electronics

MDPI

Article

Detecting Objects from Space: An Evaluation of Deep-Learning Modern Approaches

Khang Nguyen [1,2,3,*], Nhut T. Huynh [2,3], Phat C. Nguyen [2,3], Khanh-Duy Nguyen [1,3], Nguyen D. Vo [1,3] and Tam V. Nguyen [4]

[1] Multimedia Communications Laboratory, University of Information Technology, Ho Chi Minh City 700000, Vietnam; khanhnd@uit.edu.vn (K.-D.N.); nguyenvd@uit.edu.vn (N.D.V.)
[2] Department of Software Engineering, University of Information Technology, Ho Chi Minh City 700000, Vietnam; 15520589@gm.uit.edu.vn (N.T.H.); 15520601@gm.uit.edu.vn (P.C.N.)
[3] Vietnam National University, Ho Chi Minh City 700000, Vietnam
[4] Department of Computer Science, University of Dayton, Dayton, OH 45469, USA; tamnguyen@udayton.edu
* Correspondence: khangnttm@uit.edu.vn

Received: 13 February 2020; Accepted: 26 March 2020; Published: 30 March 2020

Abstract: Unmanned aircraft systems or drones enable us to record or capture many scenes from the bird's-eye view and they have been fast deployed to a wide range of practical domains, i.e., agriculture, aerial photography, fast delivery and surveillance. Object detection task is one of the core steps in understanding videos collected from the drones. However, this task is very challenging due to the unconstrained viewpoints and low resolution of captured videos. While deep-learning modern object detectors have recently achieved great success in general benchmarks, i.e., PASCAL-VOC and MS-COCO, the robustness of these detectors on aerial images captured by drones is not well studied. In this paper, we present an evaluation of state-of-the-art deep-learning detectors including Faster R-CNN (Faster Regional CNN), RFCN (Region-based Fully Convolutional Networks), SNIPER (Scale Normalization for Image Pyramids with Efficient Resampling), Single-Shot Detector (SSD), YOLO (You Only Look Once), RetinaNet, and CenterNet for the object detection in videos captured by drones. We conduct experiments on VisDrone2019 dataset which contains 96 videos with 39,988 annotated frames and provide insights into efficient object detectors for aerial images.

Keywords: object detection; VisDrone2019; aerial imagery; Faster R-CNN; SSD; RFCN; YOLOv3; RetinaNet; SNIPER; CenterNet

1. Introduction

Object detection is a fundamental yet difficult task in image processing and computer vision research. It has been an important research topic for decades. Its development in the past two decades can be regarded as an epitome of computer vision history [1]. Since it plays a principal role in understanding and absorbing the contexts of images, therefore, object detection is considered to be a prerequisite measure that offers the computer to detect various objects. Giving a testing image, object detection could localize the coordinates of the objects and assign the corresponding labels to the objects in terms of the object category, i.e., human, dog, or cat. The coordinates of a detected object represent the object's bounding box [2,3]. Object detection has many applications in robot vision, autonomous driving, human-computer interaction, intelligent video surveillance. The deep-learning technology has brought significant breakthroughs in recent years. In particular, these techniques have produced remarkable development for object detection. Object detection can detect a specific instance, i.e., Obama's face, Eiffel Tower, Golden Gate Bridge; or objects of specific categories, i.e., humans, cars, bicycles. Historically, object detection has mainly directed on the detection of a single category,

for example, person class [4]. In recent years, the research community has started moving towards other categories than the well-known categories like person, cat, or dog. Here are some common challenges that object detectors face on aerial images: viewpoints, illuminations, scale variations, perspectives, intra-class variations, low resolutions, and occlusions. For example, the main challenges in pedestrian detection come from crowded scenes with heavy overlaps, occlusion, and low-resolution images.

Generic object detection has received significant attention. There are many competition benchmarks, i.e., PASCAL-VOC [5,6], ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [2], MS-COCO [7], and VisDrone-DET [8]. There are several notable studies on specific object detection like face detection [9], pedestrian detection [10] and vehicle detection [11]. Recently, the research community has focused on deep learning and its applications towards the object recognition/detection tasks. In the past few years, Convolutional Neural Networks (CNNs) [12,13] have brought breakthroughs in speech, audio, image, and video processing. CNNs have driven notable progress in visual recognition and object detection. Many successful CNN architectures, e.g., OverFeat [14], R-CNN [15], Fast R-CNN [16], Faster R-CNN [17], SSD [18], RFCN [19], YOLO [20], YOLOv2 [21], Faster R-CNN [17], RetinaNet [22], YOLOv3 [23], and SNIPER [24] have performed well on the task of object detection. For example, SNIPER, RetinaNet and YOLOv3 are the top models for object detection on MS-COCO dataset [7] with mAP 46.1%, 40.8%, 33.0%, respectively.

The object detection has now been widely used in many practical scenarios [1]. Its use cases ranging from protecting personal security to boosting productivity in the workplace. While the challenges of normal viewpoints have been considered to be the prevalence, in recent years, there has been increasing interest in flying drones and their applications in healthcare, video surveillance, search-and-rescue, and agriculture. The drones are now common devices that enable us to record or capture many scenes as the bird's-eye view. Visual object detection is an essential component in the drone application. However, object detection is a very challenging task since video sequences or images captured by drones vary significantly in terms of scales, perspectives, and weather conditions. Aerial images are often noisy and blurred due to the drone motion. The ratio of object size to the video resolution is also small. Therefore, in this paper, we investigate the performance of state-of-the-art object detectors in the aerial images. Please note that this paper is the extension of our earlier version which is the best paper in MITA 2019 conference [25]. Our contributions are three-fold.

- To the best of our knowledge, we are among the first ones who investigate the impact of different deep-learning object detection methods on the given problem.
- Second, we double the number of benchmarking methods compared to the MITA paper [25].
- Last but not least, we also increase the number of benchmarking classes in VisDrone dataset. In particular, we evaluate full 10 classes in this paper (vs. 2 human classes, namely *pedestrian* and *people*, in the MITA paper).

The remainder of the paper is organized as follows. In Section 2, the related works are presented. Section 3 and Section 4 present the benchmarked methods and the experimental results, respectively. Finally, Section 5 concludes our work.

2. Related Work

2.1. CNN Models

CNN-based architectures have been backbones in many detection frameworks. Popular architectures include AlexNet [12], ZFNet [26] VGGNet [27], GoogLeNet [28], ResNet [29] and DenseNet [30]. We briefly introduce these models as follows.

Known as a pioneering work, AlexNet [12] consists of eight layers: five convolutional (conv1, conv2, conv3, conv4 and conv5) layers and three fully connected (fc6, fc7 and fc8) layers. The fc8 is a SoftMax classifier. The convolutional layers are connected directly: conv3, conv4 and conv5. The convolutional layers are connected via an Overlapping Max-Pooling layer: conv1–conv2,

conv2–conv3, conv5–fc6. AlexNet used 11 × 11, 5 × 5 and 3 × 3 kernels. Later, Reference [26] proposed ZFNet by modifying AlexNet. ZFNet uses 7 × 7 kernels. The small kernels retain more information than the big kernels. By proposing a deeper network, VGGNet [27] outperforms AlexNet in the image classification task. Similarly, GoogLeNet [28] won the ILSVRC2014 competition by increasing the number of layers. In particular, it includes 22 layers: 21 convolutional layers and a fully connected layer.

The number of layers is increasing in CNN architectures. The CNNs require a lot of computational resources. There are several problems: gradient vanishing, exploding, and degrading. Degradation occurs when we add more layers into deep networks, the accuracy becomes saturated and then decrease quickly. To overcome this problem, ResNet [29] introduces many residual blocks. In a residual block, each layer is fed directly to the layers about 2–3 hops away using skip-connections.

DenseNet [30] was proposed by Huang et al. in 2017. It includes many dense blocks. A dense block consists of composite layers which are densely connected together. The input of one layer is the output of all previous layers, so input information is shared.

