Article

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

Adel Ammar [1,*], Anis Koubaa [1,2*], Mohanned Ahmed[1], and Abdulrahman Saad [1]

[1] Prince Sultan University, Riyadh, Saudi Arabia
[2] CISTER Research Centre, ISEP, Polytechnic Institute of Porto, Porto, Portugal
* Correspondence: AA: aammar@psu.eu.sa; AK: 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 YOLOv3 yields better performance in most configurations, except that it exhibits a lower recall and less confident detectionswhen object sizes and scales in the testing dataset differ largely from those in the training dataset.

Keywords: Car Detection; Convolutional Neural Networks; Deep Learning; Faster R-CNN; Unmanned Aerial Vehicles; You Only Look Once (Yolo).

1. 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 (many 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. Aerial images have different resolutions as compared to satellite images. For example, in our experiments, we reached a resolution of 2 cm/pixel (and can have even lower) of aerial images using typical DJI drones, whereas satellite images have resolutions of about 15 cm/pixel as for the dataset described in [5], and can be even larger.

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 view.

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 [6], 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 [7,8], semantic segmentation [9–11], generative adversarial networks (GANs) [12].

In this paper, we propose a comprehensive comparative study between two state-of-the-art deep learning algorithms, namely Faster R-CNN [13] and YoloV3 [14] for car detection from aerial images. The contributions of this paper are manyfold. 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 detections. 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.

2. 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. [15] applied a technique based on a hybrid deep convolutional neural network (HDNN) and a sliding window search to solve the vehicle detection problem from Google Earth images. 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. [16] 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 [17] 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 on a reduced testing dataset (5 images containing 127 car instances), 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 private 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. [18] 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) [19] provide the best tradeoff between accuracy and response latency.

Some recent works have addressed the problem of domain adaptation for CNNs on aerial images. In fact, one of the problems of CNN is that it is very prone to the domain changes. This means that if a network is trained on objects from a certain domain, its accuracy is likely to decrease a lot if the same objects provided as input come from a different domain. To address this issues, in [12], 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 Potsdam to Vaihingen domains.

In [20], 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 domains.

In [21], 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 [22] and FCRN (fully convolutional regression network) [23]. Their results show that deep learning techniques yield markedly better detection results (in terms of F1 score) when applied on higher resolution datasets.

Also for the aim of car counting, Mundhenk et al. [5] built their own Cars Overhead with Context (COWC) dataset containing 32,716 unique cars and 58,247 negative targets, standardized to a resolution of 15 cm per pixel, and annotated using single pixel points. The authors used a convolutional neural network that they called ResCeption, based on Inception synthesized with Residual Learning. The model was able to count the number of cars in test patches with a root mean square error of 0.66 at 1.3 FPS.

Other works [24,25] focused on fine-grained car detection, where the objective is to classify cars according to vehicle models and other visual differences, with a classification accuracy that attains 87%. This classification can be used for census estimation and sociological analysis of cities and countries. Liu and Mattyus [26] also classified car orientations and types in aerial images of the city of Munich, with an accuracy of 98%, but limited the classification to two classes (car or truck).

Table 1. Comparison of our paper with the related works.

Ref.	Dataset used	Algorithms	Main results
[5] Mundhenk et al., 2016	Cars Overhead with Context (COWC): 32,716 unique cars. 58,247 negative targets. 308,988 training patches and 79,447 testing patches. Annotated using single pixel points. Resolution: Standardized to 15 cm/pixel.	ResCeption (Inception with Residual Learning)	Up to 99.14% correctly classified patches (containing cars or not). F1 score of 94.34% for detection. Car counting: RMSE of 0.676.
[6] Xi et al., 2019	Parking lot dataset from aerial view. Training: 2000 images. Testing: 1000 images. Number of instances: NA. Resolution: 5456x3632.	Multi-Task Cost-sensitive Convolutional Neural Network (MTCS-CNN).	mAP of 85.3% for car detection.
[15] Chen et al., 2014	63 satellite images collected from Google Earth. Training: 31 images (3901 vehicles). Testing: 32 images (2870 vehicles). Resolution: 1368x972.	Hybrid Deep Convolutional Neural Network (HDNN).	Precision up to 98% at a recall rate of 80%.
[16] Ammour et al., 2017	8 images acquired by UAV. Training: 3 images (136 positive instances, and 1864 negative instances). Testing: 5 images (127 positive instances). Resolution: Variable from 2424x3896 to 3456x5184. Spatial resolution of 2 cm.	Pre-trained CNN coupled with a linear support vector machine (SVM).	Precision from 67% up to 100%, and recall from 74% up to 84%, on the five testing images. Inference time: between 11 and 30 min/image.
[21] Hardjono et al., 2018	4 CCTV datasets: - Dataset 1: 3 second videos at 1 FPS. Resolution: 480x360 - Dataset 2: 60 min:32 sec video at 9 FPS. Resolution: 1920x1080 - Dataset 3: 30min:27sec video at 30 FPS. Resolution: 1280x720 - Dataset 4: 32 sec video at 30 FPS. Resolution: 1280x720 Training: 1932 positive instances and 10,000 negative instances.	- Background Subtraction (BS) - Viola Jones (VJ)- YOLOv2	- BS: F1 score from 32% to 55%. Inference time from 23 to 40 ms. - VJ: F1 score from 61% to 75%. Inference time from 39 to 640 ms. - YOLOv2: F1 score from 92% to 100% on datasets 2 to 4. Inference time not reported.
[1] Benjdira et al., 2019	PSU+[27] UAV dataset: Training: 218 images (3,365 car instances). Testing: 52 images (737 car instances). Resolution: Variable from 684x547 to 4000x2250.	- YOLOv3 (input size: 608x608). - Faster R-CNN (Feature extractor: Inception ResNet v2).	- YOLOv3: F1 score of 99.9%. Inference time: 57 ms. - Faster R-CNN: F1 score of 88%. Inference time: 1.39 s. (Using an Nvidia GTX 1080 GPU).
Our paper	- Stanford UAV dataset: Training: 6872 images (74,826 car instances). Testing: 1634 images (8,131 car instances). Resolution: Variable from 1184x1759 to 1434x1982. PSU+[27] UAV dataset: Training: 218 images (3,365 car instances). Testing: 52 images (737 car instances). Resolution: Variable from 684x547 to 4000x2250.	- YOLOv3 (input sizes: 320x320, 416x416, and 608x608). - Faster R-CNN (Feature extractors: Inception v2, and Resnet50).	- YOLOv3: F1 score: up to 32.6% on Stanford dataset up to 96.0% on PSU dataset. Inference time: from 43 to 85 ms. - Faster R-CNN: F1 score: up to 31.4% on Stanford dataset up to 84.5% on PSU dataset. Inference time: from 52 to 160 ms. (Using an Nvidia GTX 1080 GPU).

Table 1 summarizes the datasets, algorithms, and results of the most similar related works on car detection, compared to the present paper. 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. We show that annotation errors in the dataset have an important effect on the detection performance.

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 [27].

3. 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.

3.1. Faster R-CNN

R-CNN, as coined by [28] 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 [29] 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. [13] 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 [29].

