn complex patterns in visual data. Another key advantage of YOLOv5 for traffic sign recognition is its ability to handle a wide range of conditions and scenarios. YOLOv5 is trained on a large dataset of images and videos that includes a wide variety of conditions, including different lighting conditions, weather, and backgrounds. This allows YOLOv5 to generalize well and handle a wide range of scenarios, making it robust and reliable for traffic sign recognition.

In addition, YOLOv5 is able to make high-quality predictions with a high level of accuracy. YOLOv5 uses a single neural network to make predictions, which allows it to make predictions that are more accurate and more consistent than other models that use multiple networks. This makes YOLOv5 a good choice for applications where accuracy is critical, such as traffic sign recognition.

In terms of limitations, one potential issue with using YOLOv5 for traffic sign recognition is its reliance on large amounts of labelled training data. This can be a challenge in some contexts, where there may be limited availability of annotated traffic sign images.

Additionally, YOLOv5 may not always be able to accurately detect and classify traffic signs in challenging conditions, such as low lighting or the presence of clutter. Further research and development is needed to improve the robustness of YOLOv5 in these scenarios.

Overall, YOLOv5 has great potential for use in traffic sign recognition, due to its ability to detect objects in real-time and its strong performance on a wide range of tasks. However, further research and development is needed to address potential limitations and improve its performance in challenging conditions.

References:

Bochkovskiy, A., Wang, C., & Liao, H. M. (2020). YOLOV4: Optimal speed and accuracy of object detection. arXiv (Cornell University). https://arxiv.org/pdf/2004.10934v1

Carranza-García, M., Torres-Mateo, J., Lara-Benítez, P., & García-Gutiérrez, J. (2020). On the performance of one-stage and two-stage object detectors in autonomous vehicles using camera data. *Remote Sensing*, *13*(1), 89. https://doi.org/10.3390/rs13010089.

Dewi, C., Chen, R., Liu, Y., Jiang, X., & Hartomo, K. D. (2021). Yolo V4 for advanced traffic sign recognition with synthetic training data generated by various GAN. IEEE Access, 9, 97228–97242. https://doi.org/10.1109/access.2021.3094201.

Jocher, G., Chaurasia, A., Stoken, A., Borovec, J., Kwon, Y., Fang, J., ... & Thanh Minh, M. (2022). ultralytics/yolov5: v6. 1-TensorRT, TensorFlow edge TPU and OpenVINO export and inference. *Zenodo*.. https://doi.org/10.5281/zenodo.6222936.

Mulyanto, A., Jatmiko, W., Mursanto, P., Prasetyawan, P., & Borman, R. I. (2021). A new Indonesian Traffic Obstacle dataset and performance evaluation of YOLOV4 for ADAS. *Journal of ICT Research and Applications, 14*(3), 286-298. https://doi.org/10.5614/itbj.ict.res.appl.2021.14.3.6.

Neuhold, G., Ollmann, T., Bulo, S. R., & Kontschieder, P. (2017, October). The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes. In *2017 IEEE International Conference on Computer Vision (ICCV)* (pp. 5000-5009). IEEE Computer Society. doi: 10.1109/ICCV.2017.534.

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). https:// doi.org/10.1109/CVPR.2016.91

Redmon, J., & Farhadi, A. (2017). *YOLO9000: Better, Faster, Stronger*, 6517-6525. https://doi.org/10.1109/CVPR.2017.690

Redmon, J., & Farhadi, A. (2018). *Yolov3: An incremental improvement.*

Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, *39*(06), 1137-1149. doi: 10.1109/TPAMI.2016.2577031.

Sang, J., Wu, Z., Guo, P., Hu, H., Xiang, H., Zhang, Q., & Cai, B. (2018). An improved YOLOv2 for vehicle detection. *Sensors*, *18*(12), 4272. https://doi.org/10.3390/s18124272.

Tan, M., Pang, R., & Le, Q. V. (2020). Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10781-10790). doi:10.1109/CVPR42600.2020.01079

Haitam Ettazi
Faculty of Sciences, University Ibn Tofail,
Kenitra
Morocco
Haitam.ettazi@uit.ac.ma
ORCID 0000-0002-7303-3044

Najat Rafalia
Faculty of Sciences, University Ibn Tofail,
Kenitra
Morocco
najar.rafalia@uit.ac.ma

Jaafar Abouchabaka
Faculty of Sciences, University Ibn Tofail,
Kenitra
Morocco
jaafar.abouchabaka@uit.ac.ma