Hindawi
Journal of Electrical and Computer Engineering
Volume 2020, Article ID 3189691, 18 pages
https://doi.org/10.1155/2020/3189691

Research Article

An Evaluation of Deep Learning Methods for Small Object Detection

Nhat-Duy Nguyen, Tien Do, Thanh Duc Ngo, and Duy-Dinh Le

University of Information Technology, Vietnam National University, Ho Chi Minh City, Vietnam

Correspondence should be addressed to Tien Do; tiendv@uit.edu.vn

Received 20 January 2020; Accepted 11 March 2020; Published 27 April 2020

Academic Editor: Cesare F. Valenti

Small object detection is an interesting topic in computer vision. With the rapid development in deep learning, it has drawn attention of several researchers with innovations in approaches to join a race. These innovations proposed comprise region proposals, divided grid cell, multiscale feature maps, and new loss function. As a result, performance of object detection has recently had significant improvements. However, most of the state-of-the-art detectors, both in one-stage and two-stage approaches, have struggled with detecting small objects. In this study, we evaluate current state-of-the-art models based on deep learning in both approaches such as Fast RCNN, Faster RCNN, RetinaNet, and YOLOv3. We provide a profound assessment of the advantages and limitations of models. Specifically, we run models with different backbones on different datasets with multiscale objects to find out what types of objects are suitable for each model along with backbones. Extensive empirical evaluation was conducted on 2 standard datasets, namely, a small object dataset and a filtered dataset from PASCAL VOC 2007. Finally, comparative results and analyses are then presented.

1. Introduction

Object detection is known as a task that locates all positions of objects of interest in an input by bounding boxes and labeling them into categories that they belong to. To do this task, several ideas have been proposed from traditional approaches to deep learning-based approaches. The approaches of object detection are mainly separated into two types, namely, approaches based on region proposal algorithms known as two-stage approaches [1–3] and approaches based on regression or classification recognized as real-time and unified networks or one-stage approaches [4–7]. Applications based on real-time object detection now draw much attention of people because of its demand for meeting the modern life and helping people to have a better life. For example, self-driving cars are an authentic one to simultaneously help people transport on streets safely, reducing car accidents by distracted drivers. The other one includes that in manufacturing industries, the need of detecting assembly parts that are defective or the uncertainty of an angle of view, size of detected object, and deformable shape that significantly changes during assembly process [8]. It illustrates that real-time object detection, applied to the most popular vision-based applications in real world, is really indispensable. However, such applications require early object detection in order to be used subsequently as inputs for other tasks [9, 10]. Due to early detection, representation of objects is usually small or even tiny. Generally, given an image of interest, the purpose of small object detection is to immediately detect what common objects belong to the image, especially in small sizes, implying that objects of interest are objects which either own a physically big appearance but just occupy a small patch on an image (train, car, bicycle, etc.) [11, 12] or are really with a small appearance (mouse, plate, jar, bottle, etc.) [13], as shown in Figure 1.

Small object detection, therefore, is a challenging task in computer vision because apart from the small representations of objects, the diversity of input images also make the task more difficult. For instance, an image can be in different

(a) (b)

Figure 1: Illustration of (a) objects such as a bus, plains, or cars that have big appearance but occupy small parts on an image taken from [11] and (b) objects that really own small appearance such as mouses or plates taken from [13].

resolutions; if the resolution is low, it can hinder the detector from detecting small objects. In this case, the visual information to highlight the locations of small objects will be significantly limited. In addition, small objects can be deformable or are overlapped by other objects. A wide variety of detection methods have been proposed in the last years from the development of deep learning. Various ideas have been presented, and attached evaluations have been made to deal with challenges of object detection, but those proposed detectors currently spend their ability on the detection of normal sizes, not just small objects. However, an evaluation of small object detection approaches is indispensable and important in the study of object detection. Lately, object detection has significantly attracted attention from state-of-the-art approaches, and these have made their efforts to tackle object detection and yield good performance on challenging and multiclass datasets such as PASCAL VOC and COCO. These cutting-edge methods are firstly trained on ImageNet and transferred to detection; for example, in [2], the authors use a proposed network which applies a spatial pyramid pooling layer to extract features and compute these over an entire image regardless of image sizes instead of employing part-based models [14]. R-CNN [1] is a pioneer of breakthrough object detection and has several innovations from previous approaches; an image is resized to a fixed size to feed into the network and then applies an external algorithm to generate object proposals. Improved from [1], Fast R-CNN [3] applies regions of interest (RoIs) to extract a fixed-length feature from the feature maps for each proposal. Faster R-CNN [15] uses its own network to generate object proposals instead of applying an external algorithm.

So far, almost detection models are all well-performed on challenging datasets such as COCO and PASCAL VOC. These datasets commonly contain objects taking medium or big parts on an image that contains a few small objects which cause an imbalance data between objects in different sizes resulting in a bias of models to objects greater in numbers. In

addition, the number of classes of current small object datasets is less than common datasets. Besides, most of the state-of-the-art detectors, both in one-stage and two-stage approaches, have struggled with detecting small objects. As a result, we have presented an in-depth evaluation of existing deep learning models in detecting small objects in our prior work [16]. We evaluate three state-of-the-art models including You Only Look Once (YOLO), Single Shot MultiBox Detector (SSD), and Faster R-CNN with related trade-off factors, i.e., accuracy, execution time, and resource constraints. In this time, we make not only an extension by continually evaluating state-of-the-art and up-to-date detection models but also summarize pros and cons as well as the design of models rather than just introducing their idea. Instead of focusing on real-time models, we evaluate state-of-the-art models both in the one-stage approach which is able to run in real time such as YOLOv3, RetinaNet, and the two-stage approach which do not meet real-time detection but high accuracy such as Fast RCNN and Faster RCNN. We add these models to our evaluation due to some reasons, and we firstly take claims from the original work of these models. Particularly, we pick up YOLOv3 because this detector is a novel and state-of-the-art model, which combines current advanced techniques such as residual blocks, skip connections, and multiscale detection. Similarly, RetinaNet is a detector that proposes an updated calculation for loss function to penalize the imbalance of classes in a dataset. Although Faster RCNN is the only one model that is evaluated in our previous work, we want to evaluate this model with different backbones to consider how well backbones work when they are combined with Faster RCNN. Furthermore, Faster RCNN is an improvement of Fast RCNN, and we still add Fast RCNN to our evaluation because this model works with an external algorithm to generate region proposals on an input image instead of on a feature map alike Faster RCNN. Besides, we evaluate these models with different backbones such as ResNet 50, ResNet 101, ResNet 152, ResNeXT 101, and FPN on small objects to consider how well these backbones are when combining them with models. We still make our evaluation on 2 datasets namely, small object dataset [13] and our filtered dataset from PASCAL VOC 2007 [11] with criteria such as accuracy, speed of processing, and resource consumption as well. However, we want to provide analyses to the design and the way models work and explore how well models can afford with multiscale objects. This helps readers have a preference of each model, and from there, they can choose a suitable model to meet their needs. Therefore, the followings are our contributions:

(i) We made an extension for evaluating deep models in two main approaches of detection, namely, the one-stage approach and two-stage approach such as YOLOv3, RetinaNet, Fast RCNN, and Faster RCNN along with popular backbones such as FPN, ResNet, or ResNeXT.

(ii) We provided not only disadvantages and advantages of the models relating to accuracy, resource consumption, and speed of processing in context of small objects as well as changes of these factors when an object size is scaled up or down but also a comparison between one-stage and two-stage methods.

2. Challenges

Overall, there are several problems relating to challenges that need to be solved with object detection. Object detection itself draws much attention from researchers, but after a period of time, challenges just tackle a part; particularly, COCO challenges provide a standard in regard to small and medium detection, and accuracy in most of detectors is still low with this standard. Therefore, in terms of small object detection, it is harder to researchers because apart from normal challenges alike object detection, it owns particular challenges for small objects. Besides, the definition of small objects is not obviously clear. The following presentation make it more obvious.

2.1. Small Appearances. Recently, small object detection has been considered as an attractive problem itself because there are many sorts of its own challenges that are very intriguing to researchers. First of all, the possibilities of the appearance of small objects are much more than other objects because of the small size that leads to a fact that detectors get confused to spot these objects among plenty of other objects which are located around or even are the same size or appearance. It is arduous when differentiating small objects from the clutter of background. Furthermore, the pixels available to represent the information of small objects are also much fewer than normal objects. It means that there are less informative representatives for detectors to perform its task. Besides, key features to obtain small objects from an image are vulnerable and even lost progressively when going thorough many kinds of different layers of deep network such as convolutional or pooling layers. For example, in VGG16, if the object of interest occupies a 32×32 size, it will be presented at most 1 pixel after 5 times of going through the pooling block. As a result, the exhausted searching such as sliding window [14] or the drastic increase in the number of bounding boxes like selective search [17] is unfeasible to achieve good outputs. Some samples of small objects are shown in Figure 1.

2.2. Small Object Definitions. The definition problem of small object detection is to clarify how small scales or sizes of objects are or how many pixels they occupy on an image. This is arduous and different if we consider objects on images of high resolution and low resolution. For example, an object is assigned as a small object as occupying a part of 400×400 resolution on 2048×2048 but being very big on 500×500 one. Therefore, it causes a difficulty to researchers when a dataset consists of images with various ranges of resolution. Up till now, there are some definitions of small objects, and these definitions are not clearly defined. It depends upon datasets that are used for evaluation and characteristics of objects of interest. Therefore, to perform the task of detecting small objects, researchers define

different definitions for different datasets instead of only using the size of bounding boxes containing objects to consider if the objects are small or not. For example, Zhu et al. [18] mentioned that small objects are objects whose sizes are filling 20% of an image when releasing their dataset about traffic signs. If the traffic sign has its square size, it is a small object when the width of the bounding box is less than 20% of an image and the height of the bounding box is less than the height of an image. In [19], Torralba et al. supposed small objects are less than or equal to 32×32 pixels. In small object dataset [13], objects are small when they have mean relative overlap (the overlap area between bounding box area and the image is) from 0.08% to 0.58%, respectively, 16×16 to 42×42 pixel in a VGA image. In this work, we reuse the above definitions, especially the definitions from [13, 18] as the main references because they are reliable resources and are widely accepted by other researchers.