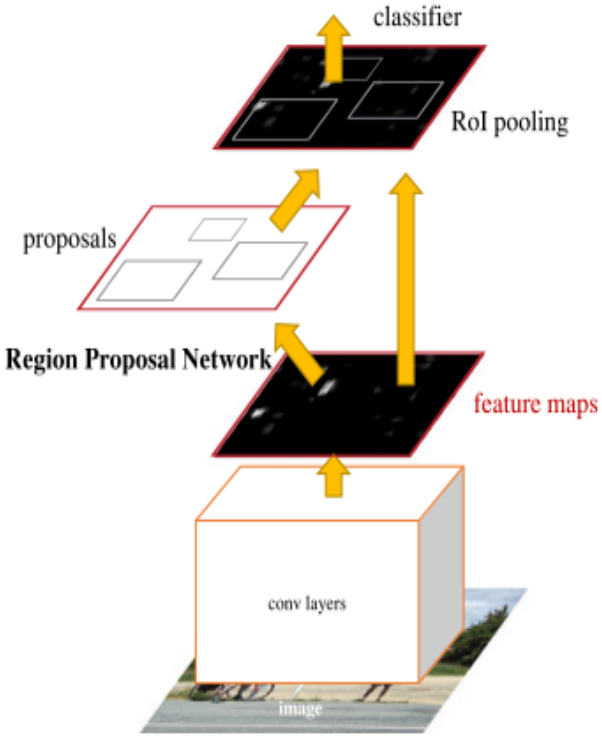

Figure 1. *Faster R-CNN basic architecture (from [1]).*

- N_{cls} and N_{reg} are normalization factors.
- λ is a balancing 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.

3.2. YOLOv3

Contrary to R-CNN variants, YOLO [30], 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 [22] 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 [14] 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.

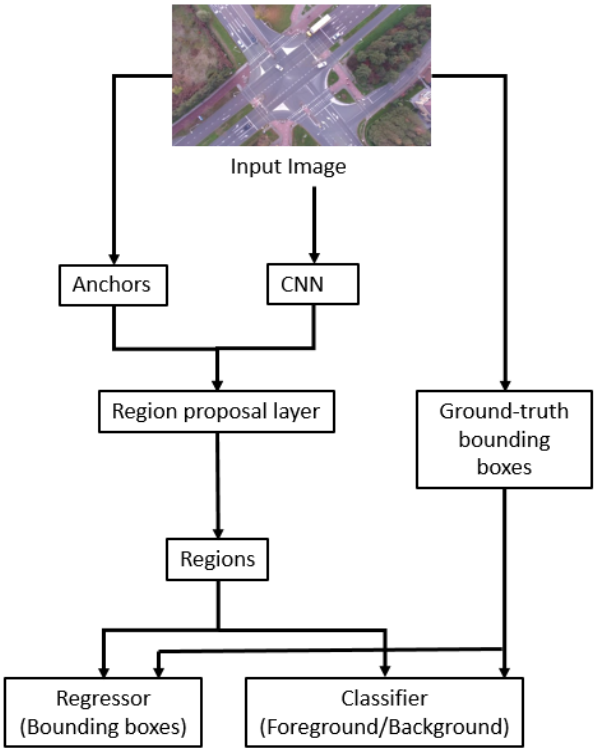

Figure 2. *Region Proposal Network (RPN) architecture.*

- 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 [31], 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 [19] in terms of accuracy but with the advantage of being three times faster [14]. 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 the three input sizes that are usually used (as in the original YOLOv3 paper [14]): 320x320, 416x416, and 608x608. Figure 6 shows an example of the output of YOLOv3 on a sample image of our PSU dataset, that will be described in section IV.A.

To summarize, Table 2 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.

4. Experimental comparison between Faster R-CNN and YOLOv3

4.1. Datasets

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

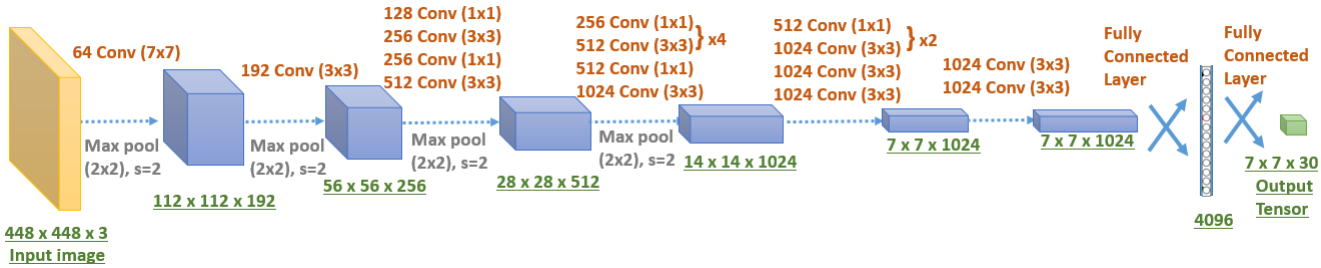

Figure 3. *YOLO (version 1) network architecture.*

	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			

Figure 4. *Darknet-53 architecture adopted by YOLOv3 (from [14]).*

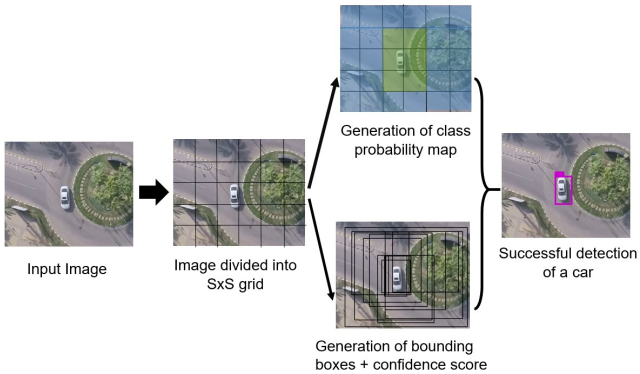

Figure 5. *Successive stages of the YOLOv3 model applied on car detection.*

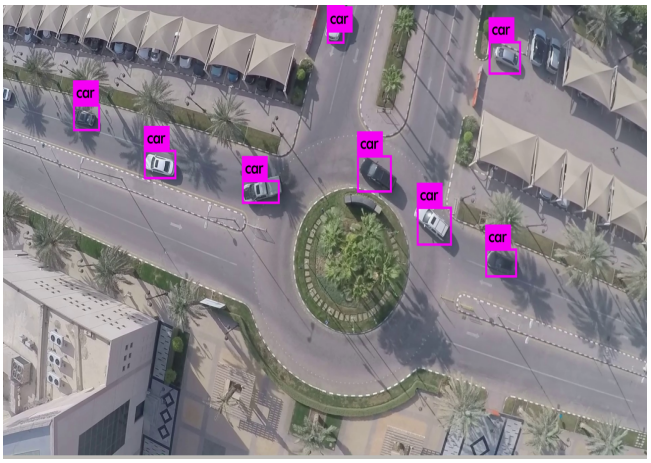

Figure 6. *Example of the output of YOLOv3 algorithm, on an image of the PSU dataset.*

Table 2. 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.
Default Anchor sizes	(10,13),(16,30),(33,23), (30,61),(62,45), (59,119),(116,90), (156,198), (373,326).	Scales: (128,128),(256,256), (512,512). Aspect ratios: 1:1, 1:2, 2:1.
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.
Batch size	Default value: 64.	Default value: 1.

