Aerial Images Processing for Car Detection using Convolutional Neural Networks: Comparison between Faster R-CNN and YoloV3

Adel Ammar*, Anis Koubaa * § ¶, Mohanned Ahmed *, Abdulrahman Saad *
* Prince Sultan University, Saudi Arabia.
§ CISTER Research Centre, ISEP, Polytechnic Institute of Porto, Porto, Portugal
¶ Gaitech Robotics, China

akoubaa@psu.edu.sa

***Abstract*—In this paper, we address the problem of car detection from aerial images using Convolutional Neural Networks (CNN). This problem presents additional challenges as compared to car (or any object) detection from ground images because features of vehicles from aerial images are more difficult to discern. To investigate this issue, we assess the performance of two state-of-the-art CNN algorithms, namely Faster R-CNN, which is the most popular region-based algorithm, and YOLOv3, which is known to be the fastest detection algorithm. We analyze two datasets with different characteristics to check the impact of various factors, such as UAV's altitude, camera resolution, and object size. The objective of this work is to conduct a robust comparison between these two cutting-edge algorithms. By using a variety of metrics, we show that none of the two algorithms outperforms the other in all cases.**

***Index Terms*—**car detection; convolutional neural networks; deep learning; you only look once (yolo); faster r-cnn; unmanned aerial vehicles

I. INTRODUCTION

Unmanned aerial vehicles (UAVs) are nowadays a key enabling technology for a large number of applications such as surveillance [1], tracking [2], disaster management [3], smart parking [4], Intelligent Transport Systems, to name a few. Thanks to their versatility, UAVs offer unique capabilities to collect visual data using high-resolution cameras from different locations, angles, and altitudes. These capabilities
allow providing rich datasets of images that can be analyzed to extract useful information that serves the purpose of the underlying applications. UAVs present several advantages in the context of aerial imagery collection, including a large field of view, high spatial resolution, flexibility, and high mobility.
Although satellite imagery also provides a bird-eye view of the earth, UAV-based aerial imagery presents several advantages as compared to satellite imagery. In fact, UAV imagery has a much lower cost and provides more updated views (several satellite maps are several months old and do not present recent changes). Besides, it can be used for real-time image/video stream analysis in a much more affordable means.

This work is supported by the Robotics and Internet-of-Things Lab at Prince Sultan University.

With the current hype of artificial intelligence and deep learning, there has been an increasing trend since 2012 (the birth of AlexNet) to use Convolutional Neural Networks (CNNs) to extract information from images and video streams. While CNNs have been proven to be the best approach for classification, detection and semantic segmentation of images, the processing, and analysis of aerial images is generally more challenging than the classical types of images (ground-level images). In fact, given that UAVs can fly at high altitudes, the feature extraction from images and detection becomes more difficult to discern. This fact is due to the small size of features and also the angle of views.

Recently, there have been several research works that addressed the problem of car detection from aerial images. In our previous work [1], we also compared between YOLOv3 and Faster R-CNN in detecting cars from aerial images. However, we only used one small dataset from low altitude UAV images collected at the premises of Prince Sultan University. However, the altitude at which the image is taken plays an essential role in the accuracy of the detection. Besides, we did not profoundly analyze advanced and essential performance metrics such as Intersection over Union (IoU) and the Mean Average Precision (mAP). In this paper, we address the gap, and we consider multiple datasets with different configurations. Our objective is to present a more comprehensive analysis of the comparison between these two state-of-the-art approaches.

In [5], the authors mentioned the challenges faced with aerial images for car detection, namely the problem of having small objects and complex backgrounds. They addressed the problem with the proposed of Multi-task Cost-sensitive-Convolutional Neural Network based on Faster R-CNN. Some other researchers addressed the problem applying deep learning techniques on aerial images, in different contexts such as object detection and classification [6], [7], semantic segmentation [8]–[10], generative adversarial networks (GANs) [11].

In this paper, we propose a comprehensive comparative study between two state-of-the-art deep learning algorithms, namely Faster R-CNN [12] and YoloV3 [13] for car detection from aerial images. The contributions of this paper

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2019 doi:10.20944/preprints201910.0195.v1

are manifold. First, we consider two different datasets of aerial images for the car detection problem with different characteristics to investigate the impact of datasets properties on the performance of the algorithms. In addition, we provide a thorough comparison between the two most sophisticated categories of CNN approaches for object detection, Faster RCCN, which is a region-based approach proposed in 2017, and YOLOv3, which is the latest version of the You-Look-Only-Once approach proposed by Joseph Redmon in 2018.

The remaining of the paper is organized as follows. Section II discusses the related works that dealt with car detection and aerial image analysis using CNN, and some comparative studies applied to other object detection. Section III sets forth the theoretical background of the two algorithms. Section IV describes the datasets and the obtained results. Finally, section V draws the main conclusions of this study.

II. RELATED WORKS

Various techniques have been proposed in the literature to solve the problem of car detection in aerial images and similar related issues. The main challenge being the small size and the large number of objects to detect in aerial views, which may lead to information loss when performing convolution operations, as well as the difficulty to discern features because of the angle of view. In this scope, Chen et al. [14] applied a technique based on a hybrid deep convolutional neural network (HDNN) and a sliding window search to solve the vehicle detection problem. The maps of particular layers of the CNN are split into blocks of variable field sizes, to be able to extract features of various scales. They obtained an improved detection rate compared to the traditional deep architectures at that time, but with the expense of high execution time (7s per image, using a GPU).

Following a different approach, Ammour et al. [15] used a pre-trained CNN coupled with a linear support vector machine (SVM) classifier to detect and count cars in high-resolution UAV images of urban areas. First, the input image is segmented into candidate regions using the mean-shift algorithm. Then, the VGG16 [16] CNN model is applied to windows that are extracted around each candidate region to generate descriptive features. Finally, these features are classified using a linear SVM binary model. This technique achieved state-of-the-art performance, but it still falls short of real-time processing, mainly due to the high computational cost of the mean-shift segmentation stage.

Xi et al. [4] also addressed the problem of vehicle detection in aerial images. They proposed a multi-task approach based on the Faster R-CNN algorithm to which they added a cost-sensitive loss. The main idea is to subdivide the object detection task into simpler subtasks with enlarged objects, thus improving the detection of small objects which are frequent in aerial views. Besides, the cost-sensitive loss gives more importance to the objects that are difficult to detect or occluded because of complex background and aims at improving the overall performance. Their method outperformed state-of-the-art techniques on their own specific dataset that was collected from surveillance cameras placed on top of buildings surrounding a parking lot. However, their approach has not been tested on other datasets, nor on UAV images.

In a similar application, Kim et al. [17] compared various implementations of YOLO, SSD, R-CNN, R-FCN and SqueezeDetPerson on the problem of person detection, trained and tested on their own in-house dataset composed of images that were captured by surveillance cameras in retail stores. They found that YOLOv3 (with 416 input size) and SSD (VGG-500) [18] provide the best tradeoff between accuracy and response latency.

Some recent works have addressed the problem of domain adaptation for convolutional neural networks on aerial images. In fact, one of the problem of CNN is that it is very prone to the domain changes. This means if a network is training on objects from a certain domain and achieves a certain accuracy, this accuracy is likely to decrease a lot if the same objects provided as input come from a different domain. To address this issues, in [11], the authors have proposed a technique for domain adaptation based on generative adversarial networks (GANs) to improve the semantic segmentation accuracy of urban environment in aerial images. The authors achieved an increase of accuracy up to 52% of the semantic segmentation when performing domain adaptation from the source domain which is the German Potsdam city to the Vaihingen city. In [19], the authors proposed another technique for domain adaptation based on Active Learning applied to animal detection in wildlife using aerial images. The core idea consists in using Transfer Sampling (TS) criterion to localize animals effectively, which uses Optimal Transport to determine the regions of interest between the source and target domain.