2.2. Object Detection Methods

Object detection methods are mainly divided into one-stage frameworks and two-stage frameworks. Two-stage frameworks are more accurate than one-stage frameworks, but one-stage frameworks usually achieve real-time detection. The two-stage approach includes two steps: the first stage creates region proposals, the second stage classifies region proposals. The one-stage approach predicts object regions and object classes at the same time.

CNN-based Two-Stage frameworks: Two-stage frameworks mainly include R-CNN [15], SPP-Net [31], Fast R-CNN [16], Faster R-CNN [17], RFCN [19], Mask R-CNN [32], and SNIPER [24].

R-CNN [15] is a method for detecting objects based on the ImageNet pre-trained model. R-CNN uses Selective Search algorithm to generate region proposals. Then, these regions are warped and fed into the pre-trained model to extract high-level features. Finally, several SVM classifiers are trained based on these features to identify object classes. Fast R-CNN [16] was introduced to solve some R-CNN's limitations, i.e., the computational speed. Fast R-CNN feeds the whole image into ConvNet to create convolutional feature map instead of 2000 regions as R-CNN.

Faster R-CNN [17] proposes a Region Proposal Network (RPN) to detect region proposals instead of Selective Search, which is used in R-CNN and Fast R-CNN. Faster R-CNN is 10× faster than Fast R-CNN, and 250× faster than R-CNN in reference time. RFCN introduces the positive sensitive score map, which improves speed but remains accurate compared to Faster R-CNN. Mask R-CNN extends Faster R-CNN with instance segmentation and introduces Align Pooling.

CNN-based one-stage frameworks: The most common examples of one-stage object frameworks are YOLO, YOLOv2, YOLOv3 [23], SSD [18], and RetinaNet. You Only Look Once (YOLO [20]) is one of the first approaches to build a one-stage detector. Unlike R-CNN family, YOLO does not use a region proposal component. Instead, it learns to regress bounding-box coordinates and class probabilities directly from image pixels. This significantly boosts the speed of the detecting process. Single-Shot MultiBox Detector (SSD [18]) is also a one-stage detector which also aims at high speed object detection. However, unlike YOLO, SSD adopts a multi-scale approach. It then adds many convolutional layers decreasing in size sequentially. This can be regarded as a pyramid representation of an image, where earlier levels contain feature maps that are useful to detect small objects and deeper levels are expected to detect larger objects. Each of these layers has a set of predefined anchor boxes (also known as default boxes or prior boxes) for every cell. The model will learn and predict the offsets corresponding to correct anchor boxes. The approach has made a successful attempt on creating an efficient detector for objects in various sizes while maintaining a low inference time.

3. Benchmarked State-of-the-Art Object Detection Methods

In this section, we provide further details of the benchmarked object detection methods. Figure 1 visualizes the methods in the chronological order. Meanwhile, Figure 2 depicts the framework structures of different state-of-the-art object detectors.

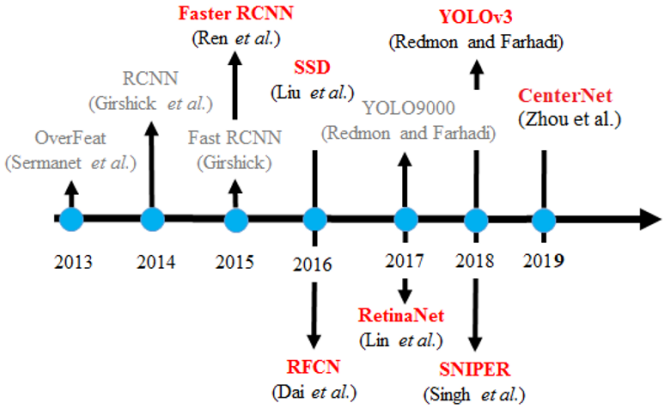


Figure 1. Timeline of state-of-the-art object detection methods. The benchmarked methods are marked in red and boldfaced font.

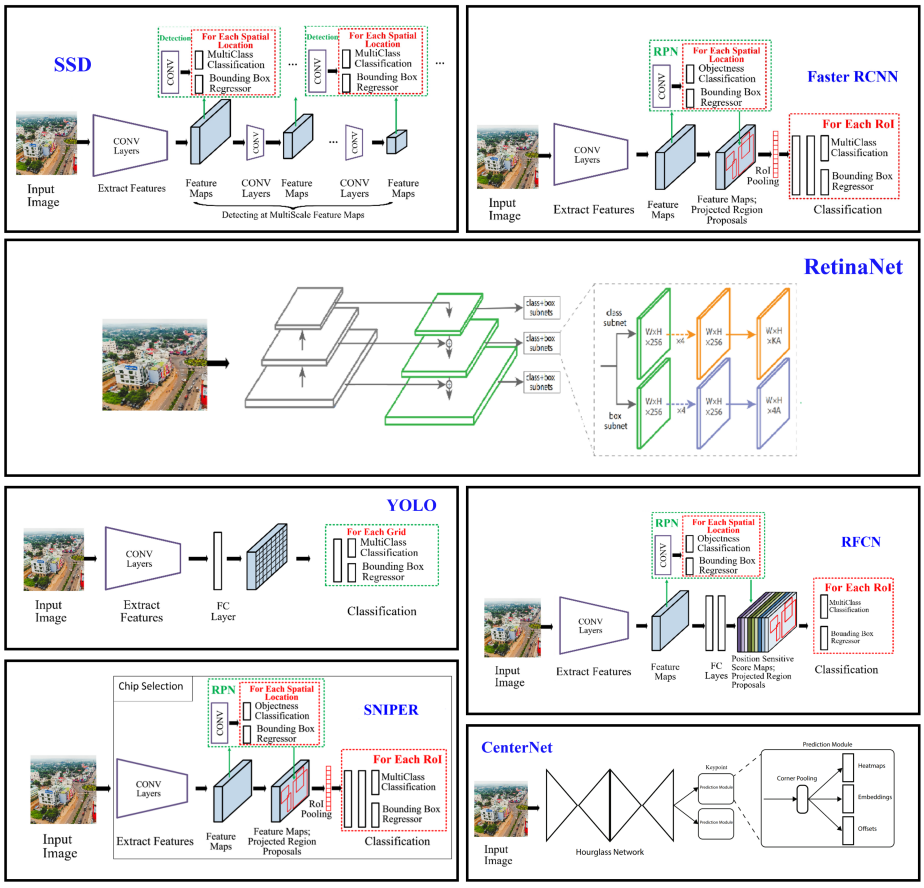


Figure 2. The benchmarked methods adopted in this paper: Faster R-CNN, SSD, RFCN, RetinaNet, SNIPER, YOLOv3, and CenterNet.

3.1. Faster R-CNN

Faster R-CNN [17] is an extension of the R-CNN [15] and Fast R-CNN [16] methods for object detection. R-CNN requires a forward pass of the CNN for around 2000 region proposals (ROI) for every single image. Later, Fast R-CNN was able to solve the problem of R-CNN by sharing the computation of convolution between different proposals (feature map). The detection process is sped up but still depends on the region proposal method (Selective Search). Region proposals were generated by additional methods, i.e., Selective Search or Edge Box. To solve this problem, Ren et al. [17] introduced the Region Proposal Network (RPN).

Faster R-CNN consists of two main components, namely the RPN and the Fast R-CNN detector. RPN initializes squared reference boxes of aspect ratios and diverse scales at each convolutional feature map location. Each squared box is mapped to a feature vector. The feature vector is fed into two fully connected layers, an object category classification layer, and a box regression layer. Faster R-CNN enables highly efficient region proposal computation because RPN shares convolutional features with Fast R-CNN. With an image of arbitrary size as an input, RPN is trained end-to-end to generate high-quality region proposals as output. The Fast R-CNN detector also uses the ROI pooling layer to extract features from each candidate box and performs object classification and bounding-box regression. The entire system is a single, unified network for object detection.