Table 3. 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 4. 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 Stanford dataset** [32] 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 three scenes that contains the largest percentage of cars: Nexus (in which 29.51% of objects are cars), Gates (1.08%), and DeathCircle (4.71%). All other scenes contain less than 1% of cars. We used the two first scences for training, and the third one for testing. 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 8, 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 3 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 4, and contain cars of various sizes, as depicted in Figure 7. The average car size (calculated based on the ground-truth bounding boxes) is shown in Table 8. The discrepancy observed between the training and testing datasets in terms of car sizes is explained by the fact that we used different scenes for the training and testing datasets, as explained above. This discrepancy will constitute an additional challenge for the considered object detection algorithms.
- **The PSU dataset** was collected from two sources: an open dataset of aerial images available on Github [33]; 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. The training/testing split

Table 5. 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 6. Image size in the PSU dataset

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

Table 7. Details of Experiments

#	Algorithm	Feature Extractor	Dataset	Average input size	Number of iterations
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	320,000
6	YOLO v3	Darknet-53	Stanford	608*608	1,088,000
7	YOLO v3	Darknet-53	Stanford	320*320	896,000
8	YOLO v3	Darknet-53	PSU	416*416	640,000
9	YOLO v3	Darknet-53	PSU	608*608	640,000
10	YOLO v3	Darknet-53	PSU	320*320	640,000

was made randomly. Figure 9 shows a sample image of the PSU dataset, and Table 5 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 6, and contain cars of various sizes, as depicted in Figure 7. The average car size (calculated based on the ground-truth bounding boxes) in the training and testing datasets is shown in Table 8.

4.2. Hyperparameters

The main hyperparameter for the YOLO network is the input size, for which we tested three values (320x320, 416x416, and 608x608), as explained in section III.B. On the other hand, the main hyperparameter for Faster R-CNN is the feature extractor. We tested two different feature extractors: Inception-v2 [34] (also called BN-inception in the literature [35]) and Resnet50 [36]. 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 7. We kept the default values for the momentum (0.9), weight decay (0.0005), learning rate (initial rate of 0.001 for YOLOv3, 0.0002 for Faster R-CNN with Inception-v2, and 0.0003 with Resnet50), batch size (64 for YOLOv3 and 1 for Faster R-CNN), and anchor sizes (see Table 2). And we trained each network for the number of iterations necessary to its convergence. We notice, for example, in Table 7 that YOLOv3 necessitated a higher number of iterations when using the largest input size (608x608) on Stanford dataset, while it reached convergence after much less iterations when using the medium input size (416x416) on the same dataset. Nevertheless,

Table 8. Average car width and length (in pixels) in the PSU and Stanford datasets, calculated based on the ground-truth bounding boxes.

Dataset	Average car width	Average car length
PSU training	48	36
PSU testing	55	46
Stanford training	72	152
Stanford testing	60	90

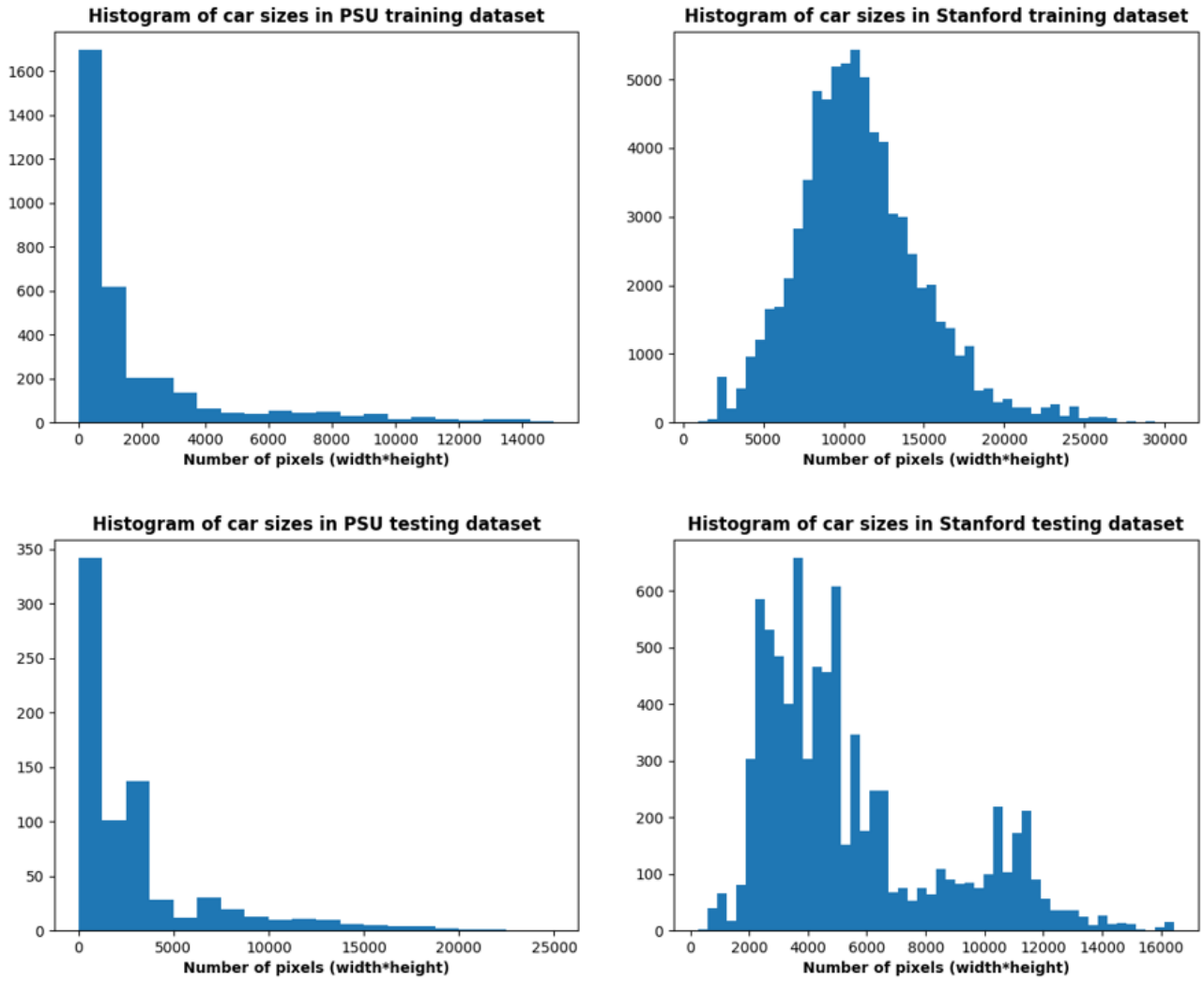

Figure 7. *Histogram of car sizes in PSU (left) and Stanford (right) training (up) and testing (down) datasets, expressed as the number pixels inside the ground truth bounding boxes (width*height).*

Table 9. 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 10. 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%**

the number of steps needed to reach convergence is non-deterministic, and depends on the initialization of the weights.

4.3. 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.

$$Inference\ time\ (ms) = \frac{1000}{FPS}$$

- $AR^{max=1}$, $AR^{max=10}$, $AR^{max=100}$: average recall, when considering a maximum number of detections per image, averaged over all values of IoU specified above. We allow only the 1, 10 or 100 top-scoring detections for each image. This metric penalizes missing detections (false negatives) and duplicates (several bounding boxes for a single object).

Table 11. Detailed results of the tested configurations of YOLOv3 and Faster R-CNN, on PSU dataset.

