Truck Detection in Multispectral Satellite Imagery using YOLOv8 Model

Rita Magalhães
Instituto Superior Técnico
Lisboa, Portugal
ritaamagalhaes7@tecnico.ulisboa.pt
ist195967

Rodrigo Lopes
Instituto Superior Técnico
Lisboa, Portugal
rodrigo.d.lopes@tecnico.ulisboa.pt
ist107255

***Abstract*—This report presents an application of the You Only Look Once (YOLO) algorithm for the detection of moving trucks in multispectral satellite imagery captured by the Sentinel-2 mission. The global coverage and regular acquisition of Sentinel-2 images constitute the two primary advantages over other traffic monitoring methods. The project aims to contribute to the field of Remote Sensing by utilizing YOLOv8, a state-of-the-art deep learning model for Object Detection, to monitor road truck traffic in the surroundings of Porto, Portugal. The dataset is composed by 300 images from motorways featuring at least one truck, captured on different days. The report details the dataset construction, including the use of Sentinel-2 data and OpenStreetMap road data to isolate road regions and improve model accuracy. Manual annotations were performed using Roboflow to create YOLO-compatible bounding box labels, used as ground-truth. The evaluation metric, Intersection over Union (IoU), is used to compare the proposed approach with a baseline method (developed by Fisser et Al. [1]). Our approach achieved an IoU = 0.63 between its prediction boxes and the annotated boxes, surpassing the result obtained by the baseline (IoU = 0.16) The results show the effectiveness of YOLO in detecting trucks in multispectral satellite imagery, positioning it as most efficient method for identifying trucks in Sentinel-2 images.**

***Index Terms*—Remote Sensing, YOLO, Sentinel-2, truck detection, deep learning**

I. Introduction

Our project aims at implementing YOLO to detect moving trucks on satellite images from Sentinel-2. This application is part of a broader field named **Remote Sensing**, which consists on the acquisition of information about physical objects on a given area by measuring the radiation emitted or reflected by the object itself (in contrast to *in situ* observation). Remotely sensed images from the Earth are collected by special cameras typically installed on aircraft or satellites. [2]

Monitoring road truck traffic provides us with relevant data for depicting spatial patterns of: 1) economic performance and 2) emission of air pollutants. Measuring the spatio-temporal variations of cargo traffic may be used as an economic proxy of a given region. In fact, in many countries the majority of freight is transported by road cargo trucks. In addition, road traffic is one of the main sources of air pollutants emissions, as the vast majority of road trucks are still powered by fossil fuels. According to the European Environmental Agency, in 2019, 86% of the exceeded EU annual limits for NO_2 were detected at traffic stations. Collecting more relevant data for these applications justifies the importance of detecting and monitoring road trucks traffic. [3] [1]

For this project, we used images of the Earth surface that were captured by Sentinel-2's multispectral instrument. Launched in 2015, Sentinel-2 is an Earth observation mission that is part of the Copernicus program, operated by the European Space Agency (ESA). It acquires high spatial resolution images (between 10 and 60 meters of spatial resolution) in both terrestrial and maritime areas all over the globe. Sentinel-2 is a constellation of two satellites, Sentinel-2A and 2B, which are orbiting the Earth in opposite sides (i.e. are phased 180º from each other), in such a way that every place has a regular 5-daily revisit. The short time between the acquisition of two consecutive images of the same area - which allows monitoring the temporal evolution of cargo traffic - is the main advantage of using Sentinel-2. Moreover, Sentinel-2 is an open-source resource, making it the natural choice for this application. In 2024, a third satellite (Sentinel-2C) will be launched. [1] [4]

Our method takes advantage of the satellite's multispectral instrument (MSI), more specifically it exploits the fact that the time of light acquisition in different spectral bands suffers a small temporal offset - *i.e.* different bands are acquired at slightly different times. This means that large moving objects - such as moving trucks - are perceived by the MSI as a colorful sequence of pixels (see Fig. 1 in Section II), precisely what we want to detect by using YOLO.

You Only Look Once (YOLO) is the state-of-the-art deep learning model for Object Detection (OD). It has a CNN-based architecture, designed specifically for the task of OD. Recently, it has become increasingly popular due to its high accuracy and fast processing speed - these are, as well, the main reasons why we decided to adopt it in our work. In the context of our work, a pre-implemented version of the YOLO model was trained with data from Sentinel-2, in order to detect moving trucks in motorways.

II. State-of-the-Art

Our work combines an OD deep learning algorithm with traffic monitoring, namely via satellite imagery. In this section, we shall provide a brief description about the state-of-the-art, covering both these topics.

A. *Traffic monitoring*

Traditionally, traffic monitoring is performed *in situ*, namely by using automatic counting stations that register the number of vehicles that pass through a specific location. The first attempts to monitor traffic via remote sensing used systems with zeppelins and helicopters and, since then, many other systems have been used to monitor traffic, including multiple optical cameras mounted on aircraft, airborne laser scanning and satellite imagery data [1]. Very high resolution (VHR) spaceborne data have also been used for vehicle detection. However, the main problem with VHR data is that it lacks of regular acquisition. On the other hand, *in situ* counting stations not only are expensive, but also spatially limited, in the sense that they only acquire data in a specific geographic point.

Fisser et Al. [1] have introduced a methodology that overcome both these limitations by making use of data from Sentinel-2, since it provides a global coverage combined with a regular 5-days revisit. Our project was inspired in their article ”Detecting Moving Trucks on Roads Using Sentinel-2 Data” and hence here we will explore in more detail their methodology. Furthermore, their results will serve as a baseline for comparison to our method.

Given a Sentinel-2’s image, Fisser et Al. start by cutting it into several smaller images of motorways containing at least one truck. Using road data available from Open Street Map (OSM), a mask is applied to every image, such that only the pixels that lie in a location where there is a motorway are conserved. In other words, OSM possesses the coordinates of all motorways, and thus all images were masked using that information (see Fig. 3 in section III).