In [20], Hardjono et al. investigated the problem of automatic vehicle counting in CCTV images collected from four datasets with various resolutions. They tested both classical image processing techniques (Back Subtraction, Viola Jones Algorithm, and Gaussian Filters) and deep learning neural networks, namely YOLOv2 [21] and FCRN (fully convolutional regression network) [22]. Their results show that deep learning techniques yield markedly better detection results (in terms of F1 score) when applied on higher resolution datasets.

The closest work to the present study is that of Benjedira et al. [1] who presented a performance evaluation of Faster R-CNN and YOLOv3 algorithms, on a reduced UAV imagery dataset of cars. The present paper is an improvement over this work from several aspects:

1) We use two datasets with different characteristics for training and testing, whereas most previous works described above tested their technique on a single proprietary dataset.

2) We tested various hyperparameter values (three different input sizes for YOLOv3, two different feature extractors for Faster R-CNN, various values of score threshold).

3) We conducted a more detailed comparison of the results, by showing the AP at different values of IoU thresholds, comparing the tradeoff between AP and inference speed, and calculating several new metrics that have been suggested for the COCO dataset [23].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2019 doi:10.20944/preprints201910.0195.v1

III. Theoretical Overview Faster R-CNN and YOLOv3

Object detection is an old fundamental problem in image processing, for which various approaches have been applied. But since 2012, deep learning techniques markedly outperformed classical ones. While many deep learning algorithms have been tested for this purpose in the literature, we chose to focus on two recent cutting-edge neural network architectures, namely Faster R-CNN and YOLOv3, since they were proved to be successful in terms of accuracy and speed in a wide variety of applications.

A. Faster R-CNN

R-CNN, as coined by [24] is a convolutional neural network (CNN) combined with a region-proposal algorithm that hypothesizes object locations. It initially extracts a fixed number of regions (2000), by means of a selective search. Then it merges similar regions together, using a greedy algorithm, to obtain the candidate regions on which the object detection will be applied. Then, the same authors proposed an enhanced algorithm called Fast R-CNN [25] by using a shared convolutional feature map that the CNN generates directly from the input image, and from which the regions of interest (RoI) are extacted. Finally, Ren et al. [12] proposed Faster R-CNN algorithm (Figure 1) that introduced a Region Proposal Network (RPN), which is a dedicated fully convolutional neural network that is trained end-to-end (Figure 2) to predict both object bounding boxes and objectness scores in an almost computationally cost-free manner (around 10ms per image). This important algorithmic change thus replaced the selective search algorithm which was very computationally expensive and represented a bottleneck for previous object detection deep learning systems. As a further optimization, the RPN ultimately shares the convolutional features with the Fast R-CNN detector, after being first independently trained. For training the RPN, Faster R-CNN kept the multi-task loss function already used in Fast R-CNN, which is defined as follows:

$$L(p_i, t_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*)$$

Where:

- p_i is the predicted probability that an anchor i in a mini-batch is an object.
- p_i^* equals 1 if the anchor is positive (having either the highest IoU overlap with a ground-truth box, or an overlap higher than 0.7), and 0 if it is negative (IoU overlap lower than 0.3 with all ground-truth boxes).
- t_i is the vector of coordinates of the predicted bounding box.
- t_i^* is the vector of coordinates of the ground truth box corresponding to a positive anchor.
- L_{cls} is the classification log loss.
- L_{reg} is the regression loss calculated using the robust loss function, already used for Fast R-CNN [25].
- N_{cls} and N_{reg} are normalization factors.

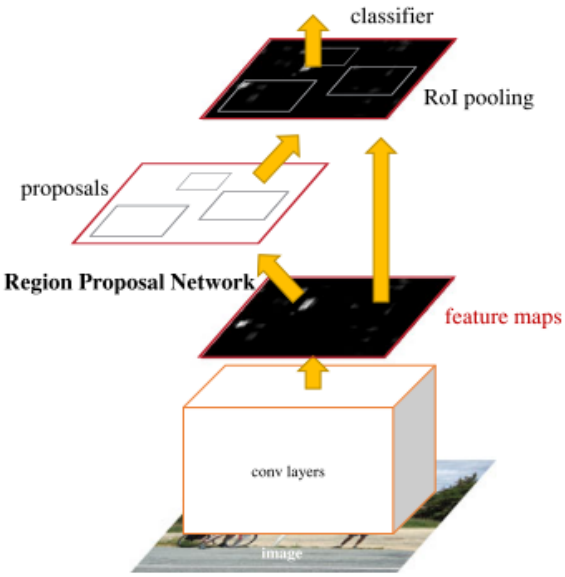

Fig. 1: *Faster R-CNN basic architecture.*

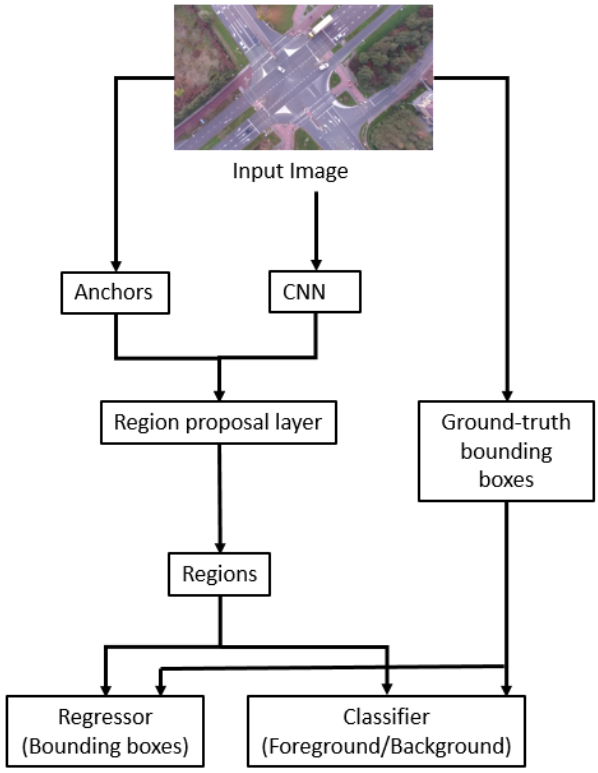

Fig. 2: *Region Proposal Network (RPN) architecture.*

- λ is a blancing weight.

Faster R-CNN uses three scales and three aspect ratios for every sliding position, and is translation invariant. Besides, it conserves the aspect ratio of the original image while resizing it, so that one of its dimension should be 1024 or 600.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2019 doi:10.20944/preprints201910.0195.v1

B. YOLOv3

Contrary to R-CNN variants, YOLO [26], which is an acronym for You Only Look Once, does not extract region proposals, but processes the complete input image only once using a fully convolutional neural network that predicts the bounding boxes and their corresponding class probabilities, based on the global context of the image. The first version was published in 2016 (Figure 3). Later on in 2017, a second version YOLOv2 [21] was proposed, which introduced batch normalization, a retuning phase for the classifier network, and dimension clusters as anchor boxes for predicting bounding boxes. Finally, in 2018, YOLOv3 [13] improved the detection further by adopting several new features:

- Replacing mean squared error by cross-entropy for the loss function. The cross-entropy loss function is calculated as follows:

$$-\sum_{c=1}^{M} \delta_{x \in c} log(p(x \in c))$$

  Where M is the number of classes, c is the class index, x is an observation, $\delta_{x \in c}$ is an indicator function that equals 1 when c is the correct class for the observation x, and $log(p(x \in c))$ is the natural logarithm of the predicted probability that observation x belongs to class c.
- Using logistic regression (instead of the softmax function) for predicting an objectness score for every bounding box.
- Using a significantly larger feature extractor network with 53 convolutional layers (Darknet-53 replacing Darknet-19). It consists mainly of 3*3 and 1*1 filters, with some skip connections inspired from ResNet [27], as illustrated in Figure 4.

