Performance Benchmarking of YOLO Architectures for Vehicle License Plate Detection from Real-Time Videos Captured by a Mobile Robot

Amir Ismail, Mehri Maroua, Anis Sahbani, Najoua Essoukri Ben Amara

Performance Benchmarking of YOLO Architectures for Vehicle License Plate Detection from Real-Time Videos Captured by a Mobile Robot

Amir Ismail[1,3], Maroua Mehri[1], Anis Sahbani[2,3] and Najoua Essoukri Ben Amara[1]

[1] *Université de Sousse, Ecole Nationale d'Ingénieurs de Sousse, LATIS-Laboratory of Advanced Technology and Intelligent Systems, 4023, Sousse, Tunisie;*

[2] *Sorbonne Université, CNRS, Institute for Intelligent Systems and Robotics (ISIR), Paris, France;*

[3] *Enova Robotics, Novation City, Technopôle de Sousse, 4000, Sousse, Tunisie;*

{*amir.ismail, maroua.mehri*}*@eniso.u-sousse.tn, anis.sahbani@enovarobotics.com, najoua.benamara@eniso.rnu.tn*

Keywords: Vehicle license plate, detection, real-time, video, mobile robot, deep neural networks, YOLO architectures.

Abstract: In this paper, we address the issue of vehicle license plate (LP) detection for a mobile robotic application. Specifically, we tackle the dynamic scenario of a robot in the physical world interacting based on its cameras. The robot is dedicated essentially to patrol and secure unconstrained environments. Counter to the most recent works of LP detection which assume controlled deploying scenario, the mobile platform requires a more robust system that is suitable for various complex scenarios. To contribute to this purpose, we propose an end-to-end detection module capable of localizing LP either in images or in live-streaming videos. The proposed system is based on deep learning based detectors, particularly the most recent *YOLOv4-tiny* one. To evaluate the proposed system, we introduce the first-ever public Tunisian dataset, called *PGTLP*, for LP detection that contains $3,000$ annotated images. This dataset was gathered using the security robot during its patrolling and surveillance of parking stations and high-risk areas. For the detection, a comparative study for the different YOLO variants has been carried out in order to select the best detector. Our experiments are performed on the *PGTLP* images and following the same experimental protocol. Among the selected models, *YOLOv4-tiny* reveals the best compromise between detection performance and complexity. Further experiments that have been conducted using the *AOLP* benchmark dataset point out that the proposed system has satisfying results.

1 INTRODUCTION

A long-standing target in the field of license plate detection and recognition (LPDR) is to develop systems that can perceive and understand a rich and huge variety of configurations of license plates (LP). Significant strides have been made towards this goal over the last few years benefiting from the parallel progress in computing, data availability and particularly deep learning architectures. Meanwhile, conventional computer vision techniques become more and more limited, and no longer reliable in terms of their ability to follow the requirements of real-time scenario applications. Therefore, most of the state-of-the-art works in LPDR have been focusing on exploring what is happening in a very fast growing field, called representation learning, more commonly known as deep learning. Hence, the progress has been rapid in this context, especially when the algorithms that were designed for visual recognition got really useful for LPDR.

Given this, the identification of vehicles through their LP has been empowering many aspects of the modern civilization from intelligent transportation (e.g. traffic flow monitoring, congestion estimation and management) to surveillance systems (e.g. vehicle identification and tracking, police interference) to car park management (e.g. automatic number plate recognition ticking, data insights, vehicle access). Evidently, these extremely delicate applications require first and foremost a tricky trade-off between precision and operating time. Furthermore, a robust solution is highly required in this context considering the large number of challenges. First up, the LP template has been considered as the foreground issue since it varies not only domestically, but as well across countries. This includes character component specifications from size to color to texture along with its background. Also, it is pointed out that the single- and double-lined LP keeps to be a major bottleneck situation. Second, when it comes to deployment, LPDR is definitely an outdoor scenario. Hence it requires

to be situation-independent in terms of camera angle, camera motion and vehicle speed.

