Internship Report - Third Project

Assessment of Intel & ImageAI networks for car detection

Ismail OUSSAID

with the supervision of: Enrico Marchetto

Summary:

Quividi

A. Current car detection models

The idea of the project is to evaluate many open-source networks. Considering that Quividi has a habit of using Intel tools and that ImageAI networks are the state-of-the-art, I found it particularly interesting to have an overview of both packs of models.

1. ImageAI models

When talking about object detection, one can easily find YOLO and Resnet, which are the state-of-the-art of object detection. The first one is available in multiple versions from v1 to v4, with alternative versions of which we find tiny YOLO, that is studied here.
These models are known to be very efficient according to their mAP (mean average precision) that one can check at https://rb.gy/e3py0c. Their efficiency in different conditions will be screened later but for now, let's dig into their computational cost.
In this paper, the main focus is on YOLO v3, tiny YOLO and Resnet COCO 50.

Model	FLOPS
YOLO v3	65.3 Bn
Tiny YOLO	5.8 Bn
Resnet COCO 50 (Retina)	4.3 Bn

These models are studied on Python by using the *ImageAI* framework and *Tensorflow* 1.13.1. Furthermore, I used a method *CustomObjects* of object customization to allow me to detect only cars, trucks, buses and motorcycles.

2.Intel OpenVino Object Detection

OpenVINO™ toolkit provides a set of pre-trained models that you can use for learning and demo purposes or for developing deep learning software. Among their pre-trained models, one can find the Object Detection models mostly SSD-based and provide reasonable accuracy/performance trade-offs.

Model	FLOPs
pedestrian-and-vehicle-detector-adas-0001	3.97 Bn
person-vehicle-bike-detection-crossroad-0078	3.96 Bn
person-vehicle-bike-detection-crossroad-1016	3.56 Bn
person-vehicle-bike-detection-crossroad-1016-FP16	3.56 Bn
vehicle-detection-adas-0002	2.79 Bn
vehicle-detection-adas-binary-0001	0.94 Bn

In simple terms, the bigger the FLOPS, the slower the inference so it is interesting to compare all these models performances and FLOPs to find a good tradeoff. As a matter of fact, all Intel models above are computationally cheaper than every ImageAI's available model. Then, if any of Intel's network reaches an equivalent or better performance than YOLOs' or Retina's, it might be a considerable piece of information.

In the next pages, I put a focus on the technical approaches I innovated with the help of Enrico to evaluate open source networks on different kinds of images (Bright images, rainy, etc.). Most importantly, many figures will be brought to prove my conclusions and that helps a lot to understand the impact of images' quality and the climate on the performance of a network.

B. Networks assessment protocol

Most of the time, object detectors are capable of detecting cars when the light is good. Here, we assume H_1 : "Yolo is a reference detector" so we will evaluate our models by confronting their performance to Yolo's.

1.Dataset

a) Bright images

To build the dataset, I used the video https://cutt.ly/zhStiz2 as its brightness is exactly what we're looking for and there are enough cars/samples to notify tremendous differences of performance between models.

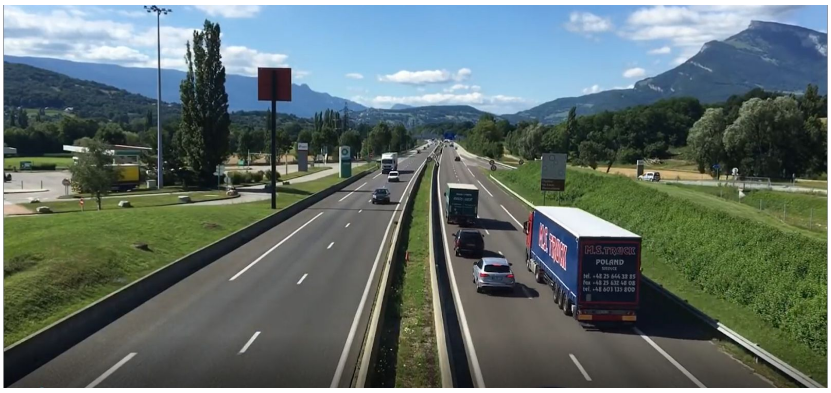


Fig.1 Bright sample from the video

As the networks need and/or accept 608x608 size images and to make a more significant test, I have decided to crop the 1280x720 size samples into random windows of 608x608 size to ensure that we have different angles of image.