2.3. Datasets and Approaches. There are limited works to concentrate on sorts of small objects, and it results in the limitation of experience and knowledge to deeply go for a comprehensive research. The previous approaches just specify to focus on big objects and ignore the existence of small objects. In fact, we do not comprehend how much existing detection approaches are well-performed when dealing with small objects. Hence, in this work, we conduct to assess the performance of existing state-of-the-art detectors to draw a general picture of their abilities for small object detection.

In terms of small object detection, there are just a few works regarding the problem of detecting small objects. So far, most of these works are just designed to detect some single categories such as traffic signs [18] or vehicles [20–22] or pedestrians [23] that do not contain common or multiclass datasets in real world. This results in a lack of evaluation for the approaches to show its ability detecting different kinds of objects and variation of their shapes as well. Fortunately, Chen et al. [13] present their small object dataset by combining the Microsoft COCO [12] and SUN datasets [24] that consist of common objects such as "mouse," "telephone," "switch," "outlet," "clock," "tissue box," "faucet," "plate," and "jar." Chen also augments the R-CNN algorithm with some modifications to improve performance of detecting small objects. Following this idea, we conduct a small survey on existing datasets and the authors find that PASCAL VOC is in common with COCO and SUN datasets which consist of small objects of various categories. So we depend on existing and common definitions of small objects to filter objects that meet these definitions and form a dataset including 4 subsets corresponding to 4 different definitions of small objects so as to objectively consider how different scales of objects affecting performance of detection are. In addition, there is recently a small object dataset in a challenge called Vision Meets Drones: A Challenge (http://aiskyeye.com/), and this dataset is considered the challenging dataset because it consists of several small objects, even tiny objects in images in different contexts and conditions in wild, but the views in images are snapshot from drones which fly above and take pictures from the high resolution cameras attached to it. Unfortunately, this dataset does not have annotations for testing, so it is hard to take it for evaluation.

Therefore, in this work, we choose small object dataset [13] and our filtered dataset to make our evaluation because these datasets contain common objects and the number of images are large, so the evaluations are objective.

3. Deep Models for Object Detection

Recently, in widespread developments of deep learning, it is known that convolutional neural network (CNN) approaches have showed lots of improvements and achieved good results in various tasks. Therefore, it is commonly applied to well-known works. Most of the works have showed significant improvements in detecting objects filling medium or big parts on an image.

RCNN [1] is one of the pioneers. The following methods are an improvement form of R-CNN such as [2, 3, 15]. Especially, Faster R-CNN [15] is considered as a state-of-the-art approach. Although this sequence of advanced works uses a lot of different and breakthrough ideas from sliding window to object proposals and mostly achieves the best results as state-of-the-art methods on challenging datasets such as COCO, PASCAL VOC, and ILSVRC, however, their representations take much time to run on an image completely and may lead to reduction in the running performance of the detector. As a result, the detectors face difficulty in using them for detecting objects in real time despite achieving high accuracy. This means they just focus on accuracy and ignore effects of speed of processing. In addition, detecting objects having small sizes in real world is as important as objects having big or medium sizes, even more necessary than we imagined. Especially, in industries of automotive, smart cars, army projects, and smart transportation, data must be promptly and precisely processed to make sure that safety is first. But in these cases, generally, the data recorded usually are far from our position and the information is a small thing.

In terms of real-time detection, the one-stage methods, instead of using object proposal to get RoI before moving to classifier like two-stage approaches such as Faster R-CNN, use local information to predict objects such as YOLO and SSD. Both methods process images in real time and detect objects correctly and still have a high point of mAP. Nevertheless, these papers just mention that the models can detect small objects and have good results, but they do not show evidences to prove how much or what extent of small objects that are solved. In this work, we evaluate these models from both approaches to find out their performance and to what extend they are good at as detecting small objects. The following are general ideas of above-mentioned approaches.

3.1. R-CNN. R-CNN [1] is a novel and simple approach as a pioneer advanced, providing more than 30% mean average precision (mAP) than the previous works on PASCAL VOC.

The overview of R-CNN architecture consists of four main phases which are known as the new advances of this method. Firstly, the R-CNN network resizes an image to 227×227 and takes it as an input. Then, selective search algorithm [17] is applied to the image and generates 2000 candidates of proposed bounding boxes as the warped regions used for the input of CNN feature network. Through the regions, the network extracts a 4096-dimensional feature vector from each region and then computes the features for each region. Finally, using the class-specific linear SVM classifier behind the last layer is to classify regions to consider if there are any objects and what the objects are.

The major key to the success of the R-CNN is the features matter. In R-CNN, the low-level image features (e.g., HOG) are replaced with the CNN features, which are arguably more discriminative representations. However, the evaluation of an image is extremely costly and wasteful because R-CNN must apply the convolutional network 2000 times. Besides, resizing the input to the low 227×227 is a problem affecting small objects which are easy to deform or even lose information as changing the resolution far from its original sizes. The region proposals overlapped, thus leading to computation of familiar features many times, and with every region proposal, it must be stored to disk before performing the extraction of features. In addition, lots of bounding boxes overlapped will result in a drop of mAP if small objects are close to big objects because there is a bias to choose the bounding boxes which contain big objects and ignorance of bounding boxes for small objects.

3.2. Spatial Pyramid Pooling (SPP). The primary ideas of SPP [2] are motivated from limitations of CNN architecture, such as the original CNN receiving the size of input images must be a fixed size (224×224 of AlexNet), so the actual use of the raw picture often needs cropping (a fixed-size patch that truncates the original image) or warping (RoI of an image input must be a fixed size of the patch). The fully connected layer needs a fixed-length input and convolutional layer that can be adapted to the arbitrary input size; thus, it needs a bridge as a mediate layer between the convolutional layer and the fully connected layer and that is the SPP layer. Particularly, SPP-net firstly finds 2000 candidates of region proposals like the R-CNN method and then extracts the feature maps from the entire image. SPP maps each window of the features corresponding to region proposals as a fixed-length representation regardless of the input size. Finally, 2 fully connected layers are used to classify by SVM.

In short, SPP-net versus R-CNN: detection task is better 100× faster than R-CNN, but training time is very slow because of multistage training steps (fine-tuning of last layers, SVM, and regressions) and really taking a lot of disk space to save vectors of features.

3.3. Fast R-CNN. Fast R-CNN [3] is an advanced method that presents various innovations to improve the time of training and testing phase and efficiently classifying object proposals while still increasing the accuracy rate by using deep convolutional networks. The architecture of Fast R-CNN is trained end-to-end with a multitask loss. Specifically, the convolutional network takes an image at any size as an input and several RoIs. Instead of applying RoI on an input and wrapping them to feed into the network at the first step like RCNN, Fast RCNN applies these RoIs on a feature map after the several convolutional layers of the base network. Each RoI is extracted a fixed-size feature vector by a pooling layer and mapped to a feature vector by fully connected layers. The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets.

The most important feature of RoI is sharing computation and memory in the forward and backward passes from the same image. The huge contribution of Fast R-CNN is that it proposes a new training method that fixes the drawbacks of R-CNN and SPP-net, while increasing their running time and accuracy rate. The advantage is the mean average precision of detection is higher than R-CNN and SPP-net. Training phase is a single stage, using a multitask loss, and can update the entire network layers. The capacity of disk storage is not required for feature caching.

3.4. Faster R-CNN. Faster R-CNN [15] is an innovated approach improved from Fast R-CNN. Unlike two previous approaches of its own, instead of generating bounding boxes by external algorithms [17] like [1, 3], Faster R-CNN runs its own method called the region proposal network (RPN) which is trained end-to-end in order to give the generation of highly qualified region proposals. After gaining deep features from early convolutional layers, RPN is taken into the account and windows slide over the feature map to extract features for each region proposal. RPN is considered as a fully convolutional network which simultaneously predicts bounding boxes of objects and objectness scores at each position. The input of RPN is an image of any size and outputs a set of bounding boxes as rectangular object proposals, along with an objectness score for each proposal. Specifically, the RPN takes the image feature map of the fifth convolutional layer (conv5) as an input and applies a 3×3 sliding window on the feature map. Then, the intermediate layer will feed into two different branches, one for object score (determines whether the region is thing or stuff) and the other for regression (determines how should the bounding box change to become more similar to the ground truth). The RPN improves accuracy and running time as well as avoids to generate excess of proposal boxes because the RPN reduces the cost by sharing computation on convolutional features. RPN and Fast R-CNN are merged into a single network by sharing their convolutional features. This combination helps Faster R-CNN to have leading performance on accuracy but leads to its architecture as a two-stage network which reduces the speed of processing of this method.

3.5. You Only Look Once. Inherited from the advantages of the previous models which have been introduced earlier, You Only Look Once (YOLO) [4] is considered as a state-of-the-art object detection in real time with various categories at that time. YOLO currently has three versions [4–6], which are improved substantially through each version progressively.

The detail analyses of the YOLO approaches as a premise to apply it into practical applications are as follows:

YOLOv1 [4] is widely known that YOLO, an unified or one-stage network, is a completely novel approach based on an idea that aims to tackle object detection in real time proposed by Redmon et al,. implying that, instead of performing the task of object detection like the previous techniques based on complex tasks such as [1, 4], which use exhausted sliding window and then feed outputs of this to classifiers performing at equally spaced locations over the whole image or region proposals to generate bounding boxes which possibly contain objects and then feed them to convolutional neural networks, YOLO considers object detection to be a regression problem, simultaneously giving the prediction for various coordinates of bounding boxes and class probabilities for these boxes. The key idea to perform the detection of YOLO is that YOLO separates images into grid views which push the running time as well as accuracy in localizing objects of YOLO. The goal of YOLO is to deal with two problems, namely, what objects are presented and where they are in an image. The summarization of YOLO operation proceeds with three principal steps simply and straightforwardly. Firstly, YOLO takes an input image resized to a fixed size, then works a single convolutional network as a unified network on the image, and ultimately puts a threshold on the resulting detections by the confidence score of the model. YOLO runs at 45 fps on GPU and the smaller Fast YOLO reaches 150 fps. This processing can run steaming video in real time. Although the design of YOLO architecture affords end-to-end training and real-time detection, it still keeps high average precision.

