




Article

YOLO-Fine: One-Stage Detector of Small Objects Under Various Backgrounds in Remote Sensing Images

Minh-Tan Pham ^{1,*}, Luc Courtrai ¹, Chloé Friguet ¹, Sébastien Lefèvre ¹
and Alexandre Baussard ²

¹ IRISA, Université Bretagne Sud, 56000 Vannes, France; luc.courtrai@irisa.fr (L.C.); chloe.friguet@irisa.fr (C.F.); sebastien.lefevre@irisa.fr (S.L.)

² Institut Charles Delaunay, Université de Technologie de Troyes, 10000 Troyes, France; alexandre.baussard@utt.fr

* Correspondence: minh-tan.pham@irisa.fr

Received: 9 June 2020; Accepted: 2 August 2020; Published: 4 August 2020



Abstract: Object detection from aerial and satellite remote sensing images has been an active research topic over the past decade. Thanks to the increase in computational resources and data availability, deep learning-based object detection methods have achieved numerous successes in computer vision, and more recently in remote sensing. However, the ability of current detectors to deal with (very) small objects still remains limited. In particular, the fast detection of small objects from a large observed scene is still an open question. In this work, we address this challenge and introduce an enhanced one-stage deep learning-based detection model, called You Only Look Once (YOLO)-fine, which is based on the structure of YOLOv3. Our detector is designed to be capable of detecting small objects with high accuracy and high speed, allowing further real-time applications within operational contexts. We also investigate its robustness to the appearance of new backgrounds in the validation set, thus tackling the issue of domain adaptation that is critical in remote sensing. Experimental studies that were conducted on both aerial and satellite benchmark datasets show some significant improvement of YOLO-fine as compared to other state-of-the-art object detectors.

Keywords: small object detection; remote sensing; background variability; deep learning; one-stage detector

1. Introduction

Object detection consists in both estimating the exact object position (object localization) and determining which category it belongs to (object classification). It is a central component in many real-world applications, including detection of pedestrians and vehicles in autonomous driving, detection of people in order to analyze their behaviors, detection, and further recognition of faces for surveillance purposes, etc. (see [1] for a recent survey). However, due to the high variations of viewpoints, poses, occlusion, and lighting conditions, object detection remains a challenging problem. Before the advent of deep learning, the standard object detection pipeline in computer vision was made of three main steps: (i) selecting regions of interest (ROIs), where objects may appear or be present, (ii) extracting some characteristics from these regions/objects (a.k.a. feature extraction), and (iii) feeding a supervised classifier with these features. These so-called handcrafted feature design-based approaches have generally provided good results for real-time applications in operational systems [1]. However, they were mostly unable to exceed a certain level of accuracy. Within the last decade, the emerging development of deep neural networks, and more particularly of convolutional neural networks (CNNs), brought both a paradigm change and some significant improvements [2,3]. This gain in accuracy has been possible, thanks not only to some methodological

advances, but also to the availability of large amounts of data and to the increased computing capacity as well. One of the most remarkable milestones in object detection was achieved by the proposition of Region-based CNN (R-CNN) in 2014 [4]. Since then, object detection using deep learning approaches has been evolved at an unstoppable speed [1–3], attempting to achieve better and better detection accuracy as well as to ensure faster and faster detection for real-time applications. We briefly mention here some milestones in two-stage detection frameworks, i.e., which first generate regions of interest and then perform classification and bounding box regression for each proposal, such as Fast R-CNN [5], Faster R-CNN [6], R-FCN (Region-based Fully Convolutional Network) [7], Feature Pyramid Network [8], Mask R-CNN [9], etc., and one-stage detectors, i.e., which perform a one-pass regression of class probabilities and bounding box locations, such as YOLO (You Only Look Once) [10], YOLOv2 [11], YOLOv3 [12], SSD (Single Shot Multibox Detector) [13], DSOD (Deeply Supervised Object Detector) [14], RetinaNet [15], RON (Reverse connection with Objectness prior Networks) [16], M2Det [17], EfficientDet [18], etc. In general, two-stage detectors favor detection accuracy while one-stage models are faster. To illustrate, we recall the results reported in [14]: the inference speed of Faster R-CNN, R-FCN, SSD, and YOLOv2 for PASCAL VOC data is, respectively, 7, 11, 46, and 81 FPS (frames per second), while the detection rates provided by the two former (two-stage) were better than the two latter (one-stage). For more details about their concept and performance, we refer readers to the aforementioned surveys.

In remote sensing, object detection has also been an active research area due to its central role in various Earth Observation (EO) applications: detecting and counting man-made structures (such as roads, buildings, airports, etc.) [19–21]; detecting and tracking moving objects (such as vehicles, ships, etc.) [22–25]; and, detecting endangered species (e.g., wildlife animals, sea mammals, etc.) [26,27]. The popularity of CNN-based object detection methods introduced in the computer vision domain made these solutions a first choice when detecting objects from optical remote sensing data (e.g., aerial and satellite images). However, such a transfer from computer vision to remote sensing is not straightforward due to the specific characteristics of geospatial objects that appear in the observed scenes. Unlike in natural images, objects in remote sensing data are highly varied in size and shape, from very small (vehicle, animal, etc.) to very large (airport, football yard, swimming pool, etc.) ones, with also a high range of spectral signatures depending on the acquisition sensor, acquisition lighting, weather conditions, etc. Unsurprisingly, data collection and annotation play an important role in achieving good performance in addition to the design or improvement of detection methods. We refer readers to some recent surveys on object detection in remote sensing using deep learning [28–30] for more details about detectors, data, and applications in the remote sensing community.

One of the most challenging tasks pointed out by the above surveys is the limited performance of current detectors when dealing with small and very small objects (i.e. with less than 10 pixels in width), such as vehicles or wildlife animals in high resolution optical satellite images [28–30]. The trade-off between detection accuracy and computational time does not allow for rapidly detecting small objects in large EO scenes, in particular to perform real-time detection or to use tracking systems within an operational context. Both two-stage and one-stage detector models have their own issues to deal with small objects. While two-stage frameworks are very slow (as reported in [14]), one-stage models are often designed for fast inference and not to deal with small objects. In this paper, we aim at addressing real-time detection of small objects in remote sensing images. To do so, we propose to build a one-stage object detector that could succeed in detecting small objects with both high accuracy and high speed. The proposed detector is named YOLO-fine and it focuses on small object detection by performing finer regression to search for finer objects. Our YOLO-fine is inspired from YOLOv3 [12], which is one of the state-of-the-art detectors in computer vision as well as in remote sensing, as illustrated in the next section. We also investigate the ability of our detector to deal with new backgrounds that appear in the validation set. Experiments, conducted on both aerial and satellite data sets, show that YOLO-fine significantly improves the detection performance *w.r.t.* state-of-the-art detectors, better deals with various backgrounds to detect known objects, and it is able to perform real-time prediction.

In the remainder of the paper, Section 2 briefly goes through literature works in object detection in remote sensing with a focus on small objects. Section 3 revisits the one-stage YOLO detector family and describes our proposal, called YOLO-fine. In Section 4, we report our experimental studies on three different datasets, including two aerial (VEDAI [31] and MUNICH [32]) and one satellite (XVIEW [33]) remote sensing datasets. We also provide our experimental design to investigate the performance of our detector regarding various types of backgrounds. Finally, the main contributions and improvement prospects are considered in Section 5.

2. Related Studies

Many efforts have been devoted to tackle the detection of small objects in the computer vision domain, mostly by adapting the existing one-stage and two-stage frameworks mentioned in our introduction to better deal with small objects. We refer readers to a very recent survey on small object detectors in computer vision based on deep learning [34]. In this review, the authors have mentioned five crucial aspects that are involved in recent small object detection frameworks, including multiscale feature learning, data augmentation, training strategy, context-based learning, and generative network-based detection. They also highlighted some powerful models to detect generic small objects, such as improved Faster R-CNN [35,36], Feature-fused SSD [37], MDSSD (Multi-scale deconvolutional SSD) [38], RefineDet [39], SCAN (Semantic context aware network) [40], etc. In the remote sensing community, the detection of small objects has been mostly tackled by exploiting two-stage object detectors thanks to their capacity to generally provide more accurate detection performance as compared to one-stage detectors. In this scenario, the region-based approach appears to be more relevant since the first stage whose aim is to search candidate objects could be set to focus on small regions and ignore large ones. Many recent works have exploited state-of-the-art two-stage detectors, such as Faster R-CNN [22,41–43], deconvolution R-CNN [44], and deformable R-CNN [45,46] to detect e.g., small vehicles, airplanes, ships, man-made structures, in remote sensing datasets collected mostly from aerial or Google Earth images. In [41], the RPN (Region proposal network) of Faster-RCNN was modified by setting appropriate anchors and leveraging a single high-level feature map of finer resolution for small region searching. Authors also incorporated contextual information with the proposal region to further boost the performance of detecting small objects. In [44], Deconv R-CNN was proposed by setting a deconvolution layer after the last convolutional layer in order to recover more details and better localize the position of small targets. This simple but efficient technique helped to increase the performance of ship and plane detection compared to the original Faster R-CNN. In [46], an IoU-adaptive deformable R-CNN was developed with the goal of adapting IoU threshold according to the object size, to ease dealing with small objects whose loss (according to the authors) would be absorbed during training phase. Therefore, such an IoU-based weighted loss was adopted to train the network and improve the detection performance on small objects in the large-scale aerial DOTA dataset [47]. Recently, other developments that were based on feature pyramid detector (MCFN) [48], R2CNN [49], or SCRNet [50] were proposed based on the two-stage detection scheme to tackle small object detection. While these methods achieve a proper level of accuracy, they are still limited by their computational cost (see results from [14] recalled in introduction).

In contrast to the growing number of studies relying on two-stage detection networks to deal with remote sensing objects of small size, the use of one-stage detectors has been less explored. Few studies have been proposed in particular using the YOLO-based frameworks (i.e. including YOLOv2 and YOLOv3) to perform small object detection, such as persons [51], aircrafts [52], ships [53], and building footprints [54] from remote sensing images. This relatively limited interest could be explained by the fact that, although YOLO detectors could be seen as a first choice for real-time applications in computer vision, they provide much lower detection accuracy when compared to region-based detectors, notably in detecting small objects. In [51], UAV-YOLO was proposed to adapt the YOLOv3 to detect small objects from unmanned aerial vehicle (UAV) data. Slight modification from YOLOv3 was done by concatenating two residual blocks of the network backbone having the same size. The authors actually

focused more on training optimization of their dataset with UAV-viewed perspectives. They reported superior performance when compared to YOLOv3 and SSD models. In [52], the authors exploited YOLOv3 without any modification and showed that its performance in computational time is much better than Faster R-CNN or SSD. In [53], the authors compared YOLOv3 and YOLT (You Only Look Twice) [55], which is actually quite similar to YOLOv2, for ship detection. Again, no modification of the original architecture was proposed. In [54], the authors proposed the locally-constrained YOLO (named LOCO) to specifically detect small and dense building footprints. However, LOCO was designed for building detection with rectangular forms and could not be generalized to other objects with various shapes and forms. We argue that YOLO or other one-stage detectors, despite their limited use in remote sensing, remain appealing for real-time applications in remote sensing thanks to their high efficiency. In the sequel, we will present our enhanced one-stage architecture, called YOLO-fine, to achieve the detection of small objects with both high accuracy and efficiency, thus enabling its embedding into real-time operating systems.

