Frustratingly Easy Trade-off Optimization between Single-Stage and Two-Stage Deep Object Detectors

Petru Soviany[1] and Radu Tudor Ionescu[1,2]

[1] University of Bucharest, 14 Academiei, Bucharest, Romania,
[2] Inception Institute of Artificial Intelligence (IIAI), Abu Dhabi, UAE
petru.soviany@yahoo.com, raducu.ionescu@gmail.com

Abstract. There are mainly two types of state-of-the-art object detectors. On one hand, we have two-stage detectors, such as Faster R-CNN (Region-based Convolutional Neural Networks) or Mask R-CNN, that (*i*) use a Region Proposal Network to generate regions of interests in the first stage and (*ii*) send the region proposals down the pipeline for object classification and bounding-box regression. Such models reach the highest accuracy rates, but are typically slower. On the other hand, we have single-stage detectors, such as YOLO (You Only Look Once) and SSD (Singe Shot MultiBox Detector), that treat object detection as a simple regression problem, by taking an input image and learning the class probabilities and bounding box coordinates. Such models reach lower accuracy rates, but are much faster than two-stage object detectors. In this paper, we propose and evaluate four simple and straightforward approaches to achieve an optimal trade-off between accuracy and speed in object detection. All the approaches are based on separating the test images in two batches, an *easy* batch that is fed to a faster single-stage detector and a *difficult* batch that is fed to a more accurate two-stage detector. The difference between the four approaches is the criterion used for splitting the images in two batches. The criteria are the image difficulty score (easier images go into the easy batch), the number of detected objects (images with less objects go into the easy batch), the average size of the detected objects (images with bigger objects go into the easy batch), and the number of detected objects divided by their average size (images with less and bigger objects go into the easy batch). The first approach is based on an image difficulty predictor, while the other three approaches employ a faster single-stage detector to determine the approximate number of objects and their sizes. Our experiments on PASCAL VOC 2007 show that using image difficulty compares favorably to a random split of the images. However, splitting the images based on the number objects divided by their size, an approach that is frustratingly easy to implement, produces even better results. Remarkably, it shortens the processing time nearly by half, while reducing the mean Average Precision of Faster R-CNN by only 0.5%.

Keywords: object detection, deep neural networks, Single-Shot Multibox Detector, Faster R-CNN.

1 Introduction

Object detection, the task of predicting the location of an object along with its class in an image, is perhaps one of the most important problems in computer vision. Nowadays, there are mainly two types of state-of-the-art object detectors. On one hand, we have two-stage detectors, such as Faster R-CNN (Region-based Convolutional Neural Networks) [17] or Mask R-CNN [10], that (*i*) use a Region Proposal Network (RPN) to generate regions of interests in the first stage and (*ii*) send the region proposals down the pipeline for object classification and bounding-box regression. Such models reach the highest accuracy rates, but are typically slower. On the other hand, we have single-stage detectors, such as YOLO (You Only Look Once) [16] and SSD (Singe Shot MultiBox Detector) [15], that treat object detection as a simple regression problem in which class probabilities and bounding box coordinates are learned directly from the input images. Such models reach lower accuracy rates, but are much faster than two-stage object detectors. In this context, finding a model that provides the optimal trade-off between accuracy and speed does not seem to be a straightforward task, at first hand. Based on the principles of curriculum learning [1], we hypothesize that using more complex (two-stage) object detectors for *difficult* images and less complex (single-stage) detectors for *easy* images will provide an optimal trade-off between accuracy and speed, without ever having to change anything about the object detectors. The only problem that prevents us from testing our hypothesis in practice is finding an approach to classify the images into easy or hard. In order to be useful in practice, the approach also has to work fast enough, e.g. at least as fast as a single-stage detector. To this end, we propose and evaluate four simple and straightforward approaches to achieve an optimal trade-off between accuracy and speed in object detection. All the approaches are based on separating the test images in two batches, an *easy* batch that is fed to the faster single-stage detector and a *hard* (or *difficult*) batch that is fed to the more accurate two-stage detector. The difference between the four approaches is the criterion used for splitting the images in two batches. The first approach assigns a test image to the easy or the hard batch based on the image difficulty score, as also described in [20]. The difficulty score is estimated using a recent approach for image difficulty prediction introduced by Ionescu et al. [13]. The image difficulty predictor is obtained by training a deep neural network to regress on the difficulty scores produced by human annotators. The other three approaches used for splitting the test images (into easy or hard) employ a faster single-stage detector, namely MobileNet-SSD [11], in order estimate the number of objects and each of their sizes. The second and the third approaches independently consider the number of detected objects (images with less objects go into the easy batch) and the average size of the objects (images with bigger objects go into the easy batch), while the fourth approach is based on the number of objects divided by their average size (images with less and bigger objects go into the easy batch). If one of the latter three approaches classifies an image as easy, there is nothing left to do (we can use the detections provided by MobileNet-SSD), unless we want to use a different (slower, but more accurate) single-stage