Algorithm	**Feature extractor**	**Input size**	**AP**	**TP**	**FN**	**FP**	**Precision**	**Recall**	**F1 score**
Faster RCNN	Inception v2	992*550	0.739	548	190	11	0.980	0.743	0.845
Faster RCNN	Resnet50	992*550	0.708	524	214	**9**	**0.983**	0.710	0.825
YOLOv3	Darknet-53	320*320	0.902	672	66	35	0.950	0.911	0.930
YOLOv3	Darknet-53	416*416	0.957	710	28	40	0.947	0.962	0.954
YOLOv3	Darknet-53	608*608	**0.965**	**715**	**23**	36	0.952	**0.969**	**0.960**

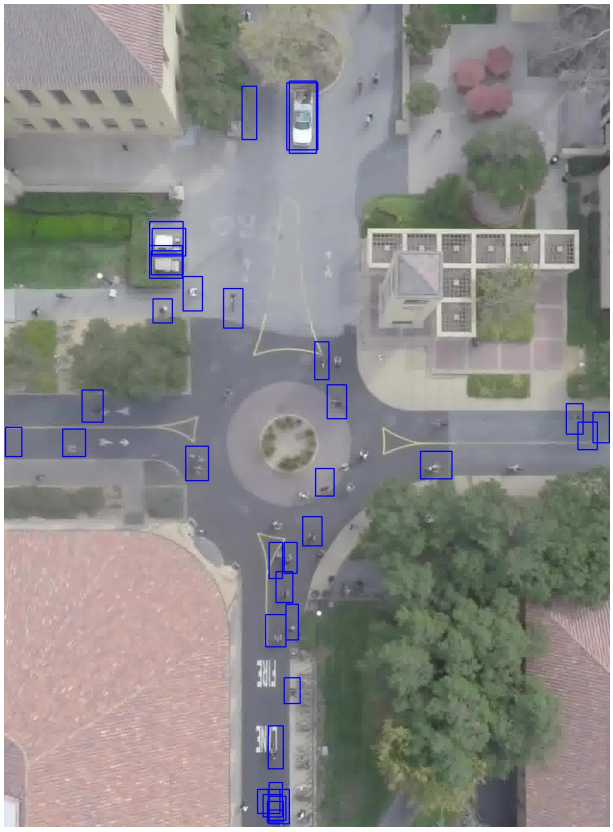

Figure 8. *A sample image of the Stanford dataset, with ground-truth bounding boxes showing some annotation errors and imprecisions.*

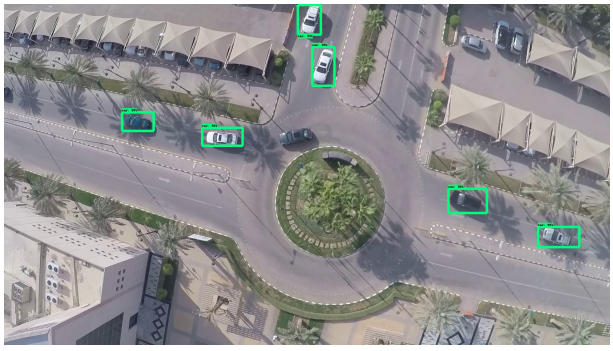

Figure 9. *Example of the output of Faster R-CNN algorithm, on an image of the PSU dataset.*

Table 12. Detailed results of all tested configurations of YOLOv3 and Faster R-CNN, on Stanford dataset.

Algorithm	Feature extractor	Input size	AP	TP	FN	FP	Precision	Recall	F1 score
Faster RCNN	Inception v2	600*816	0.202	1780	6351	1813	0.495	0.219	0.304
Faster RCNN	Resnet50	600*816	**0.219**	**1909**	**6222**	2117	0.474	**0.235**	0.314
YOLOv3	Darknet-53	320*320	0.107	876	7255	4	0.995	0.108	0.194
YOLOv3	Darknet-53	416*416	0.195	1583	6548	**1**	**0.999**	0.195	**0.326**
YOLOv3	Darknet-53	608*608	0.194	1581	6550	10	0.994	0.194	0.325

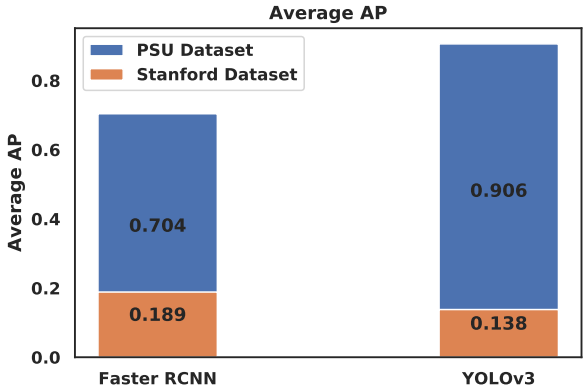

Figure 10. *Comparison of the average AP between YOLOv3 and Faster R-CNN.*

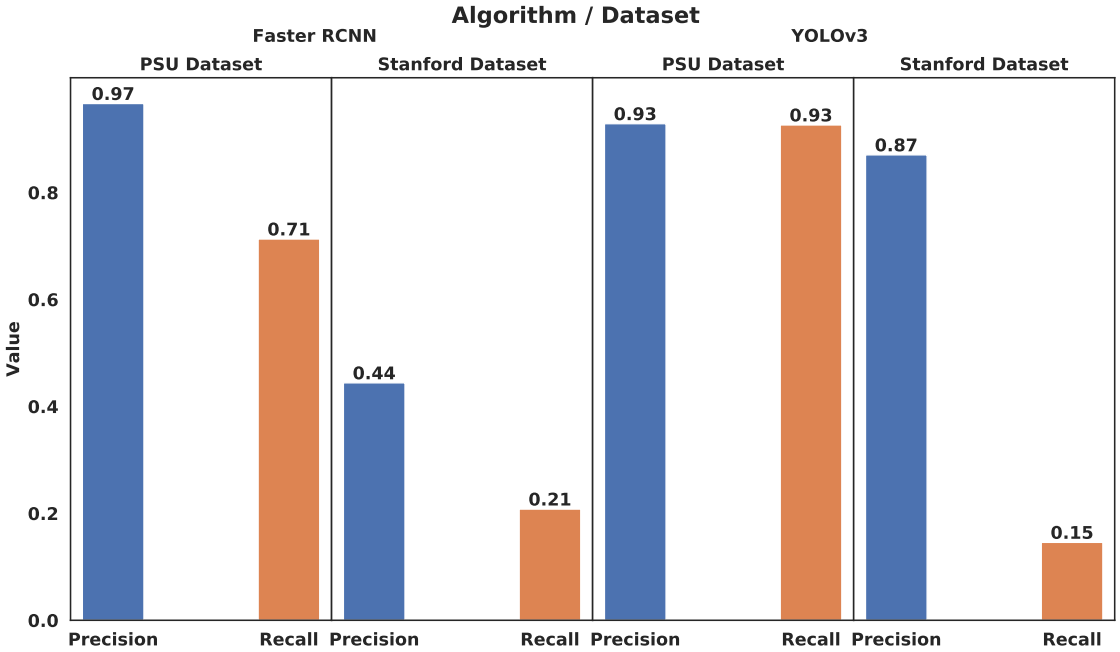

Figure 11. *Precision and recall values for YOLOv3 and Faster R-CNN, on the two datasets.*