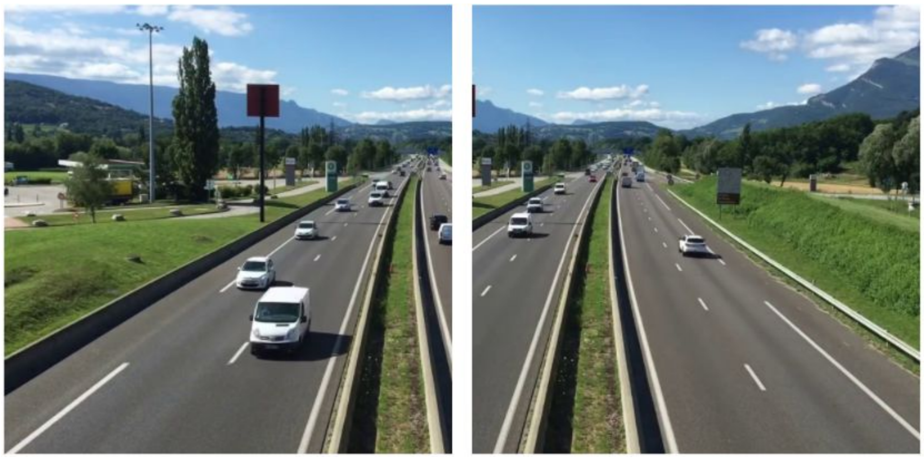


Fig.2 Two random cropped samples

b) Rainy images

The dataset was hard to build as I didn't find any long video from a camera filming routes on a rainy day. Then, I have taken the following videos https://cutt.ly/7hFSfFE, https://cutt.ly/MhFSva2, and https://cutt.ly/xhFSD02 where I have three camera qualities, angles and intensity of rain.

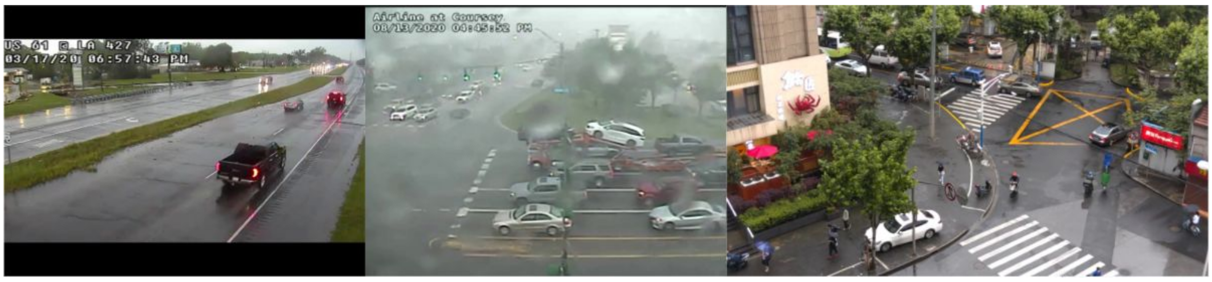


Fig.3 One sample from each dataset (D_1 , D_2 and D_3)*

N.B: Their respective names in the laptop are rainy_frames, rainy_frames_1 and rainy_frames_2

As the videos V_1 and V_2 are short and V_3 is long, I have chosen to save one frame each 2 frames for the two first ones and each 10 frames for the latter to have a closer amount of samples between the datasets and ensure that between each frame of a dataset, the vehicles don't have the same position.

2.Assessment Protocol

After many attempts, I have chosen to find the best model with the following protocol. **First**, I developed customized metrics to compare 2 networks N_2 & N_1 with:

- TP is the amount of vehicles detected by N_1 and by N_2
- TN is the amount of detections identified as vehicles by N_2 but not by N_1
- FN is the amount of detections identified as vehicles by N_1 and not by

First thing to notice is that if we compare two networks N_i and N_j, for each sample X, the numbers $n_{X,i}$ of vehicles detected by N_i and $n_{X,j}$ of vehicles detected by N_j satisfies the relationship:

$max(TP, TN, FN) \leq min(n_{X,i} \; ; \; n_{X,j})$ (R_1) and $TP + TN + FN = max(n_{X,i} \; ; \; n_{X,j})$ (R_2)

To compute TP, TN and FN, let's check how it is done for one sample X. In practice, the inference $I_{X,k}$ by a network N_k (with a minimum confidence level: c =40%) generates $n_{X,k}$ rows (number of vehicles detected by N_k in sample X) with features:

- x_1 : left bottom corner x-axis coordinate
- y_1 : left bottom corner y-axis coordinate
- w : width of the bounding box
- h : height of the bounding box
- p : confidence of the inference
- $object$: category (car, truck, bus, motorcycle)

To count the amount of detections for one sample, I have to avoid the False Positives that are generated after an overview of images post-inference, it happens that a car (Fig.4) is identified as multiple vehicles.

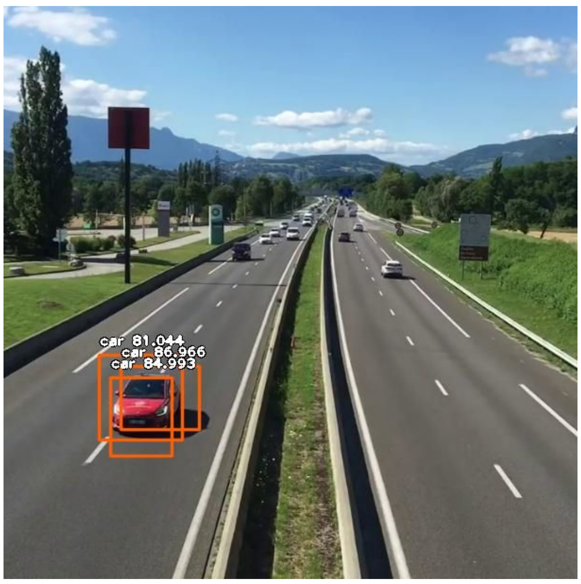


Fig.4 Sample with 2 FP

Second, to avoid this problem, I introduce a similarity measure, which is the IOU (Intersection Over Union). I decided an *average threshold* of 0.8 that allows me to consider that two boxes in an image correspond to the very same vehicle. Then, all the overlapping bounding boxes are considered as members of a cluster and the algorithm computes an average clustering that generates a new sample that has the average coordinates of all the boxes' coordinates.