detector, e.g. SSD300 [15], for the easy images. Our experiments on PASCAL VOC 2007 [4] show that using image difficulty as a primary cue for splitting the test images compares favorably to a random split of the images. However, the other three approaches, which are frustratingly easy to implement, can produce even better results. Among the four proposed methods, the best results are obtained by splitting the images based on the number objects divided by their average size. This approach shortens the processing time nearly by half, while slightly reducing the mean Average Precision of Faster R-CNN from 0.7837 to no less than 0.7779. Moreover, all our methods are simple and have the advantage that they allow us to choose the desired trade-off on a continuous scale.

The rest of this paper is organized as follows. Recent related works on object detection are presented in Section 2. Our methodology is described in Section 3. The object detection experiments are presented in Section 4. Finally, we draw our conclusions in Section 5.

2 Related Work

Although there are quite a few models for the object detection task available in the recent literature [17, 10, 16, 15, 11], it is difficult to pick one as the best model in terms of both accuracy and speed. Some [17, 10] are more accurate and require a higher computational time, while others [16, 15, 11] are much faster, but provide less accurate results. Hence, finding the optimal trade-off between accuracy and speed is not a trivial task. To our knowledge, the only work that studied the trade-off between accuracy and speed for deep object detection models is [12]. Huang et al. [12] have tested different configurations of deep object detection frameworks by changing various components and parameters in order to find optimal configurations for specific scenarios, e.g. deployment on mobile devices. Different from their approach, we treat the various object detection frameworks as black boxes. Instead of looking for certain configurations, we propose a framework that allows to set the trade-off between accuracy and speed on a continuous scale, by specifying the point of splitting the test images into easy versus hard, as desired. It is important to note that a different line of research is to build models for tasks such as object detection [22] or semantic segmentation [14], that offer an optimal trade-off between accuracy and speed.

In the rest of this section, we provide a brief description of the most recent object detectors, in chronological order. Faster R-CNN [17] is a very accurate region-based deep detection model which improves Fast R-CNN [8] by introducing the Region Proposal Networks (RPN). It uses a fully convolutional network that can predict object bounds at every location in order to solve the challenge of selecting the right regions. In the second stage, the regions proposed by the RPN are used as an input for the Fast R-CNN model, which will provide the final object detection results. On the other hand, SSD [15] is a single-shot detection method which uses a set of predefined boxes of different aspect ratios and scales in order to predict the presence of an object in a certain image. SSD does not include the traditional proposal generation and resampling stages, common for

two-stage detectors such as Faster R-CNN, but it encapsulates all computations in a single network, thus being faster than the two-stage models. YOLO [16] is another fast model, which treats the detection task as a regression problem. It uses a single neural network to predict the bounding boxes and the corresponding classes, taking the full image as an input. The fact that it does not use sliding window or region proposal techniques provides more contextual information about classes. YOLO works by dividing each image into a fixed grid, and for each grid location, it predicts a number of bounding boxes and a confidence for each bounding box. The confidence reflects the accuracy of the bounding box and whether the bounding box actually contains an object (regardless of class). YOLO also predicts the classification score for each box for every class in training. MobileNets [11] are a set of lightweight models that can be used for classification, detection and segmentation tasks. Although their accuracy is not as high as that of the state-of-the-art very deep models, they have the great advantage of being very fast and low on computational requirements, thus being suitable for mobile devices. MobileNets are built on depth-wise separable convolutions with a total of 28 layers, and can be further parameterized in order to work even faster. Mask R-CNN [10] is yet another model used in image detection and segmentation tasks, which extends the Faster R-CNN architecture. If Faster R-CNN has only two outputs, the bounding boxes and the corresponding classes, Mask R-CNN also provides, in parallel, the segmentation masks. An important missing piece of the Faster R-CNN model is a pixel alignment method. To address this problem, He et al. [10] propose a new layer (RoIAlign) that can correct the misalignments between the regions of interest and the extracted features.