As regards to the above-mentioned requirements, countless attempts keep maintaining a tremendous effort to get over the gap and deliver a scalable and robust LPDR system. Although the fascinating up-to-date outcomes, the existing solutions can operate only with respect to a predefined ground which is not always the case. We can mention for example the single-template LP, the mono-language LP, the camera placing (e.g. parking gate, highway, urban road). Few recent contributions have suggested deep learning-based frameworks to completely remove some of these dependencies (Henry et al., 2020) (Kessentini et al., 2019). However, their methods are either designed to operate on one LP per frame or are tested on a single-camera scenario.

The context of our work covers the partnership with our industrial collaborator named "Enova Robotics", a company whose business is in manufacturing that manufactures mobile robots. It responds to our partner's need in order to satisfy his multi-camera scenarios. In fact, "Pearl Guard"[1], "Enova Robotics"'s main product, is autonomously driving security mobile platform. It patrols high-security areas such as airport runways and parking lots. It is with respect to these conditions that we will introduce a novel and realistic licence plate detection (LPD) system to be implemented on the robot.

The main outcomes of this work are summarized as listed below:

1. We present a comparative study of the state-of-the-art object detection models, namely YOLO and its derivatives in the context of LP detection since many recent works rely on the YOLO extensions.
2. We propose an end-to-end Tunisian LP detection module that works on multi-templates without the need of any pre-processing functionalities. The infrastructure of the module is based on the state-of-the-art *YOLOv4-tiny* object detection model. The module remains robust to all the challenges such as multi-LP context, various templates and unusual backgrounds.
3. To the best of our knowledge, we present the first publicly-accessible Tunisian vehicle LP dataset so-called the *PGTLP* dataset containing up to $3,000$ multi-norm LP annotated images. This dataset, collected using the security robot, covers the major difficulty levels of LP detection in unconstrained environments.
4. We evaluate the proposed module on the *AOLP* benchmark dataset (Hsu et al., 2012). The achieved module is capable of running on a multi-terrain robot with real-time performance.

The remainder of this paper is structured as follows. Section 2 reviews the main recent deep architectures proposed in the literature for LP detection. Section 3 presents brief descriptions of the different YOLO architectures evaluated in this paper. In Section 4, we detail the experiments carried out to compare the different YOLO architectures by outlining the experimental corpus, the experimental protocol, and the different computed performance evaluation metrics. Section 5 analyzes the obtained qualitative and quantitative results along with the computational cost. Finally, our conclusions and further work are given in Section 6.

2 RELATED WORK

The existing LPDR related contributions can be seen as two main well-separated branches. Firstly, we detect the LP, and then we recognize its components in order to identify the vehicle. Since the recognition module is out of the scope of this work, we will only dig in from the detection part direction and we will go mostly through deep learning-based contributions.

There is a large number of related works and inspiration in the field of LP detection. In a first time, a lot of focus is on traditional computer vision techniques. Hsu et al. (Hsu et al., 2012) used the edge clustering approach to detect Taiwan LP. Kteta et al. (Ktata et al., 2015) introduced a so-called extraction module composed of horizontal and vertical edge processing together with conventional pre-processing functions such as dilatation and filtering to detect Tunisian LP.

Since the jump up of deep models, old-fashion computer vision methods are no longer reliable in terms of pattern recognition compared to the new competitive deep learning-based approaches. For instance, Li and Shen (Li and Shen, 2016) extracted candidate LP using a two 4-layer convolutional neural networks (CNN). The first CNN was used in a sliding window fashion to find the LP regions, while the second one was applied to classify them into plate/non-plate regions. Selmi et al. (Selmi et al., 2017) put forward a complex detection system that started with few pre-processing steps and ended with a CNN classifier to distinguish LP regions from non-LP ones. Bulan et al. (Bulan et al., 2017) localized the American LP using a CNN inspired by the AlexNet architecture (Krizhevsky et al., 2012) and a linear support vector machine (SVM) (Cortes and Vapnik, 1995) on top of