3. Methodology

3.1. Inspiration from the YOLO Family

YOLO (You Only Look Once) [10] is an end-to-end deep learning-based detection model that determines the bounding boxes of the objects present in the image and classifies them in a single pass. It does not involve any region proposal phase conversely to two-stage detectors, such as the R-CNN family. The YOLO network first divides the input image into a grid of $S \times S$ non-overlapping cells. For each cell, YOLO predicts three elements: (1) the probability of an object being present; (2) if an object exists in this cell, the coordinates of the box surrounding it; and, (3) the class c the object belongs to and its associated probability. As illustrated in Figure 1a, for each cell, the network predicts B bounding boxes. Each of them contains five parameters (x, y, w, h, sc) , where sc is the objectness confidence score of the box. Subsequently, the network calculates the probabilities of the classes for each cell. If C is the number of classes, the output of YOLOv1 is a tensor of size $(S, S, B \times 5 + C)$.

In the literature, it is usually reported that the first YOLO version (YOLOv1) is faster (i.e. lower computational time), but less accurate than SSD [13], another popular one-stage detector. In 2017, YOLOv2 [11] (also called YOLO9000 at the beginning) was proposed to significantly improve the detection accuracy while making it even faster. Many changes were proposed as compared to the first version. YOLOv2 only uses convolutional layers without fully-connected layers and it introduces the anchor boxes rather than arbitrary boxes present in YOLOv1. These are a set of predefined enclosing boxes of certain height and width to capture the scale and aspect ratio of the objects to be detected. Anchors are usually set based on the size (and aspect ratio) of the objects in the training set. The class probabilities are also calculated for each anchor box and not for each cell as in YOLOv1 (see Figure 1b). Finally, the detection grid is also refined (i.e. 13×13 instead of 7×7 in YOLOv1).

Inspired from the evolution of two-stage detection networks in particular the Feature Pyramid Network [8], YOLOv3 was proposed in 2018 [12] to further improve the detection accuracy, and in particular to be able to find objects of different sizes in the same image. YOLOv3 offers three detection levels instead of only one in the two previous versions, which helps to search for smaller objects (see Figure 1c). Compared to YOLOv2, the following changes were made: (1) predict three box anchors for each cell instead of five in YOLOv2; (2) detect at three different levels with the searching grids $S \times S$, $2S \times 2S$, and $4S \times 4S$; (3) exploit a deeper backbone network (Darknet-53) for feature map extraction. As a result, the number of layers highly increases to 106 as compared to 31 in the two previous versions. Obviously, because YOLOv3 offers a deeper feature extraction network with three-level prediction, it also becomes slower than YOLOv2 (the fastest in the YOLO family). However, because one-stage detectors in general, and YOLO in particular, are characterized by rather lower accuracy than their two-stage counterparts, they remain largely unexplored for detecting small objects in remote sensing, as noted in Section 2.

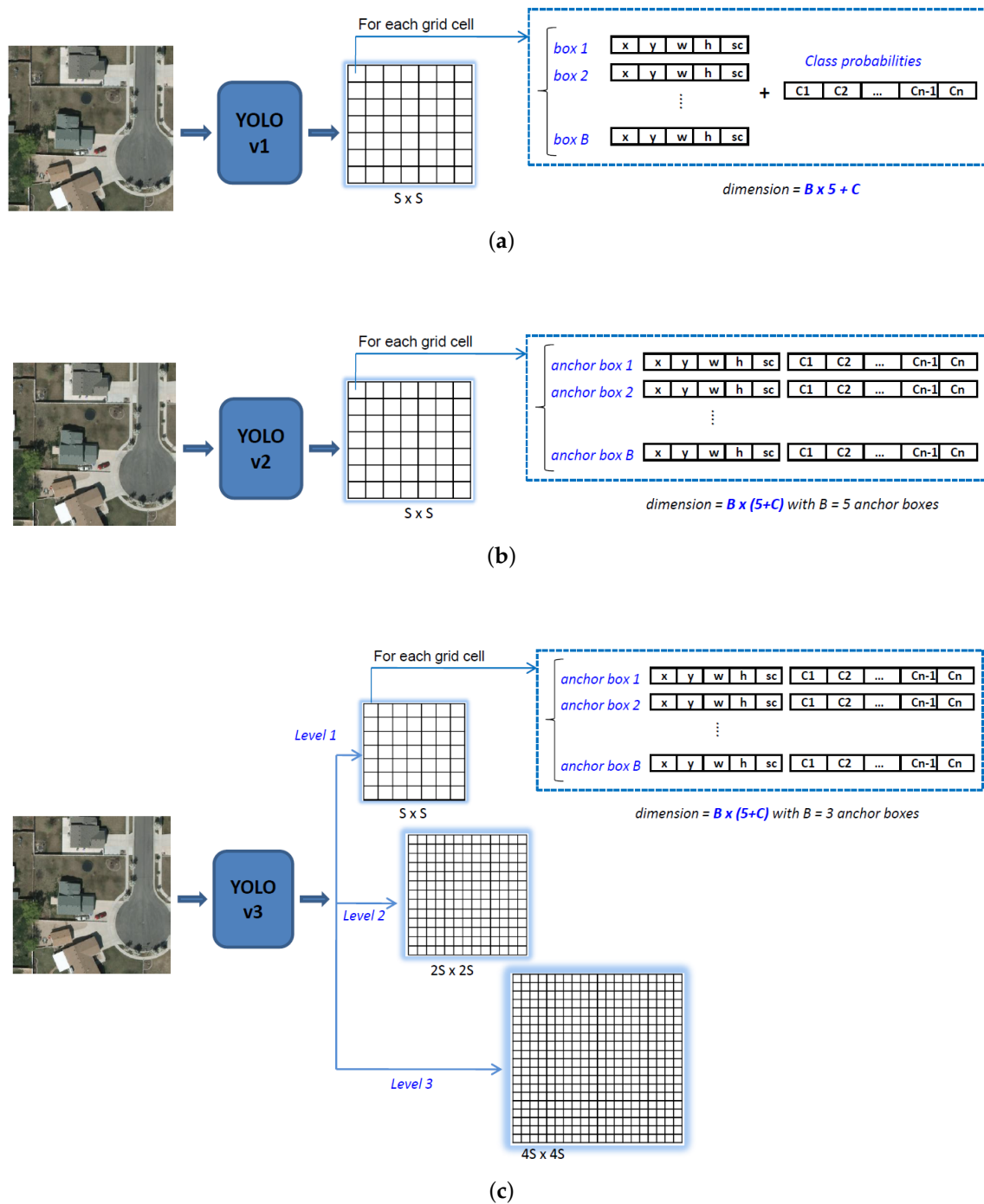


Figure 1. Overview of the state-of-the-art You Only Look Once (YOLO) family for one-stage object detection. (a) In YOLOv1, the output is a tensor of dimension $(S, S, B \times 5 + C)$ with (S, S) the size of the grid, B the number of predicted boxes for each cell and C the number of classes. By default in [10], $S = 7, B = 2$ and $C = 20$ for the PASCAL VOC dataset. For an input image of size 448×448 pixels, the output is a tensor of size $7 \times 7 \times 30$. (b) In YOLOv2, the output is a tensor of dimension $(S, S, B \times (5 + C))$. The difference is that the class probabilities are calculated for each anchor box. By default in [11], $S = 13, B = 5$ anchor boxes and $C = 20$ for the PASCAL VOC dataset. For an input image of size 416×416 pixels, the output is a tensor of size $13 \times 13 \times 125$. (c) In YOLOv3, the output consists of 3 tensors of dimension $(S, S, B \times (5 + C))$, $(2S, 2S, B \times (5 + C))$ and $(4S, 4S, B \times (5 + C))$ which correspond to the 3 detection levels (scales). By default in [12], $S = 13, B = 3$ anchor boxes and $C = 80$ for the COCO dataset. For an input image of size 416×416 pixels, the outputs are three tensors of size $13 \times 13 \times 255$, $26 \times 26 \times 255$ and $52 \times 52 \times 255$.

3.2. Proposed Model: YOLO-Fine

The YOLO-fine network proposed in this paper relies on the structure of YOLOv3 and pursues three objectives: (1) detect small and very small objects (dimension, i.e., height or width lower than 10 pixels) better than any other one-stage detector; (2) be efficient enough to enable prediction in real-time applications; and, (3) be lighter in parameters and weights to facilitate the implementation within an operational context. The proposed network is summarized in Figure 2 and compared to the three reference YOLO versions in Table 1.

Table 1. Comparison of the three YOLO versions and our YOLO-fine model in terms of architecture parameters. We note that the reported input image size of each model was the one originally proposed by the authors. In our experiments, we set all to 512×512 in order to perform fair comparison (cf. Section 4).

Parameters	Models			
	YOLO [10]	YOLOv2 [11]	YOLOv3 [12]	YOLO-Fine (Ours)
Number of layers	31	31	106	68
Multilevel prediction	-	-	3 levels	3 levels
Anchor boxes	-	5	3	3
Input image size	448×448	416×416	320×320	512×512
		544×544	416×416	
		608×608	608×608	
Size of weight file	753 MB	258 MB	237 MB	18 MB

In order to achieve the aforementioned objectives, we have designed YOLO-fine with the following improvements.

1. While YOLOv3 is appealing for operational contexts due to its speed, its performance in detecting small objects remains limited because the input image is divided into three detection grids with subsampling factors of 32, 16, and eight. As a result, YOLOv3 is not able to detect objects measuring less than eight pixels per dimension (height or width) or to discriminate two objects that are closer than eight pixels. The ability of YOLOv3 to detect objects of a wide range of sizes is relevant in numerous computer vision applications, but not in our context of small object detection in remote sensing. The highly sub-sampled layers are then not necessary anymore, thus our first proposition is to remove two coarse detection levels related to large-size objects often observed in natural images. We replace them by two finer detection levels dedicated to lower sub-sampling factors of four and two with skip connections to the corresponding feature maps from high-level layers to those from low-level layers but with very high spatial resolution. The objective is to refine the object search grid in order to be able to recognize and discriminate objects smaller than eight pixels per dimension from the image. Moreover, objects that are relatively close (such as building blocks, cars in a parking, small boats in a harbor, etc.) could be better discriminated.
2. In order to facilitate the storage and implementation within an operational context, we also attempt to remove unnecessary convolutional layers from the backbone Darknet-53. Based on our experiments, we found that the last two convolutional blocks of Darknet-53 include a high number of parameters (due to the high number of filters), but are not useful to characterize small objects due to their high subsampling factors. Removing these two blocks results in both a reduction of the number of parameters in YOLO-fine (making it lighter) and of the feature extracting time required by the detector.
3. The three-level detection (that does not exist in YOLOv1 and YOLOv2) strongly helps YOLOv3 to search and detect objects at different scales in the same image. However, this comes with a computational burden for both YOLOv3 and our YOLO-fine. Moreover, when refining the search

grid, the training and prediction times will also increase from which the compromise between detection accuracy and computing time. The reason that we maintain the three detection levels in YOLO-fine is to make YOLO-fine able to provide good results for various sizes of very small and small objects, as well as provide at least equivalent performance to YOLOv3 on medium or larger objects (i.e., the largest scale of YOLO-fine is the smallest of YOLOv3). To this end, thanks to the efficient behavior of YOLO in general, our YOLO-fine architecture remains able to perform detection in real-time using GPU. We report this accuracy/time trade-off later in the experimental study.