3 Methodology

Humans learn much better when the examples are not randomly presented, but organized in a meaningful order which gradually illustrates more complex concepts. This is essentially reflected in all the curricula taught in schooling systems around the world. Bengio et al. [1] have explored easy-to-hard strategies to train machine learning models, showing that machines can also benefit from learning by gradually adding more difficult examples. They introduced a general formulation of the easy-to-hard training strategies known as *curriculum learning*. However, we can hypothesize that an *easy-versus-hard* strategy can also be applied at test time in order to obtain an optimal trade-off between accuracy and processing speed. For example, if we have two types of machines (one that is simple and fast but less accurate, and one that is complex and slow but more accurate), we can devise a strategy in which the fast machine is fed with the easy test samples and the complex machine is fed with the difficult test samples. This kind of strategy will work as desired especially when the fast machine can reach an accuracy level that is close to the accuracy level of the complex machine for the easy test samples. Thus, the complex and slow machine will be used only when it really matters, i.e. when the examples are too difficult for the fast machine. The only question that remains is how to determine if an example is easy

Algorithm 1: Easy-versus-Hard Object Detection

Input:
I – an input test image;
D_{fast} – a fast (single-stage) object detector;
D_{slow} – a slow (two-stage) object detector;
C – a criterion function used for dividing the images;
t – a threshold for dividing images into easy or hard;

Computation:
if $C(I) \leq t$ **then**
    $B \leftarrow D_{fast}(I)$;
else
    $B \leftarrow D_{slow}(I)$;

Output:
B – the set of predicted bounding boxes.

or hard in the first place. If we focus our interest on image data, the answer to this question is provided by the recent work of Ionescu et al. [13], which shows that the difficulty level of an image (with respect to a visual search task) can be automatically predicted. With an image difficulty predictor at our disposal, we have a first way to test our hypothesis in the context of object detection from images. However, if we further focus our interest on the specific task of object detection in images, we can devise additional criteria for splitting the images into easy or hard, by considering the output of a very fast single-stage detector, e.g. MobileNet-SSD [11]. These criteria are the number of detected objects in the image, the average size of the detected objects, and the number of detected objects divided by their average size.

To obtain an optimal trade-off between accuracy and speed in object detection, we propose to employ a more complex (two-stage) object detector, e.g. Faster R-CNN [17], for difficult test images and a less complex (single-stage) detector, e.g. SSD [15], for easy test images. Our simple easy-versus-hard strategy is formally described in Algorithm 1. Since we apply this strategy at test time, the object detectors as well as the image difficulty predictor can be independently trained beforehand. This allows us to directly apply state-of-the-art pre-trained object detectors [17, 15, 11], essentially as black boxes. It is important to note that we propose to use one of the following four options as the criterion function C in Algorithm 1:

1. an image difficulty predictor that estimates the difficulty of the input image;
2. a fast single-stage object detector that returns the number of objects detected in the input image (*less objects* is easier);
3. a fast single-stage object detector that returns the average size of the objects detected in the input image (*bigger objects* is easier);
4. a fast single-stage object detector that returns the number of detected objects divided by their average size (*less and bigger objects* is easier).