[1] https://enovarobotics.eu/pguard/

the extracted features. Silva et al. (Silva and Jung, 2017) proposed an end-to-end fast you only look once (YOLO) based network named FV/LPD-NET to perform LP detection from car frontal-views. Hsu et al. (Hsu et al., 2017) customized two versions of one-stage detector which are YOLO (Redmon et al., 2016) and *YOLOv2* (Redmon and Farhadi, 2017) to be able to handle LP detection under, as they have called it, in-the-wild conditions. Rafique et al. (Rafique et al., 2018) came with the idea of applying region convolutional neural network (RCNN) (Girshick et al., 2014) and its derivatives such as fast RCNN (Girshick, 2015) and faster RCNN (Ren et al., 2015) using two different CNN configurations, such as ZF (Zeiler and Fergus, 2014) and VGG16 (Simonyan and Zisserman, 2014). Xie et al. (Xie et al., 2018) addressed the task of localizing multi-directional LP in Taiwanese cars. They took into consideration the rotation angle of the LP and included it as a fifth parameter of the bounding box into a YOLO-based detector. Li et al. (Li et al., 2018) attempted to localize the LP using a unified deep neural network (DNN). Their model extracted convolutional features, generated proposal bounding boxes followed by integrating and pooling operations to output regions features that were used by fully connected (FC) layers to regress the LP class score and the bounding boxes offsets. Weidog et al. (Min et al., 2019) introduced a framework composed of K-means++ clustering algorithm (Arthur and Vassilvitskii, 2006) and YOLO-L detector. Initially, the clustering algorithm selected candidate boxes and forwarded them to the YOLO-L. The detector was a modified version of the *YOLOv2* (Redmon and Farhadi, 2017) that took care of the detection step. Meng et al. (Meng et al., 2018) designed a CNN model named LocateNet to regress the four vertices of the LP bounding box. Safie et al. (Safie et al., 2019) proposed a detection system for a surveillance camera installed at a fixed position. They combined RetinaNet (Lin et al., 2017) and residual networks (He et al., 2015) to detect the car plaque numbers.

Recently, Kessentini et al. (Kessentini et al., 2019) designed a two-stage DNN in the sake of detecting multi-norm and multilingual LP. The first stage was dedicated to extract LP regions from natural scene images based on *YOLOv2* detector (Redmon and Farhadi, 2017). In their scenario, they considered only one vehicle instance per image. More recently, Selmi et al. (Selmi et al., 2020) put available a deep learning-based setting to find LP in images. It was developed on top of the two-stage mask R-CNN object detector (He et al., 2017). Inspired by the GoogLeNet architecture (Szegedy et al., 2014), they proposed a stylish feature extractor. After the generation of proposals, a Softmax classifier was trained to differentiate LP from non-LP. To tackle the issue of various LP templates, Henry et al. (Henry et al., 2020) presented an end-to-end LPDR system. For LPD, they modified a tiny version of *YOLOv3* (Redmon and Farhadi, 2018) to make their system responds to real-time requirements. After applying pre-processing and enhancement tools, Omar et al. (Omar et al., 2020) used the SegNet (Kendall et al., 2015) architecture to segment three regions of Iraqian LP for further processing. Pustokhina et al. (Pustokhina et al., 2020) proposed a full-path technique for LPDR which is mainly applicable with its first stage to find LP using improved Bernsen algorithm (IBA) (Latha and Chakravarthy, 2012) and connected component analysis (CCA).

3 YOLO ARCHITECTURES

The majority of computer vision applications are based on using YOLO detectors due to its fast inference. Since we prioritize the real-time performance to meet the needs of the mobile robot, our main focus is to investigate the YOLO detector category. From the early *YOLOv2* to the most recent *YOLOv4*, YOLO derivates keep revealing great compromise between accuracy and runtime speed. Given this, we propose to bring to the table six versions of YOLO and explore their performances in the context of LP detection. The ultimate goal of YOLO is to close the gap of runtimes in working implementations. The fundamental idea about YOLO is that they are one-stage detectors and thereby they treat the detection as a straight regression problem. In fact, YOLO detector is made up of three well-independent parts.

- **Backbone**: is the network responsible for features formation. Trained on ImageNet classification, it learns relevant features that will be tweaked in the new task of detection.
- **Neck**: mixes and combines the features which are formed in the CNN backbone in order to capture both spatial and semantic information and feed them to the detection step.
- **Head**: detects multiple-size objects in an anchor-based fashion by using three different scales of the network.