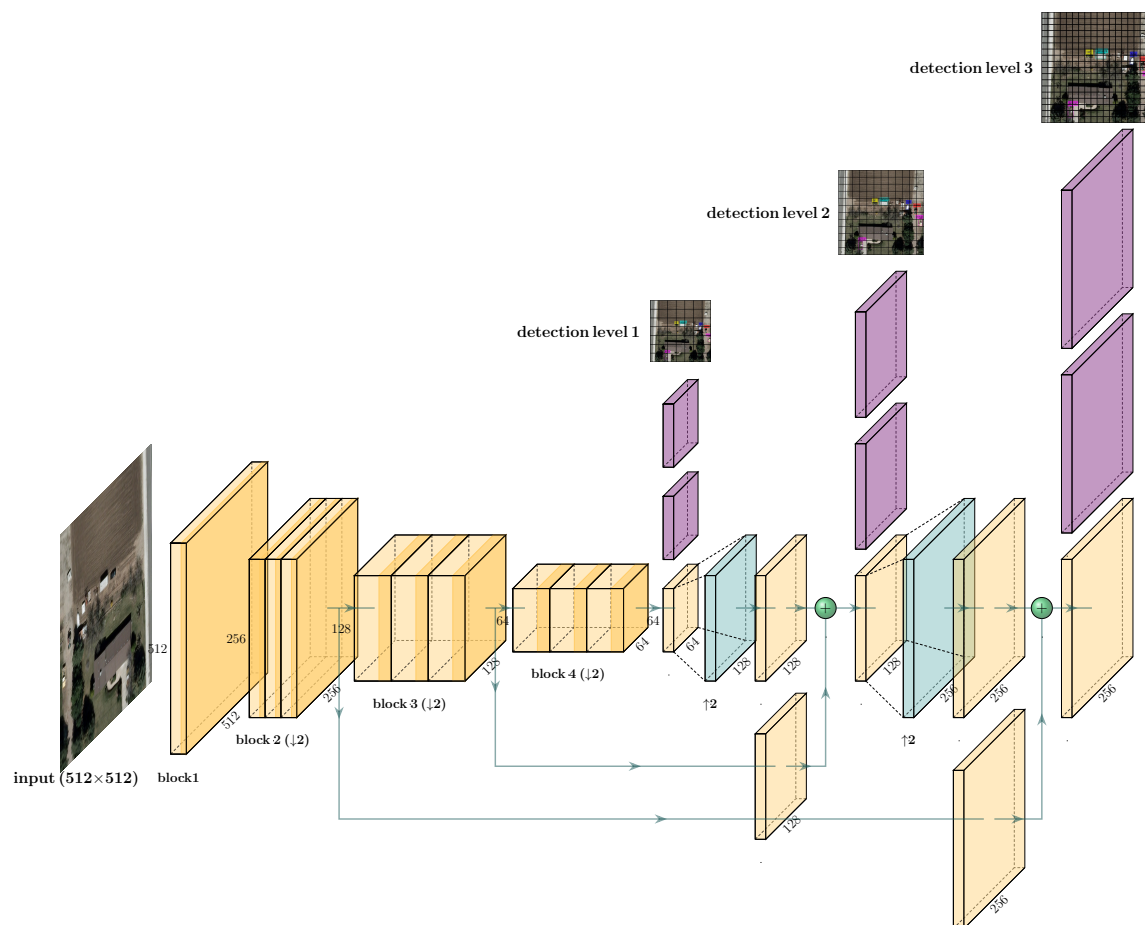


Figure 2. The YOLO-fine framework, including residual blocks (yellow), detection layers (magenta), and upsampling layers (cyan).

Table 1 provides a comparison of YOLO-fine with the three YOLO versions from the literature. We can see that our YOLO-fine model contains significantly fewer layers than YOLOv3 (68 versus 103) due to the removal of two final convolution blocks of Darknet-53 as mentioned above. Although our model is deeper than YOLOv1 and YOLOv2 (31 layers each), it provides multilevel detection, as in YOLOv3. More importantly, the size of the weight file remains the smallest with only 18 MB, as compared to 237 MB for YOLOv3 and even higher for the two older versions. This weight file size depends on the number of convolution filters of each layer more than on the number of layers. The size of the input image is set to 512×512 pixels, since this choice allow for us to work with experimental datasets that will be described in Section 4. This size can be easily set in the network during implementation. It is also important to note that even if YOLO-fine is developed to target the detection of small objects (dimension ≤ 20 pixels in favor) in operational mode, it can still detect medium or large objects. In this case, its accuracy is comparable to YOLOv3 but with a

higher prediction time because of its finer searching grids. We support these observations by a set of experiments reported in the next section.

4. Experimental Study

In this section, we first provide details of the public datasets used in our experiments, namely VEDAI (color and infrared versions) [31], MUNICH [32] and XVIEW [33]. The first two datasets are made of aerial images while the last one contains satellite images. Next, we describe the experimental setup and recall the standard evaluation metrics used in our study. We then show and discuss the detection accuracy obtained by our model as compared against its counterparts, as well as provide a comparison on their performance in terms of storage requirement and computational time. More interestingly, we also set up and study the effect of new backgrounds appearing in validation sets.

4.1. Datasets

4.1.1. VEDAI

The VEDAI aerial image dataset [31] serves as a reference public source for multiple small-vehicle detection under various environments. The objects in VEDAI, in addition to being small, present different variabilities, such as multiple orientations, changes in lighting, shading, and occlusions. Additionally, different types of background can be found in this database, including urban, peri-urban, and rural areas, or even more varied environments (desert, forest, etc.). Each image in VEDAI is available in both color and infrared versions. The spatial resolutions are 12.5 cm for the 1024×1024 images and 25 cm for the 512×512 images, while object dimension varies between 16 and 40 pixels (for the 12.5 cm version) and between eight and 20 pixels (for the 25 cm version). In our experiments, we exploit both 25-cm and 12.5-cm versions (called VEDAI512 and VEDAI1024 in the sequel), and also both color and infrared versions to study the performance of our proposed model. Note that, for the 12.5 cm version (VEDAI1024), the images were cut into patches of 512×512 pixels to fit the network's input size. The original dataset contains 3 757 objects belonging to nine different classes: car, truck, pickup, tractor, camper, ship, van, plane, other. However, since there are only few objects of the class “plane”, we decided to merge them with the class “other”. An illustrative image is shown in Figure 3.



Figure 3. Illustrations of the VEDAI dataset [31] in color version (**top**) and infrared version (**bottom**).

A precise experimental protocol by cross-validation is also provided by the authors of [31], ensuring that the experimental results obtained by different models can be properly replicated and compared. Indeed, the dataset that contains 1250 images is divided into training and validation sets using 10-fold cross-validation (it is divided into 10 subsets and at each experiment, nine are used for training, and the rest is used for validation). We also adopted this cross-validation protocol within our experiments.

4.1.2. MUNICH

We also consider the DLR 3K Munich Vehicle dataset [32] (called MUNICH in this section). Similarly to VEDAI, MUNICH offers a large number of small vehicles and a large variability of backgrounds. The dataset contains 20 large images of size 5616×3744 pixels taken by a 1000 m-high RGB sensor with a spatial resolution of 13 cm. Two classes of vehicles are considered: 9300 cars and 160 trucks. MUNICH is particularly interesting to evaluate detectors in a context where the number of objects of different categories is largely unbalanced due to its high class imbalance property.

In this work, we first cut the large images into patches of 1024×1024 pixels, then resize them into 512×512 pixels (which corresponds to a spatial resolution of 26 cm), so that the vehicles would have smaller sizes (from 8 to 20 pixels). Our dataset finally contains 1226 training images and 340 validation images. Figure 4 shows an illustrative image.



Figure 4. Example of an image crop from the MUNICH data set [32].

4.1.3. XVIEW

The third and last dataset considered in our study is XVIEW [33]. It was collected from WorldView-3 satellites at 30-cm spatial resolution (ground sample distance). A total number of 60 classes are available, but since we focus here on small objects, we gather 19 classes of vehicles, including {17, 18, 19, 20, 21, 23, 24, 26, 27, 28, 32, 41, 60, 62, 63, 64, 65, 66, 91} (these numbers correspond to the initial classes from the original XVIEW data) to create only one vehicle class. Our purpose is not to achieve state-of-the-art detection rate on the XVIEW dataset, but to experiment and validate the capacity of YOLO-fine to detect vehicles from such high resolution satellite images. This single-class dataset presents the highest number of training and validation images with a total of 35k vehicles whose size varies from six to 15 pixels. Three sample images are given in Figure 5.

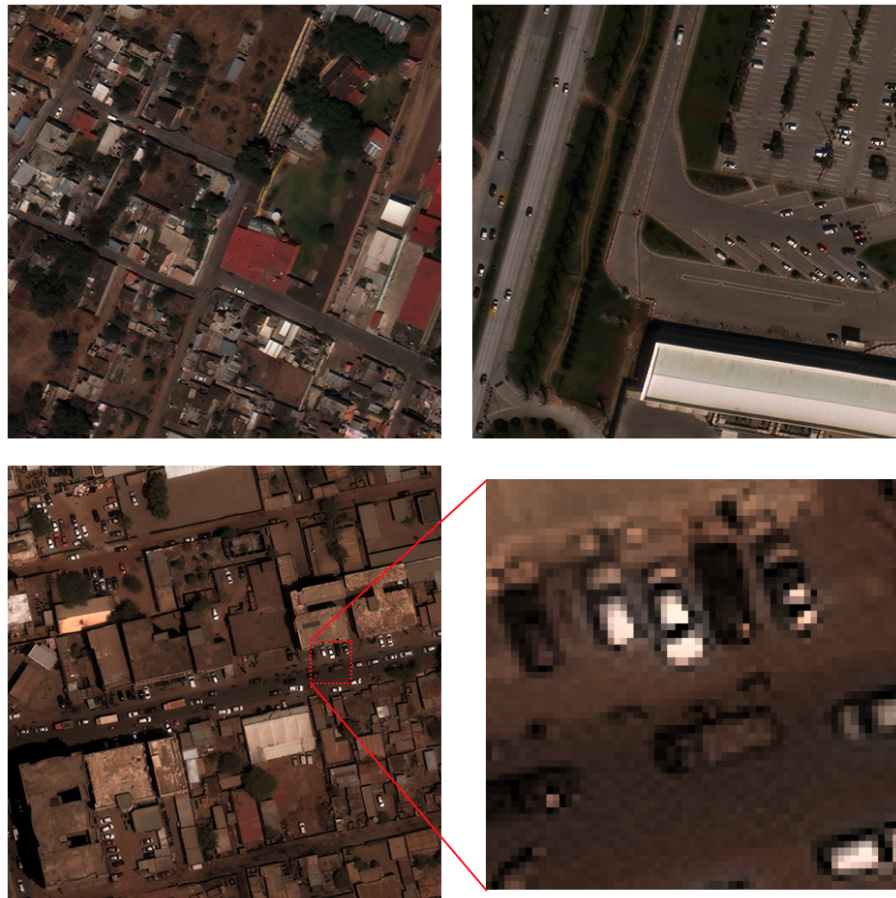


Figure 5. Examples of 30-cm resolution images in the XVIEW data [33].

Table 2 summarizes the datasets used in our experiments.

Table 2. Experimental datasets used in our study.

Parameters	Datasets			
	VEDAI1024	VEDAI512	MUNICH	XVIEW
Number of images	1200	1200	1566	7.4 k
<i>in the train set</i>	1089	1089	1226	5.5 k
<i>in the validation set</i>	121	121	340	1.9 k
Number of classes	8	8	2	1
Number of objects	3757	3757	9460	35 k
Spatial resolution	12.5 cm	25 cm	26 cm	30 cm
Object size (pixels)	16 → 40	8 → 20	8 → 20	6 → 15

4.2. Experimental Setup and Evaluation Criteria

Experiments were conducted on our three datasets to study the performance of the proposed YOLO-fine as compared to several state-of-the-art object detection frameworks. We considered several detection models from the YOLO family (<https://github.com/AlexeyAB/darknet>), including YOLOv2 [11], YOLOv3 [12], YOLOv3-tiny (which is a fast and light version of YOLOv3) and YOLOv3-spp (which is YOLOv3 with spatial pyramid pooling operator [56]). We also investigated other one-stage and two-stage methods, such as SSD (<https://github.com/lufficc/SSD>) [13,57], Faster R-CNN (<https://github.com/facebookresearch/maskrcnn-benchmark>) [6,58] (both with

VGG16 backbone), RetinaNet (<https://github.com/yhenon/pytorch-retinanet>) [15] (with ResNet-50 backbone), and EfficientDet (<https://github.com/zylo117/Yet-Another-EfficientDet-Pytorch>) [18] to enrich our comparative study. Although different implementations were adopted (mentioned in the provided links), equivalent parameters were set in order to perform a fair comparison. We retained the standard parameter setting of each framework and adapted the anchor sizes with regard to the training set of each dataset. The input size to all networks was fixed to 512×512 and batch size was set to 64. All of the detectors were trained with a learning rate of 0.001 with a momentum of 0.9 during a fixed number of 200 epochs from which the best detection result was reported for each detector.