We note that if either one of the last three criteria are employed in Algorithm 1, and if the single-stage object detector used in the criterion function C is the same as D_{fast}, we can slightly optimize Algorithm 1 by applying the single-stage object detector only once, when the input image I turns out to be easy. Another important note is that, for the last three criteria, we consider an image to be *difficult* if the single-stage detector does not detect any object. Our algorithm has only one parameter, namely the threshold t used for dividing images into easy or hard. This parameter depends on the criterion function and it needs to be tuned on a validation set in order to achieve a desired trade-off between accuracy and time. While the last three splitting criteria are frustratingly easy to implement when a pre-trained single stage object detector is available, we have to train our own image difficulty predictor as described below.

Image difficulty predictor. We build our image difficulty prediction model based on CNN features and linear regression with ν-Support Vector Regression (ν-SVR) [2]. For a faster processing time, we consider a rather shallow pre-trained CNN architecture, namely VGG-f [3]. The CNN model is trained on the ILSVRC benchmark [18]. We remove the last layer of the CNN model and use it to extract deep features from the fully-connected layer known as *fc7*. The 4096 CNN features extracted from each image are normalized using the L_2-norm. The normalized feature vectors are then used to train a ν-SVR model to regress to the ground-truth difficulty scores provided by Ionescu et al. [13] for the PASCAL VOC 2012 data set [5]. We use the learned model as a continuous measure to automatically predict image difficulty. Our predictor attains a Kendall's τ correlation coefficient [21] of 0.441 on the test set of Ionescu et al. [13]. We note that Ionescu et al. [13] obtain a higher Kendall's τ score (0.472) using a deeper CNN architecture [19] along with VGG-f. However, we are interested in using an image difficulty predictor that is faster than all object detectors, even faster than MobileNet-SSD [11], so we stick with the shallower VGG-f architecture, which reduces the computational overhead at test time.

4 Experiments

4.1 Data Set

We perform object detection experiments on the PASCAL VOC 2007 data set [4], which consists of 9963 images that contain 20 object classes. The training and validation sets have roughly 2500 images each, while the test set contains about 5000 images.

4.2 Evaluation Details

Evaluation Measure. The performance of object detectors is typically evaluated using the mean Average Precision (mAP) over classes, which is based on the ranking of detection scores for each class [7]. For each object class, the Average Precision is given by the area under the precision-recall (PR) curve for the

detected objects. The PR curve is constructed by first mapping each detected bounding box to the most-overlapping ground-truth bounding box, according to the Intersection over Union (IoU) measure, but only if the IoU is higher than 50% [6]. Then, the detections are sorted in decreasing order of their scores. Precision and recall values are computed each time a new positive sample is recalled. The PR curve is given by plotting the precision and recall pairs as lower scored detections are progressively included.

Models and Baselines. We choose Faster R-CNN [17] based on the ResNet-101 [9] architecture as our two-stage object detector that provides accurate bounding boxes. We set its confidence threshold to 0.6. In the experiments, we use the pre-trained Faster R-CNN model from https://github.com/endernewton/tf-faster-rcnn. We experiment with two single-shot detectors able to provide fast object detections, namely MobileNet-SSD [11] and SSD300 [15]. We use the pre-trained MobileNet-SSD model from https://github.com/chuanqi305/MobileNet-SSD. We use the model provided at https://github.com/weiliu89/caffe/tree/ssd for SSD300, which is based on the VGG-16 [19] architecture. SSD300 takes input images of 300×300 pixels and performs the detection task in a single step. We also tried the SSD512 detector, but we did not find it interesting for our experiments, since its speed is a bit too high for a fast object detector (1.57 seconds per image).

The main goal of the experiments is to compare our four different strategies for splitting the images between the single-stage detector (MobileNet-SSD or SSD300) and the two-stage detector (Faster R-CNN) with a baseline strategy that splits the images randomly. To reduce the accuracy variation introduced by the random selection of the baseline strategy, we repeat the experiments for 5 times and average the resulted mAP scores. We note that all standard deviations are lower than 0.5%. We consider various splitting points starting with a $100\% - 0\%$ split (applying the single-stage detector only), going through three intermediate splits ($75\% - 25\%$, $50\% - 50\%$, $25\% - 75\%$) and ending with a $0\% - 100\%$ split (applying the two-stage detector only). To obtain these splitting points, we individually tune the parameter t of Algorithm 1 on the PASCAL VOC 2007 validation set, for each splitting criterion.