For such a sample (Fig.4), we have this inference:

file	object	x1	y1	w	h	p
img_38.jpg	car	101	370	88	90	81.043541431427
img_38.jpg	car	124	382	83	67	86.96645498275757
img_38.jpg	car	113	394	65	83	84.9931001663208

After average clustering, we have:

file	object	x1	y1	w	h	p
img_38.jpg	car	113	382	79	80	

Third, each networks is capable of generating average clusters and now, we are capable of finding real inferences to compare two networks N_1 and N_2, I make a correspondence between the average clusters made by both networks.
For instance, in this very example (Fig.3), I made an algorithm identifying the red car detected by YOLO v3 as the same car detected by Resnet. Next, all the other bounding boxes do not find their homologue in Renset's inference.

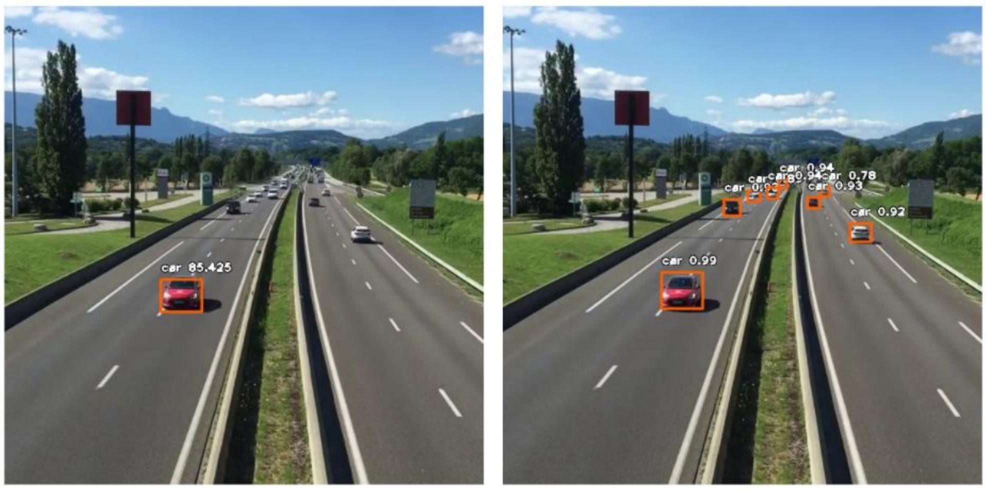
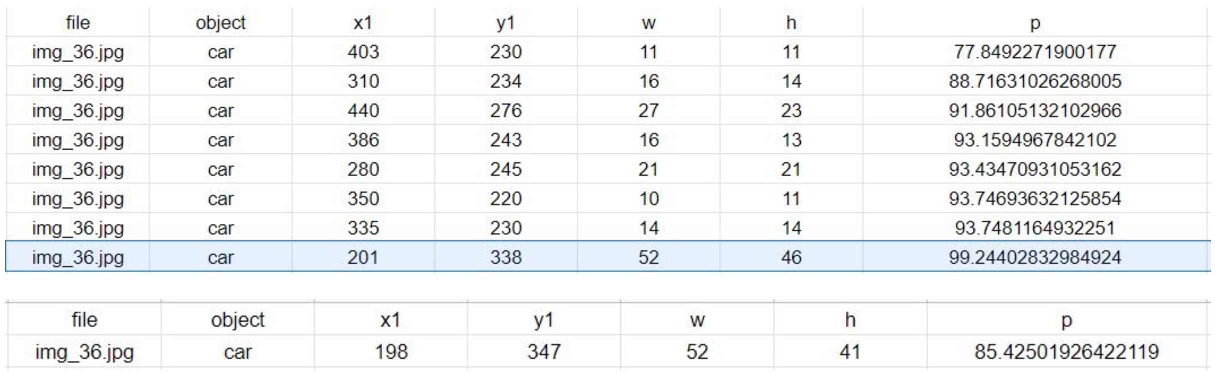


Fig.5 A sample's generated inference by Resnet (left) and YOLO v3 (right)

The algorithm is based on the IOU and I decided to use a *comparison threshold* of 0.2 that allows me to consider that two bounding boxes from two inferences correspond to the same object. The parameter's value is estimated with an heuristic approach as Enrico & I have tested some values and it happens this value is realistic.



file	object	x1	y1	w	h	p
img_36.jpg	car	403	230	11	11	77.8492271900177
img_36.jpg	car	310	234	16	14	88.71631026268005
img_36.jpg	car	440	276	27	23	91.86105132102966
img_36.jpg	car	386	243	16	13	93.1594967842102
img_36.jpg	car	280	245	21	21	93.43470931053162
img_36.jpg	car	350	220	10	11	93.74693632125854
img_36.jpg	car	335	230	14	14	93.7481164932251
img_36.jpg	car	201	338	52	46	99.24402832984924

file	object	x1	y1	w	h	p
img_36.jpg	car	198	347	52	41	85.42501926422119

Fig.6 One sample's inference by YOLO v3 (top) and Resnet (bottom)

For the example above, YOLO finds 8 cars (as you can see in Fig.5) and the one marked in blue corresponds to the only car found by Resnet (Fig.4) because the IOU is 0.57.
So, for this sample, there is only one car detected by YOLO and Resnet, 7 vehicles detected by YOLO and not Resnet and 0 vehicle detected by Resnet and not YOLO. Thus, TP=1, FN=7 and FP=0.