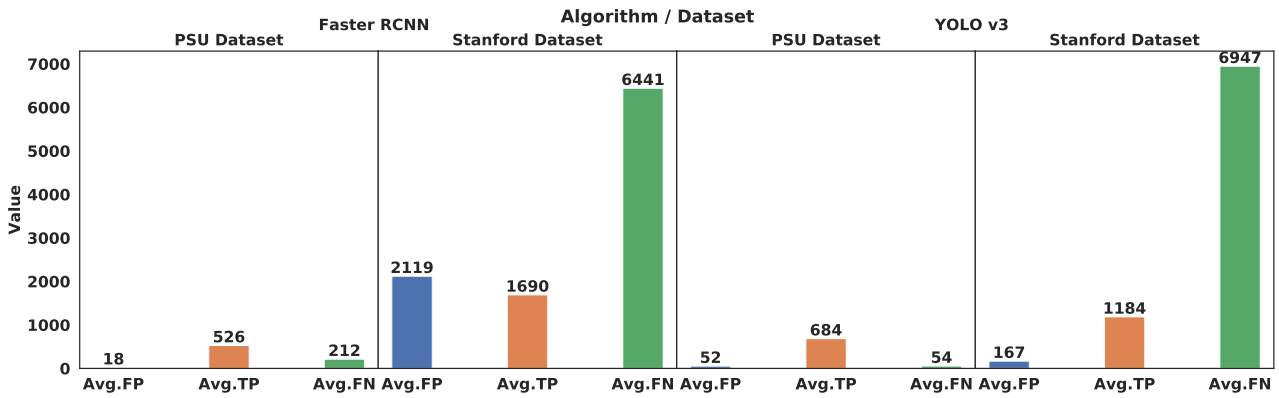

Figure 12. *Average number of false positives (FP), false negatives (FN), and true positives (TP) for YOLOv3 and Faster R-CNN, on the two datasets.*

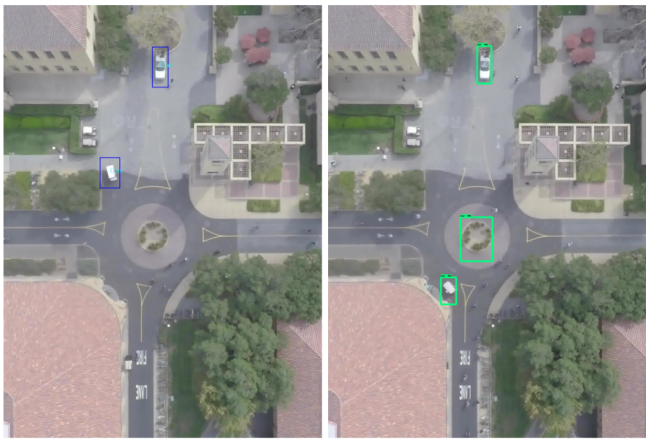

Figure 13. *Example of (a) YOLOv3 and (b) Faster R-CNN's output on a sample image of the Stanford dataset.*

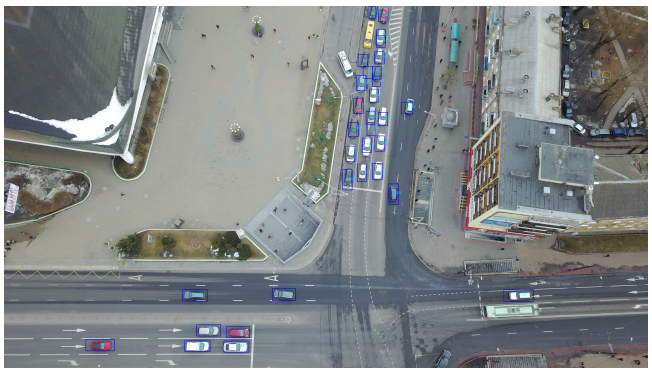

Figure 14. *Example of YOLOv3's output on an image of the PSU dataset, showing a few false negatives (non detected cars).*

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

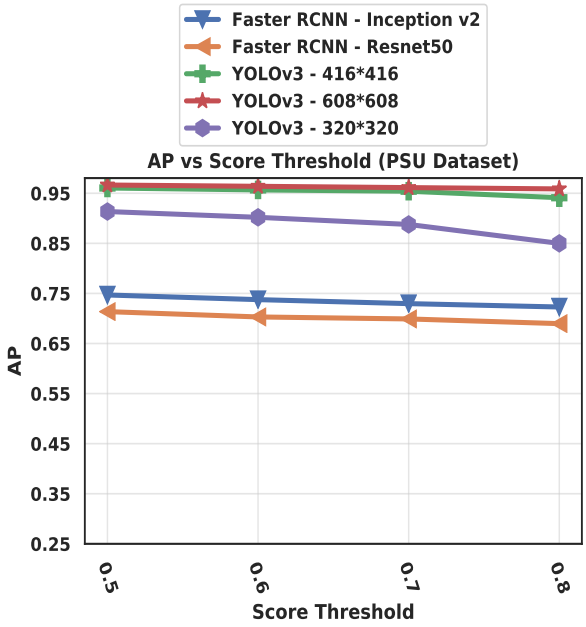

Figure 16. *AP for different values of score threshold, for the two algorithms on PSU dataset (IoU threshold fixed at 0.6).*

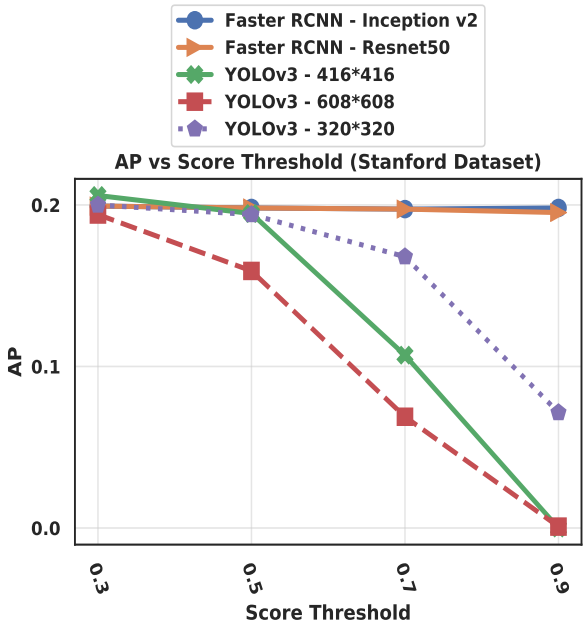

Figure 17. *AP for different values of score threshold, for the two algorithms on Stanford dataset (IoU threshold fixed at 0.6).*