The following standard criteria are exploited to quantitatively evaluate and compare the detection accuracy:

- Intersection over Union (IoU) is one of the most widely used criteria in the literature to evaluate the object detection task. This criterion measures the overlapping ratio between the detected box and the ground truth box. IoU varies between 0 (no overlap) and 1 (total overlap) and decides if the predicted box is an object or not (according to a threshold to be set). Within our specific case of small object detection, the IoU threshold should be set very low. In our experiments, this threshold was set to 0.1 for all of the detectors.
- Precision, Recall, F1-score, and mean average precision (mAP) are widely used criteria to evaluate the performance of a model. We remind they are computed, as follows:

$$\text{Recall} = \frac{\text{number of true detections}}{\text{number of existing objects}} = \frac{TP}{TP + FN} \quad (1)$$

$$\text{Precision} = \frac{\text{number of true detections}}{\text{number of detected objects}} = \frac{TP}{TP + FP} \quad (2)$$

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (3)$$

with TP, FP, and FN denoting True Positive, False Positive, and False Negative, respectively.

- Precision/Recall curve plots the precision in function of the recall rate. It is a decreasing curve. When we lower the threshold of the detector (the confidence index), the recall increases (as we will detect more objects), but the precision decreases (as our detections are more likely to be false alarms), and vice versa. The visualization of the precision/recall curve gives us a global vision of the compromise between precision and recall.

4.3. Detection Performance

The detection performance is now assessed on each datasets (VEDAI, MUNICH, and XVIEW). Note that for VEDAI, we focus on VEDAI512 dataset (with 25-cm resolution) as it is more relevant to our paper scope (objects size is smaller (from eight to 20 pixels)). Nevertheless, detection results on VEDAI1024 dataset are provided in order to show that YOLO-fine provides slightly better or at least equivalent accuracy for detecting medium and larger objects. Finally, the computational time of YOLO-fine compared to other YOLO versions are reported, performed on three different GPUs.

4.3.1. VEDAI512 Color and Infrared

The detection results on VEDAI512 obtained with different detection models compared to the proposed YOLO-fine are shown in Tables 3 and 4 for the color and infrared versions, respectively. We remind that the object size varies here from eight to 20 pixels (Table 2). As observed in Table 3a, our YOLO-fine reached $mAP = 68.18\%$, showing an increase of 6.09% when compared to the second-best counterpart, which is YOLOv3 ($mAP = 62.09\%$). The YOLOv3-spp obtained equivalent results as YOLOv3 with $mAP = 61.57\%$, while YOLOv3-tiny provides the poorest results ($mAP = 44.97\%$) among the YOLO-based models. This is not surprising, since a YOLOv3-tiny model was designed to perform very fast detection for tracking systems. We report its performance in

computational time later within the results on XVIEW. Other detectors, including SSD, EfficientDet, Faster R-CNN, and RetinaNet, provided lower detection accuracy, since they were not designed to detect small objects. Even their anchor size was adapted according to the training set, since they could not naturally yield good detection results for small objects without any modification and adaptation of network architectures. However, we will observe later their good performance in detecting medium and larger objects in VEDAI1024. Back to Table 3a, we observe the best results for 4/8 classes (pickup, tractor, boat and van) were achieved by the proposed YOLO-fine. It is important to note that very good results were obtained for the three classes of small vehicles: car (76.77%, second-best after YOLOv3-spp), pickup (74.35%, best), and tractor (78.12%, best), especially for the tractor class which is often difficult to be detected with all other detectors (the second-best result was only 67.46%, thus more than 10% lower than YOLO-fine model). From Table 3b, YOLO-fine globally produced a good number of true positives and not many false positives/negatives, thus achieved a good balance of precision (0.67) and recall (0.70) rates, as well as the highest F1-score (0.69). This behavior is confirmed in Figure 6a, which shows the precision/recall curves obtained by all detectors for the color VEDAI512, when the detector confidence threshold ranges from 0 and 1. We observe that the YOLO-fine curve (red) remains higher than the others, thus confirming the superior performance of YOLO-fine w.r.t. existing methods, regardless of the detection threshold.

Table 3. Detection results on VEDAI512 (color version).

(a) Classwise accuracy and mean average precision. Best results in bold.									
Model	Car	Truck	Pickup	Tractor	Camping	Boat	Van	Other	mAP
SSD	73.92	49.70	67.31	52.03	57.00	41.52	53.36	39.99	54.35
EfficientDet(D0)	64.45	38.63	51.57	27.91	57.14	17.19	15.34	27.81	37.50
EfficientDet(D1)	69.08	61.20	65.74	47.18	69.08	33.65	16.55	36.67	51.36
Faster R-CNN	72.91	48.98	66.38	55.61	61.61	30.29	35.95	35.34	50.88
RetinaNet(50)	60.61	35.98	63.82	19.36	65.16	21.55	46.60	09.51	40.33
YOLOv2	72.65	50.24	62.83	67.46	56.53	42.33	60.95	56.73	58.72
YOLOv3	75.22	73.53	65.69	57.02	59.27	47.20	71.55	47.20	62.09
YOLOv3-tiny	64.11	41.21	48.38	30.04	42.37	24.64	68.25	40.77	44.97
YOLOv3-spp	79.03	68.57	72.30	61.67	63.41	44.26	60.68	42.43	61.57
YOLO-fine	76.77	63.45	74.35	78.12	64.74	70.04	77.91	45.04	68.18
(b) Overall evaluation metrics. Best result of F1-score in bold.									
Model	TP	FP	FN	Precision	Recall	F1-Score			
SSD	188	81	178	0.69	0.51	0.59			
EfficientDet(D0)	198	143	168	0.58	0.54	0.56			
EfficientDet(D1)	250	164	116	0.60	0.68	0.64			
Faster R-CNN	260	277	106	0.48	0.71	0.57			
RetinaNet(50)	216	236	150	0.47	0.59	0.52			
YOLOv2	209	162	157	0.56	0.57	0.57			
YOLOv3	246	139	120	0.64	0.67	0.66			
YOLOv3-tiny	163	151	203	0.52	0.45	0.48			
YOLOv3-spp	244	103	122	0.69	0.67	0.68			
YOLO-fine	258	133	108	0.67	0.70	0.69			

From Table 4, similar results were obtained on the infrared VEDAI512. Let us note that this specific VEDAI infrared version has rarely been investigated in the literature. We are interested here to assess the performance of our model on infrared images, which are of high importance in many remote sensing scenarios. Our first observation is that, for this dataset, only exploiting the

infrared band also provided equivalent results to the use of color bands. This remark shows that infrared VEDAI images also contain rich information for discriminating different vehicle classes, as good as the three color bands. Back to the detection results from Table 4a, YOLO-fine achieved the highest mAP of 68.83%, i.e., a gain of 5.27% when compared to the second-best result of YOLOv3-spp (mAP = 63.56%). Again, YOLOv3 yielded equivalent result to YOLOv3-spp, while SSD, Faster R-CNN, and EfficientDet achieved equivalent results to YOLOv2 (mAP = 50.36%), being much lower than the proposed YOLO-fine. In terms of classwise accuracy, the proposed model again achieved the best results for pickup (70.65%) and boat (60.84%), as the previous case of color VEDAI512. Good results were yielded as well for other small vehicle classes, which are car (79.68%) and pickup (74.49%), even if they were not the best. Finally, the results from Table 4b and Figure 6b also confirm good behavior of recall/precision balance provided by YOLO-fine. The best F1-score of 0.71 was achieved by our model, significantly higher than the other detectors, such as EfficientDet(D1) and YOLOv3 (both 0.64), YOLOv3-spp (0.65), or SSD (0.63).

Table 4. Detection results on VEDAI512 (infrared version).

(a) Classwise accuracy and mean average precision. Best results in bold.									
Model	Car	Truck	Pickup	Tractor	Camping	Boat	Van	Other	mAP
SSD	78.72	51.27	65.50	55.98	59.25	42.99	61.21	36.68	56.45
EfficientDet(D0)	64.48	44.08	51.57	27.72	53.32	18.88	4.94	24.59	36.20
EfficientDet(D1)	78.72	51.43	63.86	43.11	68.79	17.80	42.99	27.92	49.33
Faster R-CNN	75.24	48.87	62.95	35.55	60.14	31.07	39.20	13.11	45.77
RetinaNet(50)	73.58	44.50	65.98	21.65	64.37	32.93	28.04	22.20	44.16
YOLOv2	73.47	57.18	68.67	46.25	63.70	34.71	36.55	22.35	50.36
YOLOv3	85.58	56.46	79.12	54.77	55.79	55.02	75.96	38.49	62.65
YOLOv3-tiny	66.63	34.36	56.28	37.42	43.70	31.59	75.52	31.30	46.72
YOLOv3-spp	84.23	71.54	73.12	55.92	65.08	50.07	70.00	38.52	63.56
YOLO-fine	79.68	80.97	74.49	70.65	77.09	60.84	63.56	37.33	68.83
(b) Overall evaluation metrics. Best result of F1-score in bold.									
Model	TP	FP	FN	Precision	Recall	F1-Score			
SSD	196	59	170	0.76	0.53	0.63			
EfficientDet(D0)	188	139	178	0.57	0.51	0.54			
EfficientDet(D1)	233	125	133	0.65	0.63	0.64			
Faster R-CNN	247	314	119	0.44	0.67	0.53			
RetinaNet(50)	248	180	118	0.58	0.67	0.62			
YOLOv2	203	84	163	0.71	0.55	0.62			
YOLOv3	241	146	125	0.62	0.66	0.64			
YOLOv3-tiny	176	181	190	0.49	0.48	0.49			
YOLOv3-spp	239	129	127	0.65	0.65	0.65			
YOLO-fine	262	112	104	0.70	0.72	0.71			