Contrary to Faster R-CNN's approach, each ground-truth object in YOLOv3 is assigned only one bounding box prior. These successive variants of YOLO were developed with the objective of obtaining a maximum mAP while keeping the fastest execution which makes it suitable for real-time applications. Special emphasis has been put on execution time so that YOLOv3 is equivalent to state-of-the-art detection algorithms like SSD [18] in terms of accuracy but with the advantage of being three times faster [13]. Figure 5 depicts the main stages of YOLOv3 algorithm when applied to the car detection problem. Variable input sizes are allowed in YOLO. We have tested three input sizes: 320x320, 416x416, and 608x608. Figure 6 shows an example of the output of YOLOv3 on a sample image of the PSU dataset.

To summarize, Table I compares the features and parameters of Faster R-CNN and YOLOv3. While successive optimizations and mutual inspirations made the methodology of the two algorithms relatively close, the main difference remains that Faster R-CNN has two separate phases of region proposals and classification (although now with shared features), whereas YOLO has always combined the classification and bounding-box regression processes.

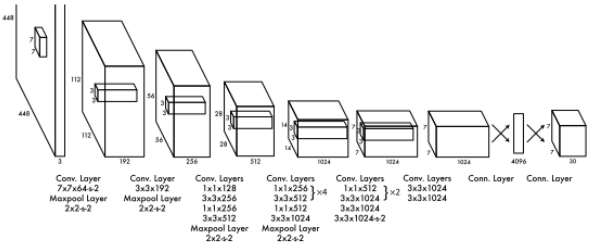

Fig. 3: *YOLO (version 1) network architecture.*

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Fig. 4: *Darknet-53 architecture adopted by YOLOv3 (from [13]).*

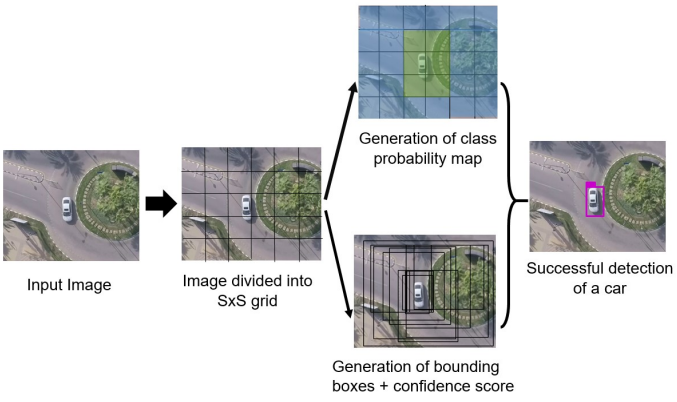

Fig. 5: *Successive stages of the YOLOv3 model applied on car detection.*

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2019 doi:10.20944/preprints201910.0195.v1

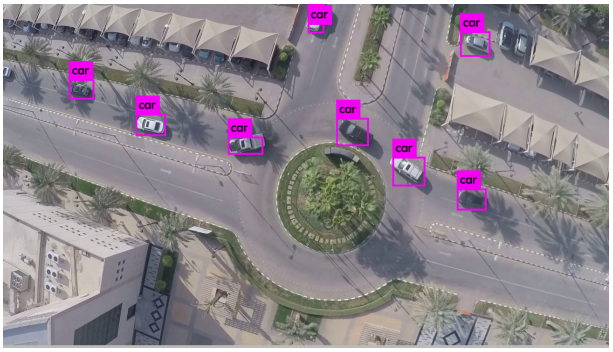

Fig. 6: *Example of the output of YOLOv3 algorithm, on an image of the PSU dataset.*

TABLE I: Theoretical comparison of Faster R-CNN and YOLOv3

	YOLOv3	**Faster R-CNN**
Phases	Concurrent bounding-box regression, and classification.	RPN + Fast R-CNN object detector.
Neural network type	Fully convolutional.	Fully convolutional (RPN and 4 detection network).
Backbone feature extractor	Darknet-53 (53 convolutional layers).	VGG-16 or Zeiler & Fergus (ZF). Other feature extractors can also be incorporated.
Location detection	Anchor-based (dimension clusters).	Anchor-based.
Number of anchors boxes	Only one bounding-box prior for each ground-truth object.	3 scales and 3 aspect ratios, yielding k = 9 anchors at each sliding position.
IoU thresholds	One (at 0.5).	Two (at 0.3 and 0.7).
Loss function	Binary cross-entropy loss.	Multi-task loss: - Log loss for classification. - Smooth L1 for regression.
Input size	Three possible input sizes (320x320, 416x416, and 608x608).	- Conserves the aspect ratio of the original image. - Either the smallest dimension is 600, or the largest dimension is 1024.
Momentum	Default value: 0.9.	Default value: 0.9.
Weight decay	Default value: 0.0005.	Default value: 0.0005.

IV. Experimental comparison between Faster R-CNN and YOLOv3

A. Datasets

In order to obtain a robust comparison, we test Faster R-CNN and YOLOv3 algorithms on two datasets of aerial images showing completely different characteristics.

- **The Stanford dataset** [28] consists of a large-scale collection of aerial images and videos of a university campus containing various agents (cars, buses, bicycles, golf carts, skateboarders and pedestrians). It was obtained using a 3DR solo quadcopter (equipped with a 4k camera) that flew over various crowded campus scenes, at an altitude of around 80 meters. It is originally composed of eight scenes, but since we are exclusively interested in car detection, we chose only one scene (Nexus) that contains the largest percentage of cars (29.51%). All other scenes contain less than 5% of cars. Besides, we have removed images that contain no cars. We noticed that the ground-truth bounding boxes in some images contain some mistakes (bounding boxes containing no objects) and imprecisions (many bounding boxes are much larger than the objects inside them), as can be seen in Figure 7, but we used them as they are in order to assess the impact of annotation errors on detection performance. In fact, the Stanford Drone Dataset was not primarily designed for object detection, but for trajectory forecasting and tracking. Table II shows the number of images and instances in the training and testing datasets. The images in the selected scene have variable sizes, as shown in Table III.

TABLE II: Stanford Dataset

	Training set	Testing set	Total
Number of images	6872	1634	8506
Percentage	80.8%	19.2%	100%
Number of car instances	74,826	8,131	82,957

TABLE III: Image size in the Stanford dataset

Size	Number of images
1409x1916	1634
1331x1962	1558
1330x1947	1557
1411x1980	1494
1311x1980	1490
1334x1982	295
1434x1982	142
1284x1759	138
1425x1973	128
1184x1759	70

- **The PSU dataset** was collected from two sources: an open dataset of aerial images available on Github [29]; and our own images acquired after flying a 3DR SOLO drone equipped with a GoPro Hero 4 camera, in an outdoor environment at PSU parking lot. The drone recorded videos from which frames were extracted and manually labeled. Since we are only interested in a single class, images with no cars have been removed from the dataset. Figure 8 shows a sample image of the PSU

TABLE IV: PSU Dataset

	Training set	Testing set	Total
Number of images	218	52	270
Percentage	80.7%	19.3%	100%
Number of car instances	3,364	738	4,102

TABLE V: Image size in the PSU dataset

Size	Number of images
1920x1080	172
1764x430	26
684x547	21
1284x377	20
1280x720	19
4000x2250	12

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2019 doi:10.20944/preprints201910.0195.v1

TABLE VI: Details of Experiments

#	Algorithm	Feature Extractor	Dataset	Average input size	Number of steps
1	Faster R-CNN	Inception v2	Stanford	816*600	600,000
2	Faster R-CNN	Inception v2	PSU	992*550	600,000
3	Faster R-CNN	Resnet50	Stanford	816*600	600,000
4	Faster R-CNN	Resnet50	PSU	992*550	600,000
5	YOLO v3	Darknet-53	Stanford	416*416	200,000
6	YOLO v3	Darknet-53	Stanford	608*608	200,000
7	YOLO v3	Darknet-53	Stanford	320*320	200,000
8	YOLO v3	Darknet-53	PSU	416*416	10,000
9	YOLO v3	Darknet-53	PSU	608*608	10,000
10	YOLO v3	Darknet-53	PSU	320*320	10,000

dataset, and Table IV shows the number of images and instances in the training and testing datasets. The dataset thus obtained contains images of different sizes, as shown in Table V.