Finally, The computation process is repeated for every sample and the TP, TN FN counts are summed over all the images of the dataset. Especially, if there is no detection by both networks for a sample, TP, FN and FN are null.

All in all, at this step, we are able to identify all the real detections made by a network, generating a tab with all the comparisons one to one between all the models we want. Next step is to do so with ImageAI & Intel Open Vino networks.

C. Results of the assessments

For the assessment, we assume H_1 : "Yolo is a reference detector" so we will evaluate our models by confronting their performance to Yolo's.

After making inferences with all the networks, I decided to analyze their performances thanks to TP, TN & FN. I need to find an index that evaluates how a network N_2 performs in comparison with a network N_1.

The basic rule here is that N_2 is better than N_1 if TN is large, FN small. Thus, $TN - FN$ must be large so by normalizing, $\frac{TN-FN}{TP+TN+FN}$ must also be large. Then, to amplify the difference $TN - FN$, I propose: $I(N_1, N_2) = e^{\frac{TN-FN}{TP+TN+FN}}$.

Such measure is interesting as it is equal to one when N_1 and N_2 are the same model Also, it is larger than one if TN is higher than FN, which means that the amount of vehicles not identified as such by N_1 but detected as such by N_2 is higher than the amount of vehicles not identified as such by N_2 but detected as such by N_1 . In simple terms, N_1 misses more cars than N_2 . Conversely, if $I(N_1, N_2)$ is smaller than one, then N_1 performs better than N_2 .

1. Comparison on bright videos

To assess all the models, I have computed the index **I** between YOLO and all the networks N_2 (Fig.7).

YOLO being considered to have an index of 1, one can deduce that the two crossroad models perform better than YOLO while all others don't. However, Vehicle Adas 0002 performs very well, even if it is weaker than YOLO.

In particular, let's dig the meaning of having an index of 0.91.

$$e^{\frac{TN-FN}{TP+TN+FN}} = 0.91 \Leftrightarrow -0,094 \simeq ln(0.91) = \frac{TN-FN}{TP+TN+FN}$$

$$\Leftrightarrow |TN - FN| \simeq 0.094 . Total\ amount\ of\ predictions$$

In simpler terms, for Vehicle Adas 0002, the gap between its amount of misdetections and the misdetections of YOLO v3 is almost the tenth of the maximum number of detections of the two models (R_2) and the gap $TN - FN$ is negative so there are more misclassifications for Vehicle Adas 0002.

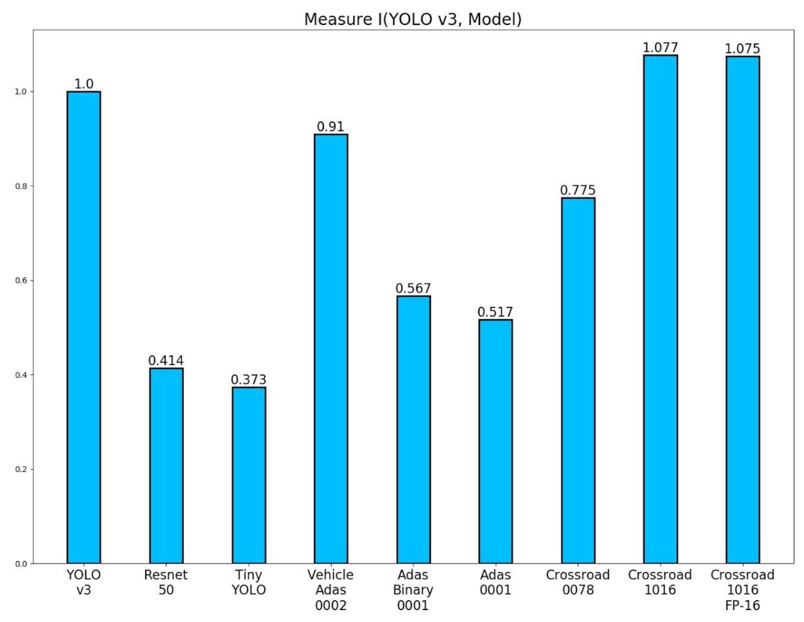


Fig.7 Bar plot of performance measures of every network for the bright dataset

To have an overview of the tradeoff Index/FLOPs, I have chosen to compute the ratio:

$$ratio = \frac{I(YOLO, N_2)^2}{flops(N_2)}$$

$$\text{with } flops(N_2) = \frac{FLOPS(N_2)}{max(FLOPS)}$$

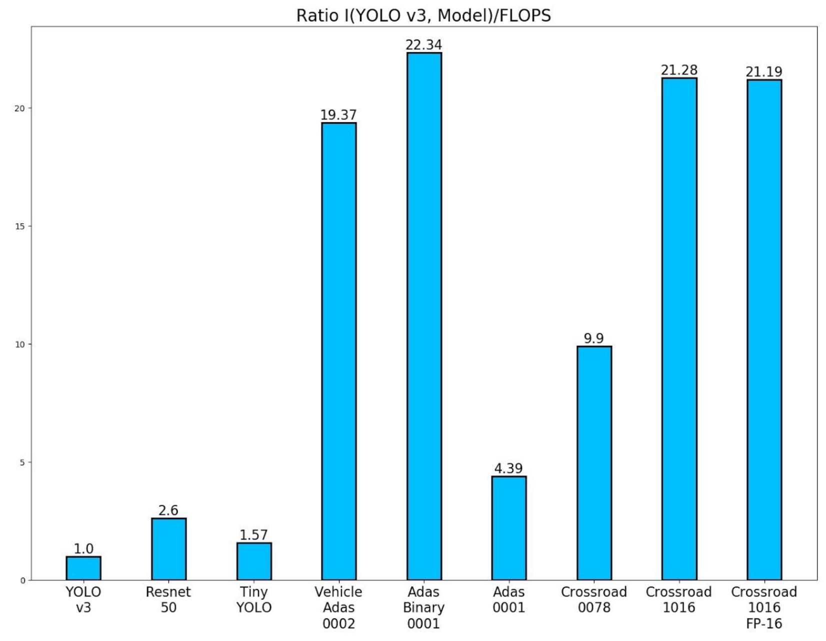


Fig.8 Bar plot of ratio measure/flops for the bright dataset

The plot (Fig.8) stresses out that even if Adas binary 0001 is less efficient than Crossroad models, its low FLOPS makes it worth using as its ratio is better. Surprisingly, tiny YOLO and Resnet don't perform well in this situation and are the worst.

2. Comparison on rainy videos

Considering the images we have, it's expectable to watch a decrease of the networks' performances as the quality of images and the brightness have decreased. Let's check!

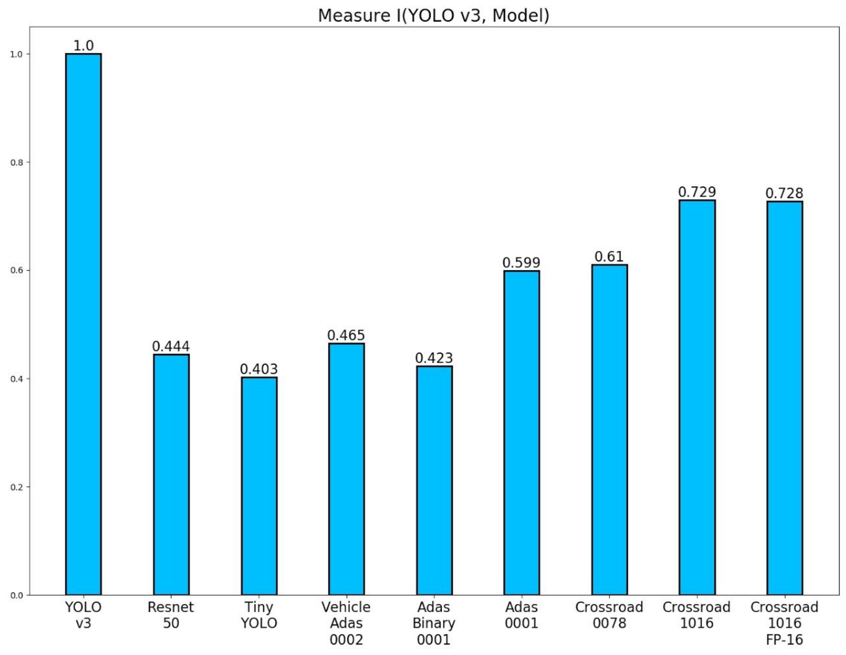


Fig.9 Bar plot of the performance of every network for D_1

There is an obvious decrease of the performance of the Crossroad 1016 models are very close and these are the best models compared to YOLO v3 for D_1. Here, it means, that the gap between its amount of misdetections and the misdetections of YOLO v3 is almost the third of the maximum number of detections of the two models and there are more misclassifications for Crossroad models. Furthermore, Crossroad 1016 FP-16 is almost as good as Crossroad 1016 so it is worth exploiting it. Here, Adas Binary 0001 doesn't perform as well, but considering its FLOPs, it's surely due to its limited number of layers, that might not.