The network divides the input image into a $S \times S$ grid, where $S \times S$ is equal to the width and height of the tensor which presents the final prediction. In case the center of an object is in a grid cell, the gird cell takes responsibility for detecting that object. Moreover, each gird cell is simultaneously responsible for predicting bounding boxes and confidence scores which present how confident the model of bounding box contains an object as well as how accurate it indicates the bounding box is predicted.

The drawback of YOLO is that it lags behind the state-of-the-art detection systems about accuracy but is better than those about running time. It makes less than half the number of background errors compared to Fast R-CNN. YOLO is highly generalizable, so it can quickly identify objects in an image, but it usually struggles to precisely localize some objects, especially small ones. Therefore, the author introduced YOLOv2 to improve performance and fix drawbacks of YOLO as well.

YOLOv2 [5] has a number of various improvements from YOLOv1. Similarly to the origin, YOLOv2 runs on different fixed sizes of an input image, but it introduced several new training methods for object detection and classification such as batch normalization, multiscale training with the higher resolutions of input images, predicting final detection on higher spatial output, and using good default bounding boxes instead of fully connected layers.

However, this offers a trade-off between speed and accuracy. The details of the mAP improvements in PASCAL VOC 2007 are shown in Figure 2.

These novel improvements allow YOLOv2 to train on multiclass datasets like COCO or ImageNet. In addition, it was attempted to train the detector to detect over 9000 different object classes. YOLOv2 uses a network architecture customized from the original network. YOLOv2 mainly concentrates on a way of improving recall and localization while still receiving high accuracy of classification in comparison with state-of-the-art detectors, and the origin YOLO significantly makes more localization errors but is far less likely to predict false detections on places where nothing exists. Although YOLOv2 has accuracy improvements, YOLOv2 does not work well on small objects because the input downsampling results in the low dimension of the feature map which is used for the final prediction. To solve these problems, recently, the author introduces YOLOv3 with significant improvements on object detection, especially on small object detection. Generally, a variety of latest networks tend to be toward deeper and yield good performance on their tasks with deep features learned from numerous layers.

YOLOv3 [6] is one of these approaches; instead of using Darknet-19 like two old versions [4, 5], YOLOv3 develops a deeper network with 53 layers called Darknet-53 and combines the network with state-of-the-art techniques such as residual blocks, skip connections, and upsampling. The residual blocks and skip connections are very popular in ResNet and relative approaches, and the upsampling recently also improves the recall, precision, and IOU metrics for object detection [25]. For the task of detection, 53 more layers are stacked onto it, giving a 106-layer fully convolutional underlying architecture for YOLOv3. This is the reason behind the slowness of YOLOv3 compared to YOLOv2.

Second, YOLOv3 enables the detector to predict objects at three different outputs with three different scales rather than just one prediction at the last layer of the network similar to its competitor SSD [26] which has improved a lot of performance on a low resolution image. This is useful to pick up diverse outcomes in order to improve performance of detection. The final output is created by applying a 1×1 kernel on a feature map. Particularly, the detection is done by applying 1×1 detection kernels on feature maps of three different sizes at three different places in the network partly similar to feature pyramid networks (FPNs) [27].

Third, YOLOv3 still keeps using K-means to generate anchor boxes, but instead of fully applying 5 anchor boxes at the last detection, YOLOv3 generates 9 anchor boxes and separates them into 3 locations. Each location applies 3 anchor boxes; hence, there are more bounding boxes per image. For example, if we have an image of 416×416, YOLOv2 predicts $13 \times 13 \times 5 = 845$ boxes; in YOLOv3, the number of boxes is 10647, implying that YOLOv3 predicts 10 times the number of boxes compared to YOLOv2.

Fourth, YOLOv3 also changes the way to calculate the cost function. If the anchor overlaps a ground truth more than other bounding boxes, the corresponding objectness score should be 1. For other anchor boxes with overlap greater than a predefined threshold 0.5, they incur no cost. Each ground truth is only associated with one boundary box. If a bounding box is not assigned, it incurs no classification

FIGURE 2: The visualization of detectors with the strongest backbones on subsets of PASCAL, VOC_MRA_0.58, VOC_MRA_10, VOC_MRA_20, and VOC_WH_20 in order, respectively: (a) YOLO Darknet-53; (b) Faster RCNN ResNeXT-101-64 × 4d-FPN; (c) RetinaNet ResNeXT-101-64 × 4d-FPN; (d) Fast RCNN ResNeXT-101-64 × 4d-FPN.

and localization lost, just confidence loss on objectness. The loss function in previous YOLO looks like

$$\begin{aligned}
&\lambda_{\mathbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\mathrm{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\mathbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\mathrm{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\mathrm{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\mathrm{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\mathrm{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} 1_i^{\mathrm{obj}} \sum_{c \in \mathrm{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2.
\end{aligned} \tag{1}$$

Currently, instead of using mean square error in calculating the classification loss at the last three terms, YOLOv3 uses binary cross-entropy loss for each label. In other words, YOLOv3 makes its prediction of an objectness score and class prediction for each bounding box using logistic regression.

There is no more softmax function for class prediction. The reason is that the most currently used classifiers assume that predicted labels are independent and mutually exclusive implying that if an object belongs to one class, then it cannot belong to the other and this is solely true if output prediction is really mutual, nevertheless, in case dataset has multilabel classes and there are labels which are not nonexclusive such as pedestrian and person. At the time, the sum of possibility scores may be greater than 1 if the classifier is softmax, so YOLOv3 alternates the classifier for class prediction from the softmax function to independent logistic classifiers to calculate the likeliness of the input belonging to a specific label.

3.6. Single Shot MultiBox Detector. Single Shot MultiBox Detector (SSD) [26] is a single shot detector using a single and one-stage deep neural network designed for object detection in real time. By comparison, the state-of-the-art method in two-stage processing, Faster RCNN, uses its proposed network to generate object proposals and utilizes those to classify objects in order to be toward real-time detection instead of using an external method, but the whole process runs at 7 FPS. SSD enhances the speed of running

time faster than the previous detectors by eliminating the need of the proposal network. Therefore, it causes a few drop in mAP, and SSD compensates this by applying some improvements including multiscale features and default boxes. These improvements allow SSD to gain the same of Faster RCNN using lower resolution images, which then further speeds up the processing of SSD. For 300×300 input image as the best version, SSD gets 77.2% mAP at 46 FPS better than Faster R-CNN 73.2% and a little smaller than the best version of YOLOv2, 554×554 input image, and 78.6% mAP at 40 FPS on VOC 2007 on Nvidia Titan X.

Similarly, SSD consists of 2 parts, namely, extraction of feature maps and use of convolution filters to detect objects. SSD uses VGG16 as a base network to extract feature maps. Then, it combines 6 convolutional layers to make prediction. Each prediction contains a bounding box and $N+1$ scores for each class, where N is the number of classes and one for extraclass for no object. Instead of using a region proposal network to generate boxes and feed to a classifier for computing the object location and class scores, SSD simply uses small convolution filters. After the VGG16 base network extracts features from feature maps, SSD applies 3×3 convolution filters for each cell to predict objects. Each filter gives an output including $N+1$ scores for each class and 4 attributes for one boundary box.

SSD has a difference from previous approaches at the same time, and it makes prediction on multiscale feature maps for detection independently rather than just one last layer. The CNN network spatially reduces the dimension of the image gradually, leading to the decrease in the resolution of the feature maps. As mentioned, SSD uses a lower input image to detect objects; hence, early layers are used to detect small objects and lower resolution layers to detect larger scale objects progressively. Besides, SSD applies different scales of default boxes to different layers, and for intuitive visualization in Figure 3. Particularly, the only blue default box on 8×8 feature map fits to the ground truth of the cat, and the only red one on 4×4 feature map matches to the ground truth of the dog.

Although SSD has significant improvements in object detection as integrating with these above parts, SSD is not good at detecting small objects which can be improved by adding deconvolution layers with skip connections to introduce additional large-scale context [28]. Generally, SSD outperforms Faster RCNN, which is a state-of-the-art approach about accuracy, on PASCAL VOC and COCO while running at real-time detection.

3.7. CNN Drawbacks. Most of the CNN models are currently designed by the hierarchy of various layers such as convolutional and pooling layers that are arranged in a certain order, not only on small networks but also on multilayer networks to state-of-the-art networks. Along with these layers, fully connected layers are added behind and known as FC layers. The block consisting of FC layers and previous layers is designated as feature extractors, and it outputs key features of objects of interest as an input for classifiers coming behind. However, deeply going through many kinds of layers is a way that is not good for small object detection because in the task of small object detection, objects of interest are objects owning small sizes and appearance. Besides, small objects, unlike normal or big objects which are less affected by resizing the image or passing lots of different layers, are very vulnerable to the changes in image sizes. When an image passes a convolutional layer, the size of the image will be decreased by receptive fields that slide over the image to extract useful features. This does not affect small objects if there are just a few layers, but in a CNN network, we have many layers like this, and it is very hard for small objects. Still, if small objects just go through convolutional layers, it will not be anything to mention. Small objects which just have a few informative presence have to pass pooling layers, which help in avoiding, overfitting, and reducing computational costs by decreasing a number of parameters. To do this, these layers use fixed sliding windows that care about a fixed target that is identified before such as maximum or average calculations of valuables. For these reasons, GAN is an approach that may alter the CNN approach because of its advantages. We can take advantages of a way that the approach generates data to overcome the limitations of data of small objects for the training phase. Although images still have to pass layers such as convolutional and pooling layers, in this context, the network just has less layers compared to others. Bai el al. [29] have proposed to apply MTGAN to detect small objects by taking crop inputs from a processing step made by baseline detectors such as Faster RCNN [15] or Mask RCNN [9].