Broadly speaking, YOLO descendants respect the same scheme. Indeed, they access the whole image and split it into an $S \times S$ grid. Instead of predicting arbitrary boxes, they predict offsets to a bunch of pre-selected boxes more known as anchors presented in

Table 1. *YOLOv2* happens to use mutli-scale training by removing fully connected layers which makes it able to accept images of different sizes. Later on, *YOLOv3* comes essentially with feature pyramid network (FPN) approach which allows it to make predictions at three different scales. A modified version of *YOLOv3* called *YOLOv3-SPP* detects objects with different scales with a slightly different strategy just by adding a spatial pyramid pooling (SPP) layer. The SPP block is integrated just after the final features map in order to concatenate mutli-scale local and global features. Just recently, the final version of YOLO termed *YOLOv4* comes out with countless additional fascinating blocks in particular the path aggregation network (PAN) to be used as a way to propagate information from low layers to the top ones. The takeaway from this is the modularity of YOLO in a way small blocks can be arranged and interconnected in various ways so that they jointly process the data.

In this work, we propose a comparative study of the six following YOLO variants which are :

- ***YOLOv2***: was a breakthrough in object detection. It provides a smooth tradeoff between speed and accuracy.

- ***YOLOv3***, ***YOLOv3-SPP***, ***YOLOv3-tiny***: add numerous connections to the backbone layers and makes predictions at three separate levels to be suitable for small object detection. *YOLOv3-SPP* is a robust version of *YOLOv3* which plugs in SPP modules in front of the detection headers. *YOLOv3-tiny* is a reduced version of *YOLOv3*, much faster and less accurate.

- ***YOLOv4***, ***YOLOv4-tiny***: represent the mature versions of YOLO detectors. Many technologies have been integrated into *YOLOv4* making it the state-of-the-art detector with a great compromise between accuracy and processing frame rate. *YOLOv4-tiny* is a tiny version of *YOLOv4* with a compressed backbone layers. It is 8 times faster and about $2/3$ more efficient.

Table 1 reviews the key components and modules in the evaluated variants of YOLO detectors and their performances. In the tables below, the values which are quoted in red and green colors, are considered as the lowest and highest, respectively.

Figure 1: Samples from the *PGTLP* dataset. The resolution of the images in the left column is 1920×1080 pixels, while in the right column is 800×600 pixels.

4 EXPERIMENTS

4.1 Experimental corpus

We propose to train the deep models on our proper dataset. To do so, we take advantage of the mobile robot, called "Pearl Guard", to navigate in different environments while recording vehicles. During the patrolling, the robot "Pearl Guard" supervises parking slots and high-risk areas. We consider two scenarios: when the robot is stopped and the vehicle is moving or both of them are in motion. We have used the cameras of the mobile platform to collect images with different resolutions: 1920×1080, 800×600 and 640×480 pixels (cf. 1). We have tried to cover the most Tunisian LP templates used for vehicles. Also, the proposed dataset considers the multiple LP per frame situation so that images do not have only one plate but also two and three instances. Until now, the dataset is composed of $3,000$ annotated images for LP detection. The training and validation sets contain $2,000$ and 500 images, respectively. The remaining 500 images are for the test phase. All the models in this paper are trained on the training dataset and evaluated on the validation dataset. Images are labeled and annotated with bounding boxes (BB) using *LabelImg* tool[2]. *LabelImg* is a graphical image annotation tool that supports YOLO annotation format. The *PGTLP* dataset is the first and only Tunisian dataset publicly and freely available for the research purpose.

4.2 Experimental protocol

Following the default configurations in Darknet [3], we have trained all the versions using stochastic gradient descent (SGD) algorithm with the momentum of 0.9 except for *YOLOv4* where we have used 0.949 and weight decay of 0.0005. We have set the learning rate initially to be of 0.001 and it is decayed by a factor of 10 at the iteration step of $1,600$ and $1,800$. We have selected a maximum number of training iterations

[2]https://github.com/tzutalin/labelImg

[3]https://github.com/AlexeyAB/darknet

Table 1: The characteristics and modules specific to YOLO detectors.