However, surprisingly, Resnet and Tiny YOLO are not very consistent can't confront such a bad quality of image (Fig.3 left)

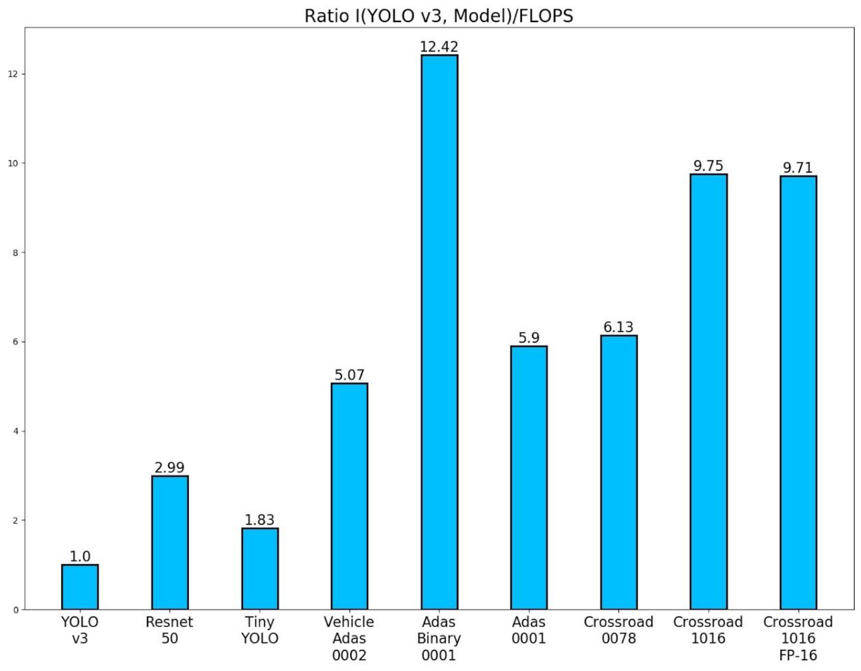


Fig.10 Bar plot of ratio measure/flops of every network for D_1

But, considering the FLOPs (Fig.10) of each network, the ratio is especially high for Adas Binary 0001. Even if the performance is not very good, one can use an approach where we guess the real number of vehicles based on the capacity of the network. For instance, if we know that the Adas detects 40% of the cars, we multiply by 2.5 to have an estimated real amount of cars.

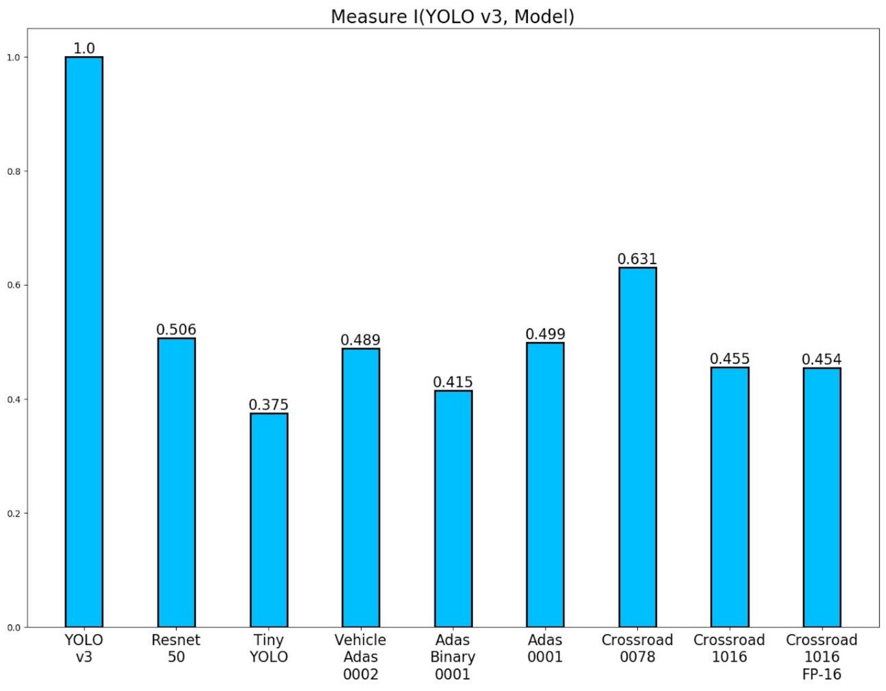


Fig.11 Bar plot of the performance of every network for D_2

On the D_2 dataset, things get even worse as the bar plot (Fig.11) shows it. Due to the rain, there are lots of drops on the camera that add blur and make detections harder. Crossroad 0078 happens to be the best model when confronted with YOLO.
For the best model, the gap between its amount of misdetections and the misdetections of YOLO v3 is almost the half of the maximum number of detections of the two models and there are more misclassifications for Crossroad models.