Because of mentioned reasons and following the survey [30], Liu et al. have presented numerous works of survey and evaluation, but there are no works that do with small objects in them. Therefore, in this work, we assess popular and state-of-the-art models to find out pros and cons of these models. Particularly, we evaluate 4 deep models such as YOLOv3, RetinaNet, Fast RCNN, and Faster RCNN with several base networks for small object detection with different scales of objects. In these models, YOLOv3 and RetinaNet belong to the one-stage approach; Fast RCNN and Faster RCNN are in the two-stage approach. We choose these models because YOLOv3 is the model with combination of state-of-the-art techniques, and RetinaNet is the model with a new loss function which penalizes the imbalance of classes in a dataset. Besides, we choose RetinaNet to make comparisons between models in the same approach. Similarly, Fast RCNN and Faster RCNN are the same, and both models are in the same approach and have nearly the similar pipeline in object detection. There is a difference is that Fast RCNN utilizes an external proposal to generate object proposals based on input images. However, Faster RCNN proposes its own network to generate object proposals on feature maps, and this makes Faster RCNN train end-to-end easily and work better.

4. Experimental Evaluation

In this section, we present the information of our experimental setting and datasets which we use for evaluation.

	YOLO								YOLOv2
Batch norm?		✓	✓	✓	✓	✓	✓	✓	✓
Hi-res classifier?			✓	✓	✓	✓	✓	✓	✓
Convolutional?				✓	✓	✓	✓	✓	✓
Anchor boxes?				✓	✓				
New nework?					✓	✓	✓	✓	✓
Dimension priors?						✓	✓	✓	✓
Location prediction?						✓	✓	✓	✓
Passthrough?							✓	✓	✓
Multiscale?								✓	✓
Hi-res detector?									✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	78.6

Figure 3: mAP of YOLOv2 at each added part [5].

4.1. Experimental Setting. We continually train and evaluate various object detectors on the two datasets such as PASCAL VOC [11] and a newly generated dataset [16]. The evaluated approaches in this time consist of Faster RCNN [15], YOLOv3 [6], and RetinaNet [7] with different backbones. Except for YOLOv3, the others are trained and evaluated by the Detectron python code.

Currently, the original datasets which commonly are used in object detection are PASCAL VOC [11] and COCO [12]. Both datasets are constructed by almost large objects or other kinds of objects whose size fill a big part in the image. These two datasets are not suitable for small object detection. In addition, there is another dataset, which is large-scale, and includes a lot of classes for small object detection, collected by drones, and named VisDrone dataset [31]. However, it does not publish the labels for test set to evaluate, and the views of images are topdown which is not our case. As a result, in order to evaluate the detection performance of the models, we use a dataset which was published in [13]. This dataset is called small object dataset which is the combination between COCO [12] and SUN [24] dataset. There are 10 classes in small object dataset including mouse, telephone, switch, outlet, clock, toilet paper (t. paper), tissue box (t. box), faucet, plate, and jar. The whole dataset consists of 4925 images in total, and there are 3296 images for training and 1629 images for testing. The mouse class owns the largest number of instances in images: 2137 instances in 1739 images, and the tissue box class has the smallest number of instances: 103 instances in 100 images. Apart from the small object dataset, we also filter subsets from PASCAL VOC 2007 following standard definitions. On PASCAL VOC, there are 20 classes, but with small object detection, there are fewer classes on strict definitions of small objects. Table 1 lists the details of the number of small objects and images containing them for subsets of the dataset.

We trained all models on small object dataset with the same parameters. Particularly, in the training phase, we trained the models with $70k$ iterations with the parameters including momentum, decay, gamma, learning rate, batch size, step size, and training days in Table 2. At the first moment, we attempted to start off the models with a higher learning rate 10^{-2}, but the models diverged leading to the loss value being NaN or Inf after 100 first iterations. Then, we tried at a lower learning rate 10^{-3} at 100 first iterations and rise to 10^{-2} to consider if the models can converge as starting off at a lower learning rate. However, it remained unchanged anything. We also saw that the models converged quickly during $10k$ first iterations with 10^{-3} and then progressively slow down after $20k$. Therefore, we decided to start off the training with a learning rate at 10^{-3} and decrease to 10^{-4} and 10^{-5} at $25k$ and $35k$ iterations, respectively. This setting shows that the loss value was stable from $40k$, but we set the training up to $70k$ to consider how the loss value changes and saw that it did not change a lot after $40k$ iterations. We tried to evaluate the models from $30k$ to $70k$, and generally, the performance of the models was not stable after $40k$ iterations. For this reason, we picked up the weight for evaluation at $30k$ and $40k$ iterations. At $30k$ iterations, YOLO achieves the best results and others get the best one at $40k$ iterations. In case of subsets of PASCAL VOC 2007, we combine train and valid set from PASCAL VOC 2007 and 2012 to form a training set. PASCAL VOC 2012 works as a data augmentation set for PASCAL VOC 2007. We use this combined training set to train all models and test them on subsets. All models train the same parameter. First, due to the limitation of memory, we rescale all the size of images to the same size with the shortest side 600 and the lengthiest side 1000 as in [15].

In YOLOv3, we run the K-means clustering algorithm in order to initialize 9 suitable default bounding boxes for training and testing phases of our selected datasets, and we changed the anchors value. The following are 9 anchors for small object dataset after running the K-means algorithm: [10.3459, 14.4216], [26.2937, 19.0947], [21.4024, 36.3180], [47.9317, 29.1237], [40.4932, 63.7489], [83.6447, 51.3203], [72.2167, 119.9181], [172.7416, 117.0773], and [124.6597, 252.8465].

In Faster R-CNN, to fairly compare with the prior work and deploy on different backbones, we also reuse directly the anchor scales and aspect ratios following the paper [13] such as anchor scales = 16×16, 40×40, and 100×100 pixels and aspect ratio = 0.5, 1, and 2, instead of having to cluster a set of default bounding boxes similar to YOLOv3. Similarly, in RetinaNet, we keep the default setting for training such as gamma loss = 2.0, alpha loss = 0.25, anchor scale = 4, and scalers per octave = 3 because of following authors, and this configuration is the optimized valuables.

4.2. Our Newly Generated Dataset. In this time, to have an objective comparison, we also use our newly generated dataset, and the information of this dataset is shown in Table 1. We use it to consider the effects of object sizes among factors including models, time of processing, accuracy, and resource consumption. The dataset consists of 4 subsets filtered from PASCAL VOC 2007 such as

Table 1: The information of the subsets.

Subsets	Classes	Images	Instances
VOC_MRA_0.58	16	329	529
VOC_MRA_10	20	2231	5893
VOC_MRA_20	20	2970	7867
VOC_WH_20	18	1070	2313

Table 2: The parameters of models.

Method	Momentum	Decay	Gamma	Learning_rate	Batch_size	Training_days	Stepsize
YOLOv2 [16]	0.9	0.0005		0.001	8	5	25000
YOLOv3	0.9	0.0005		0.001	32	3–4	25000
SSD300 [16]	0.9	0.0005	0.1	0.000004	12	9	40000, 80000
SSD512 [16]	0.9	0.0005	0.1	0.000004	12	12	100000, 120000
RetinaNet	0.9	0.0005	0.1	0.001	64	4->12 h	25000, 35000
Fast RCNN	0.9	0.0005	0.1	0.001	64	4->12 h	25000, 35000
Faster RCNN	0.9	0.0005	0.1	0.001	64	4->12 h	25000, 35000

VOC_WH_20, VOC_MRA_0.58, VOC_MRA_10, and VOC_MRA_20, and detail information is provided as follows:

(i) VOC2007_WH_0.2 contains objects whose width and height are less than 20% of an image's width and height. This one has fewer than PASCAL VOC 2007 two classes such as dining table and sofa because of the constraint of the definition.

(ii) VOC_MRA_0.58, VOC_MRA_10, and VOC_MRA_20 compose objects occupying the maximum mean relative area of the original image under 0.58%, 10%, and 20%, respectively. Two of them have the same number of PASCAL VOC 2007 classes except for VOC_MRA_0.58 and the one has fewer four classes such as dining table, dog, sofa, and train.

5. Results and Analyses

In this section, we show results that we achieved through the experimental phase. All models mentioned in this section except for models cited from other papers are trained on the same environment and 1 GPU: Ubuntu 16.04.4 LTS, Intel (R) Xeon (R) Gold 6152 CPU @ 2.10 GHz, GPU Tesla P100. In addition to the comparative accuracy, other comparisons are also provided to make our objective and clear assessment results.

5.1. Accuracy

5.1.1. Small Object Dataset. Following the detection results in Table 3, methods which belong to two-stage approaches outperform ones in one-stage approaches about 8–10%. Specifically, Faster RCNN with ResNeXT-101-64 × 4d-FPN backbone achieved the top mAP in two-stage approaches and the top of the table as well, 41.2%. In comparison with the top in one-stage approaches, YOLOv3 608 × 608 with Darknet-53 obtained 33.1%. Following [32], methods based on region proposal such as Faster RCNN are better than methods based on regression or classification such as YOLO and SSD. Actually, this is also right once again as in context of small object dataset.