The following step is to train a Random Forest (RF) model to predict the color (R, G or B) of each pixel. To do so, the RF model considers as inputs 4 of the 13 bands provided by the MSI. More specifically, bands B04, B03, B02 and B08 (corresponding to red, green, blue and near-infrared spectra) were combined into six different variables that serve as input to the RF model. The model outputs four probabilities for each pixel, which correspond to the probability of each one being classified as red, green, blue or background. Once every pixel has been classified, an Object Extraction algorithm is implemented. This algorithm, developed by Fisser et Al., consists in a set of explicit rules that attempt to determine if a certain pixel is part of a truck, considering its class and neighbors. It is important to note that these rules were explicitly programmed - contrarily to our approach, instead, that uses deep learning to detect trucks, as we will explain in the following sections. The last step explained in this article is Object Characterization, in which the authors extract information about the truck’s direction and speed.

Finally, regarding their article, Fisser et Al. assume the hypothesis that the motion effect (see Fig. 1) observed in Sentinel-2 data is apparent only for large moving vehicles. In fact, the dimensions of a pixel (which has 10×10 meters in the bands mentioned) are smaller than the usual length of a truck (up to 18.75 m in the EU [1]). On the other hand, a large car measures up to 4.8 m, meaning that, in principle, it cannot be accurately detected using this technique. This information, according to them, justifies the hypothesis that the motion effect seen in the Sentinel-2 images appear due to the presence of large moving vehicles, like trucks. In this report, the word ”truck” includes any large moving object, for example, buses may also be detected.

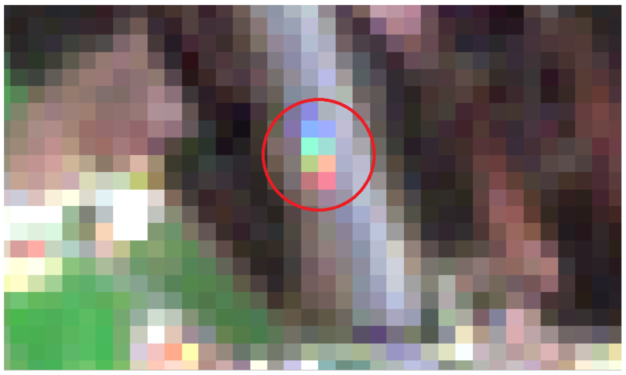

Fig. 1. Motion effect for large moving vehicles.

B. *Object Detection*

Object Detection (OD) is a computer vision task in which the general goal is to identify, locate and segment objects in an image. For the purposes of this project, we intend to locate and count moving trucks in satellite images, as previously explained.

Traditional computer vision techniques for OD were based on handcrafted features, together with a machine learning classifier (such as an SVM) to detect and classify objects. These feature extraction algorithms include Haar Features, SIFT, HOG, and others. [5]

In the recent years, the task of OD has been taken over by deep learning and CNN-based solutions, which can learn features from raw pixel data, eliminating the need for manual feature engineering. [6]

The most popular CNN-based approaches include, among others, the R-CNN model family and the YOLO model family. Despite R-CNN being generally more accurate, in this project we decided to adopt YOLOv8 (the most recent version of YOLO), as it was designed for achieving good results, within a much shorter period of time when compared to R-CNN. [7]

YOLO - which stands for ”You Only Look Once” - is a multi-layer neural network that is applied to the whole image, used not only for OD - as it is in our case - but also for object segmentation and classification. It essentially works by predicting one or more bounding boxes for each pixel, assuming that the center of the bounding box falls within that pixel. Associated with that prediction, there is a class probability and the output of the model will combine both, assigning a class to each bounding box. The fact that it ”looks” to the whole image allows that its predictions, contrarily to

R-CNN, are informed by the global context. In conclusion, YOLO has the best trade-off between accurate results and speed, and hence the reason we adopted it. [8] [9]

III. The Dataset

Our dataset is composed by 300 images from motorways in the area around the city of Porto, captured on 4 different days. These images have dimensions that vary from 107×59 to 297×152 pixels. Between these, 25 images do not have any truck and are considered "background", while the remaining 275 feature at least one truck. Regarding the model, our strategy involved splitting the dataset into three distinct sets: a training set (60%) for model training, a validation set (20%) for finetuning hyperparameters, and a test set (20%) for assessing actual performance.

Initially, we had built a dataset with 150 images, but in response to unsatisfactory results, we decided to enhance it.

A. Sentinel-2 data

The data from Sentinel-2 is available on Copernicus Data Space Ecosystem [10].

Sentinel-2 mission captures information on 13 spectral bands from all the regions in the globe every 5 days, as mentioned before. This information is available in two different levels of product processing: Level-1C and Level-2A. The first projects the image in cartographic geometry, provided in Top Of Atmosphere reflectances. [11] The latest, which we utilized for our work, provides atmospherically corrected Surface Reflectance images, derived from the associated Level-1C products. [12] Out of the 13 spectral bands, we employed the bands 4, 3 and 2 corresponding to the True Color RGB. These bands have a resolution of 10 meters, signifying that each pixel represents an area of 10×10 m^2.

To construct our dataset, we decided to focus on the city of Porto and its surrounding area. Thus, we searched among the available data from Sentinel-2 for four different cloud-free days to ensure optimal visibility of the Earth's surface. We paid also special attention to select days with an acceptable amount of trucks in order to gather sufficient images for our dataset. For instance, our observations revealed a decrease in the number of trucks during weekends. Upon downloading, all the GeoTIFF images were sourced from the same region, each with 10980×10980 pixels, equivalent to approximately 110×110 km^2.

Next, we used QGIS, an open-source geographic information system (GIS) software, to visualize the data. [13] Given the substantial size of the original image, we zoomed in to focus on motorways, facilitating the identification of moving trucks. When we identified one or more trucks within the same visualization window we applied the 'raster' function in QGIS and extracted a second smaller GeoTIFF matching the size of the window (ranging from 107×59 to 297×152 pixels).