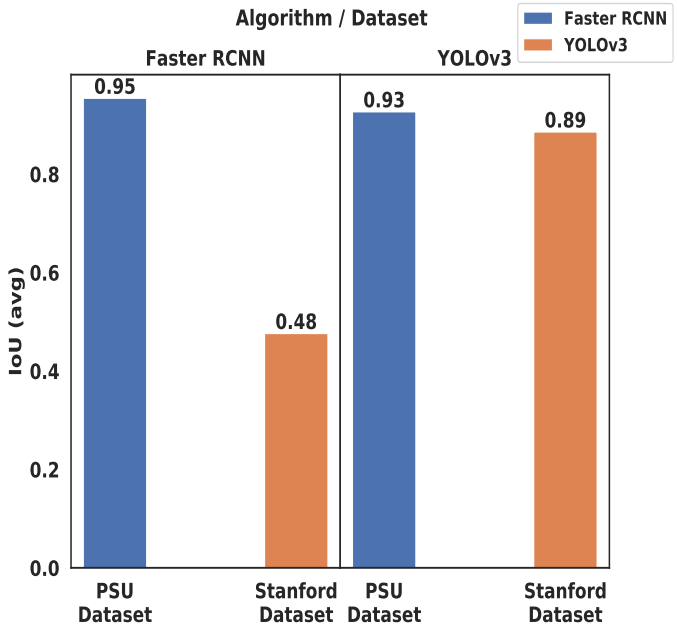

Figure 18. *Average IoU value for YOLOv3 and Faster R-CNN, on the two datasets.*

4.3.1. **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 10). This is mainly due to the fact that, contrary to the PSU dataset, the characteristics of the Stanford dataset differ largely between the training and testing images, as detailed in IV.A. This is the well known problem of domain adaptation in machine learning (see section II). The Stanford dataset contains 20 times more car instances than the PSU dataset (Tables 3 and 5), whereas the performance of Faster R-CNN and YOLOv3 was respectively 4 and 7 times better on the PSU dataset, in terms of AP. This highlights the fact that the clarity of the features, the quality of annotation, and the representativity of the learning dataset are more important than the actual size of the dataset. Figure 11 confirms this observation and shows that both precision and recall are significantly lower on the Stanford dataset. However, Figure 12 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, most probably due to the different size and aspect ratio of cars in the testing images, compared to the training images. Even though both algorithms performed poorly on the Stanford dataset as compared to PSU dataset, with less than 20% of AP, there is still a statistically significant difference between Faster R-CNN and YOLOv3 on this dataset. In fact, a T-test between the two sets of AP values of the two algorithms (for different IoU and score thresholds) yielded a p-value of 0.0020, which means that the null hypothesis (equality of the means of the two sets of AP values) can be rejected with a confidence of 99.8%.

Figure 13 shows 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

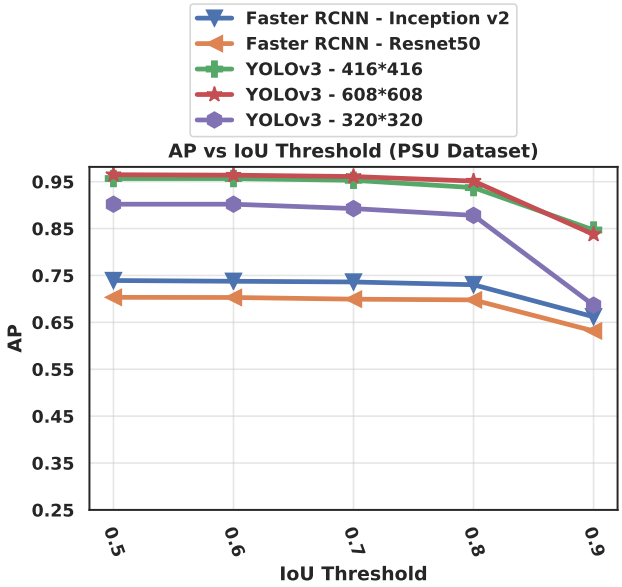

Figure 19. *Average Precision at different IoU threshold values of the tested algorithms on the PSU dataset.*

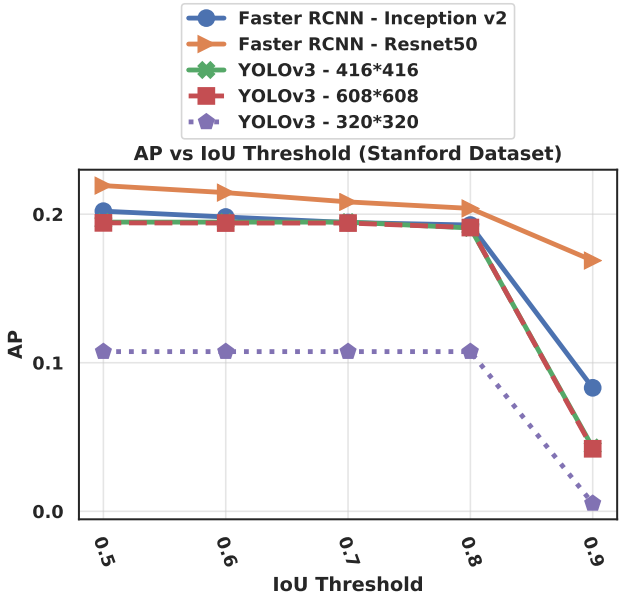

Figure 20. *Average Precision at different IoU threshold values of the tested algorithms on the Stanford dataset.*

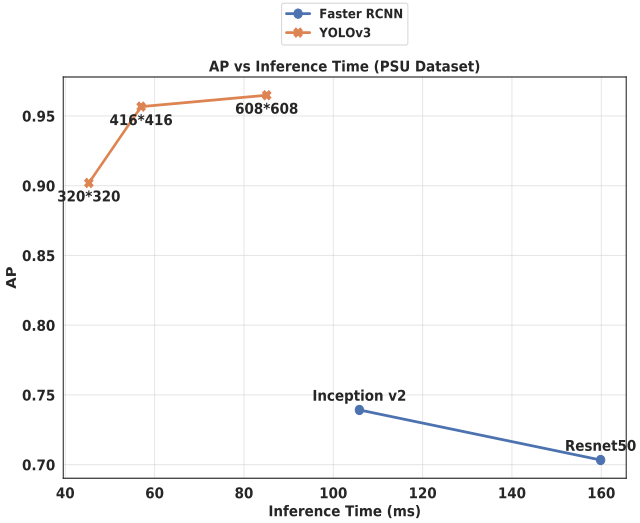

Figure 21. *Comparison of the trade-off between AP and inference time for YOLOv3 (with 3 different imput sizes) and Faster R-CNN (with two different feature extractors), on the PSU dataset.*

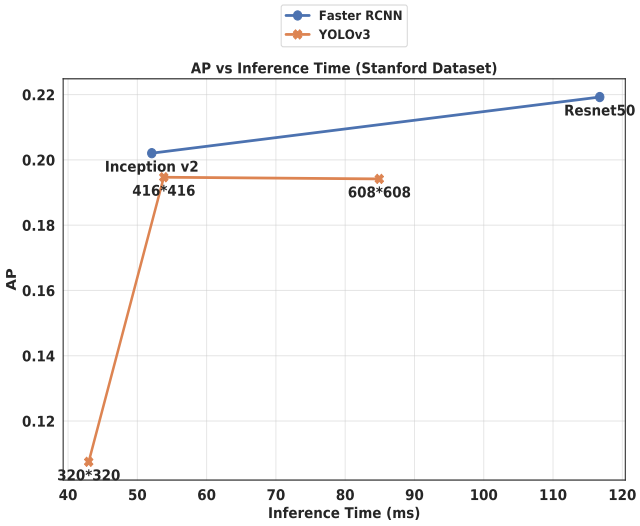

Figure 22. *Comparison of the trade-off between AP and inference time for YOLOv3 (with 3 different input sizes) and Faster R-CNN (with two different feature extractors), on the Stanford dataset.*