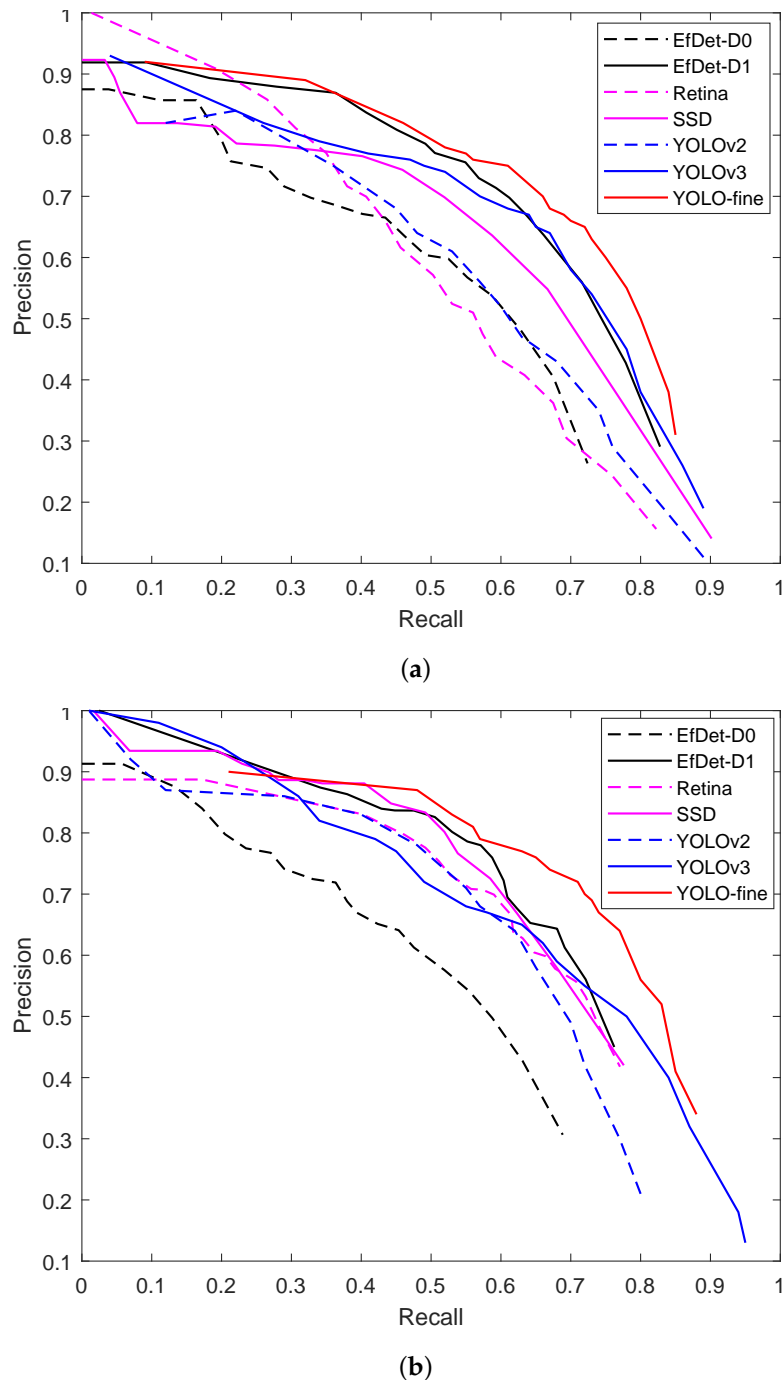


Figure 6. Comparison of recall/precision curves for the 25-cm VEDAI512 dataset. (a) color version; (b) infrared version.

4.3.2. VEDAI1024 Color and Infrared

Table 5 provides the detection results on VEDAI1024 (12.5-cm resolution). This dataset is a high-resolution version of the previous one which was also proposed by the authors in [31]. We remind the size of the vehicles in VEDAI1024 ranges from 16 to 40 pixels, thus double the size of those in the previous VEDAI512. We consider the objects in VEDAI1024 are of medium size and not small size. By conducting experiments on this dataset, we show that, although YOLO-fine is designed to mainly detect small objects, it could be able to provide at least equivalent detection accuracy as the original YOLOv3 for other object sizes. Let us observe Table 5, which shows the comparative detection results of different detectors yielded on the color and infrared versions of VEDAI1024. Globally,

the detection results of all detectors are better than those on VEDAI512 version. In particular, we now observe good results from reference detectors, including SSD, Faster R-CNN, RetinaNet, and, in particular, EfficientDet, which was not the case for VEDAI512, as previously reported. This could be explained by the fact that objects in VEDAI1024 are larger so that these state-of-the-art detectors could naturally provide their good detection capacity. For our YOLO-fine model, while providing significant improvement w.r.t. the previous VEDAI512 dataset (i.e., gain of 6.09% and 5.07% in mAP on the color and infrared versions, respectively), it now provides slightly better performance than other reference models on VEDAI1024. For the color version (left part of Table 5), it achieved a mAP = 76.0% with 0.96% better than the second-best YOLOv3-spp and 1.99% better than the third-best EfficientDet(D1). For the infrared version (right part of the table), it achieved mAP = 75.17%, again slightly better than YOLOv3-spp (gain of 1.47%). In terms of F1-score results, EfficientDet(D1) slightly outperformed YOLO-fine in both cases (i.e., 0.79 compared to 0.78 on color version, and 0.78 as compared to 0.76 on infrared version). However, these differences in F1-score are not significant and given the fact that YOLO-fine yielded better mAP in both cases (1.99% for color version and 3.94% for infrared version), we consider that YOLO-fine provide at least equivalent detection performance as EfficientDet in detecting medium and larger objects from VEDAI1024.

Table 5. Detection results on VEDAI1024 color and infrared versions. The best results of F1-score and mean average precision in bold.

Model	Color Version				Infrared Version			
	Precision	Recall	F1-Score	mAP	Precision	Recall	F1-Score	mAP
SSD	0.76	0.69	0.72	70.91	0.76	0.70	0.73	69.83
EfficientDet(D0)	0.75	0.89	0.77	70.68	0.77	0.75	0.76	69.79
EfficientDet(D1)	0.79	0.80	0.79	74.01	0.78	0.78	0.78	71.23
Faster R-CNN	0.64	0.81	0.71	70.22	0.47	0.69	0.56	56.41
RetinaNet(50)	0.59	0.75	0.66	61.47	0.60	0.80	0.69	67.59
YOLOv2	0.66	0.73	0.69	65.72	0.68	0.67	0.68	63.16
YOLOv3	0.74	0.79	0.76	73.11	0.8	0.72	0.76	71.01
YOLOv3-tiny	0.65	0.58	0.61	55.55	0.66	0.58	0.62	55.36
YOLOv3-spp	0.79	0.76	0.77	75.04	0.78	0.74	0.76	73.70
YOLO-fine	0.79	0.77	0.78	76.00	0.80	0.74	0.76	75.17

Finally, Figures 7 and 8 illustrate some detection results for the sake of qualitative assessment to enrich our experiments on VEDAI data. We can observe in Figure 7 that the YOLO-fine model often provides the best performance in terms of true positives for the five sample images (and for the entire validation set as previously observed in Table 5). For instance, a perfect detection has been achieved by YOLO-fine for the second image containing two “vans”, three “trucks”, and one “car”. YOLOv3 missed two “trucks” and yielded two false detections on this image, while YOLOv2 confused a “truck” as a “van” and also gave a false positive of class “car”. Within the whole validation set, we observe that the two classes “car” and “pickup” were often confused (for example, in the fifth image where YOLO-fine detected two “cars” as “pickups”). It is also noted that, sometimes, the false alarms produced by the network correspond to objects that visually look very similar to vehicles as shown in the first image with an object at the bottom that looks like an old vehicle; or in the third image with an object visually similar to a boat. Qualitative remarks about the infrared base from Figure 8 are similar to those made for the color version. We observe that the YOLOv3 and YOLO-fine models produce few false positives but more false negatives in infrared images than in color images. We can see that YOLO-fine missed three “tractors” on the first image, one “camping car” on the third image and two objects of class “other” on the fourth image. These false negatives are to be avoided in some applications. The false negative/false positive (FN/FP) trade-off is similar to the recall/precision trade-off observed with the

recall/precision curves. Finally, we note that all detectors, especially our YOLO-fine model, are able to deal with infrared images. Indeed, the loss of accuracy remains insignificant if the objects are well characterized by their shape rather than their color or spectral signatures.

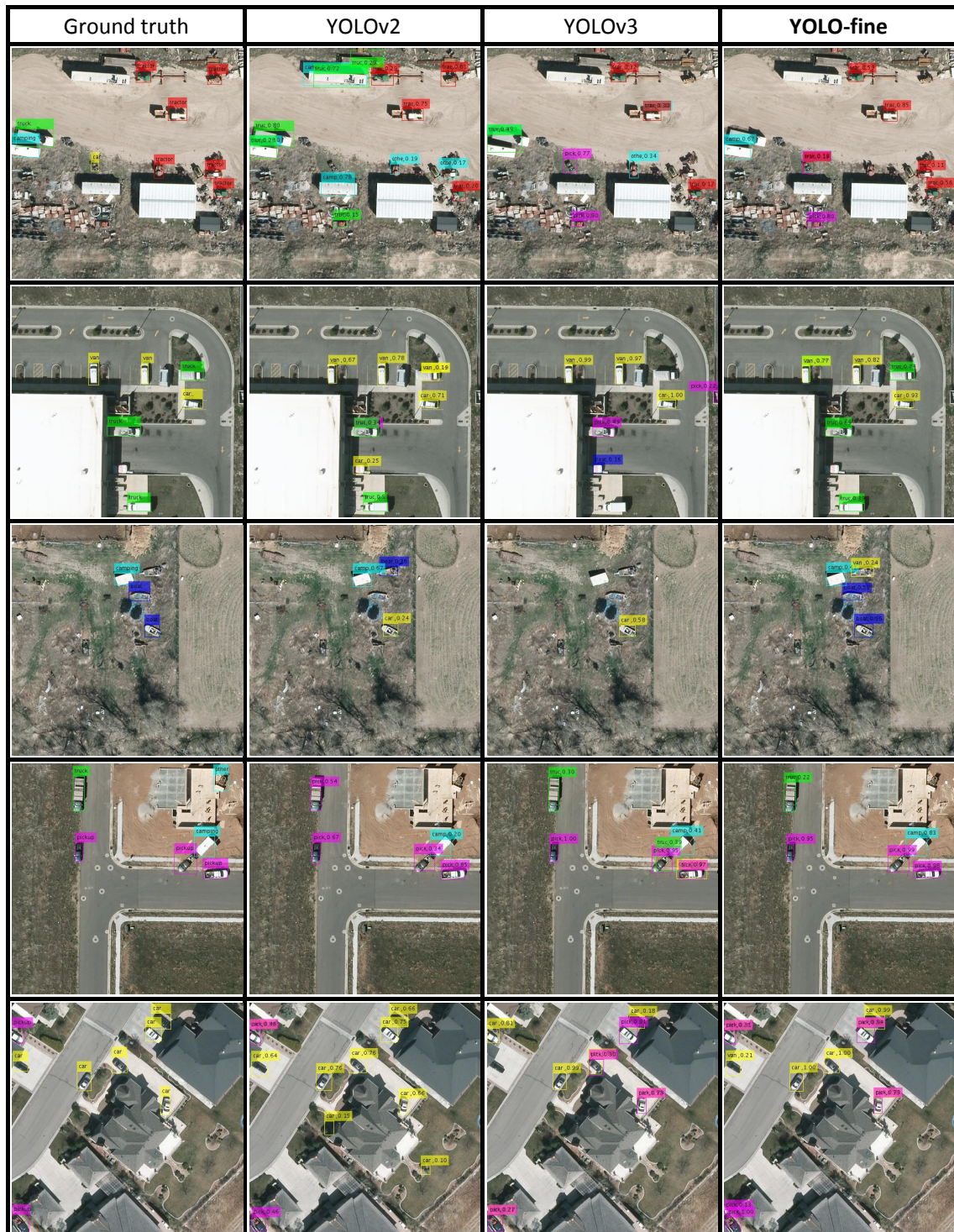


Figure 7. Illustration of detection results on VEDAI1024 (color version).