B. Hyperparameters

The main hyperparameter for the YOLO network is the input size, for which we tested three values (320x320, 416x416, and 608x608). On the other hand, the main hyperparameter for Faster R-CNN is the feature extractor. We tested two different feature extractors: Inception-v2 [30] (also called BN-inception in the literature [31]) and Resnet50 [32]. These settings make a total of 5 classifiers that we trained and tested on the two datasets described above, which amounts to 10 experiments that we summarize in Table VI. We used the number of steps necessary to the convergence of each algorithm configuration and kept the default values for the momentum (0.9), weight decay (0.0005), and learning rate (initial rate of 0.001 for YOLOv3, 0.0002 for Faster R-CNN with Inception-v2, and 0.0003 with Resnet50).

C. Results and Discussion

For the experimental setup, we used a workstation powered by an Intel core i7-8700K (3.7 GHz) processor, with 32GB RAM, and an NVIDIA GeForce 1080 (8 GB) GPU, running on Linux.

The following metrics have been used to assess the results:

- IoU: Intersection over Union measuring the overlap between the predicted and the ground-truth bounding boxes.
- mAP: mean average precision, or simply AP, since we are dealing with only one class. It corresponds to the area under the precision vs. recall curve. AP was measured for different values of IoU (0.5, 0.6, 0.7, 0.8 and 0.9).
- FPS: Number of frames per second, measuring the inference processing speed.
- Inference time (in millisecond per image): also measuring the processing speed.
- $AR^{max=1}$, $AR^{max=10}$, $AR^{max=100}$: average recall, when considering a maximum number of detections per image, averaged over all values of IoU specified above.

*1) **Average Precision**:* When analyzing the results, it appears that both YOLOv3 and Faster R-CNN gave a much better AP on PSU dataset than on Stanford dataset (Figure 9). This is mainly due to the small size of objects in the

TABLE VII: Average recall for a given maximum number of detections, averaged over all values of IoU (05, 0.65, 0.8, and 0.9), on the Stanford Dataset

Network	$AR^{max=1}$	$AR^{max=10}$	$AR^{max=100}$
Faster R-CNN (Inception-v2)	15.1%	17.1%	17.1%
Faster R-CNN (Resnet50)	16.4%	**18.6%**	**18.6%**
YOLOv3 (320x320)	9.04%	9.06%	9.06%
YOLOv3 (416x416)	17.13%	17.32%	17.32%
YOLOv3 (608x608)	**17.24%**	17.30%	17.30%

TABLE VIII: Average recall for a given maximum number of detections, averaged over all values of IoU (05, 0.65, 0.8, and 0.9), on the PSU Dataset

Network	$AR^{max=1}$	$AR^{max=10}$	$AR^{max=100}$
Faster R-CNN (Inception-v2)	6.2%	41.5%	70.8%
Faster R-CNN (Resnet50)	**6.4%**	41.5%	67.2%
YOLOv3 (320x320)	6.0%	42.2%	81.0%
YOLOv3 (416x416)	**6.4%**	44.1%	90.4%
YOLOv3 (608x608)	**6.4%**	**44.5%**	**91.9%**

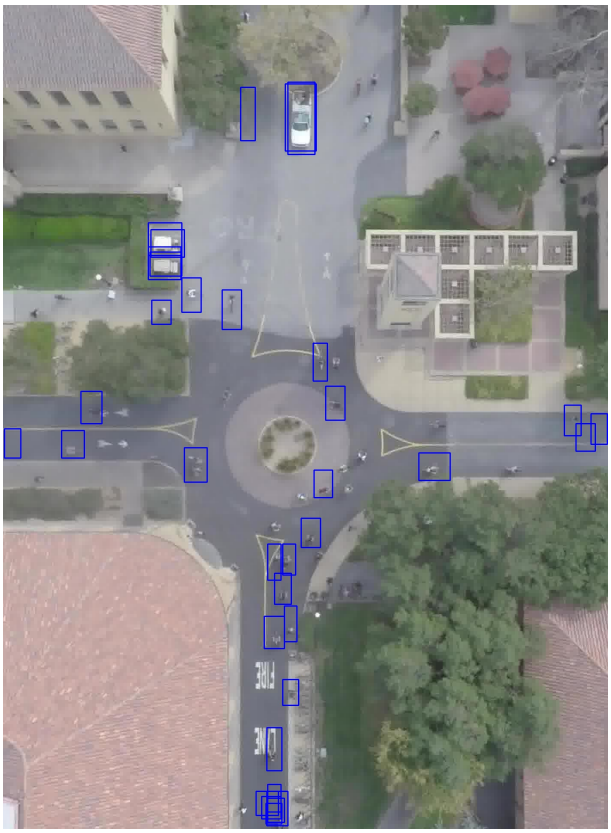

Fig. 7: *A sample image of the Stanford dataset, with ground-truth bounding boxes showing some annotation errors and imprecisions.*

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2019 doi:10.20944/preprints201910.0195.v1

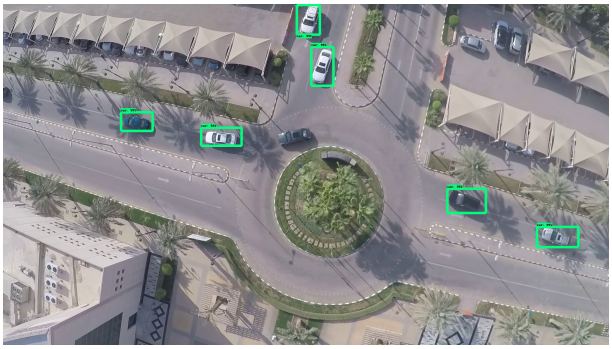

Fig. 8: *Example of the output of Faster R-CNN algorithm, on an image of the PSU dataset.*

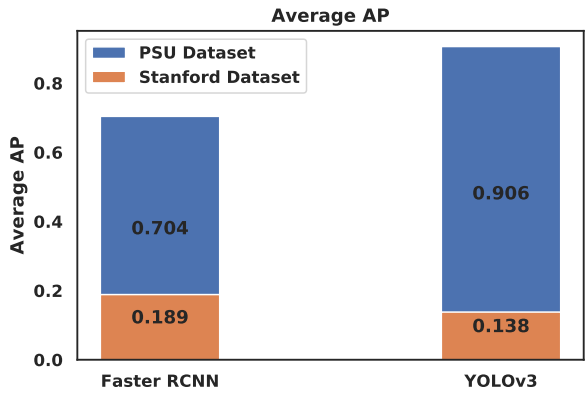

Fig. 9: *Comparison of the average AP between YOLOv3 and Faster R-CNN.*

latter dataset. Figure 10 confirms this observation and shows that both precision and recall are significantly lower on the Stanford dataset. However, Figure 11 shows that the number of false negatives (non-detected cars) is much higher than the number of false positives on the Stanford dataset (3 times higher for Faster R-CNN, and 42 times higher for YOLOv3), and also much higher than the number of true positives, which indicates that most cars go undetected in the Stanford dataset due to their small size. Figure 12 and Figure 13 respectively show examples of YOLOv3 and Faster R-CNN misclassifications on a sample image of the Stanford dataset. The false positives shown may be explained by the presence of errors of annotations in the learning dataset, as mentioned in section IV.A. Figure 14 and Figure 15 show examples of YOLOv3 and Faster R-CNN misclassifications (all of them false negatives) on a sample image of the PSU dataset, respectively.

Figures 16 and 17 show the effect of the score threshold on AP. While this effect is very reduced on Faster R-CNN for both datasets, it shows a high dependency for YOLOv3 only on Stanford dataset, decreasing to almost 0 for a score threshold of 0.9. This reveals the fact that YOLOv3 predictions on Stanford dataset are much less confident. For all other figures shown here, the score threshold has been fixed to 0.5.