B. Open Street Map road data

The initial window images included not only the motorway but also surrounding elements such as houses and trees. Aiming to improve our results and minimize extraneous details, we opted to mask the images to isolate the road.

For this purpose, we accessed OpenStreetMap (OSM) Road Data referring to the detailed and freely accessible geographic information provided by the OpenStreetMap project, a community of mappers that contribute and maintain data about roads, trails, coffees, railway stations, and much more, all over the world. [14] From here, we obtained a geopackage file containing all the portuguese motorways from which we intended to derive masks for the window images.

However, we noted that in some of the identified trucks, the colorful pixels matched the lines delimitating the roads by this geopackage. Consequently, we decided to implement a 20m buffer on the motorway file using QGIS, extending the width of the roads by 20m (approximately 2 pixels, depending on the orientation of the road). Therefore, when applying the mask, we retained the motorways while preserving the pixels from these trucks, as illustrated in Figures 2 and 3.

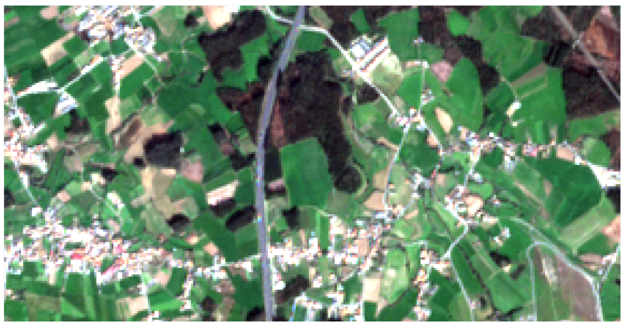

Fig. 2. Initial window image including road and surroundings

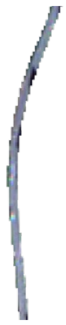

Fig. 3. Masked image with only the road

In addition it help us following the roads on QGIS while looking for moving trucks.

C. Dataset annotations

After collecting and processing the set of 300 images, the next key step was to annotate them.

The main goal in OD is to pinpoint the precise location of objects within an image. This is usually achieved through the use of bounding boxes, which are rectangular shapes that highlight the object. [15] These bounding boxes are a

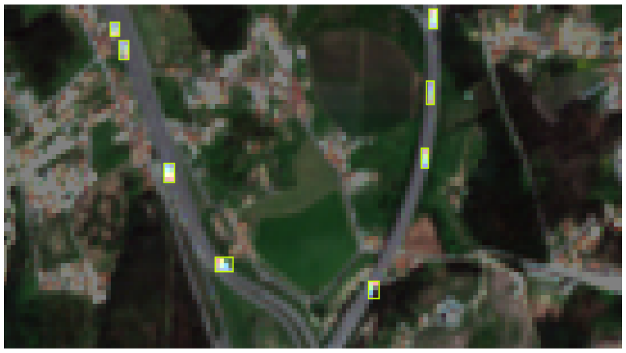

Fig. 4. Image with 8 objects labeled as "Truck" in Roboflow.

critical component in the process of accurately localizing and delineating objects, as demonstrated in Figure 4.

For this task, we used Roboflow, a computer vision developer framework offering several tools for preprocessing data and implementing model training techniques. Roboflow facilitates tasks such as image annotation, data augmentation and dataset versioning. [16] In our use case, we only utilized Roboflow for image annotation. Each observed truck within the images was identified through a bounding box and associated with a singular class, denoted as 'Truck.' This is a meticulous and crucial step to obtain positive results.

Finally, we obtained labels in a YOLO-compatible format. Each truck within an image is identified by a bounding box described as follows: class (numeric identifier or label representing the object's class), x_center (x-coordinate of the bounding box center), y_center (y-coordinate of the bounding box center), width (width of the bounding box), and height (height of the bounding box). All this measures are relative. For each image there is a correspondent `.txt` file containing this information about the trucks.

IV. Model

As we stated before, our work was inspired on [1], and it will serve as a baseline for comparison with our project. In this article, when we use the word "baseline" we are thus referring to the Fisser et Al. previously described method for truck detection. In fact, the data preprocessing pipeline we implemented was identical to theirs: obtaining Sentinel-2 images 110×110 km^2, cutting it into smaller parts containing at least one truck and finally masking them using OSM road data. However, the steps that followed were different: our next step was to annotate our dataset and then to train the YOLO model (the following subsection IV-C gives more detail on this). On the other hand, the baseline proceeded by applying a RF model to classify each pixel, followed by an object extraction algorithm. The results obtained with the baseline are explored in the subsection IV-B. However, we shall firstly define a metric that allows to compare both approaches - subsection IV-A.

A. Evaluation metric

We adopted Intersection over Union (IoU) as an evaluation metric to compare the two approaches. In fact, IoU is the metric typically used to evaluate the performance in OD tasks.

Given two bounding boxes - usually, the ground-truth box and the prediction box - the IoU corresponds to the area of the intersection of the two boxes divided by the total area that the two boxes ocupy:

$$IoU = \frac{A_{intersection}}{A_{total}}$$

This concept can be extended when we have more than one bounding box in the same image - which is the case when we labeled more than one truck. Therefore, to evaluate these algorithms, we simply summed the area in common from the two sets of bounding boxes and divided it by the total area covered by the two sets.

The usage of IoU allows to infer how close are the model predictions to the ground-truth. The closer IoU is to 1, the more accurate are the predictions of our model.

B. Baseline performance

Fisser et Al. state that their algorithm in [1] achieves an average F1-score of 0.74 in the task of identifying bounding boxes of trucks. F1-score is a widely used metric in machine learning, typically used for evaluating a classification model. It works as a combination of precision and recall, more precisely it is the harmonic mean of precision (P) and recall (R):

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

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

$$F1 = \frac{2 \times P \times R}{P + R}$$

The precision measures the portion of positive identifications that are actually correct and is defined as the ratio of true positives (TP) to the sum of true positives (TP) and false positives (FP). On the other hand, recall gives the portion of actual positives that were correctly identified and is calculated as the ratio of true positives (TP) to the sum of true positives (TP) and false negatives (FN).