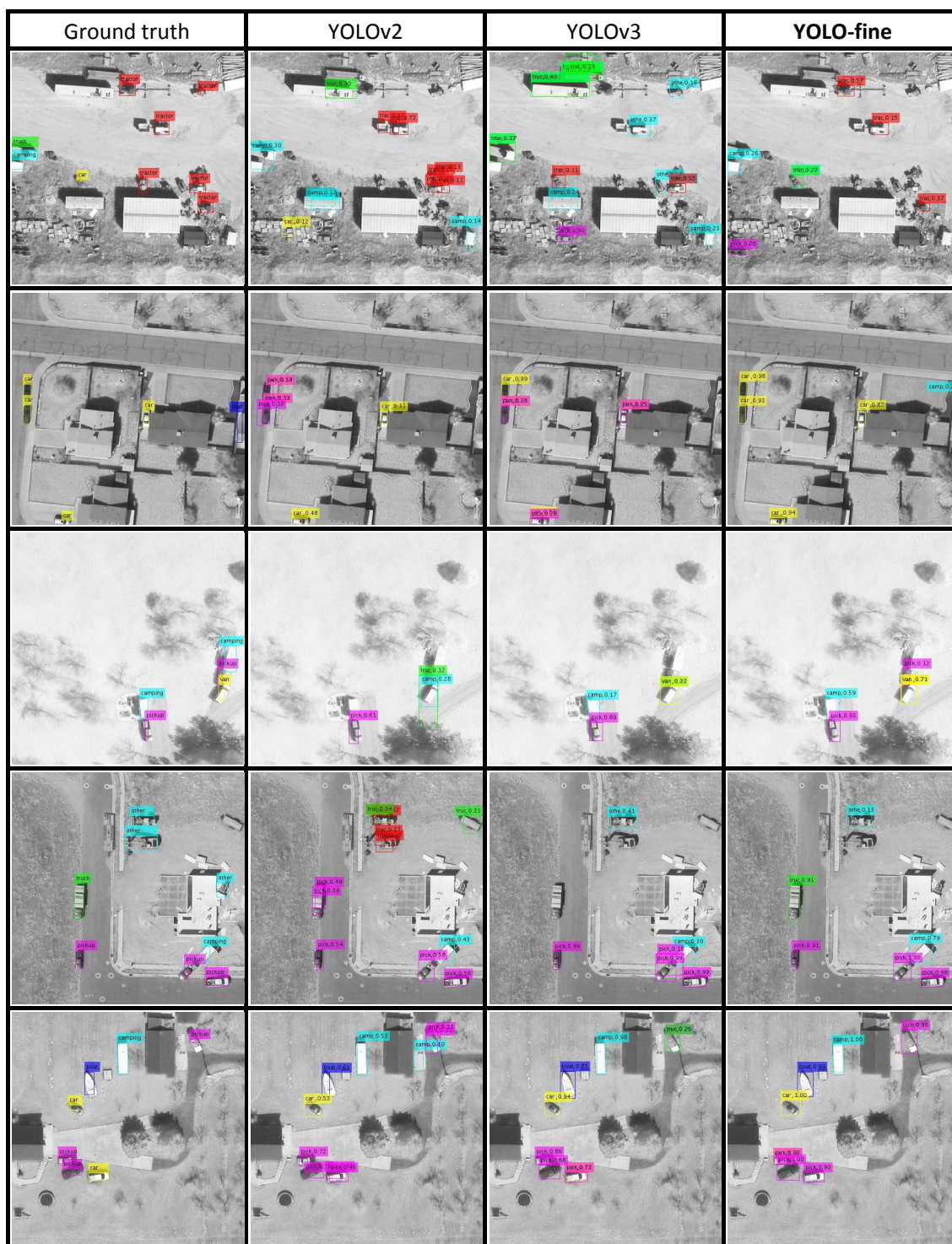


Figure 8. Illustration of detection results on VEDAI1024 (infrared version).

4.3.3. MUNICH

Table 6 provides the detection results of YOLO-fine compared to reference detectors for the MUNICH dataset. A very high detection rate was achieved by our YOLO-fine ($mAP = 99.69\%$) with an increase of 1.82% over YOLOv3 (second-best), 2.12% over SSD (third-best) and 19.93% over YOLOv2 (lowest mAP). In terms of F1-score, both YOLO-fine and SSD provided the highest value of 0.98. Compared to VEDAI512 (with similar spatial resolution and object size), MUNICH appears less

complicated, because: 1/ it contains only two very different classes, namely “car” and “truck” classes (as compared to the eight classes of vehicles that look similar in VEDAI images); 2/ the number of objects in MUNICH is higher (9460 compared to 3757 in VEDAI); and, 3/ the level of background variation is low (the images were all acquired from a semi-urban area of the city of Munich). Therefore, achieving good performance on MUNICH is a requirement to demonstrate the relevance of our proposed model. Besides, the detection rate is higher for the “car” class (99.93%) than for the “truck” class (99.45%), since there are many more cars than trucks in the training set (and also in the validation one). However, the difference is very small. This observation is confirmed by Figure 9, where some detection results are provided. We can observe that: (i) YOLO-fine detects cars and trucks with very high confidence indices (close to 1); (ii) cars are well detected even if they are on the image borders (see yellow rectangles in the third image in the third row); (iii) the image ground truth is sometimes wrongly annotated by ignoring the vehicles under the shadow, but the YOLO detectors manage to detect them (see the yellow rectangles in the second image from the second row); and, (iv) vehicles in a parking lot are also well detected and discriminated even though they are very close to each other (see the fifth image in the fifth row). In summary, the studied MUNICH dataset represents an easy small object detection task (two distinct classes from simple background) and the detection performance achieved by the proposed YOLO-fine was confirmed.

4.3.4. XVIEW

The last dataset used in our experiments is XVIEW and the detection results measured on this dataset are given in Table 7. Some illustrative results are also given in Figure 10 in order to show a good quality of detection results yielded by YOLO-fine, close to the ground truth boxes. As observed with the two previous datasets, YOLO-fine again provided best detection mAP = 84.34%, with a gain of 1.83% over EfficientDet, 5.41% over YOLOv3, and even 16.25% over SDD and 27.33% over Faster R-CNN. We note that objects in XVIEW are quite small when compared to those in MUNICH or VEDAI (spatial resolution of 30 cm compared to 25 cm in MUNICH and VEDAI512). Thus, the mAP = 84.34% achieved by YOLO-fine could be considered as successful for real-time prediction using a one-stage approach. One may wonder why this result is higher than those of VEDAI512. The answer is that for XVIEW, we only consider one class of vehicle (which was, in fact, merged by 19 vehicle classes from the original dataset as described in Section 4.1), hence the detection task becomes simpler than in the VEDAI dataset. To this end, the proposed YOLO-fine is thus able to perform small object detection from both aerial and satellite remote sensing data.

Table 6. Detection results on MUNICH. Best results of mAP and F1-score in bold.

Model	Car	Truck	mAP	Precision	Recall	F1-Score
SSD	99.31	95.83	97.57	0.98	0.97	0.98
EfficientDet(D0)	92.72	89.09	90.90	0.98	0.82	0.90
EfficientDet(D1)	97.65	94.78	96.22	0.98	0.82	0.90
Faster-RCNN	90.96	94.92	92.94	0.96	0.92	0.94
RetinaNet(50)	80.89	78.81	79.85	0.76	0.74	0.75
YOLOv2	70.23	89.28	79.76	0.90	0.59	0.71
YOLOv3	99.15	96.59	97.87	0.84	0.99	0.91
YOLOv3-tiny	92.28	93.57	92.92	0.81	0.92	0.86
YOLOv3-spp	96.39	94.77	95.58	0.90	0.88	0.89
YOLO-fine	99.93	99.45	99.69	0.96	1.00	0.98

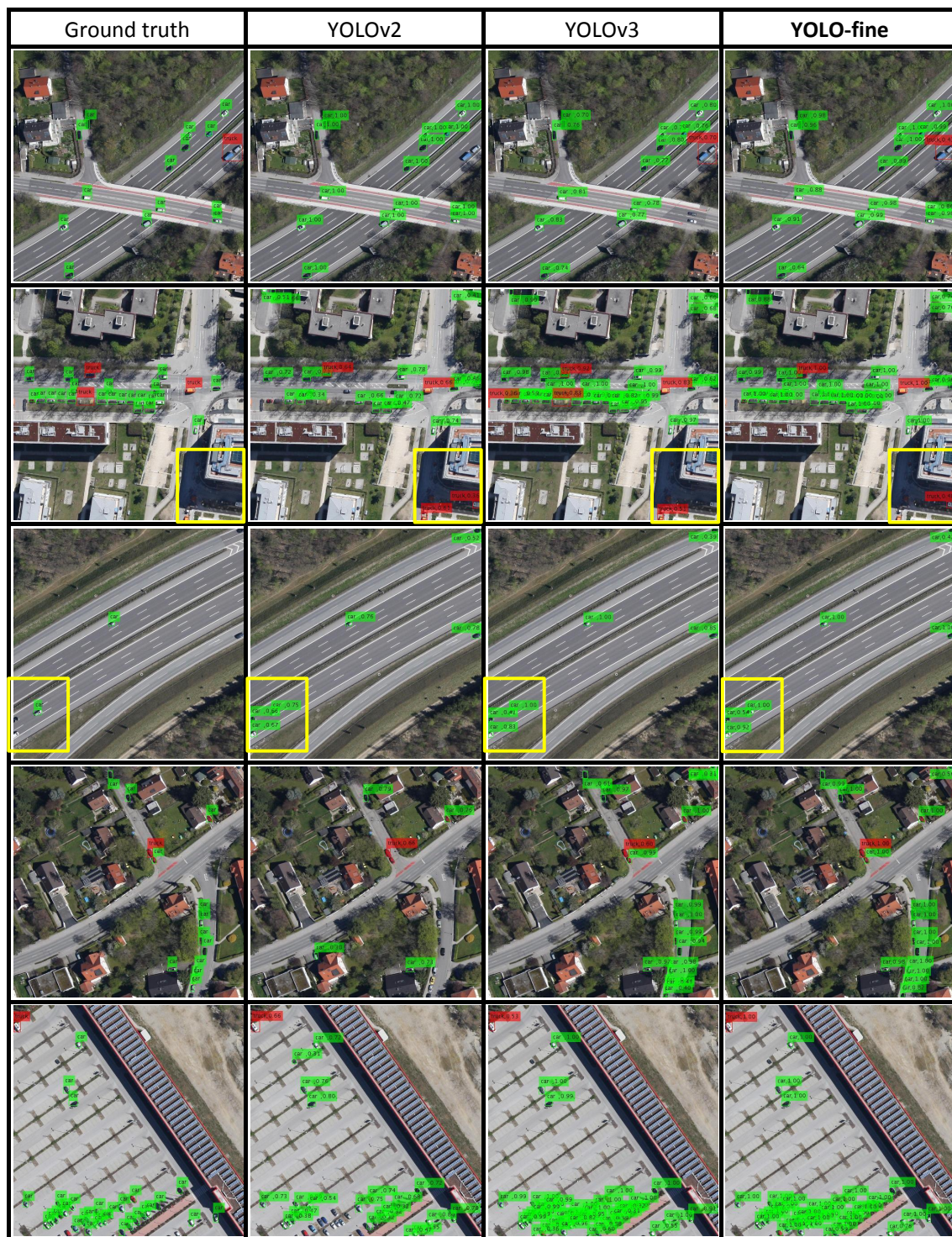
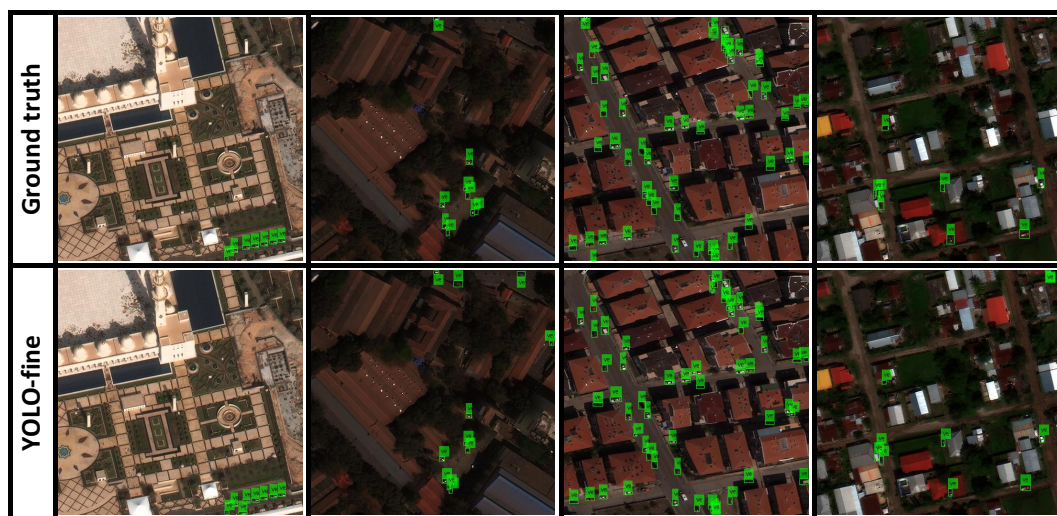


Figure 9. Illustration of detection results on MUNICH.