3.2. RFCN: Region-Based Fully Convolutional Networks

A limitation of Faster RCN is that it does not share computations after ROI pooling. The amount of computation should be shared as much as possible. Faster R-CNN overcomes the limitations of Fast R-CNN but it still contains several non-shared fully connected layers that must be computed for each of hundreds of proposals. Region-based Fully Convolutional Network [19] was proposed as an improvement to Faster R-CNN. It consists of shared, fully convolutional architectures. In RFCN, fully connected layers after ROI pooling are removed, all other layers are moved prior to the ROI pooling to generate the score maps. RFCN infers 2.5 to 20 times faster than Faster R-CNN, yet it still maintains a competitive accuracy.

Backbone Architecture. The incarnation of RFCN in [19] is based on ImageNet pre-trained ResNet-101 model. To compute feature maps, the average pooling layer and the *fc* layer are removed. Instead, RFCN only uses the convolutional layers, and attaches a randomly initialized 1024-d 1×1 convolutional layer to reduce dimension at the last convolutional block in ResNet-101. In addition, RFCN uses the $k^2(C+1)$ channel convolutional layers to generate scope maps.

Position-sensitive score maps and Position-sensitive ROI pooling. RFCN regularly divides each ROI rectangle into $k \times k$ bins, then each bin has a size of $\approx \frac{w}{k} \times \frac{h}{k}$ for an ROI rectangle of a size $w \times h$. For each category, the last convolutional layer is built to create k^2 score maps. Inside the (i,j)-th bin $(0 \leq i,j \leq k-1)$, a position-sensitive ROI pooling operation pools only over the (i,j)-th score map:

$$r_c\left(i,j \mid \ominus\right) = \sum_{(x,y) \in bin(i,j)} z_{b,j,c}\left(x + x_0, y + y_0 | \ominus\right) / n \tag{1}$$

For the c-th category, $r_c(i,j)$ is the aggregated response in the (i,j)-th bin; $z_{b,j,c}$ is a score map among $k^2(C+1)$ maps; (x_0, y_0) represents the ROI's top left corner; and the number of pixels in the bin is marked as n. $\ominus$ is the set of learnable parameters; and the (i,j)-th bin is located at $\left\lfloor i\frac{w}{k} \right\rfloor \leq x \leq \left\lfloor (i+1)\frac{w}{k} \right\rfloor$ and $\left\lfloor j\frac{h}{k} \right\rfloor \leq y \leq \left\lfloor (j+1)\frac{h}{k} \right\rfloor$.

The k^2 position-sensitive scores are averaged to obtain a $(C+1)$-dimensional vector for each ROI $r_c\left(\ominus\right) = \sum_{i,j} r_c\left(i,j|\ominus\right)$. Then the SoftMax responses for categories are computed $s_c\left(\ominus\right) = e^{r_c(\ominus)} / \sum_{c'=0}^{C} e^{r_{c'}(\ominus)}$.

The RFCN resolves bounding-box regression similar to Fast R-CNN with $k^2(C+1)$-d convolutional layer, and appends a sibling $4k^2$-d convolutional layer additionally.

The position-sensitive ROI pooling makes a $4k^2$-d vector for each ROI. Then, a bounding box as $t = (tx, ty, tw, th)$ uses a 4-d vector aggregated by average voting.

Training. The loss function is defined on each ROI and calculated by the sum of cross-entropy losses and box regression loss:

$$L(s, t_{x,y,w,h}) = L_{cls}(s_c^*) + \lambda \left[c^* > 0\right] L_{reg}(t, t^*), \tag{2}$$

where c^* is the ground-truth label of ROI. For classification, $L_{cls}(s_c^*) = -\log(s_c^*)$ denotes the cross-entropy losses, L_{reg} denotes the bounding-box regression loss and t^* is the ground-truth box. If the argument is valid, $[c^* > 0]$ receives a value of 1, otherwise, 0.

Inference. The feature maps are the results of calculations on an image with a single scale of 600 shared between RPN and RFCN (as showed in Figure 2). Then, the RFCN part evaluates score maps and regresses bounding boxes based on ROIs which are proposed by the Region Proposal Network (RPN) part.

3.3. SNIPER: Scale Normalization for Image Pyramids with Efficient Resampling

SNIPER is an effective, multi-scale training method for identification, object detection and object separation [24]. Instead of processing pixels based on the pyramid (SN), SNIPER treats the context areas around the ground truths (called chips) at an appropriate scale. This greatly increases the speed during training when it operates on low-resolution chips. Relying on the memory efficient design, SNIPER benefits from mass standardization during the training process without having to synchronize standardized statistics on the GPU.

3.3.1. Chip Generation

SNIPER generates chips C_i at multiple scales $\{s_1, s_2, ..., s_i, ..., s_n\}$ in the image. For each scale, the image is first re-sized to width (W_i) and height (H_i). On this canvas, $K \times K$ pixel chips are placed at equal intervals of d pixels.

3.3.2. Chip Selection

All favorable chips are chosen greedily to cover the highest amount of valid ground-truth boxes. If it is completely enclosed inside a chip, a ground-truth box is said to be covered. Although positive chips cover all the positive instances, a significant portion of the background is not covered by them. In multi-scale training architecture, each pixel is processed at all scales in the picture. A naive strategy is to use object suggestions to define areas where objects are likely to present. If there are no region proposals in an image, it is considered to be background.

3.4. SSD: Single-Shot Detector

SSD [18] is a one-stage solution, which has tremendously reduced inference time and resulted in an accurate, high speed detector that can be used for real-time video processing.

3.4.1. Base Network VGG-16

SSD is built on top of VGG-16 base network [27] that focuses on simplicity and depth. In particular, the model uses 16 convolutional layers with only 3×3 filters to extract features. As the model goes deeper, the number of filter doubles after each max-pooling layer. Noticeably, the convolutional layers of the same type are combined as shown in Figure 2. At the end, three fully connected layers are followed by 4096 channels. The last one contains 1000 channels for each class and is concatenated with a SoftMax layer to return the detection results. The model works well on classification and localization tasks and has achieved 89% mAP on PASCAL-VOC 2007 dataset. VGG-16 has been one of the most interesting models to research even though it is not as fast as the newer ones. Its architecture has been reused in many models because of its valuable extracted features.

3.4.2. Model Architecture

SSD extends the pre-trained VGG-16 model (on ImageNet [10]) by adding new convolutional layers conv8_2, conv9_2, conv10_2, conv11_2 in addition to using the modified conv4_3 and fc_7 layers to extract useful features. Each layer is designed to detect objects at a certain scale using k anchor boxes, where $4k$ offsets and c class probabilities are computed by using 3×3 filters. Thus, given a feature map with a size of $m \times n$, the total number of filters to be used is $kmn(c+4)$. The anchor boxes are chosen manually. Here, we use the original formula, as follows, to calculate anchor box scales at different levels.

$$s_k = s_{min} + \frac{\mathrm{s_{max}} - \mathrm{s_{min}}}{\mathrm{m} - 1}(k-1) \tag{3}$$

$k \in [1, m]$ where $s_{min} = 0.2$ and $s_{max} = 0.9$.

3.5. RetinaNet

RetinaNet [22] is another one-stage detector. It aims to tackle the class imbalance problem between foreground and background remaining in one-stage detector. RetinaNet uses two main techniques: FPN backbone and focal loss as the loss function. FPN is built on top of a convolutional neural network and is responsible for extracting convolutional feature maps from the entire image. By using focal loss, RetinaNet changes weights in the loss function, focuses on hard, misclassified examples, which improves the prediction accuracy. With ResNet (FPN) as a backbone for feature extraction and two specific subnetworks for classification and bounding-box regression, RetinaNet has achieved state-of-the-art performance.

3.5.1. Class Imbalance

As a one-stage detector, RetinaNet has a much larger set of candidate object locations which is regularly sampled across an image (∼100 k locations), covering spatial positions, scales and aspect ratios tightly. The easily classified background examples still dominate the training procedure. Bootstrapping or hard example mining is typically used as a solution for this problem. However, they are not efficient enough. To solve this, RetinaNet proposes a new loss function which can adaptively tune the contributed weights of object classes during training.