*2) **Average Recall**:* Table VII shows the average recall for a given maximum number of detections, on the Stanford Dataset. Faster R-CNN outperforms YOLOv3 in this metric except for $AR^{max=1}$, with a slight better performance for Resnet50 feature extractor over Inception-v2, and a marked inferior performance for YOLOv3 with an input size of 320x320. Whereas the input sizes 416x416 and 608x608 give similar performance, which means that YOLOv3's medium input size is sufficient for Stanford dataset. The fact that the columns $AR^{max=10}$ and $AR^{max=100}$ in this table are identical can be explained by the fact that very few images in the Stanford testing dataset contain more than 10 car instances.

Table VIII shows the same metrics on PSU dataset. While all tested networks yield a close performance in terms of $AR^{max=1}$ and $AR^{max=10}$, YOLOv3 is significantly better in terms of $AR^{max=100}$, with an increasing performance for larger input sizes, which indicates that YOLOv3 is better at detecting a high number of objects in a single image.

*3) **Effect of the dataset characteristics**:* YOLOv3 shows the largest performance discrepancy between the two datasets. While it has a very high recognition on the PSU dataset (up to 0.96 of AP), its performance markedy decreases on the Stanford dataset (Figure 9). This highlights a previously known limitation of the YOLO algorithm when dealing with small objects within the image. This is mainly due to the spatial constraints imposed by the algorithm. On the other hand, Faster R-CNN was designed to better deal with objects of various scales and aspect ratios [12].

Nevertheless, the contrary can be observed in terms of IoU (Figure 18). While the average IoU of Faster R-CNN decreases by half between PSU dataset and Stanford dataset, it decreases only by 4% times for YOLOv3. The imprecision of the ground-truth bounding boxes in the Stanford dataset partly explains the discrepancy between the two datasets in terms of IoU. YOLOv3 however manages to keep relatively precise predicted bounding boxes on both datasets.

Besides, Faster R-CNN shows a high disparity between the two datasets in terms of processing speed (2.7 times faster on Stanford dataset), mainly due to the difference in image input size. In fact, we calculated that the average number of pixels in input test images (after resizing) is 544K for PSU dataset, and 265K for Stanford dataset.

*4) **Effect of the feature extractor**:* The effect of the feature extractor for Faster R-CNN is very limited on the AP, except for a high value of IoU threshold (0.9) on the Stanford dataset, as can be seen in Figure 19 and Figure 20. Nevertheless, in terms of inference speed, the Inception-v2 feature extractor is significantly faster than Resnet50 (Figures 21 and 22), which is consistent with the findings of Bianco et al. [31] who also showed that Inception-v2 (aka BN-inception) is less computationally complex.

*5) **Effect of the input size**:* Figures 21 and 22 show a significant gain in YOLOv3's AP when moving from a 320x320 input size to 416x416, but the performance stagnates when we move further to 608x608, which means that the 416x416 resolution is sufficient to detect the objects of the two

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2019 doi:10.20944/preprints201910.0195.v1

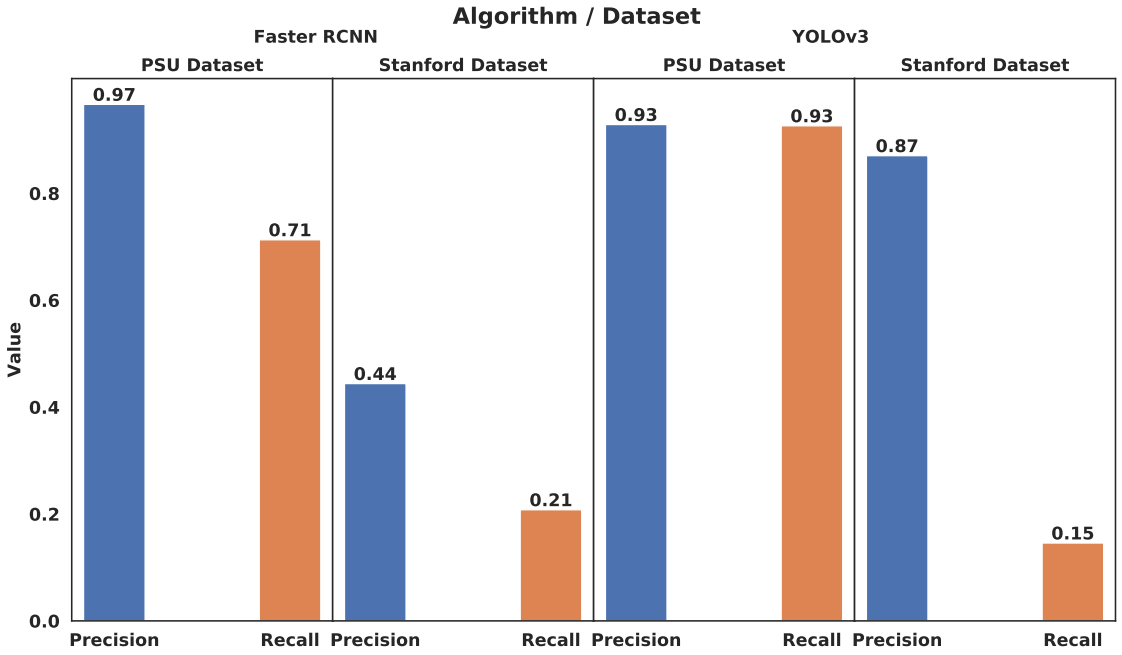

Fig. 10: *Precision and recall values for YOLOv3 and Faster R-CNN, on the two datasets.*

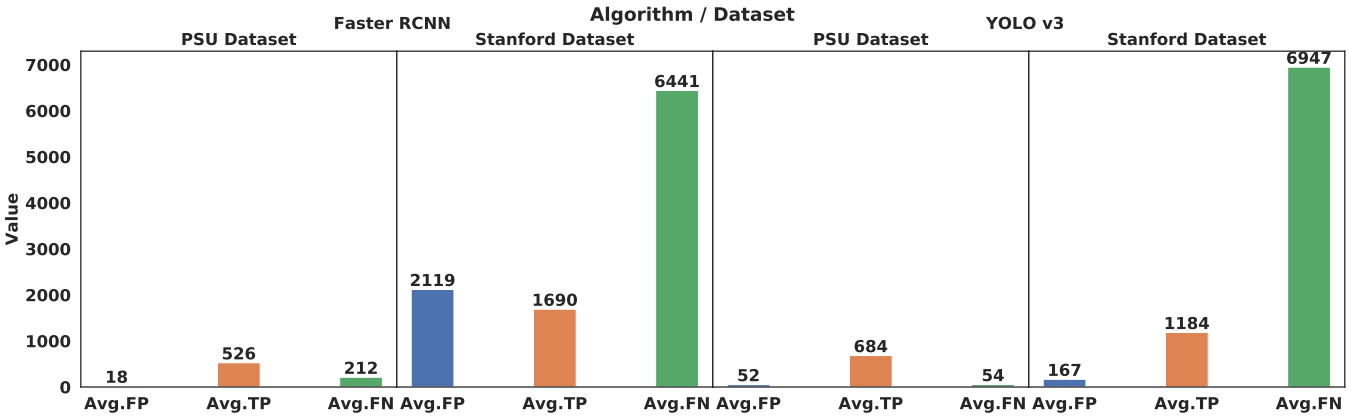

Fig. 11: *Average number of false positives (FP), false negatives (FN), and true positives (TP) for YOLOv3 and Faster R-CNN, on the two datasets.*

datasets. On another hand, the same figures show that the input size has a significant impact on the inference time, as expected, since a larger input size generates a greater number of network parameters, and hence a larger number of operations. In fact, the inference processing speed of YOLOv3 largely depends on the input size (from 12 FPS for 608*608 up to 23 FPS for 320*320), with little variation between the two datasets (Figure 23).

*6) **Main lessons learned**:* Figures 21 and 22 summarize the main results of this comparison study. They compare the trade-off between AP and inference time for YOLOv3 (with 3 different imput sizes) and Faster R-CNN (with two different feature extractors), on the PSU and Stanford datasets respectively. It can be observed that while Faster R-CNN (with Inception v2 as feature extractor) gave the best trade-off in terms of AP and inference speed on the Stanford dataset, YOLOv3 (with input size 320*320) presented the best trade-off on the PSU dataset. This lays emphasis on the fact that none of the two algorithms outperforms the other in all cases, and that the best trade-off between AP and inference time depends on the characteristics of the dataset (object size, resolution, quality of annotation, etc.).