Table 7. Detection results on XVIEW. Best results of F1-score and mAP in bold.

Model	Precision	Recall	F1-Score	mAP
SSD	0.80	0.62	0.70	68.09
EfficientDet(D0)	0.84	0.78	0.81	82.45
EfficientDet(D1)	0.80	0.75	0.78	82.51
Faster R-CNN	0.50	0.72	0.59	57.01
RetinaNet(50)	0.70	0.21	0.33	34.84
YOLOv2	0.75	0.41	0.53	47.68
YOLOv3	0.77	0.74	0.75	78.93
YOLOv3-tiny	0.66	0.61	0.64	62.03
YOLOv3-spp	0.81	0.73	0.77	77.34
YOLO-fine	0.87	0.72	0.79	84.34

**Figure 10.** Illustration of detection results on XVIEW.

Finally, using the XVIEW data, we conducted a comparative study in terms of computational performance. We note that only detector models from the YOLO-family were investigated, since the other detectors were implemented using different frameworks, hence hindering a fair comparison related to computational time. In Table 8, we report the comparison of YOLO-fine against YOLOv2, YOLOv3, YOLOv3-tiny, and YOLOv3-spp in terms of network weights (i.e., required disk space for storage), BFLOPS (billion floating point operations), and the prediction time in millisecond/image as well as in frame per second (FPS), tested on three different NVIDIA GPUs including Titan X, RTX 2080ti and Tesla V100. According to the table, YOLO-fine has the smallest weight size with only 18.5 MB compared to the original YOLOv3 with 246.3 MB. The heaviest weight size comes from YOLOv3-spp with 250.5 MB, while YOLOv2 also requires 202.4 MB of disk space. In terms of BFLOPS (the lower the better), it is also reduced in YOLO-fine with 63.16 as compared to 98.92 of YOLOv3. In terms of prediction time (average prediction time computed from 1932 XVIEW images from the validation set), YOLO-fine is the slowest (due to the accuracy/computation time trade-off), but, and as already mentioned, it can still achieve real-time performance with 34 FPS, 55 FPS, and 84 FPS on GPU Titan X, RTX 2080ti and Tesla V100, respectively. One may wonder why YOLO-fine, with several removed layers and significantly lighter weight than YOLOv3, is still slower (34 FPS compared to 37 FPS on Titan X). This is due to the fact that we refine the detection grid to look for small and very small objects. However, this slower speed is not significant w.r.t. the great gain in accuracy observed against YOLOv3 shown in previous results with our three datasets. We can also note that YOLOv2 remains the fastest detector among the YOLO family (as observed in the literature) with 57 FPS on Titan X, 84 FPS on RTX 2080i, and 97 FPS on Tesla V100. Subsequently, the YOLOv3-tiny is a very light version of

YOLOv3 with small weight size (34.7 MB compared to 246.4 MB), only 8.25 BFLOPS when compared to 98.92 BFLOPS and very fast prediction time (215 FPS on Tesla V100). However, both YOLOv2 and YOLOv3-tiny provided significantly inferior detection accuracy than YOLO-fine. Accordingly, the table confirms that YOLO-fine is the detector that has the best compromise between accuracy, size and time among the five comparative models.

Table 8. Comparison of YOLO-based detection models in terms of storage and computational requirements. Prediction time was tested on three different GPUs.

Model	Weight Size (MB)	BFLOPS	Prediction Time/Image (ms)			Number of Frames/Second (FPS)		
			Titan X	RTX 2080ti	V100	Titan X	RTX 2080ti	V100
YOLOv2	202.4	44.44	17.6	11.9	10.4	57	84	97
YOLOv3	246.3	98.92	26.9	14.5	11.4	37	69	88
YOLOv3-tiny	34.7	8.25	5.7	5.2	4.6	176	193	215
YOLOv3-spp	250.5	99.50	28.5	15.5	11.9	35	64	84
YOLO-fine	18.5	63.16	29.5	18.1	11.9	34	55	84

4.4. Effect of New Backgrounds in Validation Sets

4.4.1. Setup

Detecting known objects under various backgrounds (which are related to the environments and conditions of image acquisition) is an important challenge in many remote sensing applications. To do so, a deep learning model should have the capacity to generalize the characteristics of sought objects, without prior knowledge of the image background. Ideally, the same objects should be detected effectively, regardless of their environment (or image background). We aim here at evaluating the ability of the YOLO-fine model in order to address unknown backgrounds.

For these experiments, we relied on the 25-cm VEDAI512 dataset by dividing it into three subsets, each corresponding to a different type of background. Indeed, we split the data into a training set composed of images acquired in rural, forest and desert environments, and two validation sets with new backgrounds representing residential areas on the one hand, and very dense urban areas on the other hand. Figure 11 illustrates some sample images from the training set and the 2 validation sets (namely “urban” and “dense urban”). Table 9 provides some information related to the number of images and objects of each class included in the three sets. Even if VEDAI offers a limited number of images and objects, we make sure each class comes with enough samples in the training set, but also in the two validation sets in order to conduct a fair comparative study.

Table 9. Appearance of new backgrounds: information on the database built from VEDAI with a training set and two validation sets on two new environments.

Parameters	Set		
	Train Set	Validation Set 1	Validation Set 2
Environment	rural, forest, desert	urban	dense urban
Number of images	870	232	143
Number of objects	2142	815	797
from class Car	716	279	381
from class Truck	248	42	17
from class Pickup	505	235	215
from class Tractor	147	40	3
from class Camping	151	132	113
from class Ship	87	38	45
from class Van	63	25	13
from class Other	208	24	10



Figure 11. Appearance of new backgrounds in validation sets: the 25-cm VEDAI512 dataset (color version) is divided into three sets: one training set with images acquired in rural, forest, and desert environments; and, two validation sets with new backgrounds from urban and very dense urban areas.

4.4.2. Results

We report in Table 10 the detection results that were obtained by YOLOv2, YOLOv3 and our YOLO-fine model on the 2 validation sets. We first observe that all detectors led to better results for the first set than for the second set, e.g., with a respective mAP of 62.79% and 48.71% for YOLO-fine. This can be explained by the level of similarity between the background of the validation set and the background of the training set. The rural environment (training set) is more similar to the semi-urban environment (validation set 1) than to the dense urban environment (validation set 2). When comparing the results with those reported in Table 3, where YOLO-fine was achieving a mAP = 68.18% when trained/evaluated on all backgrounds, we can see that introducing new, unseen backgrounds in the validation set led to a loss in performance (i.e., 5.39% for the urban background and 19.47% for the dense urban background). However, when compared to its counterparts YOLOv2 and YOLOv3, YOLO-fine still achieves the best performance with a gain over YOLOv3 of 7.11% on the first background and 5.34% on the second background. YOLO-fine also achieved best results for 6/8 classes from each background. A satisfying accuracy is maintained for the 4 classes “car”, “truck”, “pickup”, and “tractor”, notably on the first background. On the other hand, a significant drop in detection accuracy is observed for the class “other” on which YOLO-fine reached only 6.61% of accuracy with the second background. This class is indeed quite challenging for every detector, since it gathers various objects (bulldozer, agrimotor, plane, helicopter) with various shape and size.

Table 10. Performance of YOLO-fine as compared to YOLOv2 and YOLOv3 on detection of known objects with the appearance of new backgrounds in validation sets: the first background contains residential/semi-urban areas and the second background very dense urban areas. Best results in bold.

Performance	Background 1			Background 2		
	YOLOv2	YOLOv3	YOLO-Fine	YOLOv2	YOLOv3	YOLO-Fine
car	52.48	70.42	75.23	61.58	77.55	81.32
truck	27.71	56.53	68.08	46.49	55.86	61.29
pickup	49.63	63.74	73.78	32.77	53.02	60.93
tractor	52.23	47.81	68.52	38.67	44.44	41.67
camping	46.62	56.81	62.84	43.49	51.58	62.40
boat	38.76	55.13	63.07	0.93	21.56	22.28
van	19.58	62.26	59.85	8.39	42.44	53.18
other	15.24	32.17	30.79	15.29	2.50	6.61
mAP	37.78	55.68	62.79	30.95	43.37	48.71
Precision	0.59	0.66	0.67	0.60	0.69	0.63
Recall	0.41	0.54	0.68	0.40	0.54	0.65
F1-score	0.48	0.60	0.67	0.48	0.60	0.64

5. Conclusions and Future Works

In this paper, we have presented an enhanced one-stage detection model, named YOLO-fine, to deal with small and very small objects from remote sensing images. Our detector was designed based on the state-of-the-art YOLOv3 with the main purpose of increasing the detection accuracy for small objects while being light and fast to enable real-time prediction within further operational contexts. Our experiments on three public benchmark datasets, namely VEDAI, MUNICH, and XVIEW, demonstrate the overall superiority of our YOLO-fine model over the well-established YOLOv3 solution. Indeed, YOLO-fine provides the best compromise between detection accuracy (highest mAP), network size (smallest weight size), and prediction time (able to perform real-time prediction).

While YOLO-fine brings promising results, some issues still remain and call for further research. First, as already mentioned, YOLO-fine focuses on small and very small objects, while its performances on medium and large objects remains comparable to YOLOv3, but with a slower prediction speed. In other words, YOLO-fine might not be the best choice if the sought objects have a very wide range of sizes. This bottleneck could be solved by making a transparent pass between YOLO-fine and YOLOv3 in the same model, allowing for an efficient detection of objects from very small sizes to very large sizes. Second, our investigation related to the behavior of YOLO-fine when dealing with new background appearances has provided some preliminary good results, but it remains limited to the proposed split made on the VEDAI dataset. It would be interesting to explore further this issue and consider other datasets where the difference between backgrounds in training and test scenes could be more significant. Finally, because EO sensors usually provided (close to) nadir views of the observed scenes, rotation invariance is a useful property for object detection in remote sensing. Thus, we aim to integrate the oriented bounding boxes' principle into YOLO-fine.