3.5.2. Focal Loss

Focal loss is computed by adding $(1-p_i)^\gamma$ to cross-entropy loss as a modulating factor.

$$\sum_{i=1}^{k}(y_i log(p_i)(1-p_i)^\gamma) + (1-y_i)log(1-p_i)p_i^\gamma) \tag{4}$$

3.5.3. RetinaNet Detector Architecture

RetinaNet adopts ResNet for deep feature extraction. ResNet builds a rich multi-scale feature pyramid from an input image of single resolution by using Feature Pyramid Network (FPN) [33]. It combines low-resolution, high-resolution, and semi-weak characteristics through a top-down pathway and lateral connections.

3.6. YOLO: You Only Look Once

YOLO is an object detection system targeted for real-time processing. Recently, the third version of YOLO has been published, YOLOv3 is extremely fast and accurate. In mAP measured at 0.5 IOU YOLOv3 is on par with focal loss but about 4 × faster.

YOLOv3 takes an input image to predict 3D tensors respectively to three scales and each scale is divided into N × N grid cells. During training, each grid cell considers a class that it likely is and be

responsible for detecting that class. Simultaneously, each grid cell is assigned with 3 initial prior boxes with various sizes. Finally, non-max suppression is applied to select the best boxes.

3.6.1. Feature Extraction

YOLOv3 uses a variant of Darknet, which originally has a 53-layer network trained on ImageNet. According to [23], Darknet-53 is better than ResNet-101 and 1.5 × faster. Darknet-53 has a performance similar to ResNet-152 and is 2 × faster.

3.6.2. Detection at Three Scales

YOLOv3 is different from its predecessors since it performs the detection process at three different scales. In YOLOv3, the detection is done by applying 1 × 1 detection kernels on feature maps of three different sizes at three different places in the network. YOLOv3 makes prediction at three scales, which are precisely given by down sampling the dimensions of the input image by 32, 16 and 8, respectively. Detections at different layer helps address the issue of detecting small objects, a common issue in YOLOv2.

3.6.3. Objective Score and Confidences

The object score illustrates the probability that an object is contained inside a bounding box and its value ranging from 1 to 0. A sigmoid is applied to compute the objectness scores. In terms of class confidences, they depict the probabilities of the detected object which belongs to a particular class. In YOLOv3, Non-maximum Suppression (NMS) is used to decide a class score and it is meant to alleviate the problem of multiple detections of the same object.

3.7. CenterNet

One-stage and two-stage detection have limitations: anchor box is designed with manual proportions that are easily affected by data and fixed during training. This requires a high computation cost, but the anchors are not always accurate. To address that, recently a series of anchor-free methods [34–36] are proposed. CenterNet [37] is a one-stage detector, anchor-free method. Reference [37] proposed a new center-based framework based on a single Hourglass network without FPN structure [38]. The object is represented by the central point of the bounding box. Other information is calculated by regression such as object size, dimension, and pose.

3.7.1. Object as Points

CenterNet considers the center point of an object as a prerequisite to localize the bounding box. As a result, Reference [37] use a keypoint estimator $\hat{y}$ to predict all center points and single a single size prediction for all object categories to alleviate the computational burden.

3.7.2. From Points to Bounding Boxes

CenterNet identifies the peak points in the heatmap before detecting all the values meet or greater than its 8-related neighbors and keep the top 100 points. Subsequently, each keypoint location is given by an integer coordinate (x_i, y_i) and the key point estimator value as a measure of its detection confidence is applied to produce the bounding box as below:

$$(\hat{x}_i + \delta\hat{x}_i - \hat{w}_i/2,\ \hat{y}_i + \delta\hat{y}_i - \hat{h}_i/2,\ \hat{x}_i + \delta\hat{x}_i + \hat{w}_i/2,\ \hat{y}_i + \delta\hat{y}_i + \hat{h}_i/2), \tag{5}$$

where $(\delta\hat{x}_i, \delta\hat{y}_i) = \hat{O}_{\hat{x}_i,\hat{y}_i}$ is the offset prediction and $(\hat{w}_i, \hat{h}_i) = \hat{S}_{\hat{x}_i,\hat{y}_i}$ is the size prediction [37].

4. Benchmark Experiments

In this section, we first introduce the benchmark dataset and the evaluation metrics. We then detail the model configuration and discuss the experimental results.

4.1. Dataset

VisDrone2019 dataset [39] consists of 288 videos with 261,908 frames and 10,209 static images that do not match the frames of videos. Data is collected from unmanned aerial vehicles such as DJI Mavic, Phantom series (3, 3A, 3SE, 3P, 4, 4A, 4P). The videos and images are collected at different times of day. The frames in the videos have the highest resolution of 3840 × 2160 and the still image is 2000 × 1500. Some images in the dataset are shown in the Figure 3. VisDrone2019 includes ten predefined categories of objects: *pedestrian, person car, van, bus, truck, motor, bicycle, awning-tricycle, and tricycle.* Only the training data is released by the contest organizers. In this paper, we use 56 clips of VID-train data, with 24,313 frames of the VisDrone2019 dataset as training data; and 7 clips of VID-val, with 2860 frames for model evaluation.

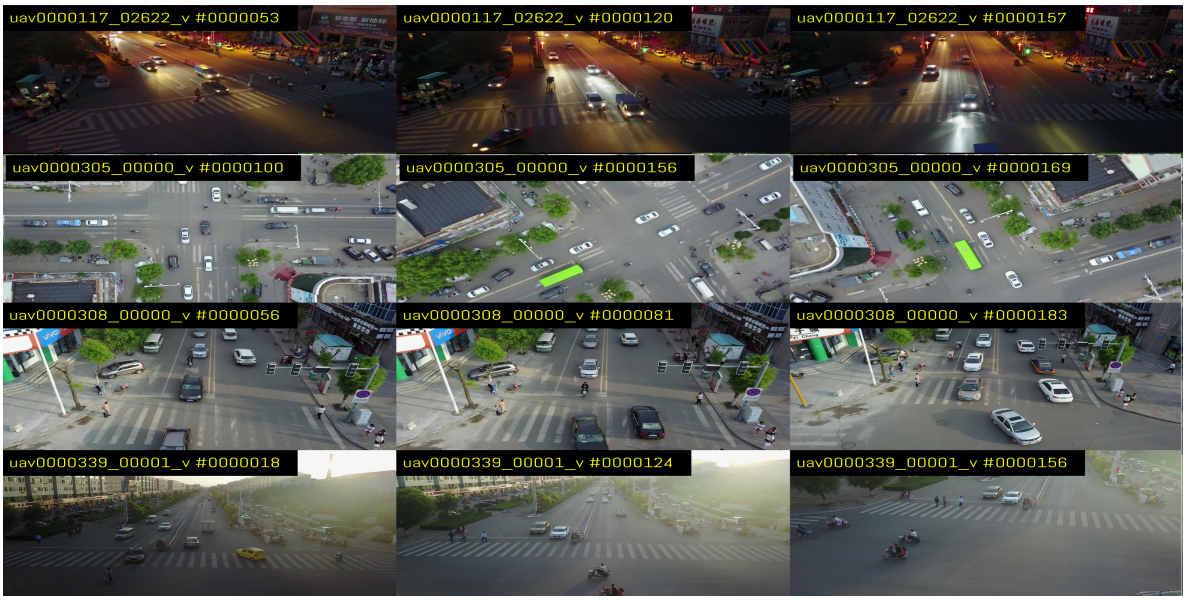


Figure 3. Some exemplary frames of videos in the VisDrone2019 dataset [39].