YOLO	***v2***	***v3***	***v3-tiny***	***v3-SPP***	***v4***	***v4-tiny***
Anchors	5	9	6	9	9	6
Backbone	VGG-16	Darknet-53	Darknet-19	Darknet-53	CSPDarknet-53	CSPNet-15
FPN	✗	✓	✓	✓	✓	✓
SPP	✗	✗	✗	✓	✓	✓
PAN	✗	✗	✗	✗	✓	✓
Head	1	3	2	3	3	2
mAP@.5 (%)	44.0	55.3	**33.1**	60.6	**62.8**	40.2
Speed (FPS)	**40**	66	**345**	38	55	330
Size (MB)	**275**	236	33.7	240	245	**23.1**

equal to 2,000 and we have used a batch size of 64. We have set the size of input image as 416 and 608. Multi-scale training is enabled by randomly rescaling on the fly the sizes of input images. The backbone networks are initialized with the weights pre-trained on ImageNet [4]. All the experiments were conducted on Google Colaboratory servers. Table 2 summarizes the selected hyperparameters for the training of the models.

Table 2: Selected hyperparameters for the training phase.

Algorithm	SGD
Momentum	0.9
Weight decay	0.0005
Learning rate	0.001
Number of iterations	2,000
Batch size	64
Subdivisions	8

Since the LP class does not exist in the ImageNet classes, we have adapted the original architectures to perform only LP detection by limiting the number of classes to only one class. Based on this, we have changed the number of filters in the convolutional layer before the YOLO layer (prediction layer) in the architecture. The number of filters is given by:

$$Nfilters = (NClasses + 1 + coor) * NbAnchors \quad (1)$$

where

- *NClasses*: denotes the total number of objects to detect;
- *NbAnchor*: denotes the number of masks for each layer;
- *coor*: denotes the four predicted offsets from a predetermined set of boxes (anchors).

As we intend to detect only one class corresponding to the LP, so the number of filters is fixed to 18. YOLO detectors are anchor-based models so that instead of directly predicting bounding boxes, they predict offsets from a dataset-specific set of priors. To identify the dimensions of anchors that have the best coverage for the training data, we have run k-means clustering on the dataset then we have injected them into the architecture of each model defining a new set of anchors.

[4]http://www.image-net.org/

5 RESULTS

To analyze the performance of the investigated YOLO architectures and provide an additional insights into their numerical complexity, quantitative and qualitative results and computational cost of each YOLO architecture are firstly presented. Then, based on the obtained results many observations and recommendations about the YOLO architecture having the best trade-off between the best performance and the lowest computational cost are discussed.

5.1 Quantitative and qualitative results

This section is divided into three parts: the quantitative description of the LP detection results, the qualitative results and the comparison with state-of-the-art LP detection models. In this work, precision (P), recall (R), mean average precision (mAP) and intersection over union (*IoU*) metrics are computed.

In Table 3, we present the results of the six evaluated YOLO models in terms of evaluation metrics. These results are obtained on the validation dataset of the *PGTLP* dataset. Two different input sizes (416 and 608) have been used during the training. As seen from Table 3, a higher input size of the architecture clearly increases the inference time while maintaining the other metrics slightly unchangeable. Based on this, we have decided to select 416×416 as input size for all the models during the testing.

Table 4 shows the results of the models evaluated on the test dataset. It is shown that the performance of some models (*YOLOv2*, *YOLOv3*, *YOLOv3-SPP* and *YOLOv3-tiny*) decreases when it comes to

different distribution of data while others (*YOLOv4* and *YOLOv4-tiny*) remain robust. Actually, *YOLOv4* and its compressed version *YOLOv4-tiny* outperform distinguishably all their previous versions. For the precision (P), *YOLOv4-tiny* reveals the best output with 95.23% whereas *YOLOv4* reach out 98.62% for the recall. This means that these two models detect correctly the LP and they do not get confused with similar-to-LP objects. As we mentioned in the metrics section, *mAP* gives a robust view on the performances. Hence, we report also in Table 4 the obtained *mAP* for each model although related works limit their metrics only to the P/R. As expected, *YOLOv4* and *YOLOv4-tiny* land to be in the same *mAP* plateau where the first one outdoes slightly by around 1% in *mAP*.