Finally, it should be noted that although the present case study was restricted to only car objects, its conclusions can be easily generalized to any similar types of objects in aerial images, since we did not use any specific feature of cars.

V. Conclusion

In this study, we conducted a thorough experimental comparison of the two leading object detection algorithms

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2019 doi:10.20944/preprints201910.0195.v1

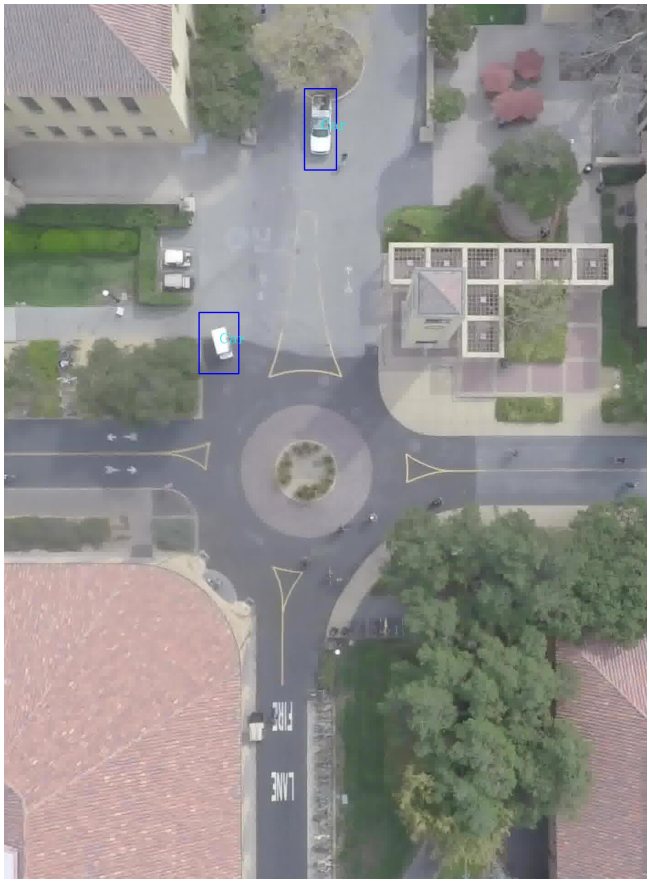

Fig. 12: *Example of YOLOv3's output on an image of the Stanford dataset, showing a true positive.*

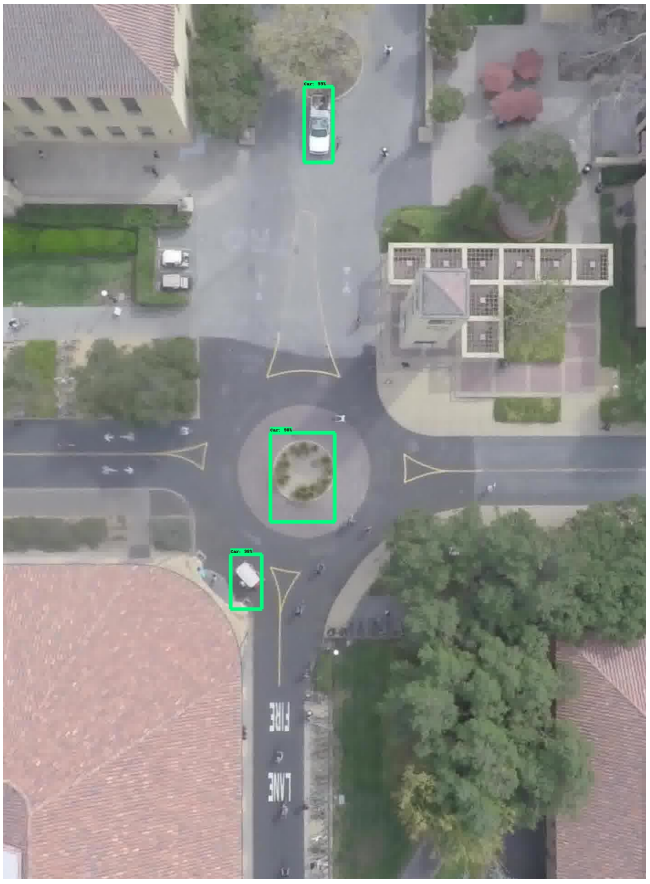

Fig. 13: *Example of Faster R-CNN's output on an image of the Stanford dataset, showing a false positive.*

(YOLOv3 and Faster R-CNN) on two UAV imaging databases that present specific challenges (tiny objects, high number of objects per image, and ambiguous features due to the angle of view) compared to classical datasets of ground images like ImageNet [33] or COCO [23]. The two databases used for performance evaluation present very different characteristics, which makes the comparison more robust. Furthermore, the performance of the two algorithms was assessed using several metrics (mAP, IoU, FPS, $AR^{max=1}$, $AR^{max=10}$, $AR^{max=100}$,...) in order to uncover their strengths and weaknesses. One of the main conclusions that we can draw from this comparative study is that the performance of these two algorithms largely depends on the characteristics of the dataset. In fact, while Faster R-CNN (with Inception v2 as feature extractor) gave the best trade-off in terms of AP and inference speed on the Stanford dataset, YOLOv3 (with input size 320*320) presented the best trade-off on the PSU dataset.

ACKNOWLEDGMENTS

This work is supported by the Robotics and Internet of Things Lab of Prince Sultan University. We also thank Mr. Bilel Ben Jdira and Mr. Taha Khursheed for working on the prior conference version of this paper.

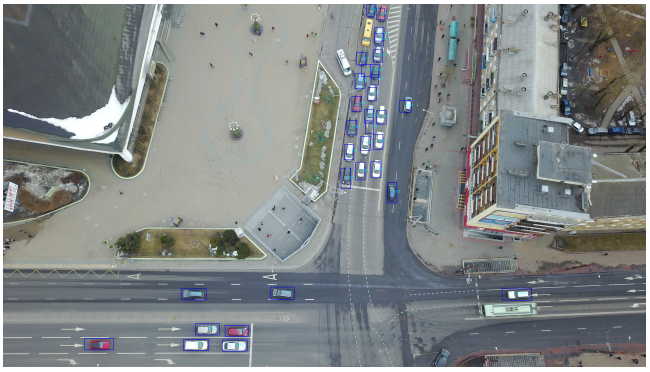

Fig. 14: *Example of YOLOv3's output on an image of the PSU dataset, showing a few false negatives (non detected cars).*

REFERENCES

[1] B. Benjdira, T. Khursheed, A. Koubaa, A. Ammar, and K. Ouni, "Car detection using unmanned aerial vehicles: Comparison between faster r-cnn and yolov3," in *2019 1st International Conference on Unmanned Vehicle Systems-Oman (UVS)*, pp. 1–6, IEEE, 2019.

[2] A. Koubaa and B. Qureshi, "Dronetrack: Cloud-based real-time object tracking using unmanned aerial vehicles over the internet," *IEEE Access*, vol. 6, pp. 13810–13824, 2018.