The computation of F1-score involved, thus, the computation of TP, FP and FN. These were computed by making use of IoU. More specifically, for each prediction bounding box intersecting a validation box, the IoU was calculated. Then, every intersection with IoU > 0.25 was counted as a positive result. This lead to an average F1-score of 0.74, with the highest result achieved in Poland (0.88) and the lowest in Kenya (0.36).

On the other hand, we run the baseline code for our dataset and evaluated it according to IoU. This means that for each image, we calculated the IoU between the baseline's prediction boxes and our ground-truth boxes. Its average value is **IoU** =

0.16 and we will compare our model with the baseline having always in mind this result.

IoU = 0.16 might seem a poor result for IoU. However, we need to consider the fact that the IoU can rapidly decrease if the two predictions are not matching too well. For instance, Fig. 5 represents a situation were the model seems to be doing a great prediction: it identifies correctly 2 of the 3 trucks in that image. However, the IoU for the predictions in this image is 0.45 (which is far from the 2/3 one might expect) due to the fact that the rectangles do not match perfectly. Furthermore, this image is one of the baseline's best predictions (there are some images where the baseline does not identify any truck at all), so when we take into account all the other predictions it is no surprise that the average value of IoU is just 0.16.

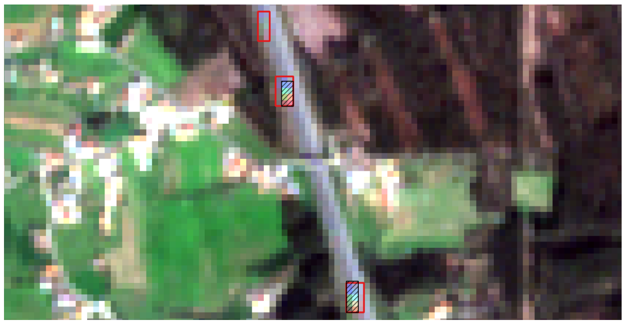
Fig. 5. Baseline's bounding boxes (black) and bounding boxes from our manually annotated dataset (red). The Intersection over Union for this image equals 0.45.

C. YOLO implementation

The YOLO model can be used for several purposes, including tasks such as OD (identifying and localizing objects or regions of interest in images or videos), image segmentation (dividing visual content into distinct regions or pixels corresponding to various objects or classes), image classification (predicting class labels for input images) and pose estimation (identifying objects and estimating their keypoints in images or videos). As previously mentioned, we specifically applied YOLO for OD in our work.

In this section, we intend to explore YOLO's hyperparameters and settings, recognizing their crucial role in the model's performance, speed and accuracy. These hyperparameters can be adjusted to improve the model's results, tuning its performance to specific tasks and dataset characteristics.

The training settings for YOLO models include various hyperparameters and configurations used during training. We will now focus on providing detail on the hyperparameters we explored during this project. To begin with, YOLOv8 is available in several sizes which differ in the complexity of their variants and the number of parameters. There are 5 different sizes available - nano, small, medium, large, x-large. [17] As a general guideline, larger models have more complex variants and higher accuracy, however, they demand more processing time. Therefore, our objective is to find the best trade-off between accuracy and processing time for the data and task at hand. [18] Additionally, it is important to choose an optimizer. Options such as SGD, Adam, AdamW, Adamax, NAdam, RAdam, and RMSProp present varied mechanisms for guiding parameter adjustments to minimize the loss function. In this work we focused on the first two optimizers. [8]

Another relevant feature is the image size for training our YOLO model. Given that YOLO involves downsample operations that eventually reduce the original image to 1/32 of its size, it's necessary to chose a multiple of 32 as the size of input image. Our biggest image has dimensions 297x152 pixels, so we selected a size of 320x320. We also experimented with the standard size of 640x640. [19]

Moreover, our task of identifying colorful pixels as trucks is highly specific and using the pretrained model proved not to be helpful for our dataset. Thus, we set this parameter to false.

Finally, the learning rate and the number of epochs are also adjustable parameters and in this case, we experimented with values of 0.1, 0.01, and 0.001 for the learning rate, and 10 and 15 for the number of epochs. For all the others hyperparameters we kept the predefined value. [8]

After training, YOLO provides a set of metrics such as precision (P), recall (R), mean Average Precision at 50 (mAP50) and mean Average Precision between 50-95 (mAP50-95) at each epoch for both the training and validation phases. [20] The mAP50 focuses on the Average Precision (AP) at an IoU threshold of 0.5 for the specific class, while mAP50-95 measures the mean average precision across IoU thresholds ranging from 0.5 to 0.95. In the context of a singular class as is our case, these metrics provide a measure of precision and recall for the model's predictions related to that class. [21] The goal is to evaluate the model's performance in detecting instances, considering different levels of overlap between predicted bounding boxes and the ground truth annotations for that class.

It also returns visual outputs, including plots like the Precision-Recall Curve, among others that will be explored later. These will help evaluating the effectiveness of different hyperparameter combinations.

The prediction mode also involves adjustable hyperparameters, such as the confidence threshold - determines the minimum confidence score required for an object to be considered in the predictions - and IoU threshold for Non-Maximum Suppression (NMS) - specifies the degree of overlap required for one of the boxes to be suppressed. [8] In our case, all parameters kept their predefined value.

V. Results

As previously mentioned, we began by training the YOLO with a dataset of 150 images. We had very unsatisfactory results in that scenario: considering our best model, we obtained an IoU = 0.07 for the test set, which is inferior to the baseline result. As an attempt to improve our results, we increased the dataset size, from 150 to 300 images. In fact, models like YOLO - which have millions of parameters - should be trained with large datasets [26]. Even though 300 images might not be a large number of examples to train the model, it should definitely perform better than 150. Our hypothesis ended up to be true, as we obtained much better results after having trained YOLO with a larger dataset.