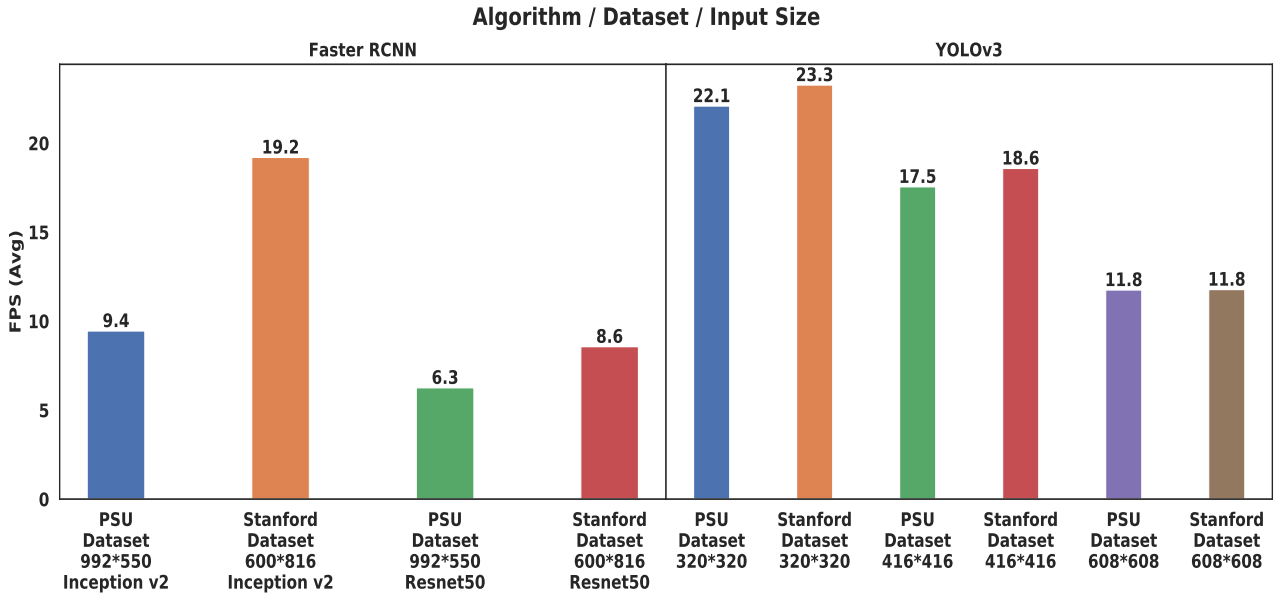

Figure 23. *Inference speed measured in Frames per Second (FPS), for each of the tested algorithms.*

Stanford dataset are much less confident. For all other figures shown here, the score threshold has been fixed to 0.5.

4.3.2. **Average Recall**

Table 9 shows the average recall for a given maximum number of detections (as described in the introduction of section C), 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. Nevertheless, we have kept this duplicated column to compare it to Table 10 which 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.

4.3.3. **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 markedly decreases on the Stanford dataset (Figure 10). 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 [13].

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, and the discrepancy between training and testing features could explains the difference 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.

4.3.4. **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. [35] who also showed that Inception-v2 (aka BN-inception) is less computationally complex.

4.3.5. **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 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).

4.3.6. **Effect of the anchor scales**

The anchor scales used for the two algorithms are the default values specified in Table 2. As we suspected that the anchor values could be the reason for the poor performance of both algorithm on Stanford dataset, we subsequently tested a different set of anchor scales. For YOLOv3 The new anchor scales were calculated using K-means clustering on the Stanford training dataset, and yielded smaller anchor sizes (10x27, 25x16, 17x26, 18x35, 22x31, 35x23, 23x38, 27x34, and 31x42). And for Faster R-CNN, we used anchor scales reduced by half (64x64, 128x128, and 256x256, instead of the default 128x128, 256x256, and 51x512). Table 13 shows the results obtained after using these anchors, compared to the previous results obtained with the default anchors. The performance was markedly lower for YOLOv3, which indicates that this algorithm is very sensitive to the change of anchor scales. As for Faster R-CNN with Resnet50 as feature extractor, the AP was slightly lower (20.7% down from 21.9%), while the average IoU dropped noticeably (25% down from 47.7%). In contrast, Faster R-CNN with Inception-v2 as feature extractor was the only algorithm that showed better results with the reduced anchor scales. The two rightmost columns in Table 13 show the average width and height of the predicted bounding boxes. We notice that the dependency between the anchor scales and the predicted sizes is not straightforward. The average predicted sizes are more affected by the size of ground-truth bounding boxes in the training dataset (72x152 in average, as shown in Table 8), and adapt poorly to the different ground-truth car sizes and aspect ratios in the testing dataset (60x90 in average), which explains the low performance of all the tested algorithms on the Stanford dataset specifically. Moreover, we can observe that despite the fact that the default anchor scales for Faster R-CNN are overall larger than those of YOLOv3, the first algorithm yields better AP on the Stanford dataset than the latter, which indicates that smaller anchor scales are not the solution for the poor performance obtained on the Stanford dataset.

4.3.7. **Main lessons learned**

Tables 11 and 12 present the detailed results of all tested configurations of the two algorithms on PSU and Stanford datasets respectively. The best performance for each metric and each dataset is highlighted. We notice that YOLOv3 with high input size (608x608) and Faster R-CNN (with Resnet50 feature extractor) show the best results in terms of AP and recall, on PSU and Stanford datasets

Table 13. Effect of reducing the anchor scales of YOLOv3 and Faster R-CNN, on the Stanford Dataset.

Algorithm	Anchor scales	AP	IoU	Average predicted width	Average predicted height
YOLOv3 416x416 (default anchors)	10x13, 16x30, 33x23, 30x61, 62x45, 59x119, 116x90, 156x198, 373x326	0.195	0.89	67	170
YOLOv3 416x416 (reduced anchors)	10x27, 25x16, 17x26, 18x35, 22x31, 35x23, 23x38, 27x34, 31x42	0.082	0.55	127	282
Faster R-CNN, with ResNet50 (default anchors)	Scales: 128x128, 256x256, 512x512 Aspect ratios: 1:1, 1:2, 2:1	0.219	0.48	91	171
Faster R-CNN, with ResNet50 (reduced anchors)	Scales: 64x64, 128x128, 256x256 Aspect ratios: 1:1, 1:2, 2:1	0.207	0.25	72	131
Faster R-CNN, with Inception-v2 (default anchors)	Scales: 128x128, 256x256, 512x512 Aspect ratios: 1:1, 1:2, 2:1	0.202	0.48	74	140
Faster R-CNN, with Inception-v2 (reduced anchors)	Scales: 64x64, 128x128, 256x256 Aspect ratios: 1:1, 1:2, 2:1	0.255	0.50	92	174

respectively. While in terms of precision, Faster R-CNN (with Resnet50 feature extractor) and YOLOv3 with medium input size (416x416) perform better on PSU and Stanford datasets respectively. 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, representativity of the training dataset, 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.

5. Conclusion

In this study, we conducted a thorough experimental comparison of the two leading object detection algorithms (YOLOv3 and Faster R-CNN) on two UAV imaging datasets 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 [37] or COCO [27]. The two datasets 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, and the representativity of the training images. 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.

Author Contributions: conceptualization, A.K. and A.A.; methodology, A.A. and A.K.; software, M.A., A.S., and A.A; validation, A.A. and A.K.; formal analysis, A.A. and A.K.; investigation, A.A., A.K., M.A., and A.S.; resources, A.K., A.A., M.A., and A.S.; data curation, M.A. and A.S.; writing–original draft preparation, A.A.; writing–review and editing, A.A.; visualization, A.A., M.A., and A.S.; supervision, A.K. and A.A.; project administration, A.K.; funding acquisition, A.K.