[3] E. T. Alotaibi, S. S. Alqefari, and A. Koubaa, "Lsar: Multi-uav collaboration for search and rescue missions," *IEEE Access*, vol. 7, pp. 55817–55832, 2019.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2019 doi:10.20944/preprints201910.0195.v1

Fig. 15: *Example of Faster R-CNN misclassifications on an image of the PSU dataset, showing several false negatives (non detected cars).*

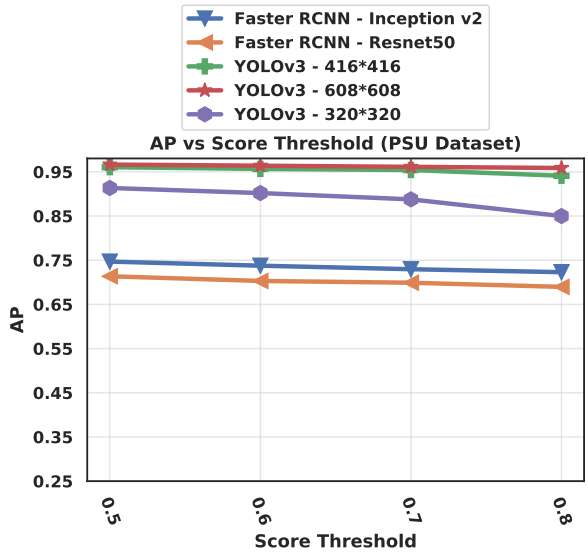

Fig. 16: *AP for different values of score threshold, for the two algorithms on PSU dataset (IoU threshold fixed at 0.6).*

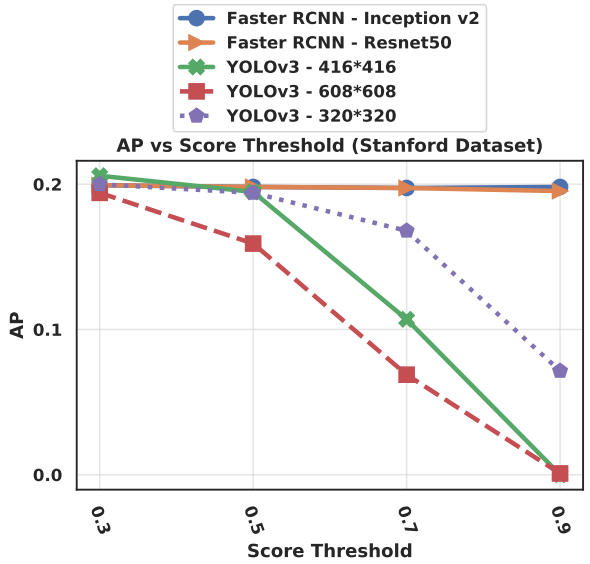

Fig. 17: *AP for different values of score threshold, for the two algorithms on Stanford dataset (IoU threshold fixed at 0.6).*

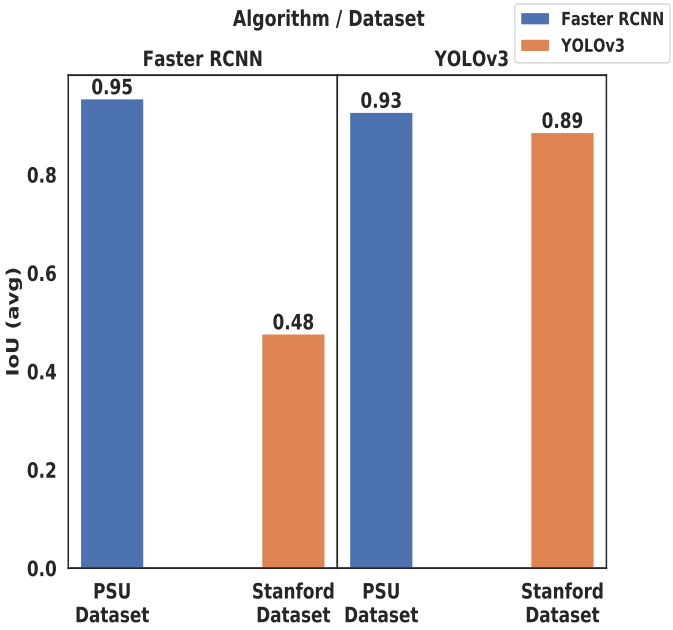

Fig. 18: *Average IoU value for YOLOv3 and Faster R-CNN, on the two datasets.*

[4] X. Xi, Z. Yu, Z. Zhan, C. Tian, and Y. Yin, "Multi-task cost-sensitive-convolutional neural network for car detection," *IEEE Access*, pp. 1–1, 2019.

[5] X. Xi, Z. Yu, Z. Zhan, Y. Yin, and C. Tian, "Multi-task cost-sensitive-convolutional neural network for car detection," *IEEE Access*, vol. 7, pp. 98061–98068, 2019.

[6] I. evo and A. Avramovi, "Convolutional neural network based automatic object detection on aerial images," *IEEE Geoscience and Remote Sensing Letters*, vol. 13, pp. 740–744, May 2016.

[7] K. S. Ochoa and Z. Guo, "A framework for the management of agricultural resources with automated aerial imagery detection," *Computers and Electronics in Agriculture*, vol. 162, pp. 53 – 69, 2019.

[8] M. Kampffmeyer, A. Salberg, and R. Jenssen, "Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images using deep convolutional neural networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 680–688, June 2016.

[9] S. M. Azimi, P. Fischer, M. Krner, and P. Reinartz, "Aerial lanenet: Lane-marking semantic segmentation in aerial imagery using wavelet-enhanced cost-sensitive symmetric fully convolutional neural networks," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, pp. 2920–2938, May 2019.

[10] L. Mou and X. X. Zhu, "Vehicle instance segmentation from aerial image and video using a multitask learning residual fully convolutional network," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, pp. 6699–6711, Nov 2018.

[11] B. Benjdira, Y. Bazi, A. Koubaa, and K. Ouni, "Unsupervised domain adaptation using generative adversarial networks for semantic segmentation of aerial images," *Remote Sensing*, vol. 11, no. 11, 2019.

[12] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with," *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 2017.

[13] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018.

[14] X. Y. Chen, S. M. Xiang, C. L. Liu, and C. H. Pan, "Vehicle Detection in Satellite Images by Hybrid Deep Convolutional Neural Networks," *Ieee Geoscience and Remote Sensing Letters*, 2014.

[15] N. Ammour, H. Alhichri, Y. Bazi, B. Benjdira, N. Alajlan, and M. Zuair,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2019 doi:10.20944/preprints201910.0195.v1

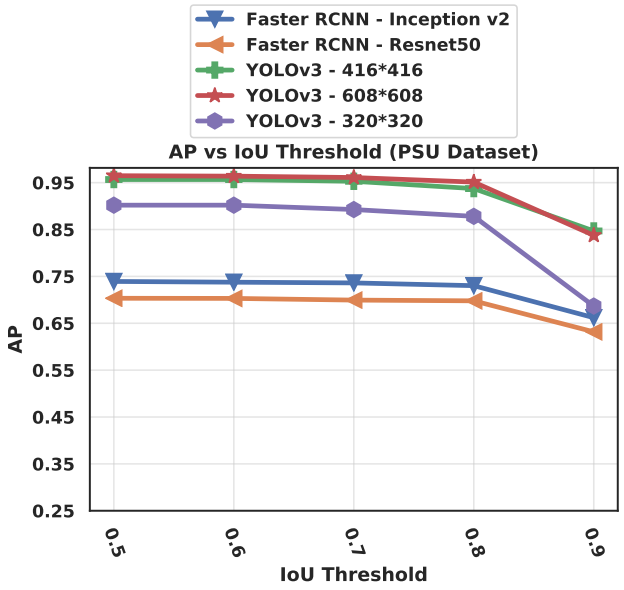

Fig. 19: *Average Precision at different IoU threshold values of the tested algorithms on the PSU dataset.*

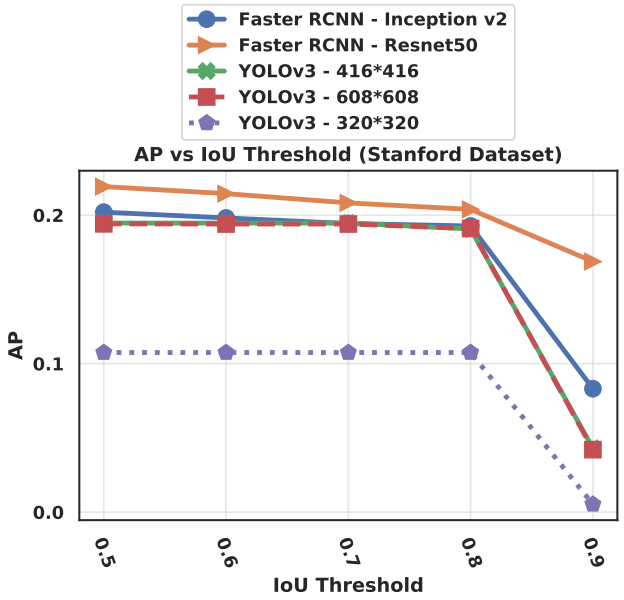

Fig. 20: *Average Precision at different IoU threshold values of the tested algorithms on the Stanford dataset.*