Model size	Optimizer	learning rate	imgsz	Epochs	IoU
small	SGD	0.01	320	10	0.41
small	SGD	0.01	320	15	0.43
small	SGD	0.01	640	10	0.53
small	SGD	0.01	640	15	0.48
small	SGD	0.001	640	10	0.35
small	Adam	0.01	320	10	0.41
small	Adam	0.01	320	15	0.20
small	Adam	0.01	640	10	0.37
small	Adam	0.01	640	15	0.59
small	Adam	0.001	320	10	0.28
small	Adam	0.001	640	10	0.61
small	**Adam**	**0.001**	**640**	**15**	**0.63**
medium	SGD	0.01	320	10	0.33

TABLE I
IoU COMPUTED IN THE TEST SET, CONSIDERING EVERY CONFIGURATION.

The Table II (in Appendix) summarizes the results we obtained for all the configurations of hyperparameters we tried. Hyperparameter tuning was made according to the results achieved for the validation set - namely the validation precision, recall, mAP50 and mAP50-95, since these were the metrics given automatically by YOLO. Note that two of the lines in the table are empty and this is because the model simply did not run and stopped (it was maybe to heavy for the Google Colab environment we used).

In addition, for the best epoch considering each configuration, we computed the IoU for the test set - see Table I. One thing we noticed is that for every configuration, we obtain an IoU greater than the one obtained with the baseline.

Our best model was achieved with the *small* version of YOLO, using *Adam* as an optimizer, a learning rate of 0.001 and image size = 640, after training it for 15 epochs. It was our *best*, in the sense that it yielded the higher IoU value for the test set, **IoU** = **0.63**, which is undeniably larger than the result obtained with the baseline. Accordingly, it yielded the highest values in all the four metrics during validation, see Table II.

The next figure (Fig. 6) serves as a visual example for the predictions of our model. It identifies accurately 3 of the 4 trucks present in the image and these 3 boxes almost overlap perfectly. In fact, its IoU for this particular image is IoU = 0.58.

For this configuration, we obtained some insights provided by YOLO. To begin with, Fig 7 represents the Precision-Recall Curve, which shows us the trade-off between precision and recall at different thresholds. The closer precision and recall are to 1, the better is the classifier - and Fig 7 shows satisfactory results, as the curve is quite close to a perfect classifier. The area under the curve is 0.745 (an ideal classifier has area under the curve equal to 1), for an mAP at 0.5 [22].

Fig 8 contains three plots that helps us understand the relation between the three metrics - recall, precision and F1-score - and the confidence threshold, which is a YOLO

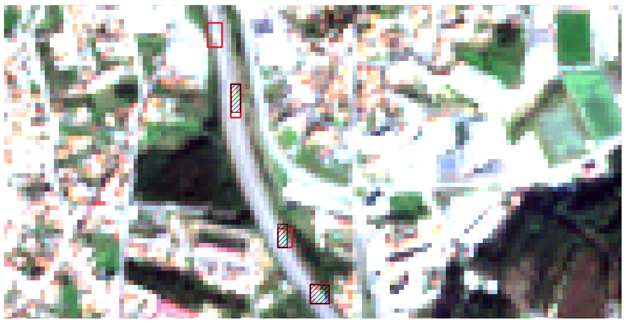

Fig. 6. Our model's prediction boxes (black) and bounding boxes from our manually annotated dataset (red). The IoU between these sets is 0.58.

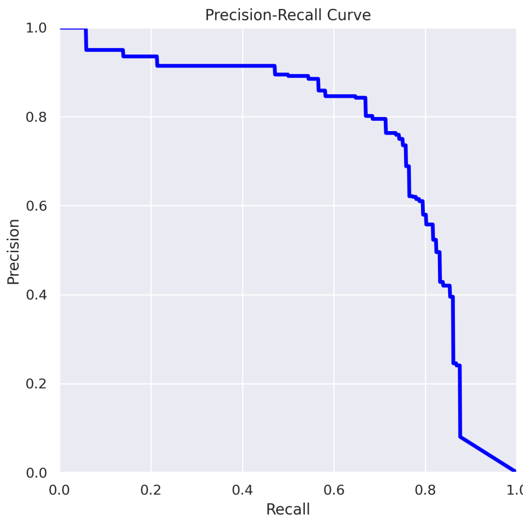

Fig. 7. Precision-Recall curve for the best model configuration

parameter set to 0.25 by default [8]. This threshold is the confidence value upon which a bounding box becomes a prediction. On the left we see how the recall decreases with the increase of the confidence, due to the reduction on the number of true positives. On the center we observe, instead, that the precision decreases with the increase of the confidence threshold, since there will be less false positives. The most relevant plot, however, is the F1-score versus confidence (on the right side of Fig. 8), because the F1 reflects the equilibrium between recall and precision. In that plot, the best F1-score is 0.75, achieved at a confidence threshold of 0.26 (which is very close to the default 0.25 and this could explain the good results obtained). We see that there is an optimal interval for the confidence, but then, approximately when it becomes larger than 0.8, F1-score goes down to zero. This makes sense because it matches the point where recall goes to zero as well. To sum up, all the three plots behave as expected, noting that the F1-score - which is the harmonic mean between recall and precision - reaches a maximum, for a specific threshold.

Furthermore, Fig 9 represents the evolution of different loss functions along the 15 epochs for train and validation sets.