The number of unbalanced objects between classes is as follows. In VID-train, *pedestrian* (234,305), people (94,396), *bicycle* (40,255), car (505,301), *van* (46,940), *truck* (30,498), tricycle (28,338), *awning-tricycle* (13,011), *bus* (9653) and *motor* (102,819). In VID-val, *pedestrian* (32,404), *people* (17,908), bicycle (6842), *car* (31,821), van (6842), *truck* (1359), *tricycle* (3769), *awning-tricycle* (1718), *bus* (264) and *motor* (12,025). The class distribution of Visdrone VID is depicted in Figure 4. As a quick glimpse through the training set, there are the wide discrepancy in terms of the weather, the light source direction and the time interval as well as the drone motion in Figure 5.

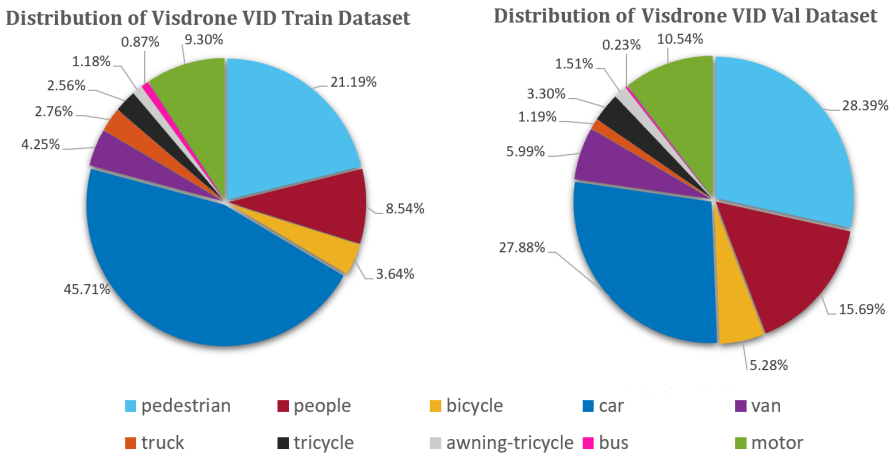


Figure 4. The class distribution in Visdrone VID-Train, Val Dataset.

Figure 5. Some example images of the challenge.

4.2. Evaluation Metrics

In this work, we use the Average Precision (AP) measurement [3,40], the commonly used metric to assess object detection accuracy. Given two bounding boxes, one for ground truth (the actual class label) and one for the detection result (the predicted class label), we use the Intersection over Union (IoU) to calculate the similarity between the two boxes and the score of the predicted box. It is computed as the intersected area (S_i) divided by the union (S_j) of the two areas. An IoU threshold η indicates whether the prediction is an object or not. If the actual class label is the predicted class label and IoU > η, it is considered a positive else it is considered a negative.

$$IoU = \frac{S_i \cap S_j}{S_i \cup S_j} \tag{6}$$

The AP computes the average precision value for recall value over 0 to 1. The mean Average Precision (mAP) is computed by taking the average over the AP of all classes. Precision is the proportion of the predicted bounding boxes matching actual ground truth. Recall is the proportion of ground-truth objects being correctly detected. For object detection, we report the performance results with AP (IoU = 0.50), AP (IoU = 0.75). The AP [3] summarizes the shape of the precision/recall curve, and is defined as the mean precision at a set of 11 equally spaced recall levels [0, 0.1, . . . , 1]:

$$AP = \frac{1}{11} \times \sum_{r \in \{0, 0.1, \ldots, 1\}} p_{interp}(r) \tag{7}$$

where:

$$p_{interp}(r) = max_{\widetilde{r}:\widetilde{r}\geq r} p(\widetilde{r}) \quad (8)$$

4.3. Model Configuration

To make a fair comparison, namely each detection model at its best, we adjust the model's parameters following the recommendation of [41–43]. We provide the detailed configuration as below.

- For **YOLOv3**, we have trained a new model which adopts darknet53 (YOLOv3 code is available at: https://github.com/AlexeyAB/darknet) as pre-trained weights for the convolutional layers. In implementation, YOLO v3 uses a total of 9 anchors with three for each scale. It assigns the three biggest anchors for the first scale, the next three for the second scale, and the last three for the third. As we aim to detect effectively on 3 scales, we recalculate nine anchors correspondingly.
- For **RetinaNet**, we used ResNet152 as backbone model and changed anchor parameters (RetinaNet code is at: https://github.com/fizyr/keras-retinanet).
- **SNIPER** is the new architecture that allows us to improve AP by using negative chip mining. We adopt the available model (SNIPER code is at: https://github.com/mahyarnajibi/SNIPER). We extracted the required proposals for chip selection.
- For **SSD**, we used VGG16 as the pre-trained model and adjusted the aspect ratio used in the original SSD300 (SSD code is at: https://github.com/pierluigiferrari/ssd_keras).
- We trained **Faster R-CNN** (Faster R-CNN code is available at:https://github.com/rbgirshick/py-faster-rcnn), **RFCN** (RFCN code is at: https://github.com/YuwenXiong/py-RFCN), **CenterNet** (CenterNet code is at: https://github.com/xingyizhou/CenterNet) with default parameters.

Table 1 summarizes the detailed configuration. All models are trained or finetuned by using GeForce RTX 2080 Ti 11GB GPU run on Ubuntu 16.04.5 LTS OS.

Table 1. Configuration of SNIPER, RetinaNet, YOLOv3 and SSD.

Architecture	Configuration		
	Attribute	**Default**	**Adjusted**
RetinaNet	Base size Strides Ratios Scales	32 64 128 256 512 8 16 32 64 128 0.5 1 2 1 1.2 1.6	32 64 128 256 512 8 16 32 64 128 0.5 1 2 **0.5 1 1.2 1.6 2.0**
YOLOv3	Anchors	10, 13, 16, 30, 33, 23, 30, 61, 62, 45, 59, 119, 116, 90, 156, 198, 373, 326	**20, 310,** **35, 626,** **20, 3751,** **69, 1135,** **92, 2038,** **54, 6129,** **50, 8045,** **107, 12908,** **117, 15654**
SSD	Ratio	[2], [2, 3], [2, 3], [2, 3], [2], [2]	**[1.0, 2.0, 0.5]** **[1.0, 2.0, 0.5, 3.0, 1.0/3.0]** **[1.0, 2.0, 0.5, 3.0, 1.0/3.0]** **[1.0, 2.0, 0.5, 3.0, 1.0/3.0]** **[1.0, 2.0, 0.5]** **[1.0, 2.0, 0.5]**
SNIPER	Scales	[1400, 2000] [800, 1280] [480, 512]	**[1600, 2200]** [800, 1280] [480, 512]

4.4. Results

As aforementioned, we benchmark six state-of-the-art methods: Faster R-CNN, RFCN, SSD with default parameters and SNIPER, YOLOv3, RetinaNet with adjusted parameters which is presented in

detail in Table 1. Table 2 shows the training time of the methods. RFCN and YOLOv3 take the least training time. Meanwhile, SSD requires a remarkable time for training. Table 3 shows the runtime performance of different methods. SSD and YOLOv3 achieve the fastest running time among the rest. In the meantime, RFCN only processes 1.75 frames per second. One-stage object detectors clearly run faster than the two-stage ones.

Table 2. Training time of Faster R-CNN, RFCN, SNIPER, RetinaNet, YOLOv3, SSD with VisDrone2018 on GeForce RTX 2080 Ti GPU.

Architecture	Training Time (hours)
Faster R-CNN	8.20
RFCN	4.30
SNIPER	62.50
RetinaNet	10.61
CenterNet	5.3
YOLOv3	4.70
SSD	251.18

Table 3. Runtime performance of Faster R-CNN, RFCN, SSD, YOLO, SNIPER, RetinaNet with VisDrone2019 on GeForce RTX 2080 Ti GPU.