In order to position our work among other related state-of-the-art ones, it is fundamentally required to evaluate the proposed module on different benchmark datasets. Actually, there are not too many public datasets to work with. The only one publicly available is the application oriented license plate (*AOLP*), thereby we will consider it as our reference dataset. This dataset contains $2,049$ images of Taiwan license plates. Images are categorized into three main subsets: access control (*AC*), traffic low enforcement (*LE*) and road patrolling (*RP*), based on their level of difficulty in particular *RP* is the toughest category in *AOLP*. *AC*, *LE* and *RP* contain respectively 681, 757 and 611 images. Table 5 summarizes the performance evaluation of the proposed module on the *AOLP* dataset. It is worth mentioning that previous works used to do both the training and testing exclusively on the *AOLP* which makes it easier for their modules to reach up high results. In our settings, things were arguably different; we trained the proposed model on our *PGTLP* dataset, we fixed the learned weights and then we passed to test on the entire *AOLP* dataset. This is extremely beneficial to check out the capacity of our model to generalize when it comes to a completely different and unseen dataset. We have noticed also that the *AOLP* image are not fully annotated. This means that numerous images have, for example, two LP instances but only one LP was annotated in the ground-truth. They were consequently a source of serious error since our model was able to detect them while considering them as false positives. Figure 2 highlights some instances of the encountered issue.

Figures 3 and 4 illustrate few result examples of LP detection in *AOLP* and *PGTLP* datasets, respectively using the *YOLOv4-tiny* architecture. By visual inspection of the obtained results on the two datasets, we note that the *YOLOv4-tiny* architecture provides satisfying results.

Figure 2: Annotation issue with the *AOLP* dataset. The left and right columns are the ground-truth annotations and the detected boxes, respectively.

(a) *AC* (b) *LE* (c) *RP*

Figure 3: Result examples of LP detection in *AOLP* dataset.

Figure 4: Result examples of LP detection in *PGTLP* dataset. Images size is 640×480 pixels.

Table 3: Evaluation results of baseline models on the *PGTLP* validation set. *IS*, *IT* and *V* denote the input size, inference time and volume, respectively.

Model	IS	F (%)	IoU (%)	mAP (%)	IT (ms)	V (MB)
V2	416	**81**	**52.35**	**83.71**	14.82	**256**
	608	83	53.65	85.69	21.40	
V3	416	95	77.24	**99.47**	26.76	235
	608	93	76.10	98.98	43.57	
V3-tiny	416	93	73.98	96.32	**4.66**	33
	608	94	75.04	98.04	7.31	
V3-SPP	416	94	77.34	99.36	27.55	239
	608	95	79.98	99.09	44.56	
V4	416	95	75.82	99.36	33.26	244
	608	94	75.60	98.81	**53.49**	
V4-tiny	416	93	69.48	94.25	5.25	**22**
	608	**96**	**81.78**	98.63	8.04	

Table 4: Evaluation results of baseline models on the *PGTLP* test set.

Model	P (%)	R (%)	mAP (%)	Speed (FPS)
V4-tiny	**95.23**	94.21	97.45	90.70
V4	88.83	**98.62**	**98.24**	**26.60**
V3	80.13	95.59	95.88	32.70
V3-SPP	69.66	91.73	89.94	30.50
V3-tiny	88.12	87.87	91.10	**92.60**
V2	**68.93**	**87.23**	**73.48**	69.60

Table 5: Performance evaluation of the proposed module on the *AOLP* dataset.

AC		***LE***		***RP***	
P	**R**	**P**	**R**	**P**	**R**
86.45	80.61	92.99	95.99	75.40	71.58

5.2 Computational cost

Since the highest priority for our application is to be relevant to deal with real-time scenarios and high-speed moving robot, we care a lot about the running time of the models. In real scenarios, more than one vehicle will exist in front of the robot therefore multiple LP will be out there. Table 6 presents the time consumption (IT) of *YOLOv4-tiny* applied to 1920×1080 images. The execution time lightly increases with the increase in the number of LP. The *YOLOv4-tiny* can process an image with three LP in $4.097ms$.

Table 6: Time consumption considering the number of LP per image. Images resolution is 1920×1080 pixels.