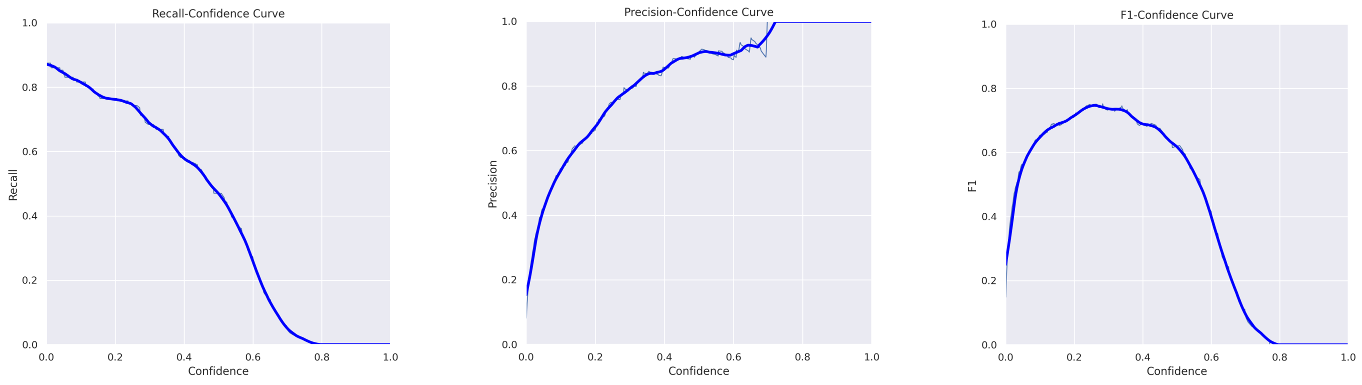

Fig. 8. The dependence of Recall (left), Precision (center) and F1-score (right) on the confidence threshold.

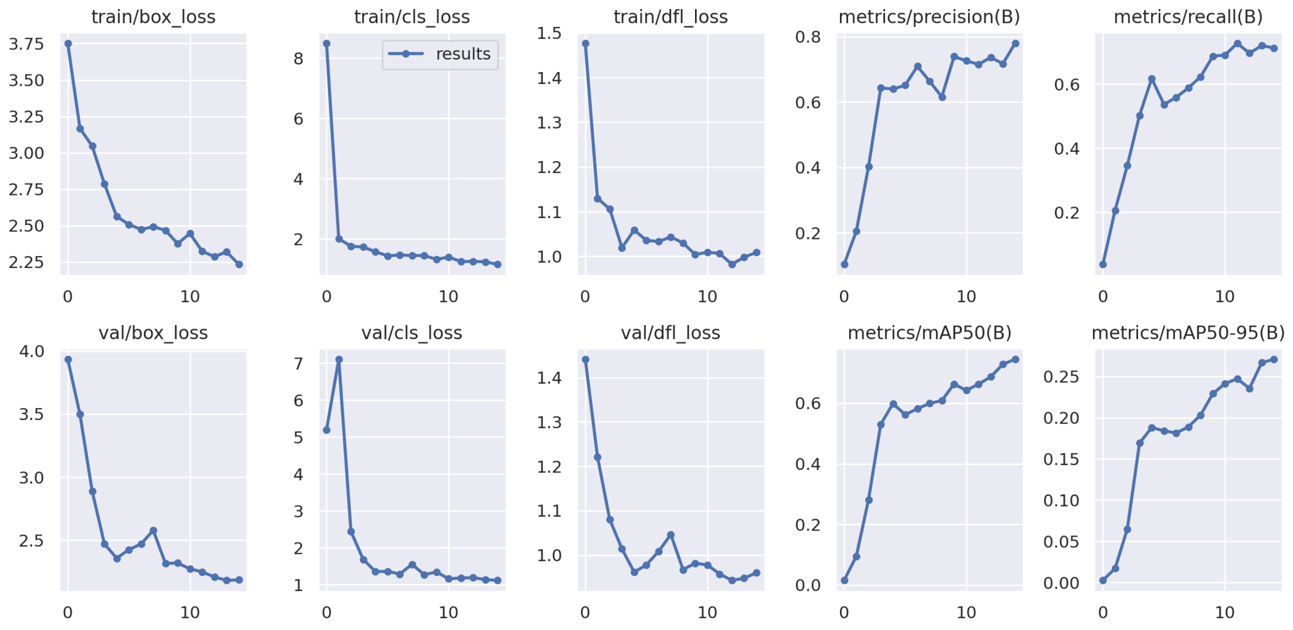

Fig. 9. Evolution of losses and metrics over the 15 epochs

In addition, it also shows on the right side, the evolution of the four metrics previously mentioned (that were used for hyperparameter tuning): precision, recall, mAP50 and mAP50-95.

There are three different types of loss shown: box loss (box_loss), classification loss (cls_loss) and dual focal loss (dfl_loss). The box loss represents the model's ability to accurately determine the centre of an object and how the predicted bounding box covers an object. Classification loss gives an idea of how well the model can predict the correct class of a object. The dual focal loss addresses the problem of class imbalance during training. In a single-class detection scenario, class imbalance refers to the unequal distribution between the detection target objects and the background. [23] [24]

We see, in fact, that our model is behaving as expected: the loss is decreasing over the epochs, both for train and validation sets and the four metrics that evaluate the model's performance are permanently increasing over the epochs. This seems to indicate that there was no overfitting during train, since the model is making overall good predictions not only for the training set but also for the validation set.

YOLO also returns Figure 10 (in Appendix), showing the predictions for some images from the validation set, alongside with corresponding confidence scores assigned by our model. In this context, the confidence score represents the probability that the identified object, as determined by the model, is indeed a truck. We observe that all the scores are higher than 0.25, as this is the predefined minimum threshold for this parameter. [25]

VI. Conclusions

Having obtained IoU = 0.63, we can confidently say that our method performed better than the baseline (IoU = 0.16).

This means that our approach is, as far as we know, the most accurate method for truck detection in Sentinel-2 images.