For methods in each approach. Firstly two-stage approaches, Faster RCNN, which is an improvement of Fast RCNN, is only greater than Fast RCNN about 1–2% but only for ResNeXT backbones and equal to Fast RCNN for the rest. The difference here is not too much, and it means that the performance of external region proposal like selective search combined with ROI pooling is as good as internal region proposal like RPN with ROI aligned in this case. Besides, compared to R-CNN, we perceive that there is a boost 8–10% when RoI pooling or RoI aligned is added because R-CNN, which uses region proposals from selective search, then feeds them into the network and directly computes features from fc (fully connected) layers, only receives 23.5% with Alexnet, and 24.8% with VGG16 combined with proposals from RPN. However, Fast RCNN and Faster RCNN with two kinds of RoIs are much better. Fast RCNN receives accuracy in a range of 31.7% to 39.6% based on different backbones. Similarly, Faster RCNN gets 30.1% to 41.2%. Secondly, in one-stage approaches, YOLO outperforms SSD and RetinaNet. However, YOLO gets the highest outcome 33.1%, and SSD and RetinaNet get 11.32% and 30%, respectively. YOLO and SSD are considered as state-of-the art methods in speed and sacrificing accuracy. However, there is a large difference in accuracy between YOLO and SSD; the difference here is that SSD adds multiple convolutional layers behind the backbone, and each layer has their own ability instead of using 2 fully connected layers like YOLO. Although RetinaNet is assigned into a method in one-stage approaches, it cannot run in real time. RetinaNet is one which is proposed to deal with the imbalance between foreground and background by the focal loss. Therefore, RetinaNet obtains a higher accuracy in comparison with others except for YOLOv3 (Darknet-53).

When it comes to the backbones, we realized that Darknet-53 is the best in one-stage and real-time methods and even far higher than ResNet-50 although it similarly has the same layers with ResNet-50. In contrast, ResNeXT combined with FPN is the most powerful one in both one-stage and two-

stage methods if we only consider accuracy. Overall, there is an increase about 1–3% for changing the simple backbone to the complex one in each type. For example, when switching from original ResNet to ResNet-FPN, the accuracy is boosted from 2 to 3%. This is clear that leveraging the advantages from multiscale features of FPN is a common way to improve detection and tackle the scale imbalance of input images and bounding boxes of different objects. Similarly, we switch ResNeXT-101-32 × 8d-FPN to ResNeXT-101-64 × 4d-FPN, and the accuracy changes from 40.5% to 41.2% for Faster RCNN and from 38.7% to 39.6% for Fast RCNN. However, when considering between ResNet-50-FPN and ResNet-101-FPN, the growth only happens in Fast RCNN from 33.3% to 35.5%. There is a little bit decrease 0.1% for Faster RCNN. This reduction also happens with RetinaNet, while the simpler backbone ResNeXT-101-32 × 8d-FPN gets 30%, and the ResNeXT-101-64 × 4d-FPN just gets 25.1%. It means that the very deeper backbones do not guarantee the increase in accuracy, and the reason is that an advantage of a deeper network needs more parameters to learn. It means the authors must have a large number of data to feed into the network to train and update parameters itself, but in this case, the data of small object dataset are not abundant too much to fit the very deep network and hence increasing the chances of overfitting. Besides, features, which are originally from the early layer of ResNet, are not well-generalized because when they are combined with FPN, the accuracy has an improvement about 2–3%. When YOLO switches from Darknet-19 to Darknet-53, it really boosts the accuracy. The highest accuracy belongs to the Darknet-19 with the resolution of 1024 × 1024 which just gets 24.02%. However, YOLO 608 × 608 with Darknet-53 gets 33.1%. The explanation for this reason is that YOLOv3 with Darknet-53 has several improvements from Darknet-19, YOLOv3 has 3 location of scales to predict objects, especially one specialized in small objects instead of only one like Darknet-19, and it is also integrated cutting-edge advantages such as residual blocks and shortcut connections. The reduction in accuracy happens again with YOLO when switching from ResNet-101 to ResNet-152 about 1–2%. In these methods, YOLO and SSD are the only ones which allow multiple input sizes. The higher the resolution of input images are, the higher accuracy the method receives. The reason is that a higher resolution image allows more pixels to describe the visual information for small objects. However, if the resolution is far from the original size of images, it results in a decrease in accuracy. For example, YOLO 1024 × 1024 with Darknet-19 gets a lower accuracy than the resolution of 800 × 800. In addition, we have tried to increase in resolution of Darknet-53 from 608 to 1024, and the mAP decreases when the resolution is over 608 × 608. Therefore, the effect of image size is clear for models like SSD and YOLO. Generally, all comparative results of mAP on this dataset have the domination of classes very great in numbers, and this is caused by the imbalance data between the number of images and instances in these images. For example, according to the statistics in [13], mouse is a major class significantly contributing to mAP in Table 3 with the highest number of instances and images as well. However, tissue has least contribution with the lowest AP originally affected by the number of data. Furthermore, the imbalance data lead models tending to detect frequent objects, implying that models will misunderstand objects having a nearly similar appearance with the domination class as the objects of interest rather than less frequent objects. As a result, false positives will increase by these problems. Figure 4 illustrates the detection with strongest backbones. Following this visualization, the domination of the classes such as mouse or faucet results in misdetection with areas which have a same appearance to them. This misunderstanding has a tendency to weaker backbones in the comparison and one-stage method like YOLO which primarily heads to speed has more misdetection than two-stage methods. A reason that causes these problems are the difference in the way of training deep networks [33]. One-stage methods such as YOLO use a soft sampling method that uses a whole dataset to update parameters rather than only choosing samples from training data. However, two-stage methods such as RCNN family tend to employ hard sampling methods that randomly sample a certain number of positive and negative bounding boxes to train its network.

5.1.2. Subsets of PASCAL. With 4 subsets of 4 different scales of objects in images, we want to find out how much the scales impact on the models. The whole results are shown in Table 4. We separate the results into 2 groups as the one-stage and two-stage approaches, and Figure 5 is a visualization for the strongest backbones in each method on subsets.

In case of different scales like our subsets, there is a difference between one-stage approaches and two-stage approaches. In this case, methods from the one-stage approach have a better performance than two-stage ones in most of scales. This is really the opposite of small object dataset. Specifically, two-stage methods are totally better than one-stage ones in case of real-time inputs and just better a bit than nonreal-time models in VOC_WH20 about 10–20% and the same result with smaller objects in VOC_MRA_0.058 and VOC_MRA_0.10. However, in bigger objects in VOC_MRA_0.20, methods in one-stage approaches have significant outcomes rather than two-stage ones. In addition, there is just Faster RCNN that has good performance in most cases to compare to methods in one-stage ones. Fast RCNN is only good at big objects in VOC_MRA_0.20 and fails to have good detection in smaller objects.

In the one-stage approach, in methods which allow multiple inputs like YOLO and SSD, there are 2 kinds, namely, ones that can run in real time and the others that cannot, if the resolution is over 640 or 512 for YOLO and SSD, respectively. For real-time ones, YOLO outperforms SSD for all scales of objects. Specifically, YOLOv2 with Darknet-19 is better than SSD 26% with objects in VOC_MRA_0.058 and VOC_MRA_0.10 and 4–15% for larger objects in VOC_MRA_0.20 and VOC_WH_20. YOLOv3 with Darknet-53 gets higher results about 3–5% in comparison with YOLOv2; hence, YOLOv3 also gets higher results compared to SSD. However, if we consider nonreal-time input images, SSD is greater than YOLO with objects in VOC_MRA_0.10. However, RetinaNet, which is the one that cannot run in real time in the one-stage approach, performs the same results compared to ones in nonreal time in YOLO

Table 3: Comparative results on small object dataset.