4.3 Results and Discussion

Table 1 presents the mAP scores and the processing times of MobileNet-SSD [11], Faster R-CNN [17] and several combinations of the two object detectors, on the PASCAL VOC 2007 data set. Different model combinations are obtained by varying the percentage of images processed by each detector. The table includes results starting with a $100\% - 0\%$ split (equivalent with MobileNet-SSD [11] only), going through three intermediate splits ($75\% - 25\%$, $50\% - 50\%$, $25\% - 75\%$) and ending with a $0\% - 100\%$ split (equivalent with Faster R-CNN [11] only). In the same manner, Table 2 shows the results for similar combinations of SSD300 [15] and Faster R-CNN [17]. While the results of various model combinations are listed on different columns in Table 1 and Table 2, the results of various splitting strategies are listed on separate rows.

Table 1. Mean Average Precision (mAP) and time comparison between MobileNet-SSD [11], Faster R-CNN [17] and various combinations of the two object detectors on PASCAL VOC 2007. The test data is partitioned based on a random split (baseline) or four easy-versus-hard splits given by: the image difficulty score, the number of objects (n), the average size of the objects (avg), and the number of objects divided by their average size (n/avg). For the random split, we report the mAP over 5 runs to reduce bias. The reported times are measured on a computer with Intel Core i7 2.5 GHz CPU and 16 GB of RAM.

	MobileNet-SSD (left) to Faster-RCNN (right)				
	100%−0%	75%−25%	50%−50%	25%−75%	0%−100%
Splitting Criterion	Mean Average Precision (mAP)				
(1) Random (baseline)	0.6668	0.6895	0.7131	0.7450	0.7837
(2) Image difficulty	0.6668	0.6981	0.7431	0.7640	0.7837
(3) Number of objects (n)	0.6668	0.7091	0.7382	0.7551	0.7837
(4) Average object size (avg)	0.6668	0.7202	0.7503	0.7680	0.7837
(5) n/avg	0.6668	0.7336	0.7574	0.7775	0.7837
Component	Time (seconds)				
(2) Image difficulty	-	0.05	0.05	0.05	-
$(3, 4, 5)$ Estimation of n, avg	-	0.07	0.07	0.07	-
Object detection	0.07	2.38	4.08	6.07	7.74
Object detection + (2)	0.07	2.43	4.13	6.12	7.74
Object detection + $(3, 4, 5)$	0.07	2.40	4.12	6.12	7.74

We first analyze the mAP scores and the processing time of the three individual object detectors, namely Faster R-CNN [17], MobileNet-SSD [11] and SSD300 [15]. Faster R-CNN reaches a mAP score of 0.7837 in about 7.74 seconds per image, while SSD300 reaches a mAP score of 0.69 in 0.56 seconds per image. MobileNet-SDD is even faster, attaining a mAP score of 0.6668 in just 0.07 seconds per image. We hereby note that we also considered the SSD512 object detector, but its results (a mAP score of 0.7046 in 1.57 seconds per image) did not convince us to include it in the evaluation.

We next analyze the average object detection times per image of the various model combinations. As expected, the time improves by about 21% when running MobileNet-SSD on 25% of the test set and Faster R-CNN on the rest of 75%. On the $50\% - 50\%$ split, the processing time is nearly 47% shorter than the time required for processing the entire test set with Faster R-CNN only ($0\% - 100\%$ split). On the $75\% - 25\%$ split, the processing time further improves by 69%. As SSD300 is slower than MobileNet-SSD, the time improvements are close, but not as high. The improvements in terms of time are 20% for the $25\% - 75\%$ split, 44% for the $50\% - 50\%$ split, and 68% for the $75\% - 25\%$ split. We note that unlike the random splitting strategy, the easy-versus-hard splitting strategies require additional processing time, either for computing the difficulty scores or for estimating the number of objects and their average size. The image difficulty predictor runs in about 0.05 seconds per image, while the MobileNet-SSD detector (used for estimating the number of objects and their average size) runs in about 0.07 seconds per image. Hence, the extra time required by our