Here, we will make several final considerations that should also be taken into account to get the full perspective of our work:

- We tested the different configurations of our model for 10 and 15 epochs. By inspecting Tables I and II we see that increasing the number of epochs results in a better performance, in most of the times and specifically in our best scenario as well. This means that increasing further the number of epochs should probably increase the performance of the model and should be subject of future work.
- The YOLO model was trained with the annotations *we* made. Also, we evaluated both the baseline and our model based in *our* annotations. The point here is that this could affect the way we are evaluating the baseline. More concretely, the baseline identifies the trucks based on a set of rules that the authors designed. Despite that, it is being evaluated by comparing to an annotation made by *us*, that could have had slightly different criteria (*e.g.* covering different area around each truck, deciding whether or not a set of pixels constitutes a truck) and could affect the baseline performance in the IoU metric.
- The downsides of using IoU, considering the way we computed it, are that if the model predicts two or more overlapping boxes, which are on top of the ground-truth box, it will count for the IoU as if several objects were in that position - which should not happen because it would mean that the model was identifying more than one truck in the same position. Another downside is if there are two ground-truth annotations very close to each other, but there is only one prediction box that overlaps both, the IoU value would increase but it would not reflect the model's true performance. Succinctly, these problems might arise when there is no one-to-one correspondence between boxes, for some reason. However, after inspecting some of the predictions yielded by YOLO we do not believe that is happening in our method.

VII. Future work

In general, our model yielded positive results when compared to the baseline on our dataset. Nevertheless, space for additional fine-tuning to improve these results. In the pursuit of improvement, we present several suggestions below.

Beyond the discussed hyperparameters in subsection IV-C, additional key parameters may benefit further exploration. To begin with, increasing the number of epochs should improve the model performance, as already mentioned in section VI. During the training phase, the batch size, momentum, and weight decay are factors that could play a crucial role in the model improvement. Exploring different configurations for confidence thresholds and IoU threshold for Non-Maximum Suppression (NMS) during the prediction phase is also worth investigating.

As we mentioned before, expanding our dataset showed a substantial impact on model performance. Thus, we consider further augmentation a potential step for even greater improvements. In fact, Ultralytics recommends a minimum of 1500 images per class in order to take full advantage of the performance of the models. [26]

Using a more complex model such as the large or x-large YOLOv8 could potentially yield improved results, however, we lack of sufficient computational power to test this options.

Additionally, it would be interesting to incorporate national roads into the study, not exclusively focusing on motorways. Considering the significant presence of truck traffic, including data from national roads would provide a more comprehensive perspective for the purposes of this study.

Acknowledgments

We would like to express our sincere gratitude to CEiiA for providing us with the opportunity to undertake our internship.

Special thanks to Jorge Cabral, André Dias, Nelson Pires, Sofia Oliveira and Alberto Barbosa, whose guidance and mentorship significantly contributed to the project development.

References

[1] Fisser, H.; Khorsandi, E.; Wegmann, M.; Baier, F. Detecting Moving Trucks on Roads Using Sentinel-2 Data. Remote Sens. 2022, 14, 1595. https://doi.org/10.3390/rs14071595 .

[2] "What is remote sensing and what is it used for?", U.S. Geological Survey. Available at: https://www.usgs.gov/faqs/what-remote-sensing-and-what-it-used#:~:text=Remote%20sensing%20is%20the%20process,sense%22%20things%20about%20the%20Earth .

[3] European Environment Agency. Air Quality in Europe—2019 Report; European Environment Agency: Copenhagen, Denmark, 2019.

[4] "Sentinel-2", (2023) Wikipedia. Available at: https://en.wikipedia.org/wiki/Sentinel-2.

[5] "Object Detection", (2023) Wikipedia. Available at: https://en.wikipedia.org/wiki/Object_detection .

[6] "Introduction to deep learning: Machine learning vs. Deep Learning", Mathworks Videos, MATLAB. Available at: https://www.mathworks.com/videos/introduction-to-deep-learning-machine-learning-vs-deep-learning-1489503513018.html.

[7] Brownlee, J. (2021), "A gentle introduction to object recognition with deep learning", Machine Learning Mastery. Available at: https://machinelearningmastery.com/object-recognition-with-deep-learning/ .

[8] "Configuration", Ultralytics YOLOv8 Docs. Available at: https://docs.ultralytics.com/usage/cfg/

[9] Redmon, J., "YOLO: Real-Time Object Detection". Available at: https://pjreddie.com/darknet/yolo/.

[10] Copernicus browser. Available at: http://dataspace.copernicus.eu/browser/ .

[11] "Level-1C", Sentinel Online. Available at: https://sentinel.esa.int/web/sentinel/user-guides/sentinel-2-msi/product-types/level-1c

[12] "Level-2A", Sentinel Online. Available at: https://sentinel.esa.int/web/sentinel/user-guides/sentinel-2-msi/product-types/level-2a

[13] Qgis, "Welcome to the QGIS project!", Available at: https://qgis.org/en/site/ .

[14] OpenStreetMap. Available at: https://www.openstreetmap.org/#map=7/39.602/-7.839 .

[15] Keita, Z. (2022), "Yolo Object Detection explained: A beginner's guide", DataCamp. Available at: https://www.datacamp.com/blog/yolo-object-detection-explained.

[16] "Give your software the power to see objects in images and video", Roboflow. Available at: https://roboflow.com/

[17] Ultralytics, "Yolov8", Ultralytics YOLOv8 Docs. Available at: https://docs.ultralytics.com/models/yolov8/ .

[18] Abdullah, M. (2023), ”Yolo working principle, difference between its DDIFFERENT variants and versions”, Medium. Available at: https://medium.com/@muhabd51/yolo-working-principle-difference-between-its-ddifferent-variants-and-versions-95b8ad7b95ab.

[19] Ultralytics, ”Why can’t I train and inference my model on my original image size? · issue #1113 · Ultralytics/ultralytics”, GitHub. Available at: https://github.com/ultralytics/ultralytics/issues/1113 .

[20] Ultralytics, ”Yolo Performance Metrics, YOLO Performance Metrics”, Ultralytics YOLOv8 Docs. Available at: https://docs.ultralytics.com/guides/yolo-performance-metrics/ .

[21] ”What is the difference between MAP50 and MAP50-95?: 4 answers from research papers”, SciSpace - Question. Available at: https://typeset.io/questions/what-is-the-difference-between-map50-and-map50-95-5asxtut1cj .