Method	Backbone	Clock	Faucet	Jar	Mouse	Outlet	Plate	Switch	Tel.	t. box	t. paper	mAP
YOLO 416 [16]	Darknet-19	22.8	30.8	4	52	20.4	13.1	13	6.1	0	35.3	19.39
YOLO 448 [16]		23	36.9	9	52.5	18.4	13.6	17.5	4.2	0	34.3	20.13
YOLO 480 [16]		34.2	37.3	9.1	53.3	21.4	13.6	15.8	9.1	9.1	34.2	23.71
YOLO 512 [16]		23.1	36.6	6.1	59.8	24.6	14.2	15.7	9.1	4.5	32.4	22.61
YOLO 554 [16]		23.4	37.2	9.1	60.1	27.2	13.4	19.9	9.1	4.5	34.5	23.84
YOLO 640 [16]		20.2	36.2	3.2	59.8	27.8	11.7	18.1	8.2	4.5	35.6	22.53
YOLO 800 [16]		27.6	36	2.3	60.2	32.8	13.1	23.3	9.1	9.1	26.7	24.02
YOLO 1024 [16]		21.7	29.3	1.4	58.3	26.4	11.8	17.5	9.1	9.1	15.7	20.03
YOLO 320	Darknet-53	26.22	38.38	4.55	56.46	36.42	13.34	24.8	10.65	4.55	42.96	25.83
YOLO 416		28.47	47.15	10.83	60.49	43.15	15.87	**30.73**	15.15	2.62	48.3	**30.28**
YOLO 608		29.98	47.89	10.76	65.88	**48.02**	18.09	31.22	14.62	**17.99**	46.56	33.1
YOLO 320	ResNet-50	19.57	25.73	0.67	45.17	14.37	9.38	13.84	9.09	9.09	23.7	17.06
YOLO 416		23.78	36.65	0.4	54.23	18.37	13.75	19.78	9.84	9.42	35.68	22.19
YOLO 608		26.92	40.65	1.77	61.86	29.18	15.04	20.24	10.09	13.29	36.01	25.5
YOLO 320	ResNet-101	20.52	27.9	0.57	44.68	16.98	13.05	13.66	9.66	9.09	24.36	18.05
YOLO 416		25.72	35.6	3.03	55.73	22.4	15.61	17.26	9.32	3.03	38.71	22.64
YOLO 608		28.79	44.59	9.42	62.18	33.34	15.53	23.88	13.24	15.83	39.17	28.6
YOLO 320	ResNet-152	21.64	27.56	3.03	48.06	17.39	11.12	14.51	9.09	4.55	31.88	18.88
YOLO 416		25.7	36.54	0.89	53.81	20.6	14.13	20.21	11.49	0.29	33.06	21.67
YOLO 608		26.01	44.54	4.55	61	31.76	13.02	22.67	12.35	9.93	39.99	26.58
SSD300 [16]	ResNet-101	5.5	9.1	0	25.5	6.1	4.5	0	4.5	9.1	18.2	8.25
SSD300 [16]	VGG16	9.1	17.1	0	26.1	9.1	9.1	0	4.5	0	16.7	9.16
SSD512 [16]	VGG16	9.1	17.1	0	43	9.1	9.1	9.1	9.1	0	7.6	11.32
RetinaNet	ResNet-50-FPN	30.7	49.3	2	65.5	21.3	16.1	8.5	12.9	1	25.7	23.3
RetinaNet	ResNet-101-FPN	30.6	48.7	7.1	64.7	20	15.9	11.8	10.7	2.9	38.7	25.1
RetinaNet	ResNeXT-101-32 × 8d-FPN	**35.5**	**55**	**12.1**	**66.5**	23.9	**18.4**	9.8	**16.2**	9.4	**53.7**	30
RetinaNet	ResNeXT-101-64 × 4d-FPN	31.4	50.2	8.9	66.3	20.8	15.3	9.4	14	2.2	32.4	25.1
R-CNN [13]	RPN prop. + VGG16	31.9	31.3	4.2	56.8	31.1	9.3	14.2	16.4	23.4	29.4	24.8
R-CNN [13]	Alexnet, 7×, 300 pro	32.4	27.2	5.1	56.9	28	9.8	13.6	12.4	17.9	35.6	23.9
R-CNN [13]	VGG16, 7×, 300 pro	37.3	30.3	7.2	60.6	41.5	15.8	21.5	13.7	22	33.3	28.4
R-CNN [13]	ContextNet (Alexnet, 7×)	32.7	26.8	4.6	56.4	26.3	9.9	12.9	12.2	18.7	34	23.5
Fast RCNN	ResNet-50-C4	32.4	46.3	6.5	65.8	38.3	20.1	25.3	16.6	14.1	52	31.7
Fast RCNN	ResNet-50-FPN	37.4	47.3	7.3	68.9	46.7	21	32.1	17.1	9.3	45.9	33.3
Fast RCNN	ResNet-101-FPN	39.3	50.3	10.6	68.3	47.1	20.4	33.3	18.6	15.4	51.4	35.5
Fast RCNN	ResNeXT-101-32 × 8d-FPN	47.5	54.8	10.3	71.8	54	21.4	34.4	21.7	17.7	53.5	38.7
Fast RCNN	ResNeXT-101-64 × 4d-FPN	45.4	55.7	10.9	72.5	53.3	24	36.9	22.9	16	58.1	39.6
Faster R-CNN [16]	VGG16	23.76	37.65	8.03	54	16.16	11.88	15.12	9.1	6.25	37.29	21.92
Faster RCNN	ResNet-50-C4	32.2	44.6	6.6	65.9	35.2	17.5	25.7	19.6	13.7	40	30.1
Faster RCNN	ResNet-50-FPN	35.7	49.9	7.3	68.4	48.9	18.8	29.6	14.7	11.4	53.3	33.8
Faster RCNN	ResNet-101-FPN	39.8	49.2	4.9	68.2	47	18.5	29.7	14	12.9	52.2	33.7
Faster RCNN	ResNeXT-101-32 × 8d-FPN	*49.8*	56.6	11.4	72.1	*56.3*	23.2	*37*	*20.8*	18.8	58.7	40.5
Faster RCNN	ResNeXT-101-64 × 4d-FPN	49.6	*58.6*	12.2	*72.5*	54.5	*23.2*	36.9	20.8	*20.1*	*63.1*	*41.2*

The values in bold represent the best in one-stage methods, and the ones in italics represent the highest in two-stage methods.

and better than SSD. RetinaNet is more stable than SSD and YOLO when the scales are changed. The bigger the objects are, the more the stability is. For example, the change is so much about 33% when the scale increases from objects in VOC_MRA_0.058 to ones in VOC_MRA_0.10 and VOC_MRA_0.20. However, this change is not much about 10% with bigger objects in comparison with YOLO 15–25%. In case of YOLO, this remarkable increase in accuracy when objects are larger is obviously good for a model. The change in SSD resembles the change in RetinaNet.

Concerning resolutions in YOLO and SSD, we see that when image resolution is increased, they push the accuracy to improve in general. For YOLOv2 with Darknet-19 and YOLOv3 with Darknet-53 and SSD, they all have an increase in accuracy when the resolution is large, except for YOLOv2 with objects belonging to VOC_MRA_0.10 and VOC_MRA_0.20 when the image is over 800. In addition, YOLOv2 has a fluctuation with those objects in VOC_WH20. As mentioned in our previous work, YOLO is better than SSD in those objects less than 10% of the images; however, in this case, YOLOv3 is good at all scales of objects. This is because YOLOv3 has 3 detection locations coming with more ratios of default boxes, and it leads to a significant outcome when combining results from 3 locations.

When we switch to the two-stage approaches, Faster RCNN has a significant improvement in most scales rather than Fast RCNN except for objects in VOC_MRA_0.20 which have the same accuracy. This shows that if objects are completely separated into different scales, the RoI pooling does not work well with smaller objects and ones in VOC_WH20. In

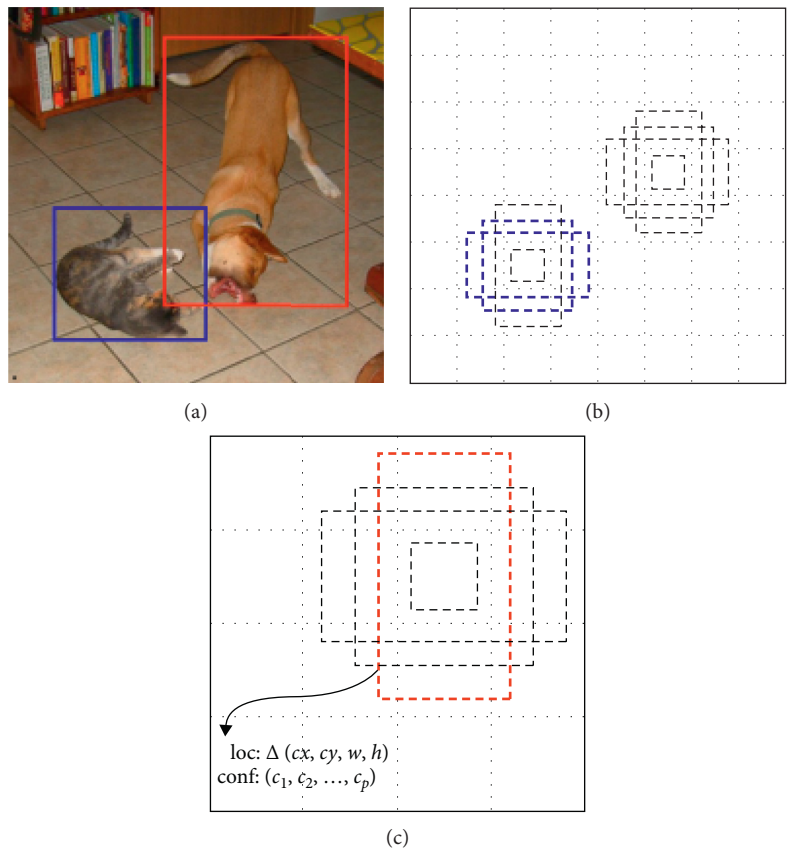


Figure 4: The location of the default boxes in different scales: (a) image with GT boxes; (b) 8×8 feature map; (c) 4×4 feature map.

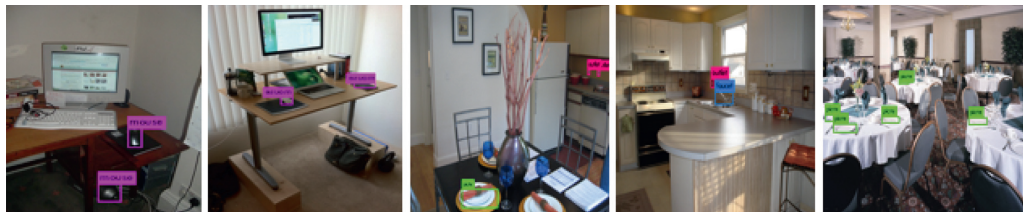
addition, if we compare with one-stage methods, it is significantly lower than them. However, RoI align along with RPN is well performed when scales are changed. When it comes to the backbones, there is a few decrease in accuracy when changing from ResNet-50-FPN to ResNet-101-FPN or from ResNeXT-101-32 × 8d-FPN to ResNeXT-101-64 × 4d-FPN with objects from all scales for both Faster RCNN and Fast RCNN. The VGG16 backbone has an impressive outcome rather than strong backbones such as ResNet or ResNeXT. Although the accuracy is less than two strong backbones, VGG16 is still better with objects in VOC_WH20 and has a few change in accuracy when changing objects with big sizes.

5.2. Time Processing and Resource Consumption. Tables 5 and 6 show us the performance comparison of the evaluated models with base networks that belong to the models. Generally, we see that when RAM consumption in testing and training increases, more layers are added. This means that if the network is more deeper, the need of processing also increases because this leads to the increase in parameters and time to process data as well. YOLO is the model consuming the least memory in both two-phase training and testing. Particularly, YOLO is only from 4G to 5G for training and from 1.6G to 1.8G for testing with Darknet-53. YOLO is the only one which is able to run in real time. YOLO just needs about 0.3 ms to 0.4 ms to process an image in comparison to more than 0.1 s and 0.2 s with Faster RCNN and RetinaNet. This allows us to pick up these models on devices which own the modest memory. While RetinaNet is assigned to the one-stage approach, it is not good enough to meet real-time detection. The inference time in Fast RCNN is lower a little bit than Faster RCNN and RetinaNet. In contrast, the RAM consumption in training and testing of RetinaNet is lower than Fast RCNN and Faster RCNN. Of all architectures, the ResNet-50-C4 is the one requiring the highest memory and time to process data because the output size of ResNet-50-C4 is bigger a bit than others [9]. If we consider ResNet or ResNeXT combined with FPN, Faster RCNN is over 100 Mb compared to Fast RCNN and 300 Mb with RetinaNet. In addition, according to Table 2, the number of training days of Faster RCNN and RetinaNet need less time for training only a few hours to 1 day rather than YOLO 3–4 days. This demonstrates that if we pay our attention to performance and do not have much time for training, we choose Faster RCNN or RetinaNet instead of YOLO one. In contrast, if we only focus on processing speed and still achieve good performance, one-stage methods are always the good one. In the same context of backbones, RetinaNet uses a lower resource than Fast RCNN and Faster RCNN about 100 Mb and 300 Mb for Fast RCNN and Faster RCNN, respectively, in testing time. However, the training time of RetinaNet uses much memory more than Fast RCNN about 2.8 G and Faster RCNN about 2.3 G for ResNeXT-101-32 × 8d-FPN and ResNeXT-101-64 × 4d-FPN. If we consider this on

Table 4: The comparative results on subsets of PASCAL VOC 2007.