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

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.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Benjdira, B.; Khursheed, T.; Koubaa, A.; Ammar, A.; Ouni, K. Car Detection using Unmanned Aerial Vehicles: Comparison between Faster R-CNN and YOLOv3. 2019 1st International Conference on Unmanned Vehicle Systems-Oman (UVS). IEEE, 2019, pp. 1–6.
2. Koubaa, A.; Qureshi, B. DroneTrack: Cloud-Based Real-Time Object Tracking Using Unmanned Aerial Vehicles Over the Internet. *IEEE Access* **2018**, *6*, 13810–13824. doi:10.1109/ACCESS.2018.2811762.
3. Alotaibi, E.T.; Alqefari, S.S.; Koubaa, A. LSAR: Multi-UAV Collaboration for Search and Rescue Missions. *IEEE Access* **2019**, *7*, 55817–55832. doi:10.1109/ACCESS.2019.2912306.
4. Xi, X.; Yu, Z.; Zhan, Z.; Tian, C.; Yin, Y. Multi-task Cost-sensitive-Convolutional Neural Network for Car Detection. *IEEE Access* **2019**, pp. 1–1. doi:10.1109/ACCESS.2019.2927866.
5. Mundhenk, T.N.; Konjevod, G.; Sakla, W.A.; Boakye, K. A large contextual dataset for classification, detection and counting of cars with deep learning. European Conference on Computer Vision. Springer, 2016, pp. 785–800.
6. Xi, X.; Yu, Z.; Zhan, Z.; Yin, Y.; Tian, C. Multi-Task Cost-Sensitive-Convolutional Neural Network for Car Detection. *IEEE Access* **2019**, *7*, 98061–98068.
7. Ševo, I.; Avramović, A. Convolutional Neural Network Based Automatic Object Detection on Aerial Images. *IEEE Geoscience and Remote Sensing Letters* **2016**, *13*, 740–744. doi:10.1109/LGRS.2016.2542358.
8. Ochoa, K.S.; Guo, Z. A framework for the management of agricultural resources with automated aerial imagery detection. *Computers and Electronics in Agriculture* **2019**, *162*, 53 – 69. doi:https://doi.org/10.1016/j.compag.2019.03.028.
9. Kampffmeyer, M.; Salberg, A.; Jenssen, R. Semantic Segmentation of Small Objects and Modeling of Uncertainty in Urban Remote Sensing Images Using Deep Convolutional Neural Networks. 2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2016, pp. 680–688. doi:10.1109/CVPRW.2016.90.
10. Azimi, S.M.; Fischer, P.; Körner, M.; Reinartz, P. 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* **2019**, *57*, 2920–2938. doi:10.1109/TGRS.2018.2878510.
11. Mou, L.; Zhu, X.X. Vehicle Instance Segmentation From Aerial Image and Video Using a Multitask Learning Residual Fully Convolutional Network. *IEEE Transactions on Geoscience and Remote Sensing* **2018**, *56*, 6699–6711. doi:10.1109/TGRS.2018.2841808.
12. Benjdira, B.; Bazi, Y.; Koubaa, A.; Ouni, K. Unsupervised Domain Adaptation Using Generative Adversarial Networks for Semantic Segmentation of Aerial Images. *Remote Sensing* **2019**, *11*. doi:10.3390/rs11111369.
13. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE* **2017**, [1506.01497]. doi:10.1109/TPAMI.2016.2577031.
14. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *CoRR* **2018**, *abs/1804.02767*, [1804.02767].
15. Chen, X.Y.; Xiang, S.M.; Liu, C.L.; Pan, C.H. Vehicle Detection in Satellite Images by Hybrid Deep Convolutional Neural Networks. *Ieee Geoscience and Remote Sensing Letters* **2014**. doi:10.1109/Lgrs.2014.2309695.
16. Ammour, N.; Alhichri, H.; Bazi, Y.; Benjdira, B.; Alajlan, N.; Zuair, M. Deep Learning Approach for Car Detection in UAV Imagery. *Remote Sensing* **2017**, *9*, 312. doi:10.3390/rs9040312.
17. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations (ICRL)* **2015**, [1409.1556]. doi:10.1016/j.infsof.2008.09.005.

18. Kim, C.E.; Oghaz, M.M.D.; Fajtl, J.; Argyriou, V.; Remagnino, P. A comparison of embedded deep learning methods for person detection. *arXiv preprint arXiv:1812.03451* **2018**.
19. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. SSD: Single shot multibox detector. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2016, [1512.02325]. doi:10.1007/978-3-319-46448-0_2.
20. 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)* **2019**, pp. 1–1.
21. Hardjono, B.; Tjahyadi, H.; Rhizma, M.G.A.; Widjaja, A.E.; Kondorura, R.; Halim, A.M. Vehicle Counting Quantitative Comparison Using Background Subtraction, Viola Jones and Deep Learning Methods. 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), 2018, pp. 556–562.
22. Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, 2017, pp. 6517–6525. doi:10.1109/CVPR.2017.690.
23. Tayara, H.; Gil Soo, K.; Chong, K.T. Vehicle Detection and Counting in High-Resolution Aerial Images Using Convolutional Regression Neural Network. *IEEE Access* **2018**, *6*, 2220–2230. doi:10.1109/ACCESS.2017.2782260.
24. Liang, J.; Chen, X.; He, M.l.; Chen, L.; Cai, T.; Zhu, N. Car detection and classification using cascade model. *IET Intelligent Transport Systems* **2018**, *12*, 1201–1209.
25. Gebru, T.; Krause, J.; Wang, Y.; Chen, D.; Deng, J.; Fei-Fei, L. Fine-grained car detection for visual census estimation. Thirty-First AAAI Conference on Artificial Intelligence, 2017.
26. Liu, K.; Mattyus, G. Fast Multiclass Vehicle Detection on Aerial Images. *IEEE Geoscience and Remote Sensing Letters* **2015**, *12*, 1938–1942. doi:10.1109/LGRS.2015.2439517.
27. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft coco: Common objects in context. European conference on computer vision. Springer, 2014, pp. 740–755.
28. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2014, pp. 580–587, [1311.2524]. doi:10.1109/CVPR.2014.81.
29. Girshick, R. Fast R-CNN. Proceedings of the IEEE International Conference on Computer Vision, 2015, [1504.08083]. doi:10.1109/ICCV.2015.169.
30. Redmon, J.; Divvala, S.K.; Girshick, R.B.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, 2016, pp. 779–788. doi:10.1109/CVPR.2016.91.
31. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. *Arxiv.Org* **2015**, [1512.03385]. doi:10.3389/fpsyg.2013.00124.
32. Robicquet, A.; Sadeghian, A.; Alahi, A.; Savarese, S. Learning social etiquette: Human trajectory understanding in crowded scenes. European conference on computer vision. Springer, 2016, pp. 549–565.
33. Aerial-car-dataset, available online on: https://github.com/jekhor/aerial-cars-dataset, accessed on (16-10-2018).
34. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *CoRR* **2015**, *abs/1502.03167*, [1502.03167].
35. Bianco, S.; Cadene, R.; Celona, L.; Napoletano, P. Benchmark analysis of representative deep neural network architectures. *IEEE Access* **2018**, *6*, 64270–64277.
36. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
37. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. Imagenet: A large-scale hierarchical image database. 2009 IEEE conference on computer vision and pattern recognition. Ieee, 2009, pp. 248–255.