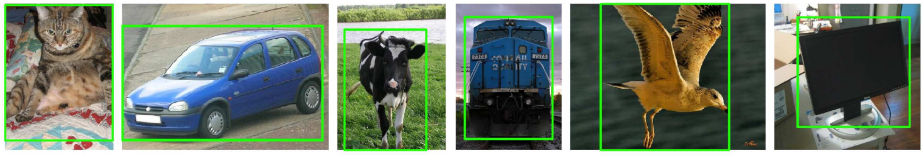
Table 2. Mean Average Precision (mAP) and time comparison between SSD300 [15], Faster R-CNN [17] and various combinations of the two object detectors on PASCAL VOC 2007. The test data is partitioned based on a random split (baseline) or four easy-versus-hard splits given by: the image difficulty score, the number of objects (n), the average size of the objects (avg), and the number of objects divided by their average size (n/avg). For the random split, we report the average mAP over 5 runs to reduce bias. The reported times are measured on a computer with Intel Core i7 2.5 GHz CPU and 16 GB of RAM.

	SSD300 (left) to Faster-RCNN (right)				
	100% − 0%	75% − 25%	50% − 50%	25% − 75%	0% − 100%
Splitting Criterion	Mean Average Precision (mAP)				
(1) Random (baseline)	0.6900	0.7003	0.7178	0.7561	0.7837
(2) Image difficulty	0.6900	0.7117	0.7513	0.7732	0.7837
(3) Number of objects (n)	0.6900	0.7386	0.7619	0.7725	0.7837
(4) Average object size (avg)	0.6900	0.7078	0.7413	0.7664	0.7837
(5) n/avg	0.6900	0.7464	0.7779	0.7840	0.7837
Component	Time (seconds)				
(2) Image difficulty	-	0.05	0.05	0.05	-
(3, 4, 5) Estimation of n, avg	-	0.07	0.07	0.07	-
Object detection	0.56	2.46	4.33	6.12	7.74
Object detection + (2)	0.56	2.51	4.38	6.17	7.74
Object detection + (3, 4, 5)	0.56	2.53	4.40	6.19	7.74

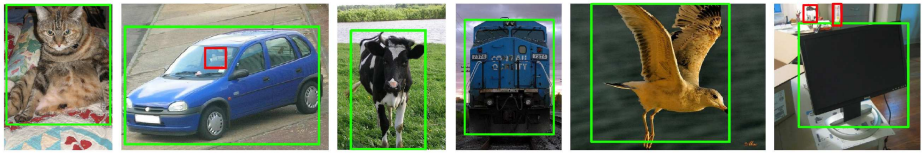
easy-versus-hard splitting strategies is almost insignificant with respect to the total time required by the various combinations of object detectors. For instance, in the 50% − 50% split with MobileNet-SSD and Faster R-CNN, the difficulty predictor accounts for roughly 1% of the total processing time (0.05 out of 4.13 seconds per image).

Regarding our four easy-versus-hard strategies for combining object detectors, the empirical results indicate that all the proposed splitting strategies give better performance than the random splitting strategy, for all model combinations. For the splitting strategy based on image difficulty, the highest improvements over the random strategy can be observed for the 50% − 50% split. When using MobileNet-SSD for the easy images (Table 1), our strategy based on image difficulty gives a performance boost of 3% (from 0.7131 to 0.7431) over the random splitting strategy. However, the mAP of the MobileNet-SSD and Faster R-CNN 50% − 50% combination is 4.06% under the mAP of the standalone Faster R-CNN. When using SSD300 for the easy images (Table 2), our strategy based on image difficulty gives a performance boost of 3.35% (from 0.7178 to 0.7513) over the baseline strategy, for the 50% − 50% split. This time, the mAP of the SSD300 and Faster R-CNN 50% − 50% combination is 3.24% under the mAP of the standalone Faster R-CNN, although the processing time is reduced by almost half. While using the average size of the objects as splitting criterion gives better results than using the image difficulty scores when the fast detector is MobileNet-SSD (Table 1), it seems that using the number of objects as splitting criterion gives better results when the fast detector is SSD300 (Table 2).