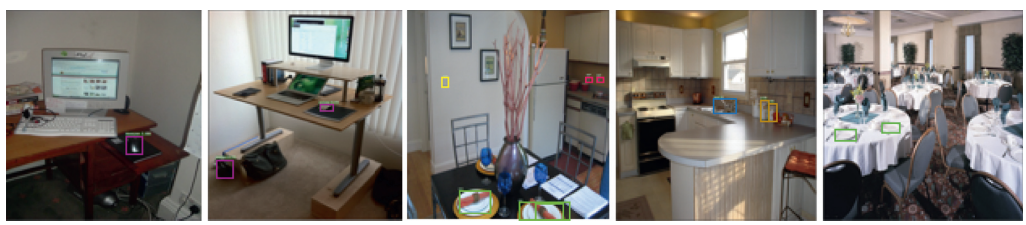
Approach	Method	VOC_MRA_0.058	VOC_MRA_0.10	VOC_MRA_0.20	VOC_WH20
One stage	YOLOv2 416 [16]	3.02	31.38	42.89	18.52
	YOLOv2 448 [16]	4.47	32.9	60.15	21.96
	YOLOv2 480 [16]	4.26	33.48	60.78	26.67
	YOLOv2 512 [16]	5.42	35.74	61.12	24.63
	YOLOv2 544 [16]	6.97	36.56	63	26.62
	YOLOv2 640 [16]	7.7	37.97	61.29	23.41
	YOLOv2 800 [16]	10.24	37.3	61.91	26.9
	YOLOv2 1024 [16]	10.69	29.93	55.14	28.97
	YOLOv3 320	7.18	34.58	60.36	20.4
	YOLOv3 416	10.2	38.97	62.53	24.12
	YOLOv3 608	11.7	42.65	**68.56**	28.86
	SSD 300 [16]	1.71	32.76	46.26	16.91
	SSD 512 [16]	2.9	43.46	57.11	19.87
	RetinaNet-ResNet-50-FPN	8.84	41.5	50.2	28.14
	RetinaNet-ResNet-101-FPN	8.95	42.5	51.9	27.46
	RetinaNet-ResNeXT-101-32 × 8d-FPN	10.29	45.4	54.5	30.08
	RetinaNet-ResNeXT-101-64 × 4d-FPN	**10.71**	**45.5**	55.1	**31.32**
Two stage	Fast RCNN-ResNet-50-C4	0.23	13.2	49.9	3.93
	Fast RCNN-ResNet-50-FPN	0.63	13.5	55.6	3.45
	Fast RCNN-ResNet-101-FPN	0.39	15.9	57.6	3.12
	Fast RCNN-ResNeXT-101-32 × 8d-FPN	0.51	14.4	*57.9*	3.33
	Fast RCNN-ResNeXT-101-64 × 4d-FPN	0.29	14.2	57.3	3.76
	Faster RCNN-ResNet-50-C4	6.98	39.9	48.7	26.04
	Faster RCNN-ResNet-50-FPN	10.74	45.6	56.3	29.79
	Faster RCNN-ResNet-101-FPN	10.63	46.9	57.6	30.57
	Faster RCNN-ResNeXT-101-32 × 8d-FPN	*11.64*	*47.3*	57.6	32.12
	Faster RCNN-ResNeXT-101-64 × 4d-FPN	10.54	47.1	56.9	31.64
	Faster RCNN-VGG16 [16]	5.73	35.58	44.14	*41.11*

This table illustrates how well models adapt to different scales of objects. The values in bold represent the best in one-stage methods, and the ones in italics represent the highest in two-stage methods.



(a)

(b)



(c)

Figure 5: Continued.

(d)

Figure 5: Highlight of bounding boxes from comparative backbones on small object dataset. We here select YOLO with Darknet-53 and ResNet-50 for objective comparison because there have obviously the same layers in their networks along with the significant techniques such as skip connections and residual blocks. The bounding boxes show that ResNet-50 has the sensitivity to areas which resembles the objects of interest than Darknet-53. Similarly, ResNet-50-FPN and ResNet-50-C4 are chosen to consider. The detection shows that combining ResNet-50 with FPN outputs a better performance rather than the original one. Particularly, misdetection happens in more density than ResNet-50-FPN such as in columns 4 and 5. Zoom in to see more detail.

Table 5: The comparison of consumption on small object dataset.

Model	Backbone	Inference time (s)	Test RAM (MiB)	Train RAM (MiB)
YOLOv3	Darknet-53	0.0331	1825	4759
YOLOv3	ResNet-50	**0.027**	1285	3479
YOLOv3	ResNet-101	0.0356	1829	5383
YOLOv3	ResNet-152	0.0454	2443	7531
RetinaNet	ResNet-50-FPN	0.102	2075	4435
RetinaNet	ResNet-101-FPN	0.127	2723	5577
RetinaNet	ResNeXT-101-32 × 8d-FPN	0.229	3767	**7863**
RetinaNet	ResNeXT-101-64 × 4d-FPN	0.292	3719	7813
Fast RCNN	ResNet-50-C4	0.3	6449	5877
Fast RCNN	ResNet-50-FPN	0.089	2277	4455
Fast RCNN	ResNet-101-FPN	0.113	2947	5627
Fast RCNN	ResNeXT-101-32 × 8d-FPN	0.212	3987	4961
Fast RCNN	ResNeXT-101-64 × 4d-FPN	0.269	3885	4799
Faster RCNN	ResNet-50-C4	0.412	**6609**	6129
Faster RCNN	ResNet-50-FPN	0.101	2387	5381
Faster RCNN	ResNet-101-FPN	0.124	3001	6487
Faster RCNN	ResNeXT-101-32 × 8d-FPN	0.256	4027	5333
Faster RCNN	ResNeXT-101-64 × 4d-FPN	0.286	4003	5246

Table 6: The comparison of consumption on subsets filtered from PASCAL VOC.

Model	Backbone	Inference time (s)	Test RAM (MiB)	Train RAM (MiB)
YOLOv3	Darknet-53	**0.027**	1645	4079
RetinaNet	ResNet-50-FPN	0.1	1935	4133
RetinaNet	ResNet-101-FPN	0.116	2585	5435
RetinaNet	ResNeXT-101-32 × 8d-FPN	0.222	3641	**7723**
RetinaNet	ResNeXT-101-64 × 4d-FPN	0.284	3561	7599
Fast RCNN	ResNet-50-C4	0.495	6371	5677
Fast RCNN	ResNet-50-FPN	0.092	2131	4387
Fast RCNN	ResNet-101-FPN	0.114	2819	5463
Fast RCNN	ResNeXT-101-32 × 8d-FPN	0.213	3873	4637
Fast RCNN	ResNeXT-101-64 × 4d-FPN	0.265	3735	4575
Faster RCNN	ResNet-50-C4	0.26	**6141**	5991
Faster RCNN	ResNet-50-FPN	0.1	2245	5207
Faster RCNN	ResNet-101-FPN	0.13	2855	6335
Faster RCNN	ResNeXT-101-32 × 8d-FPN	0.225	3943	5087
Faster RCNN	ResNeXT-101-64 × 4d-FPN	0.276	3885	4909

small object dataset, it does not work too much because RetinaNet is lower than Faster RCNN about 10% in performance. Otherwise, on different scales of subsets, RetinaNet works well when comparing to Faster RCNN, and the difference is just 2–4% percentages. Although ResNet backbones combined with the others yield an improvement in accuracy, they do not work for YOLO on small object datasets. YOLO with Darknet-53 utilizes more resource

than ResNet ones, but it has the best accuracy among models. Therefore, we only test YOLO with Darknet-53 in subsets of PASCAL.

5.3. Analyses of the Trade-Off among Detectors. Network designs and approaches alike the one-stage approach prove its performance as applying them to detect general objects both small scales and other kinds of scales. Although they are fast and accurate, there is still a drawback always existing in these models, that is, the trade-off between accuracy and speed of processing. For example, YOLOv3 proposes the idea that performs detection at three different scales, and this result is obviously impressive and yields good performance. However, to gain this advantage, YOLOv3 has to sacrifice the time to process. Instead of all inputs of the model normally processing one time for detection like YOLOv2, this idea must work 3 times. This trade-off is also partly affected by resolution as we change it during training or testing our models. In our previous work, we have mentioned that we have to choose a right resolution to ensure our models to work properly. In case of the two-stage approaches, the idea that proposes region proposals to improve the localization of objects to serve for detection is good as well. This is useful, but we have to take it into the account that we should generate proposals on feature maps or directly on input images because this affects a lot on the way, which models intend to run and identify representations of objects. If objects are normal or have a big or medium appearance, it is good for models to work, but if objects are in multiscales, this is a problem to consider and research deeply in order to balance the performance as well as improve it. Therefore, to partly fix this problem, the one-stage approach allows us to choose a fixed size of an input for training and testing, but the support still depends on characteristics of datasets which we evaluate or the image size. After all, all models we choose to evaluate are affected by the scales of objects when we change the scale, and accuracy of models change a lot, except for Faster RCNN, the only one model that seems to be stable with the scale, especially when combining with the VGG16 architecture. Although the accuracy of VGG16 is not better than the other architectures, the difference here is that it does not change too much in accuracy. This is only right for big objects having the overlap of the bounding box and the image greater than 10%; if not, this is not assured.