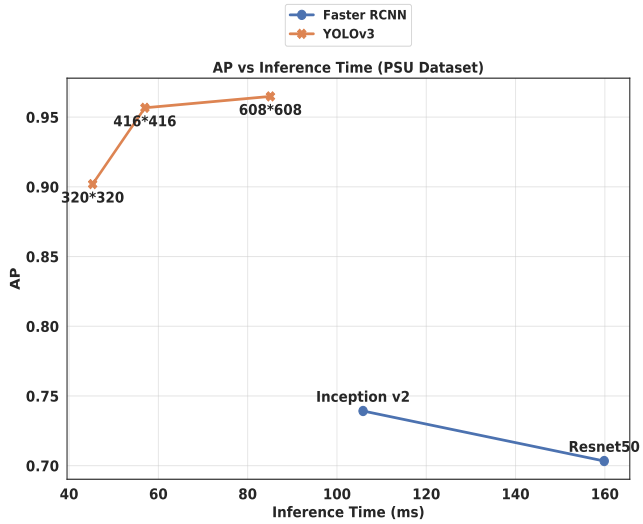

Fig. 21: *Comparison of the trade-off between AP and inference time for YOLOv3 (with 3 different imput sizes) and Faster R-CNN (with two different feature extractors), on the PSU dataset.*

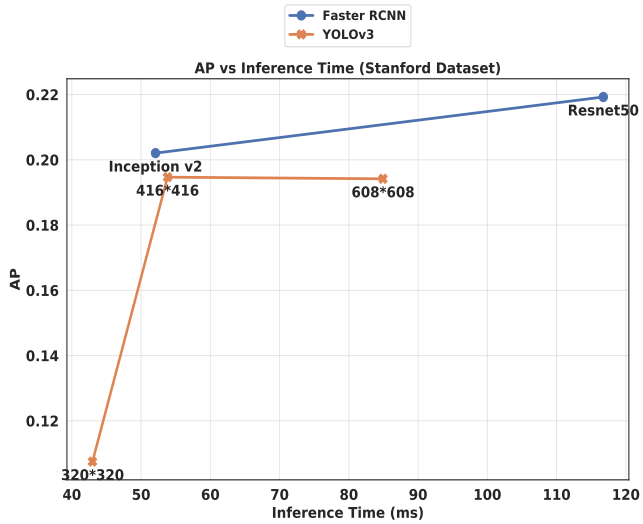

Fig. 22: *Comparison of the trade-off between AP and inference time for YOLOv3 (with 3 different input sizes) and Faster R-CNN (with two different feature extractors), on the Stanford dataset.*

"Deep Learning Approach for Car Detection in UAV Imagery," *Remote Sensing*, vol. 9, p. 312, mar 2017.

[16] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *International Conference on Learning Representations (ICRL)*, 2015.

[17] C. E. Kim, M. M. D. Oghaz, J. Fajtl, V. Argyriou, and P. Remagnino, "A comparison of embedded deep learning methods for person detection," *arXiv preprint arXiv:1812.03451*, 2018.

[18] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016.

[19] Benjamin Kellenberger, Diego Marcos, Sylvain Lobry, Devis Tuia, "Half a percent of labels is enough: Efficient animal detection in uav imagery using deep cnns and active learning," *In press at IEEE Transactions on Geoscience and Remote Sensing (TGRS)*, pp. 1–1, 7 2019.

[20] B. Hardjono, H. Tjahyadi, M. G. A. Rhizma, A. E. Widjaja, R. Kondorura, and A. M. Halim, "Vehicle counting quantitative comparison using background subtraction, viola jones and deep learning methods," in *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pp. 556–562, Nov 2018.

[21] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 6517–6525, 2017.

[22] H. Tayara, K. Gil Soo, and K. T. Chong, "Vehicle detection and counting in high-resolution aerial images using convolutional regression neural network," *IEEE Access*, vol. 6, pp. 2220–2230, 2018.

[23] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, pp. 740–755, Springer, 2014.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2019 doi:10.20944/preprints201910.0195.v1

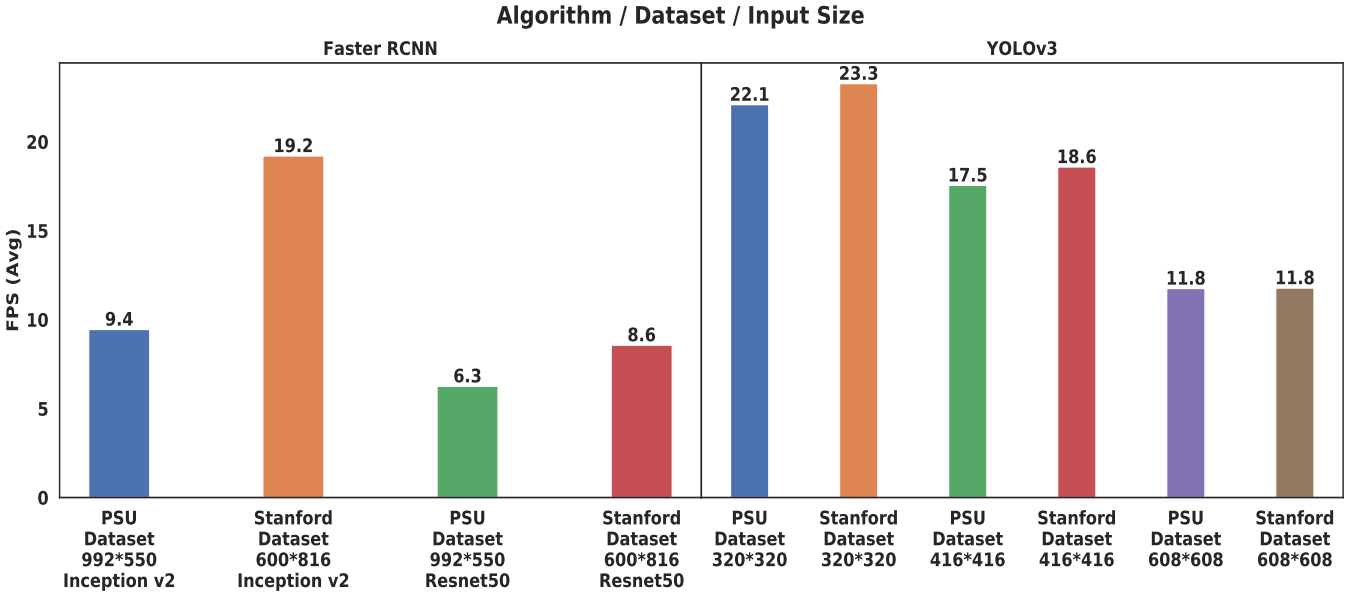

Fig. 23: *Inference speed measured in Frames per Second (FPS), for each of the tested algorithms.*

[24] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 580–587, 2014.

[25] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[26] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 779–788, 2016.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *Arxiv.Org*, 2015.

[28] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese, "Learning social etiquette: Human trajectory understanding in crowded scenes," in *European conference on computer vision*, pp. 549–565, Springer, 2016.

[29] "Aerial-car-dataset, available online on: https://github.com/jekhor/aerial-cars-dataset, accessed on (16-10-2018)."

[30] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[31] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark analysis of representative deep neural network architectures," *IEEE Access*, vol. 6, pp. 64270–64277, 2018.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[33] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.