Architecture	FPS (Frames Per Second)
Faster R-CNN	2.78
RFCN	1.75
SNIPER	7.6
SSD	76.9
YOLOv3	7.5
RetinaNet	5.9
CenterNet	7.1

Figure 6 visualizes the detection results of benchmarking methods. As a closer look, Tables 4 and 5 show the detailed results of six methods and the average performance with a threshold of IoU set as 0.5 and 0.75, respectively. In accordance with Table 4 (IoU = 0.5), CenterNet, RetinaNet and SNIPER are the only three algorithms achieving more than 25% mAP score. YOLOv3 ranks in the fourth with more than 20% mAP score. We observe that SSD is inferior, and RFCN performs better than Faster R-CNN. SSD performs the worst, only producing 10.80% mAP score. However, SSD ranks in the third with 9.10% on the *bus* class. As seen in Table 5 (IoU = 0.75), SNIPER, CenterNet and RetinaNet are the only three algorithms achieving more than 11% mAP score. RFCN ranks in the fourth with more than 8% mAP score. We observe that YOLOv3 performs the worst (3.20% mAP score). In the meantime, Faster R-CNN performs better than SSD.

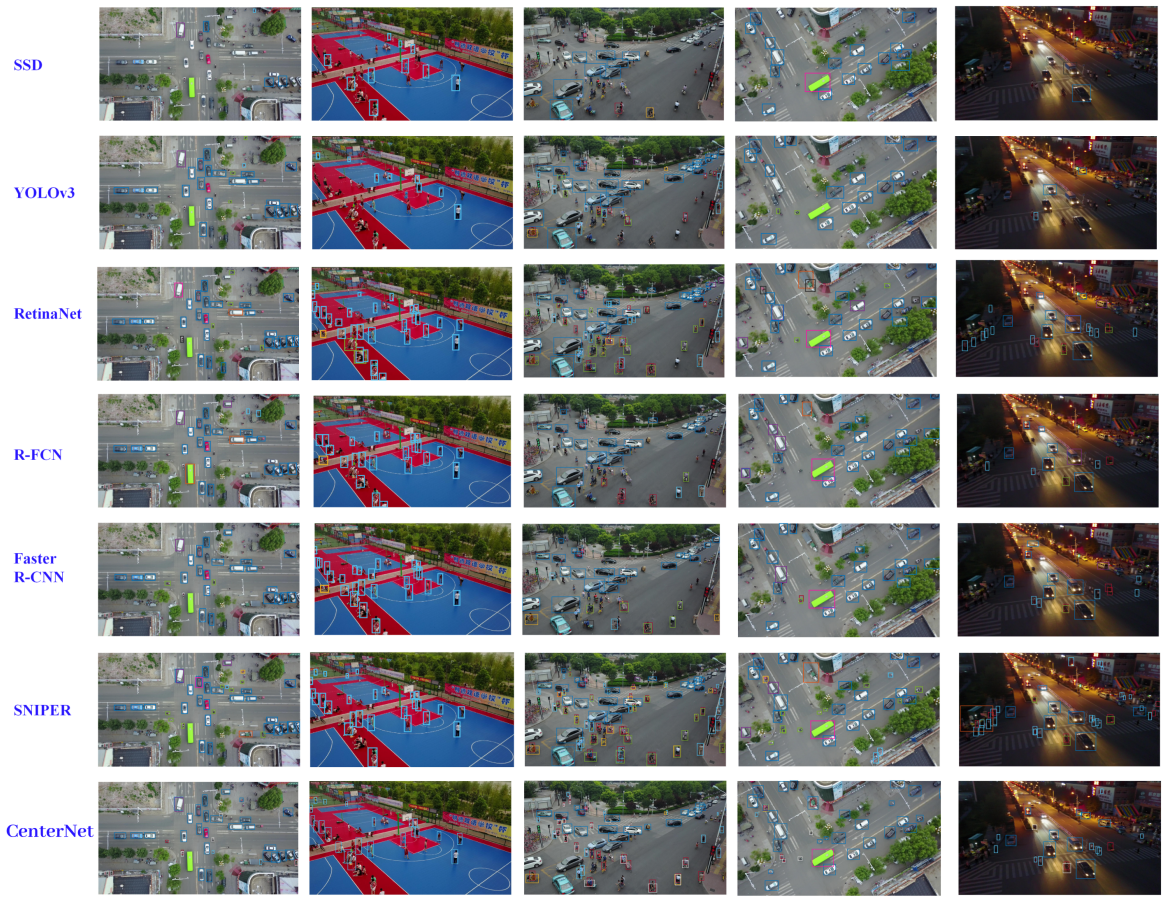


Figure 6. Visualization results of different object detection methods. Color legend: car, truck, bicycle, van, moto, pedestrian, people, bus, tricycle, awning-tricycle. Best view in high 400% resolution.

Regarding the performance, YOLOv3 has good performance with 7.5 FPS and 25.08 mAP (IoU = 0.5) then it drops rapidly to 3.2 mAP (IoU = 0.75) because YOLOv3 is does not perform well at localization. Instead, YOLOv3 is well-known for its runtime performance. We have performed the detection of YOLOv3 and realized the small confidence score for each object (<30%) due to the similarity of features. Therefore, we literally set the low confidence for the object detection demand with this YOLOv3. Meanwhile, CenterNet, RetinaNet and SNIPER achieve better detection results.

Table 4. The AP (IoU = 0.50) scores on the VisDrone2019 Validation set of each object category. The top three results are highlighted in red, blue and green fonts.

Detectors	Pedestrian	People	Bicycle	Car	Van	Truck	Tricycle	A-Tricycle	Bus	Motor	mAP
Faster R-CNN	21.36	11.02	16.97	46.85	18.72	16.96	12.22	10.81	9.09	11.13	17.51
RFCN	21.20	7.05	16.16	44.22	20.12	28.10	19.45	15.21	13.28	10.76	19.55
SNIPER	37.97	23.74	34.73	56.51	31.90	6.72	21.05	14.60	12.86	22.39	26.24
SSD	13.20	5.50	12.70	36.90	9.60	7.70	1.30	9.10	9.10	4.60	10.97
YOLOv3	32.64	16.92	36.94	51.67	33.65	12.83	8.97	26.01	4.87	26.36	25.08
RetinaNet	50.21	31.14	23.53	52.05	27.27	23.92	21.16	23.55	8.97	20.81	28.26
CenterNet	56.74	37.37	27.06	59.86	30	27.51	24.33	26.61	9.87	23.52	32.28

Table 5. The AP (IoU = 0.75) scores on the VisDrone2019 Validation set of each object category. The top three results are highlighted in red, blue and green fonts.

Detectors	Pedestrian	People	Bicycle	Car	Van	Truck	Tricycle	A-Tricycle	Bus	Motor	mAP
Faster R-CNN	3.30	0.71	9.09	19.12	9.09	9.09	1.64	3.03	9.09	0.95	6.51
RFCN	3.03	1.40	2.60	24.85	11.31	12.45	9.09	2.13	9.30	9.09	8.52
SNIPER	17.71	12.16	20.08	46.10	27.21	5.21	12.46	8.39	12.24	8.06	16.96
SSD	6.10	0.10	9.10	18.00	7.10	3.40	0.60	9.10	9.10	0.60	6.32
YOLOv3	4.03	1.22	1.27	9.44	1.94	4.67	2.36	4.78	1.43	0.95	3.20
RetinaNet	11.01	2.20	5.89	34.86	17.58	17.62	4.76	9.86	6.28	2.94	11.30
CenterNet	12.55	2.64	6.77	41.48	20.57	20.97	5.47	11.44	7.16	3.26	13.23

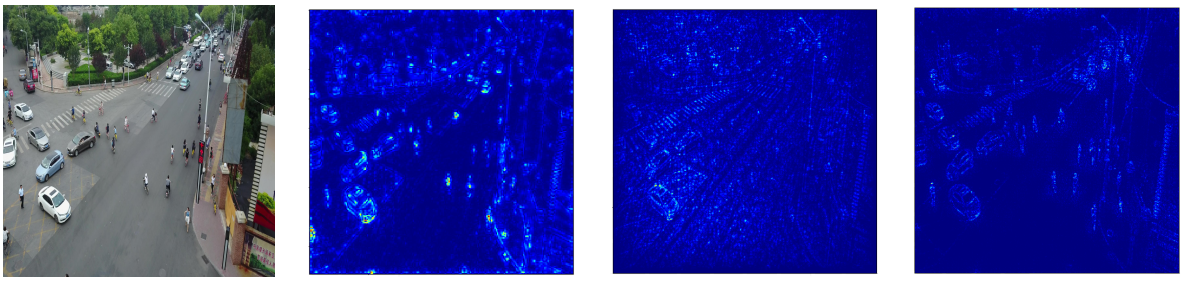
4.5. Analysis of Feature Maps Extraction

Figure 7 depicts feature maps of Faster R-CNN, RFCN, SSD, YOLO, SNIPER, RetinaNet, and CenterNet, on a real aerial image. We extracted feature maps at the final convolution layer. These feature maps offer deeper insights into how different methods capture object features from the aerial viewpoint.