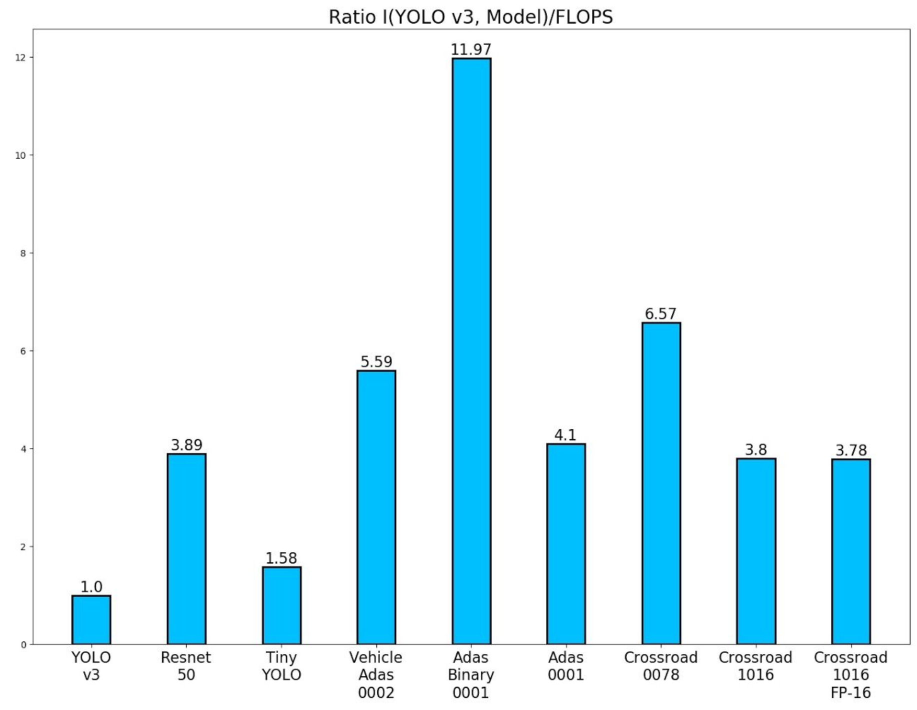


Fig.12 Bar plot of ratio measure/flops of every network for D_2

However, here the very low FLOPs of Adas Binary 0001 still makes it a very good competitor while Crossroad 0078 is the second due to its raw performance. This still proves that those non-Intel networks are very weak when it comes to the tradeoff accuracy/FLOPs.

For the final analysis, I went through the dataset D_3 (Fig.13) with high definition (720p) images on a rainy day and results are very interesting as it confirms that having good cameras can significantly improve results. Indeed, all the Crossroad models perform very well with +0.91 of performance index and FP16 version of Crossroad 1016 is better than the 32-bit version.

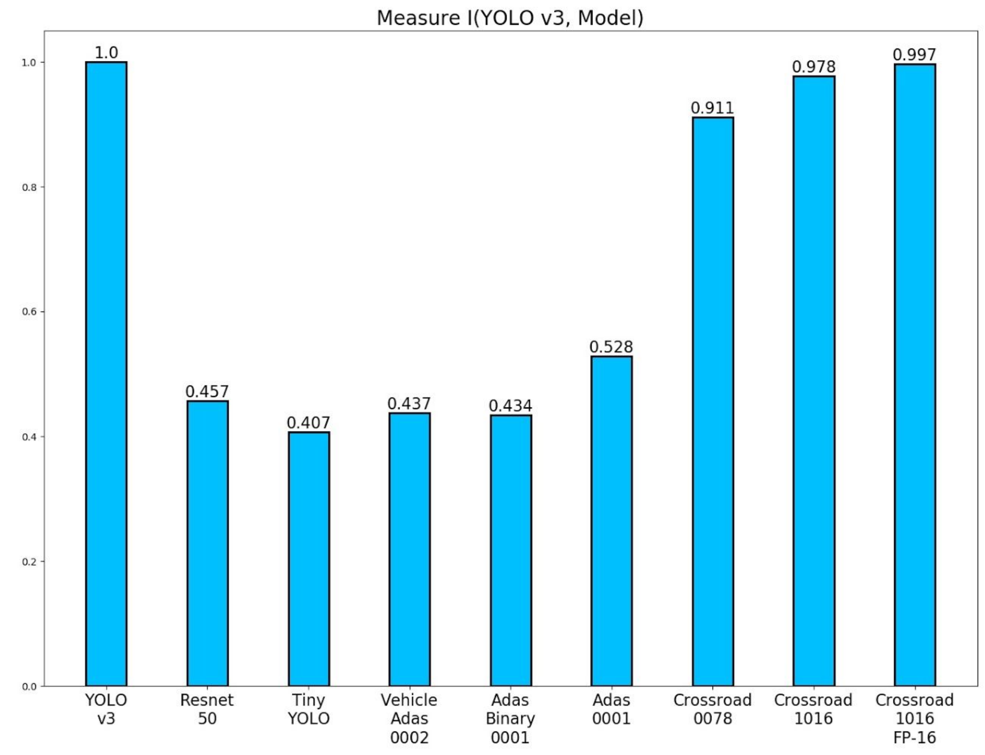


Fig.13 Bar plot of the performance of every network for D_3

Furthermore, the ratios' bar plots (Fig.14) highlight the outstanding results of the Crossroad models when it comes to the tradeoff performance/FLOPs with the best ratios by far, especially for Crossroad FP-16.