a) Images with less and bigger objects according to MobileNet-SSD

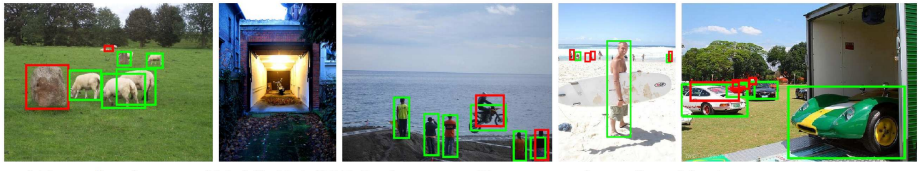


b) Bounding boxes of MobileNet-SSD for images with less and bigger objects

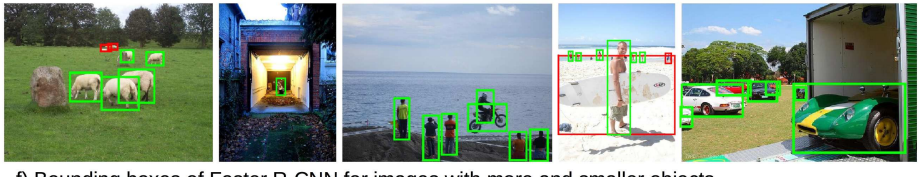


c) Bounding boxes of Faster R-CNN for images with less and bigger objects

d) Images with more and smaller objects according to MobileNet-SSD



e) Bounding boxes of MobileNet-SSD for images with more and smaller objects



f) Bounding boxes of Faster R-CNN for images with more and smaller objects

Fig. 1. Examples of easy (top three rows) and hard images (bottom three rows) from PASCAL VOC 2007 according to the number of objects divided by their average size. For each set of images, the bounding boxes predicted by the MobileNet-SSD [11] and the Faster R-CNN [17] detectors are also presented. The correctly predicted bounding boxes are shown in green, while the wrongly predicted bounding boxes are shown in red. Best viewed in color.

However, the best splitting strategy in all cases is the one based on the score obtained by dividing the number of objects by their average size (n/avg). When the fast detector is MobileNet-SSD (Table 1), our strategy based on n/avg gives a performance boost of more than 4% (from 0.7131 to 0.7574) over the random splitting strategy for the $50\% - 50\%$ split, but the results are even better when SSD300 is used as the fast detector. Indeed, the results presented in Table 2 show that our strategy based on n/avg attains a mAP score of 0.7779 for the $50\% - 50\%$ combination of SSD300 and Faster R-CNN, and a mAP score of 0.7840 for the $25\% - 75\%$ combination of SSD300 and Faster R-CNN. Remarkably, for the $50\% - 50\%$ split, the n/avg strategy reduces the time by almost half with a very small reduction in performance (from 0.7837 to 0.7779). Moreover, the n/avg strategy slashes about 1.55 seconds from the Faster R-CNN execution time, while slightly improving the mAP score from 0.7837 to 0.7840.

To understand why our splitting strategy based on n/avg gives better results than the random splitting strategy, we randomly select a few easy examples (with less and bigger objects) and a few difficult examples from the PASCAL VOC 2007 data set, and we display them in Figure 1 along with the bounding boxes predicted by the MobileNet-SSD and the Faster R-CNN object detectors. On the easy images, the bounding boxes of the two detectors are almost identical. However, we can observe some false positive detections provided by Faster R-CNN on two easy images containing a car and a monitor, respectively. Nevertheless, we can perceive a lot more differences between MobileNet-SSD and Faster R-CNN on the hard images. In the left-most hard image, MobileNet-SSD wrongly detects a large rock as an object, while in the second image MobileNet-SSD fails to detect the motorbike. In the third and fifth images, MobileNet-SSD provides some wrong detections, while Faster R-CNN provides accurate detections (without any false positives). In the fourth image, Faster R-CNN is able to detect more people in the background, but it also detects the surfboard and wrongly labels it as a boat. We thus conclude that the difference between MobileNet-SSD and Faster R-CNN is less noticeable on the easy images than on the hard images. This could explain why our splitting strategy based on n/avg is effective in choosing an optimal trade-off between accuracy and speed.