Number of vehicles	**Time (ms per image)**
1	3.224
2	3.342
3	4.097

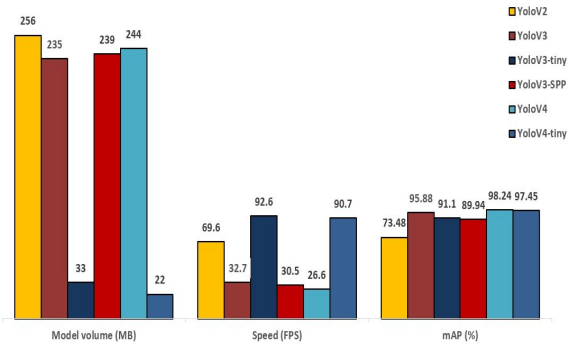

Figure 5: Comparison of baseline models in model volume, speed and *mAP* score when input size is 416×416.

When it comes to videos, even though *YOLOv4* has the best performance in terms of accuracy, it treats video data slowly. On the other hand, the tiny version of *YOLOv3* is the fastest version and capable of running with 96 frames per second (FPS). This goes the same with *YOLOv4-tiny* which runs at $90FPS$. Another important factor to consider is the memory consumption of the model towards deploying and production. Considering the limits imposed by computation capability, a small model is extremely required. As shown in Figure 5, *YOLOv3-tiny* and *YOLOv4-tiny* versions are the smallest with respectively $33MB$ and $22MB$ of volume (V). Hence, they seem to be suitable for on-edge applications such as the security robot.

5.3 Observations and recommendations

To sum things up, the choice of the YOLO model really depends on the type of application to work on. It is highly recommended to find a great compromise between the metrics to identify the suitable detector. For example, if we care, for the most part, about making only correct predictions, then *YOLOv4* is the ultimate choice. In our context, we have two constrains: processing power and memory space. Given this, we opt for the *YOLOv4-tiny* detector. On one hand, it has a great detection accuracy and high running speed. On the other hand, it is very small in terms of storage volume.

6 CONCLUSIONS AND FURTHER WORK

In this paper, we take a step into LP detection. We have presented a one-stage real-time LP detection module in order to empower a mobile security robot. Previous works used to deal with static scenarios which is a narrow view of the problem. However, in our work, we address the dynamic scenarios. To do so, we put available the first Tunisian LP dataset. The *PGTLP* dataset contains up to $3,000$ annotated high resolution images captured by the "Pearl Guard" which is a mobile robot of our industrial collaborator "Enova Robotics". Our dataset covers numerous challenges such as different templates, angles, environment backgrounds making it a major contribution and subject to further experiments and contributions. The main focal point of our work is to explore YOLO detectors in the context of LP detection. Therefore, we have conducted a thorough comparative study of six variants of YOLO models from their theoretical mechanisms to their working implementations. These models have been subject to the same training/testing procedure and evaluated in terms of precision, speed and memory storage. We experimentally demonstrate the effectiveness of the *YOLOv4-tiny* model to detect LP in real-time videos which lands to be small ($22MB$), precise (97.45%) and very fast ($90.70FPS$). In the future, we are willing to develop the recognition module to pull together the LPDR system puzzle. In the same aim, we will enlarge the *PGTLP* to reach up $10,000$ fully annotated images.

ACKNOWLEDGMENTS

This work has been supported by the VRR research fund from the Tunisian Ministry of Higher Education and Scientific Research that is gratefully acknowledged. The authors would like also to thank our industrial partner "Enova Robotics" for providing access to the *PGTLP* dataset.

REFERENCES

Arthur, D. and Vassilvitskii, S. (2006). k-means++: the advantages of careful seeding. Technical report, Stanford.

Bulan, O., Kozitsky, V., Ramesh, P., and Shreve, M. (2017). Segmentation-and annotation-free license plate recognition with deep localization and failure identification. *ITS*, 18(9):2351–2363.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *ML*, 20(3):273–297.

Girshick, R. (2015). Fast R-CNN. In *CVPR*, pages 1440–1448.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, pages 580–587.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask R-CNN. In *ICCV*, pages 2961–2969.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*.

Henry, C., Ahn, S. Y., and Lee, S. (2020). Multinational license plate recognition using generalized character sequence detection. *IEEE Access*, 8:35185–35199.