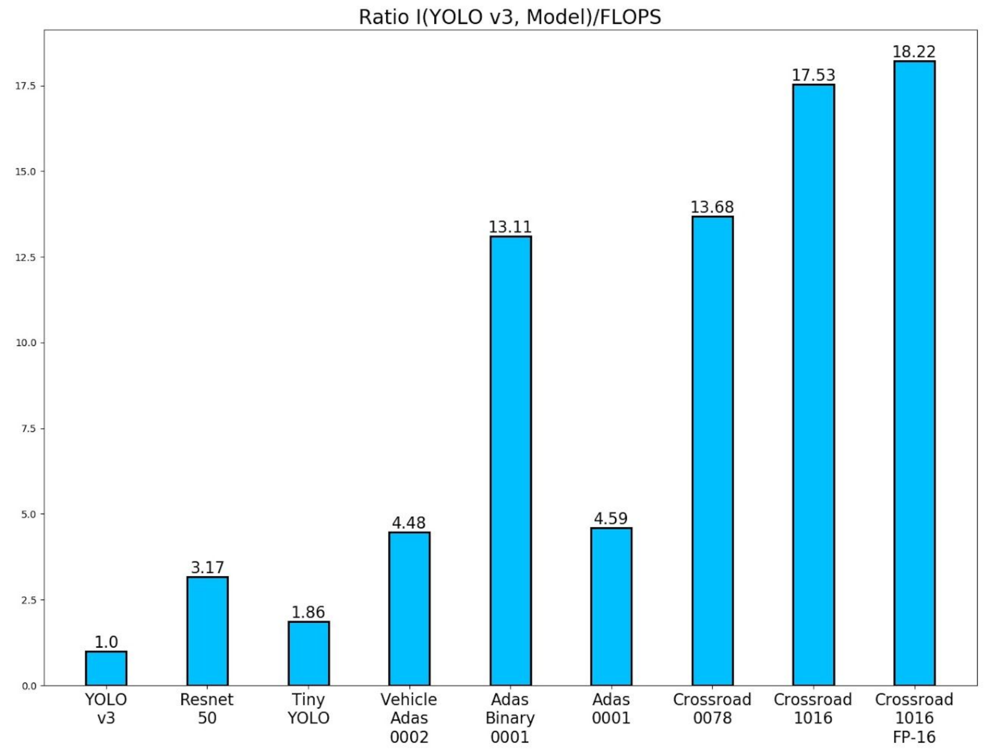


Fig.14 Bar plot of ratio measure/flops of every network for D_3

D. Conclusions of the research

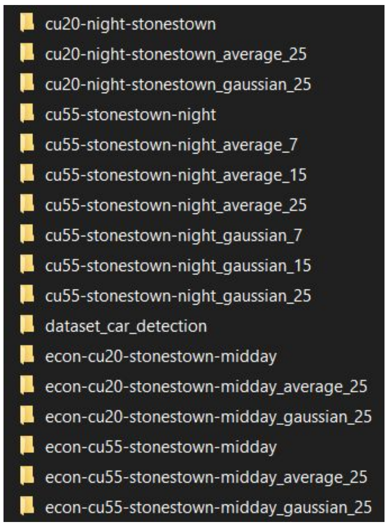
All in all, there are many networks to test on the long term but with the protocol I built, you'll be able to test them by changing the parameters *comparison threshold* and *averaging threshold* to avoid the false positives and identify well when a car is identified by a network and/or another.
Also, throughout my research on the net, we have a good database to test open source networks or maybe Quividi's future networks. For now, among the 9 networks I scrutinized, we have lots of details to exploit later. Indeed, there are two main evaluation metrics: INDEX (performance measure) and RATIO (metric to assess the tradeoff INDEX/FLOPs) that can allow us to compare models with YOLO v3 and ensure they're worth using.
Finally, thanks to the research made, we know that we need to pay attention to the image quality and climate effects on the performance of a network. Clearly, Crossroad 1016 models are quite satisfying when it comes to performance and Adas Binary 0001 is the best when it comes to the tradeoff.

Notes for Enrico:

I put all the folder in : “C:/frames”

Then, you can find data-car-detection (bright images) and rainy_frames, rainy_frames_1 and rainy_frames_2 (rainy images)

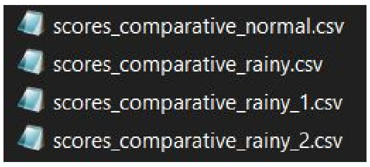
Also, all these folders:



cu20-night-stonestown
cu20-night-stonestown_average_25
cu20-night-stonestown_gaussian_25
cu55-stonestown-night
cu55-stonestown-night_average_7
cu55-stonestown-night_average_15
cu55-stonestown-night_average_25
cu55-stonestown-night_gaussian_7
cu55-stonestown-night_gaussian_15
cu55-stonestown-night_gaussian_25
dataset_car_detection
econ-cu20-stonestown-midday
econ-cu20-stonestown-midday_average_25
econ-cu20-stonestown-midday_gaussian_25
econ-cu55-stonestown-midday
econ-cu55-stonestown-midday_average_25
econ-cu55-stonestown-midday_gaussian_25

Except dataset_car_detection, these are the folders containing frames of the Volta videos normal version, gaussian blurred and average filter blurred.

When the indexes are measured, these are the files of indexes for each dataset:



scores_comparative_normal.csv
scores_comparative_rainy.csv
scores_comparative_rainy_1.csv
scores_comparative_rainy_2.csv

All the files in “D:/Documents/Projects/Detect Cars” and “C:/frames” with tab in the beginning of their names are tabs generated with all the samples and detections