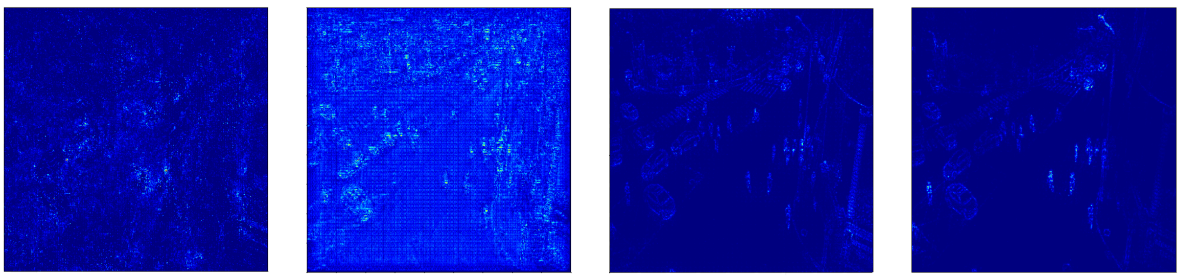
As also seen from Figure 7, SNIPER, RetinaNet as well as CenterNet produce optimal feature maps. We observe that the object shapes are well captured with larger values, whereas the background is with smaller values. This is because focal loss is adept at learning imbalanced classes (foreground/background) while chip mining is extracted from a proposal network trained for a short training schedule, which identifies regions where objects are likely to be present. Simultaneously, keypoint estimation of CenterNet is considered to be the essential factor that facilitates finding center points and regresses to all other object properties, such as size, location, orientation.

Regarding YOLO and SSD: in the feature map obtained by SSD, the object shapes are not clear, and the edges of objects are not preserved. This explains why this method detects aerial objects inaccurately which is ascribed to shallow layers in a neural network. Simultaneously, the features maps extracted by YOLO is better than SSD's ones where object regions are more prominent from the background. However, the edges of the objects are still blurred.

As far as Faster R-CNN and RFCN feature maps are concerned, the object shapes are well preserved but are not clearly distinguished from the background. This is due to the variance of various angles of images, which is considered to be an obstacle of early feature extractors.



(**a**) Input Image (**b**) Faster R-CNN (**c**) RFCN (**d**) SNIPER



(**e**) SSD (**f**) YOLOv3 (**g**) RetinaNet (**h**) CenterNet

Figure 7. Visualization of feature maps from different state-of-the-art object detection models.

4.6. Discussion

SSD is a unified object detector, which adopts a multi-scale approach. SSD uses a VGG16 network as a feature extractor and adds eight convolutional layers and ten layers separately, it also uses convolutional layers to reduce spatial dimension and resolution. To detect multi-scale objects, SSD makes independent object detections from multiple feature maps. Aspect ratios in SSD which are used as the anchor box scaling factors, so we widen the ratios range to ensure most objects could be captured. The higher resolution feature maps are responsible for detecting small objects, the first layer for object detection is conv4_3 which has a spatial dimension of 38×38, a pretty large reduction from the original input image. Furthermore, small objects can only be detected in left most feature maps. However, those maps contain low-level features, like edges or color patches that are less informative for classification. Shallow layers in a neural network may not generate enough high-level features to predict small objects [27]. Therefore, SSD usually performs worse for small objects compared to other detection methods. Although SSD is the penultimate detector, it achieves the second with 12.7% on the *bicycle* class, the third with 5.5% on the *people* class, 9.1% on the *awning-tricycle* class, 9.10% on the *bus* class and 4.6% on the *motor* class.

SSD is competitive with Faster R-CNN, RFCN on more substantial objects, which has poor performance on small objects. Faster R-CNN combines the Region Proposal Network (RPN) into Fast R-CNN. RPN produces box proposals based on the feature extractor. These box proposals are used to crop features from the same intermediate feature map. They fed to the remainder of the feature extractor to predict a class and refine box for each proposal. RFCN is similar to Faster R-CNN. RFCN crops features from the last feature layer before prediction to reduce the amount of computation. RFCN proposed a position-sensitive mechanism to keep translation variance for localization representations. Faster R-CNN has a mAP of 17.51%. Faster R-CNN ranks in the third with 16.96% on the *truck* class. RFCN has much better performance than Faster R-CNN and SSD, producing 19.55% AP. RFCN achieves the first with 28.10% on the *truck* class and 13.28% on the *bus* class, which ranks in the third with 19.45% on the *people* and 15.21% on the *awing-tricycle* class.

As far as the outstanding detectors are concerned, CenterNet, RetinaNet and SNIPER are the three algorithms that top the statistics in both IoU thresholds (0.5 and 0.75). CenterNet ranks first with 32.28%, followed by RetinaNet with 28.26% in case the threshold of IoU = 0.5. Paradoxically, in case the threshold of IoU = 0.75, which favors high accurate results, the figure for SNIPER overtakes that for CenterNet and achieves the best performance.

In particular, CenterNet achieves outstanding results in the benchmark dataset with both IoU thresholds. Please note that CenterNet object detector builds on keypoint estimation networks, finds object centers, and regresses to their size. The experimental results show that CenterNet works well with small IoU threshold, 0.5. Regarding SNIPER, the valid range, boxes, which the square root of their area lies in each range are marked as valid in that scale. Therefore, we increased the valid range to significantly detect objects in various sizes (small, medium and large objects) as shown in Table 1. Simultaneously, chip mining plays an important role in eliminating regions that are likely to contain the background and this measure could adapt to each viewpoint hence alleviating the drawback of diverse scales. As a result, these enhancements cooperate with the pyramid feature map to surpass other detectors in terms of average precision. In particular, at the 0.75 IOU threshold, SNIPER outperforms YOLOv3, with 16.96% and 3.2% respectively. This is mainly because YOLOv3 is inferior in terms of localization. Regarding Retina, by changing anchors, RetinaNet has an increment in terms of AP, 2.3% for *people* class. A scale adjustment has widened the variety of scaling factors to use per anchor location which could improve detection for the diverse size objects. Concurrently, the focal loss is designed to address a severe imbalance between foreground and background classes during training, as a result, this approach could tackle the problems of the unbalanced dataset, in which the number of training samples for car and bus classes outnumbers those from other classes.

5. Conclusion and Future Work

In this paper, we experimented the state-of-the-art object detection methods, namely Faster R-CNN, RFCN, SSD, YOLO, SNIPER, RetinaNet, and CenterNet, on aerial images. Among them, CenterNet, SNIPER and RetinaNet achieve the best performance in terms of average precision. Concurrently, YOLO is considered to be the optimal choice for real-time object detection applications which require the high FPS and moderate precision in detecting object. We notice the main challenges in the problem, for example, occlusion, scale, and class imbalance. From the aerial view, many objects are occluded, and their sizes are varied. We also notice the class imbalance issue during the training process. For example, most of the detectors perform much better on the car and pedestrian classes than on the awning-tricycle, tricycle, and bus classes due to more instances collected in the car and pedestrian classes.

In the future, we would like to investigate the fusion of different object detectors to even boost the state-of-the-art performance. In addition, we are interested in the task of aerial image segmentation. Obviously, the bounding boxes provided by the object detectors are very useful for the segmentation task. We also consider adopting the use of transfer learning [44] to consolidate and enhance the efficiency of training time.