[22] Steen, D. (2020), ”Precision-recall curves”, Medium. Available at: https://medium.com/@douglaspsteen/precision-recall-curves-d32e5b290248 .

[23] Kasper-Eulaers, M. et al. (2021), ”Short communication: Detecting heavy goods vehicles in rest areas in winter conditions using yolov5”, MDPI. Available at: https://www.mdpi.com/1999-4893/14/4/114).

[24] Ultralytics, ”About the functionality of DFL_loss · issue #4219 · Ultralytics/ultralytics”, GitHub. Available at: https://github.com/ultralytics/ultralytics/issues/4219 .

[25] ”Confidence Score - an overview”, ScienceDirect Topics. Available at: https://www.sciencedirect.com/topics/computer-science/confidence-score#:~:text=A%20confidence%20score%20is%20calculated,(Intersection%20over%20Union)%20thresholds. .

[26] Ultralytics, ”Tips for Best Training Results”, Ultralytics YOLOv8 Docs. Available at: https://docs.ultralytics.com/yolov5/tutorials/tips_for_best_training_results/ .

APPENDIX

Model size	Optimizer	Learning Rate	Image Size	Epochs	Precision	Recall	mAP50	mAP50-95
small	SGD	0.1	320	10	0	0	0	0
small	SGD	0.1	640	10	0.00653	0.0515	0.00493	0.00119
small	SGD	0.01	320	10	0.508	0.397	0.338	0.073
small	SGD	0.01	320	15	0.425	0.368	0.285	0.0776
small	SGD	0.01	640	10	0.634	0.691	0.657	0.211
small	SGD	0.01	640	15	0.694	0.584	0.632	0.225
small	SGD	0.001	320	10	0.173	0.125	0.0571	0.00895
small	SGD	0.001	640	10	0.475	0.39	0.373	0.0859
small	Adam	0.1	320	10	0	0	0	0
small	Adam	0.1	640	10	0	0	0	0
small	Adam	0.01	320	10	0.386	0.272	0.196	0.0419
small	Adam	0.01	320	15	0.473	0.279	0.249	0.0628
small	Adam	0.01	640	10	0.367	0.375	0.259	0.0601
small	Adam	0.01	640	15	0.71	0.676	0.662	0.205
small	Adam	0.001	320	10	0.492	0.399	0.353	0.0884
small	Adam	0.001	640	10	0.639	0.632	0.615	0.195
small	**Adam**	**0.001**	**640**	**15**	**0.791**	**0.723**	**0.749**	**0.273**
medium	SGD	0.01	320	10	0.474	0.331	0.315	0.0822
medium	SGD	0.01	640	10	-	-	-	-
medium	Adam	0.01	320	10	0	0	0	0
medium	Adam	0.01	640	10	-	-	-	-

TABLE II
RESULTS FOR THE VALIDATION SET, CONSIDERING EVERY CONFIGURATION.

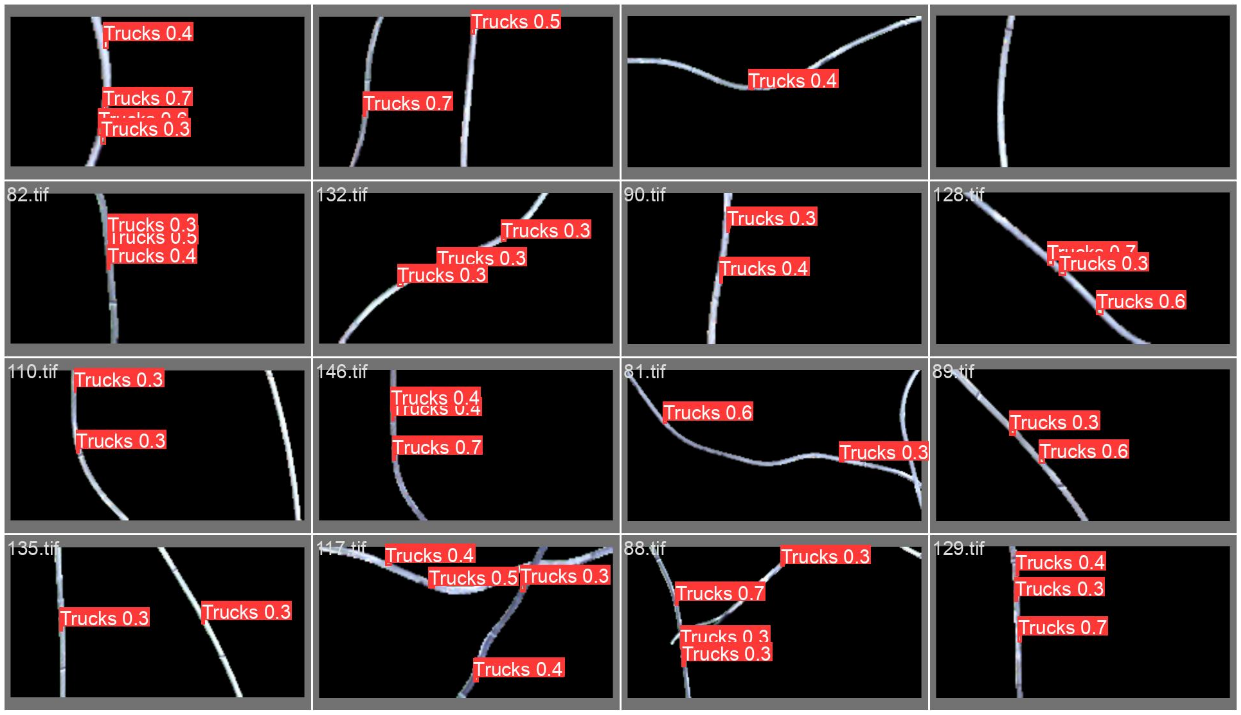

Fig. 10. Prediction bounding boxes for a batch and the confidence value for each detection.