Hsu, G.-S., Ambikapathi, A., Chung, S.-L., and Su, C.-P. (2017). Robust license plate detection in the wild. In *AVSS*, pages 1–6.

Hsu, G.-S., Chen, J.-C., and Chung, Y.-Z. (2012). Application-oriented license plate recognition. *VT*, 62(2):552–561.

Kendall, A., Badrinarayanan, V., and Cipolla, R. (2015). Bayesian SegNet: model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *arXiv preprint arXiv:1511.02680*.

Kessentini, Y., Besbes, M. D., Ammar, S., and Chabbouh, A. (2019). A two-stage deep neural network for multi-norm license plate detection and recognition. *ESA*, 136:159–170.

Krizhevsky, A., Sutskever, I., and Hinton, G. (2012). 2012 AlexNet. In *NIPS*, pages 1–9.

Ktata, S., Khadhraoui, T., Benzarti, F., and Amiri, H. (2015). Tunisian license plate number recognition. *PCS*, 73:312–319.

Latha, C. and Chakravarthy, G. (2012). An improved Bernsen algorithm approaches for license plate recognition. *IOSR*, 3(4):01–05.

Li, H. and Shen, C. (2016). Reading car license plates using deep convolutional neural networks and LSTMs. *arXiv preprint arXiv:1601.05610*.

Li, H., Wang, P., You, M., and Shen, C. (2018). Reading car license plates using deep neural networks. *IVC*, 72:14–23.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017). Focal loss for dense object detection. In *ICCV*, pages 2980–2988.

Meng, A., Yang, W., Xu, Z., Huang, H., Huang, L., and Ying, C. (2018). A robust and efficient method for license plate recognition. In *ICPR*, pages 1713–1718.

Min, W., Li, X., Wang, Q., Zeng, Q., and Liao, Y. (2019). New approach to vehicle license plate location based on new model YOLO-L and plate pre-identification. *IP*, 13(7):1041–1049.

Omar, N., Sengur, A., and Al-Ali, S. G. S. (2020). Cascaded deep learning-based efficient approach for license plate detection and recognition. *ESA*, 149:113280.

Pustokhina, I. V., Pustokhin, D. A., Rodrigues, J. J., Gupta, D., Khanna, A., Shankar, K., Seo, C., and Joshi, G. P. (2020). Automatic vehicle license plate recognition using optimal K-Means with convolutional neural network for intelligent transportation systems. *IEEE Access*.

Rafique, M. A., Pedrycz, W., and Jeon, M. (2018). Vehicle license plate detection using region-based convolutional neural networks. *SC*, 22(19):6429–6440.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: unified, real-time object detection. In *CVPR*, pages 779–788.

Redmon, J. and Farhadi, A. (2017). Yolo9000: better, faster, stronger. In *CVPR*, pages 7263–7271.

Redmon, J. and Farhadi, A. (2018). Yolov3: an incremental improvement. *arXiv preprint arXiv:1804.02767*.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster R-CNN: towards real-time object detection with region proposal networks. In *NIPS*, pages 91–99.

Safie, S., Azmi, N. M. A. N., Yusof, R., Yunus, M. R. M., Sayuti, M. F. Z. C., and Fai, K. K. (2019). Object localization and detection for real-time automatic license plate detection (ALPR) system using RetinaNet algorithm. In *SAI*, pages 760–768.

Selmi, Z., Halima, M. B., and Alimi, A. M. (2017). Deep learning system for automatic license plate detection and recognition. In *ICDAR*, volume 1, pages 1132–1138.

Selmi, Z., Halima, M. B., Pal, U., and Alimi, M. A. (2020). DELP-DAR system for license plate detection and recognition. *PRL*, 129:213–223.

Silva, S. M. and Jung, C. R. (2017). Real-time Brazilian license plate detection and recognition using deep convolutional neural networks. In *CGPI*, pages 55–62.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., et al. (2014). Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*.

Xie, L., Ahmad, T., Jin, L., Liu, Y., and Zhang, S. (2018). A new cnn-based method for multi-directional car license plate detection. *ITS*, 19(2):507–517.

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *ECCV*, pages 818–833.