Author Contributions: K.N. is responsible for discussion, paper writing and revising. N.T.H. and P.C.N. focus on training model, benchmark evaluation, and paper writing. K.-D.N. and N.D.V. are in charge of ideas, evaluation and paper writing. T.V.N. is responsible for ideas, discussion, paper structure, paper writing and revising. All authors have read and agreed to the published version of the manuscript.

Funding: This research is funded by Vietnam National University HoChiMinh City (VNU-HCM) under grant number C2018-26-03.

Acknowledgments: We are grateful to the NVIDIA corporation for supporting our research in this area. The authors would like to thank the editors and the reviewers for their professional suggestions.

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

Abbreviations

The following abbreviations are used in this manuscript:

PASCAL	Pattern Analysis, Statistical modelling and ComputAtional Learning
PASCAL-VOC	PASCAL Visual Object Classes Dataset
MS-COCO	Microsoft Common Objects in Context Dataset
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
ROI	Region of Interests
VGGNet	Visual Geometry Group Network
ZFNet	Zeiler Fergus Networks
CNN	Convolutional Neural Networks
R-CNN	Regional CNN
SSD	Single-Shot Detector
YOLO	You Only Look Once Detector
RFCN	Region-based Fully Convolutional Networks
SNIPER	Scale Normalization for Image Pyramids with Efficient Resampling

References

1. Zou, Z.; Shi, Z.; Guo, Y.; Ye, J. Object detection in 20 years: A survey. *arXiv* **2019**, arXiv:1905.05055.
2. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis.* **2015**, *115*, 211–252. [CrossRef]
3. Everingham, M.; Van Gool, L.; Williams, C.K.; Winn, J.; Zisserman, A. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results 2007. Available online: http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html/ (accessed on 5 October 2007)
4. Zhang, X.; Yang, Y.H.; Han, Z.; Wang, H.; Gao, C. Object class detection: A survey. *ACM Comput. Surv.* **2013**, *46*, 10. [CrossRef]

5. Everingham, M.; Van Gool, L.; Williams, C.K.; Winn, J.; Zisserman, A. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vis.* **2010**, *88*, 303–338. [CrossRef]
6. Everingham, M.; Eslami, S.A.; Van Gool, L.; Williams, C.K.; Winn, J.; Zisserman, A. The pascal visual object classes challenge: A retrospective. *Int. J. Comput. Vis.* **2015**, *111*, 98–136. [CrossRef]
7. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft coco: Common objects in context. *Comput. Vis. ECCV* **2014**, *8693*, 740–755.
8. Zhu, P.; Wen, L.; Du, D.; Bian, X.; Ling, H.; Hu, Q.; Wu, H.; Nie, Q.; Cheng, H.; Liu, C. VisDrone-VDT2018: The vision meets drone video detection and tracking challenge results. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 10 September 2018.
9. Zafeiriou, S.; Zhang, C.; Zhang, Z. A survey on face detection in the wild: Past, present and future. *Comput. Vis. Image Underst.* **2015**, *138*, 1–24. [CrossRef]
10. Dollar, P.; Wojek, C.; Schiele, B.; Perona, P. Pedestrian detection: An evaluation of the state of the art. *IEEE Trans. Pattern Anal. Mach. Intell.* **2011**, *34*, 743–761. [CrossRef]
11. Yang, Z.; Pun-Cheng, L.S. Vehicle detection in intelligent transportation systems and its applications under varying environments: A review. *Image Vis. Comput.* **2018**, *69*, 143–154. [CrossRef]
12. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25 (NIPS 2012)*; Neural Information Processing Systems Foundation, Inc.: Vancouver, BC, Canada, 2012; pp. 1097–1105.
13. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef]
14. Sermanet, P.; Eigen, D.; Zhang, X.; Mathieu, M.; Fergus, R.; LeCun, Y. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv* **2013**, arXiv:1312.6229.
15. 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, Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
16. Girshick, R. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 1440–1448.
17. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv* **2015**, arXiv:1506.01497.
18. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single shot multibox detector. *Comput. Vis. ECCV* **2016**, *9905*, 21–37.
19. Dai, J.; Li, Y.; He, K.; Sun, J. R-fcn: Object detection via region-based fully convolutional networks. In Proceedings of the 30th International Conference on Neural Information Processing Systems, Red Hook, NY, USA, 5–10 December 2016; pp. 379–387. Available online: https://arxiv.org/abs/1605.06409 (accessed on 21 June 2016)
20. 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.
21. 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. 7263–7271.
22. 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 (ICCV), Venice, Italy, 22–29 October 2017; pp. 2980–2988.
23. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
24. Singh, B.; Najibi, M.; Davis, L.S. SNIPER: Efficient multi-scale training. *arXiv* **2018**, arXiv:1805.09300.
25. Huynh, N.; Nguyen, P.; Vo, N.; Nguyen, K. Detecting Human from Space: An Evaluation of Deep Learning Modern Approaches. In Proceedings of the 15th International Conference on Multimedia Information Technology and Applications, University of Economics and Law, Vietnam National University, Ho Chi Minh City, Vietnam, 27 June 2019.
26. Zeiler, M.D.; Fergus, R. Visualizing and understanding convolutional networks. *Comput. Vis. ECCV* **2014**, *8689*, 818–833.
27. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *Int. Conf. Learn. Represent.* **2015**, arXiv:1409.1556.

28. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) recognition, Boston, MA, USA, 7–12 June 2015; pp. 1–9.
29. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
30. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4700–4708.
31. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *37*, 1904–1916. [CrossRef]
32. 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.
33. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 2117–2125.
34. Law, H.; Teng, Y.; Russakovsky, O.; Deng, J. Cornernet-lite: Efficient keypoint based object detection. *arXiv* **2019**, arXiv:1904.08900.
35. Duan, K.; Bai, S.; Xie, L.; Qi, H.; Huang, Q.; Tian, Q. Centernet: Keypoint triplets for object detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October–2 November 2019; pp. 6569–6578.
36. Zhou, X.; Zhuo, J.; Krahenbuhl, P. Bottom-up object detection by grouping extreme and center points. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 850–859.
37. Zhou, X.; Wang, D.; Krähenbühl, P. Objects as points. *arXiv* **2019**, arXiv:1904.07850.
38. Wu, X.; Sahoo, D.; Hoi, S.C. Recent advances in deep learning for object detection. *Neurocomputing* **2020**. [CrossRef]
39. Zhu, P.; Wen, L.; Bian, X.; Ling, H.; Hu, Q. Vision meets drones: A challenge. *arXiv* **2018**, arXiv:1804.07437.
40. Vo, N.D.; Nguyen, K.; Nguyen, T.V.; Nguyen, K. Ensemble of Deep Object Detectors for Page Object Detection. In Proceedings of the 12th International Conference on Ubiquitous Information Management and Communication (IMCOM '18), Langkawi, Malaysia, 5–7 January 2018; Association for Computing Machinery: New York, NY, USA, 2018. [CrossRef]
41. Redmon, J. Windows and Linux Version of Darknet Yolo v3 & v2 Neural Networks for Object Detection. 2019. Available online: https://github.com/AlexeyAB/darknet (accessed on 13 February 2020).
42. Gihub, F. Keras Implementation of RetinaNet Object Detection. 2019. Available online: https://github.com/fizyr/keras-retinanet. (accessed on 13 February 2020).
43. Ferrari, P. SSD300 Training Tutorial. 2019. Available online: https://github.com/pierluigiferrari/ssd_keras (accessed on 13 February 2020)
44. Chauhan, V.; Joshi, K.D.; Surgenor, B. Image Classification Using Deep Neural Networks: Transfer Learning and the Handling of Unknown Images. In Proceedings of the International Conference on Engineering Applications of Neural Networks, Crete, Greece, 24–26 May 2019; pp. 274–285. [CrossRef]