5 Conclusion

In this paper, we have presented four easy-versus-hard strategies to obtain an optimal trade-off between accuracy and speed in object detection from images. Our strategies are based on dispatching each test image either to a fast and less accurate single-shot detector or to a slow and more accurate two-stage detector, according to the image difficulty score, the number of objects contained in the image, the average size of the objects, or the number of objects divided by their average size. We have conducted experiments using state-of-the-art object detectors such as SSD300 [15] or Faster R-CNN [17] on the PASCAL VOC 2007 [4] data set. The empirical results indicate that using the strategy based on dividing the number of objects by their average size for splitting the test images

compares favorably to a random split of the images, and perhaps surprisingly, it also provides better results than splitting the images according to their difficulty score. Since our best splitting strategy is simple and easy to implement, it can be immediately adopted by anyone that needs a continuous accuracy versus speed trade-off optimization strategy in object detection. In future work, we aim to investigate whether training object detectors to specifically deal with easy or hard image samples can help to further improve our results.

Acknowledgments

The work of Petru Soviany was supported through project grant PN-III-P2-2.1-PED-2016-1842. The work of Radu Tudor Ionescu was supported through project grant PN-III-P1-1.1-PD-2016-0787.

References

1. Bengio, Y., Louradour, J., Collobert, R., Weston, J.: Curriculum learning. In: Proceedings of ICML. pp. 41–48 (2009)
2. Chang, C.C., Lin, C.J.: Training ν-Support Vector Regression: Theory and Algorithms. Neural Computation **14**, 1959–1977 (2002)
3. Chatfield, K., Simonyan, K., Vedaldi, A., Zisserman, A.: Return of the devil in the details: Delving deep into convolutional nets. In: Proceedings of BMVC (2014)
4. Everingham, M., Van Gool, L., Williams, C., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2007 Results (2007)
5. Everingham, M., Van Gool, L., Williams, C., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2012 Results (2012)
6. Everingham, M., Eslami, S.M., Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The Pascal Visual Object Classes Challenge: A Retrospective. International Journal of Computer Vision **111**(1), 98–136 (2015)
7. Everingham, M., van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The Pascal Visual Object Classes (VOC) Challenge. International Journal of Computer Vision **88**(2), 303–338 (Jun 2010)
8. Girshick, R.: Fast R-CNN. In: Proceedings of ICCV. pp. 1440–1448 (2015)
9. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. In: Proceedings of CVPR. pp. 770–778 (2016)
10. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: Proceedings of ICCV. pp. 2961–2969 (2017)
11. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
12. Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., et al.: Speed/accuracy trade-offs for modern convolutional object detectors. In: Proceedings of CVPR. pp. 7310–7319 (2017)
13. Ionescu, R., Alexe, B., Leordeanu, M., Popescu, M., Papadopoulos, D.P., Ferrari, V.: How hard can it be? estimating the difficulty of visual search in an image. In: Proceedings of CVPR. pp. 2157–2166 (2016)
14. Li, X., Liu, Z., Luo, P., Loy, C.C., Tang, X.: Not All Pixels Are Equal: Difficulty-Aware Semantic Segmentation via Deep Layer Cascade. In: Proceedings of CVPR. pp. 3193–3202 (2017)

15. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: SSD: Single Shot MultiBox Detector. In: Proceedings of ECCV. pp. 21–37 (2016)
16. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You Only Look Once: Unified, Real-Time Object Detection. In: Proceedings of CVPR. pp. 779–788 (2016)
17. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In: Proceedings of NIPS. pp. 91–99 (2015)
18. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., A., K., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (2015)
19. Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. In: Proceedings of ICLR (2014)
20. Soviany, P., Ionescu, R.T.: Optimizing the Trade-off between Single-Stage and Two-Stage Deep Object Detectors using Image Difficulty Prediction. In: Proceedings of SYNASC (2018)
21. Upton, G., Cook, I.: A Dictionary of Statistics. Oxford University Press, Oxford (2004)
22. Zhou, P., Ni, B., Geng, C., Hu, J., Xu, Y.: Scale-Transferrable Object Detection. In: Proceedings of CVPR. pp. 528–538 (2018)