Author Contributions: M.-T.P. proposed the framework and wrote the manuscript; M.-T.P. and L.C. prepared datasets and conducted several experiments; C.F., S.L. and A.B. provided suggestions, then reviewed and edited the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by ANR/DGA through the DEEPDETECT project (ANR-17-ASTR-0016).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zou, Z.; Shi, Z.; Guo, Y.; Ye, J. Object detection in 20 years: A survey. *arXiv* **2019**, arXiv:1905.05055.
2. Liu, L.; Ouyang, W.; Wang, X.; Fieguth, P.; Chen, J.; Liu, X.; Pietikäinen, M. Deep learning for generic object detection: A survey. *Int. J. Comput. Vis.* **2020**, *128*, 261–318. [[CrossRef](#)]
3. Jiao, L.; Zhang, F.; Liu, F.; Yang, S.; Li, L.; Feng, Z.; Qu, R. A survey of deep learning-based object detection. *IEEE Access* **2019**, *7*, 128837–128868. [[CrossRef](#)]
4. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
5. Girshick, R. Fast R-CNN. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1440–1448.
6. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. In Proceedings of the Advances in Neural Information Processing Systems (NIPS), Montreal, QC, Canada, 7–12 December 2015; pp. 91–99.
7. Dai, J.; Li, Y.; He, K.; Sun, J. R-FCN: Object detection via region-based fully convolutional networks. In Proceedings of the Advances in Neural Information Processing Systems (NIPS), Barcelona, Spain, 5–10 December 2016; pp. 379–387.
8. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 2117–2125.
9. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask R-CNN. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2961–2969.
10. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
11. Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 6517–6525.
12. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
13. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. SSD: Single shot multibox detector. In Proceedings of the European Conference on Computer Vision (ECCV), Amsterdam, The Netherlands, 11–14 October 2016; pp. 21–37.
14. Shen, Z.; Liu, Z.; Li, J.; Jiang, Y.G.; Chen, Y.; Xue, X. DSOD: Learning deeply supervised object detectors from scratch. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 1919–1927.
15. Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2980–2988.
16. Kong, T.; Sun, F.; Yao, A.; Liu, H.; Lu, M.; Chen, Y. Ron: Reverse connection with objectness prior networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 5936–5944.
17. Zhao, Q.; Sheng, T.; Wang, Y.; Tang, Z.; Chen, Y.; Cai, L.; Ling, H. M2det: A single-shot object detector based on multi-level feature pyramid network. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33; pp. 9259–9266.
18. Tan, M.; Pang, R.; Le, Q.V. Efficientdet: Scalable and efficient object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 16–18 June 2020; pp. 10781–10790.
19. Saito, S.; Aoki, Y. Building and road detection from large aerial imagery. *SPIE/IS&T Electron. Imaging* **2015**, *9405*, 94050K.
20. Cheng, G.; Wang, Y.; Xu, S.; Wang, H.; Xiang, S.; Pan, C. Automatic road detection and centerline extraction via cascaded end-to-end convolutional neural network. *IEEE Trans. Geosci. Remote Sens.* **2017**, *55*, 3322–3337. [[CrossRef](#)]
21. Chen, F.; Ren, R.; Van de Voorde, T.; Xu, W.; Zhou, G.; Zhou, Y. Fast automatic airport detection in remote sensing images using convolutional neural networks. *Remote Sens.* **2018**, *10*, 443. [[CrossRef](#)]

22. Chen, X.; Xiang, S.; Liu, C.L.; Pan, C.H. Vehicle detection in satellite images by hybrid deep convolutional neural networks. *IEEE Geosci. Remote Sens. Lett.* **2014**, *11*, 1797–1801. [CrossRef]
23. Tang, T.; Zhou, S.; Deng, Z.; Lei, L.; Zou, H. Arbitrary-oriented vehicle detection in aerial imagery with single convolutional neural networks. *Remote Sens.* **2017**, *9*, 1170. [CrossRef]
24. Froidevaux, A.; Julier, A.; Lifschitz, A.; Pham, M.T.; Dambreville, R.; Lefèvre, S.; Lassalle, P. Vehicle detection and counting from VHR satellite images: Efforts and open issues. *arXiv* **2019**, arXiv:1910.10017.
25. Wu, Y.; Ma, W.; Gong, M.; Bai, Z.; Zhao, W.; Guo, Q.; Chen, X.; Miao, Q. A Coarse-to-Fine Network for Ship Detection in Optical Remote Sensing Images. *Remote Sens.* **2020**, *12*, 246. [CrossRef]
26. Kellenberger, B.; Marcos, D.; Tuia, D. Detecting mammals in UAV images: Best practices to address a substantially imbalanced dataset with deep learning. *Remote Sens. Environ.* **2018**, *216*, 139–153. [CrossRef]
27. Shiu, Y.; Palmer, K.; Roch, M.A.; Fleishman, E.; Liu, X.; Nosal, E.M.; Helble, T.; Cholewiak, D.; Gillespie, D.; Klinck, H. Deep neural networks for automated detection of marine mammal species. *Sci. Rep.* **2020**, *10*, 1–12. [CrossRef] [PubMed]
28. Cheng, G.; Han, J. A survey on object detection in optical remote sensing images. *ISPRS J. Photogramm. Remote Sens.* **2016**, *117*, 11–28. [CrossRef]
29. Zhu, X.X.; Tuia, D.; Mou, L.; Xia, G.S.; Zhang, L.; Xu, F.; Fraundorfer, F. Deep learning in remote sensing: A comprehensive review and list of resources. *IEEE Geosci. Remote Sens. Mag.* **2017**, *5*, 8–36. [CrossRef]
30. Li, K.; Wan, G.; Cheng, G.; Meng, L.; Han, J. Object detection in optical remote sensing images: A survey and a new benchmark. *ISPRS J. Photogramm. Remote Sens.* **2020**, *159*, 296–307. [CrossRef]
31. Razakarivony, S.; Jurie, F. Vehicle detection in aerial imagery: A small target detection benchmark. *J. Vis. Commun. Image Represent.* **2016**, *34*, 187–203. [CrossRef]
32. Liu, K.; Mattyus, G. DLR 3k Munich Vehicle Aerial Image Dataset. 2015. Available online: <https://pba-freesoftware.eoc.dlr.de/MunichDatasetVehicleDetection-2015-old.zip> (accessed on 4 August 2020).
33. Lam, D.; Kuzma, R.; McGee, K.; Dooley, S.; Laielli, M.; Klaric, M.; Bulatov, Y.; McCord, B. XView: Objects in context in overhead imagery. *arXiv* **2018**, arXiv:1802.07856.
34. Tong, K.; Wu, Y.; Zhou, F. Recent advances in small object detection based on deep learning: A review. *Image Vis. Comput.* **2020**, 103910. [CrossRef]
35. Eggert, C.; Brehm, S.; Winschel, A.; Zecha, D.; Lienhart, R. A closer look: Small object detection in faster R-CNN. In Proceedings of the 2017 IEEE International Conference on Multimedia and Expo (ICME), Hong Kong, China, 10–14 July 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 421–426.
36. Cao, C.; Wang, B.; Zhang, W.; Zeng, X.; Yan, X.; Feng, Z.; Liu, Y.; Wu, Z. An improved faster R-CNN for small object detection. *IEEE Access* **2019**, *7*, 106838–106846. [CrossRef]
37. Cao, G.; Xie, X.; Yang, W.; Liao, Q.; Shi, G.; Wu, J. Feature-fused SSD: Fast detection for small objects. In Proceedings of the Ninth International Conference on Graphic and Image Processing (ICGIP 2017), Qingdao, China, 14–16 October 2017; International Society for Optics and Photonics: Bellingham, WA, USA, 2018; Volume 10615, p. 106151E.
38. Cui, L.; Ma, R.; Lv, P.; Jiang, X.; Gao, Z.; Zhou, B.; Xu, M. Mdssd: Multi-scale deconvolutional single shot detector for small objects. *arXiv* **2018**, arXiv:1805.07009.
39. Zhang, S.; Wen, L.; Bian, X.; Lei, Z.; Li, S.Z. Single-shot refinement neural network for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 4203–4212.
40. Guan, L.; Wu, Y.; Zhao, J. Scan: Semantic context aware network for accurate small object detection. *Int. J. Comput. Intell. Syst.* **2018**, *11*, 951–961. [CrossRef]
41. Ren, Y.; Zhu, C.; Xiao, S. Small object detection in optical remote sensing images via modified faster R-CNN. *Appl. Sci.* **2018**, *8*, 813. [CrossRef]
42. Ding, P.; Zhang, Y.; Deng, W.J.; Jia, P.; Kuijper, A. A light and faster regional convolutional neural network for object detection in optical remote sensing images. *ISPRS J. Photogramm. Remote Sens.* **2018**, *141*, 208–218. [CrossRef]
43. Zhang, S.; He, G.; Chen, H.B.; Jing, N.; Wang, Q. Scale adaptive proposal network for object detection in remote sensing images. *IEEE Geosci. Remote Sens. Lett.* **2019**, *16*, 864–868. [CrossRef]
44. Zhang, W.; Wang, S.; Thachan, S.; Chen, J.; Qian, Y. Deconv R-CNN for small object detection on remote sensing images. In Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Valencia, Spain, 22–27 July 2018; pp. 2483–2486.

45. Ren, Y.; Zhu, C.; Xiao, S. Deformable Faster R-CNN with aggregating multi-layer features for partially occluded object detection in optical remote sensing images. *Remote Sens.* **2018**, *10*, 1470. [[CrossRef](#)]
46. Yan, J.; Wang, H.; Yan, M.; Diao, W.; Sun, X.; Li, H. IoU-adaptive deformable R-CNN: Make full use of IoU for multi-class object detection in remote sensing imagery. *Remote Sens.* **2019**, *11*, 286. [[CrossRef](#)]
47. Xia, G.S.; Bai, X.; Ding, J.; Zhu, Z.; Belongie, S.; Luo, J.; Datcu, M.; Pelillo, M.; Zhang, L. DOTA: A large-scale dataset for object detection in aerial images. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 3974–3983.
48. Liu, J.; Yang, S.; Tian, L.; Guo, W.; Zhou, B.; Jia, J.; Ling, H. Multi-component fusion network for small object detection in remote sensing images. *IEEE Access* **2019**, *7*, 128339–128352. [[CrossRef](#)]
49. Pang, J.; Li, C.; Shi, J.; Xu, Z.; Feng, H. R2CNN: Fast tiny object detection in large-scale remote sensing images. *IEEE Trans. Geosci. Remote Sens.* **2019**, *57*, 5512–5524. [[CrossRef](#)]
50. Yang, X.; Yang, J.; Yan, J.; Zhang, Y.; Zhang, T.; Guo, Z.; Sun, X.; Fu, K. SCRDet: Towards more robust detection for small, cluttered and rotated objects. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Seoul, Korea, 3 November–27 October 2019; pp. 8232–8241.
51. Liu, M.; Wang, X.; Zhou, A.; Fu, X.; Ma, Y.; Piao, C. UAV-YOLO: Small Object Detection on Unmanned Aerial Vehicle Perspective. *Sensors* **2020**, *20*, 2238. [[CrossRef](#)]
52. Zhao, K.; Ren, X. Small aircraft detection in remote sensing images based on YOLOv3. In Proceedings of the International Conference on Electrical Engineering, Control and Robotics (EECR), Guangzhou, China, 12–14 January 2019; Volume 533.
53. Nina, W.; Condori, W.; Machaca, V.; Villegas, J.; Castro, E. Small ship detection on optical satellite imagery with YOLO and YOLT. In Proceedings of the Future of Information and Communication Conference (FICC), San Francisco, CA, USA, 5–6 March 2020; pp. 664–677.
54. Xie, Y.; Cai, J.; Bhojwani, R.; Shekhar, S.; Knight, J. A locally-constrained YOLO framework for detecting small and densely-distributed building footprints. *Int. J. Geogr. Inf. Sci.* **2020**, *34*, 777–801. [[CrossRef](#)]
55. Van Etten, A. You Only Look Twice: Rapid multi-scale object detection in satellite imagery. *arXiv* **2018**, arXiv:1805.09512.
56. Huang, Z.; Wang, J.; Fu, X.; Yu, T.; Guo, Y.; Wang, R. DC-SPP-YOLO: Dense connection and spatial pyramid pooling based YOLO for object detection. *Inf. Sci.* **2020**, *522*, 241–258. [[CrossRef](#)]
57. Li, C. High Quality, Fast, Modular Reference Implementation of SSD in PyTorch. 2018. Available online: <https://github.com/lufficc/SSD> (accessed on 4 August 2020).
58. Massa, F.; Girshick, R. Maskrcnn-Benchmark: Fast, Modular Reference Implementation of Instance Segmentation and Object Detection Algorithms in PyTorch. 2018. Available online: <https://github.com/facebookresearch/maskrcnn-benchmark> (accessed on 4 August 2020).



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).