Figure 1 shows that the possibility of small objects is more than other objects. The black length of the camera is somehow similar to the black mouse placed on a mouse pad. This possibility of small object presence causes more difficulties to detectors and leads to wrong detection. Anywhere in an image can be small objects, it results in a fact that detectors have much wrong detection with familiar appearance which they have seen. If we consider the visualization of the detection in Figure 4, the wrong detection is partly similar to the appearance of the other objects in the dataset. This problem is caused by the data imbalance between classes and instances in each class which originally is known as the foreground-foreground class imbalance. In other words, the common problems, which not only happen with small objects but also for whole datasets, are the intraclass similarity and interclass variation.

6. Conclusion

Small object detection is a challenging and interesting problem in the task of object detection and has drawn attention from researchers, thanks to the development of deep learning which is motivation to improve performance of tasks in computer vision. Although deep models belonging to detection originally tend to solve problems relating to general object detection, they still work at a particular level to the success of small object detection. As evaluation works on small object detection for deep models, our goal is to highlight remarkable achievements of popular and state-of-the-art deep models in order to provide a variety of views as applying deep models in small object detection. Particularly, we evaluate state-of-the-art real-time detectors based on deep learning from two approaches such as YOLOv3, RetinaNet, Fast RCNN, and Faster RCNN on two datasets, namely, small object dataset and subsets filtered from PASCAL VOC about effects of different factors objectively including accuracy, execution time, and resource usage.

In spite of the successful achievements in recent years, the performance of detection has improved significantly, and there is still a huge gap in accuracy between normal objects and small objects. In the criteria of the COCO dataset, the difference from the small scale to medium and big scale is too much. Most models are good at detection of normal objects, and problems are going to happen when applying them to detect small objects. As a result, to reduce the gap of small object detection, the first thing to do is invest datasets which have massive amount of data to train models and have a wide range of categories to compete with the human visual system alike [12, 34].

So far, detection models are divided into two main approaches, namely, one-stage approach and two-stage approach. Models in the one-stage approach is known as detectors which have better and more efficient detection in comparison to another approach. The efficiency here has the potential power to run in real time and is able to apply them to practical applications. However, the trade-off between accuracy and speed is a difficult challenge which needs to be taken into the account in order to balance the gap. However, models in the two-stage approach have their reputation of region-based detectors which have high accuracy but are too low in speed to apply them to real world. This drawback comes from the computation of networks.

Through our evaluation, there is a fact that architectures which are utilized as base networks to extract deep features have significant effects on frameworks. The deeper the architecture is, the higher the accuracy of detection is. Once a network has an increase in the depth, this means it has more layers than normal ones, and it will have massive parameters to train. Hence, this needs a lot of data to fine tune these parameters reasonably. If there is an increase in computation, resource consumption will also increase. As a result, it

will be difficult as we want to take them to apply in practical applications. Besides, the contextual exploit in models is definitely limited, this results cause ignoring much useful and informative data in training, especially in context of small objects. Because, small objects are able to appear anywhere in an input image, if the image is well-exploited with the context, the performance of small object detection will be improved better.

For all above reasons and according to our evaluation, if we tend to have good performance and ignore the speed of processing, two-stage methods like Faster RCNN are well-performed and demonstrate its network design with the different datasets on many contexts of objects including multiscale objects. Therefore, Faster RCNN is considered as a giant baseline in order to base on or develop from it. If our target has a balance of accuracy and speed, YOLO is a good one in case we do not care the training time because the sacrifice between the speed and accuracy is worth applying it into practical applications. Otherwise, Faster RCNN or RetinaNet is still a substitution to work on. When it comes to backbones, we have to concern about the data to choose a reasonable backbone to combine with the methods. Because the amount of data will significant impact on the model, if data are not abundant, the shallow network will fit it well. Besides, there is recently a novel approach promising in training deep models with less data that is weakly supervised learning such as zero-shot, one-shot, or few-shot learning. Therefore, these approaches will be considered in our future works, and following our recent searching to have better performance on object detection, we have to consider several factors to improve the mAP such as multiscale training, superresolution for scaling up the visual information to small objects [35], or preprocessing data to avoid the imbalance data because we have a wide range of imbalance problems relating to data [33].

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was funded by the Vietnam National University, HoChiMinh City (VNU-HCM), under grant no. B2017-26-01.

References

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

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *Proceedings of the European Conference on Computer Vision*, pp. 346–361, Springer, Zurich, Switzerland, September 2014.

[3] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1440–1448, Santiago, Chile, December 2015.

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision And Pattern Recognition*, pp. 779–788, Las Vegas, NV, USA, June 2016.

[5] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," 2016, https://arxiv.org/abs/1612.08242.

[6] J. Redmon and A. Farhadi, "YOLOv3: an incremental improvement," 2018, https://arxiv.org/abs/1804.02767.

[7] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE transactions on pattern analysis and machine intelligence*, Venice, Italy, October 2018.

[8] K. Židek, A. Hosovsky, J. Pitel', and S. Bednár, "Recognition of assembly parts by convolutional neural networks," in *Advances in Manufacturing Engineering and Materials*, pp. 281–289, Springer, Cham, Switzerland, 2019.

[9] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE International Conference on computer vision (ICCV)*, IEEE, Venice, Italy, pp. 2980–2988, October 2017.

[10] L.-C. Chen, A. Hermans, G. Papandreou et al., "Instance segmentation by refining object detection with semantic and direction features," 2017, https://arxiv.org/abs/1712.04837.

[11] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (VOC) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[12] T.-Y. Lin, M. Maire, S. Belongie et al., "Microsoft COCO: common objects in context," in *Proceedings of the European Conference on Computer Vision*, pp. 740–755, Springer, Zurich, Switzerland, September 2014.

[13] C. Chen, M.-Y. Liu, O. Tuzel, and J. Xiao, "R-CNN for small object detection," in *Proceedings of the Asian Conference on Computer Vision*, pp. 214–230, Springer, Taipei, Taiwan, November 2016.

[14] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[15] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," in *Proceedings of the 28th International Conference on Neural Information Processing Systems, Ser. NIPS'15*, pp. 91–99, MIT Press, Cambridge, MA, USA, 2015, http://dl.acm.org/citation.cfm?id=2969239.2969250.

[16] P. Pham, D. Nguyen, T. Do, T. D. Ngo, and D.-D. Le, "Evaluation of deep models for real-time small object detection," in *Proceedings of the International Conference on Neural Information Processing*, pp. 516–526, Springer, Guangzhou, China, November 2017.

[17] J. R. R. Uijlings, K. E. A. Van De Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, 2013.

[18] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li, and S. Hu, "Traffic-sign detection and classification in the wild," in *Proceedings of the IEEE Conference on Computer Vision And*

Pattern Recognition, pp. 2110–2118, Vegas, NV, USA, June 2016.

[19] A. Torralba, R. Fergus, and W. T. Freeman, "80 million tiny images: a large data set for nonparametric object and scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 11, pp. 1958–1970, 2008.

[20] A. Kembhavi, D. Harwood, and L. S. Davis, "Vehicle detection using partial least squares," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 6, pp. 1250–1265, 2011.

[21] V. I. Morariu, E. Ahmed, V. Santhanam, D. Harwood, and L. S. Davis, "Composite discriminant factor analysis," in *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, pp. 564–571, IEEE, Steamboat Springs, CO, USA, March, 2014.

[22] A. Andreas, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Proceedings of the IEEE Conference on Computer Vision And Pattern Recognition*, Providence, RI, USA, June 2012.

[23] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: human trajectory prediction in crowded spaces," in *Proceedings of the IEEE Conference on Computer Vision And Pattern Recognition*, pp. 961–971, Las Vegas, NV, USA, June 2016.

[24] J. Xiao, K. A. Ehinger, J. Hays, A. Torralba, and A. Oliva, "Sun database: exploring a large collection of scene categories," *International Journal of Computer Vision*, vol. 119, no. 1, pp. 3–22, 2016.

[25] E. Dong, Y. Zhu, Y. Ji, and S. Du, "An improved convolution neural network for object detection using YOLOv2,"in *Proceedings of the 2018 IEEE International Conference on Mechatronics and Automation (ICMA)*, pp. 1184–1188, IEEE, Changchun, China, August 2018.

[26] W. Liu, D. Anguelov, D. Erhan et al., "Single shot multibox detector," in *Proceedings of the European Conference on Computer Vision*, Springer, Amsterdam, The Netherlands, pp. 21–37, October 2016.

[27] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, no. 2, p. 4, Honolulu, HI, USA, July 2017.

[28] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, "DSSD: deconvolutional single shot detector," 2017, https://arxiv.org/abs/1701.06659.

[29] Y. Bai, Y. Zhang, M. Ding, and B. Ghanem, "SOD-MTGAN: small object detection via multi-task generative adversarial network," in *Proceedings of the European Conference on Computer Vision*, pp. 210–226, Springer, Munich, Germany, September 2018.

[30] L. Liu, W. Ouyang, X. Wang et al., "Deep learning for generic object detection: a survey," 2018, https://arxiv.org/abs/1809.02165.

[31] P. Zhu, L. Wen, X. Bian, L. Haibin, and Q. Hu, "Vision meets drones: a challenge," 2018, https://arxiv.org/abs/1804.07437.

[32] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: a review," 2018, https://arxiv.org/abs/1807.05511.

[33] K. Oksuz, B. C. Cam, S. Kalkan, and E. Akbas, "Imbalance problems in object detection: a review," 2019, https://arxiv.org/abs/1909.00169.

[34] O. Russakovsky, J. Deng, H. Su et al., "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[35] Y. Bai, Y. Zhang, M. Ding, and B. Ghanem, "SOD-MTGAN: small object detection via multi-task generative adversarial network," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 206–221, Munich, Germany, September 2018.