RIGA TECHNICAL UNIVERSITY

Information Technology and Management Science

ISSN 2255-9094 (online)
ISSN 2255-9086 (print)
December 2018, vol. 21, pp. 15–23
doi: 10.7250/itms-2018-0002
https://itms-journals.rtu.lv

Anchor Box Parameters and Bounding Box Overlap Ratios for the Faster R-CNN Detector in Detecting a Single Object by the Masking Background

Vadim Romanuke
Polish Naval Academy, Gdynia, Poland

Abstract **– Anchor box parameters and bounding box overlap ratios are studied in order to set them appropriately for the Faster R-CNN detector. The benchmark detection is based on monochrome images whose background may mask a small dark object. Three object detection tasks are generated, where every image either contains a small black square/rectangle or does not contain the object, representing thus class "background". If this class is represented, the ratios are recommended to be tried at 0.7. The ratio for positive training samples is to be tried at a less value, but greater than 0.4, for the task every image of which contains an object. The minimum anchor box size is better to be tried at a lesser value from a range of object sizes. The anchor box pyramid scale factor and the number of levels are better to try at 2 and 8, respectively. Subsequently, these parameters may be corrected as their influence is fuzzier than that of the ratios.**

Keywords **– Anchor box, bounding box overlap ratio, object detection, R-CNN.**

I. Introduction to an Object Detection Task Problem

Object detection is an important part of machine vision, artificial intelligence, medical diagnostics, and industrial automation. The task of object detection involves images, and it seems far simpler than the task of object tracking that involves sequences of images or videos [1], [2]. Although detecting and recognising objects in an image are not as hard as tracking objects in motion, both tasks require many parameters for fine-tuning object detectors and trackers. Without setting these parameters to their effective values, even detectors of very simple objects do not recognise them [3]. It is principally essential to learn how to adjust the parameters for achieving proper accuracy of detection because this is a particular basis for object tracking, apart from object detection itself.

II. Related Works and Motivation

As CNNs are widely used for image categorization, they can serve a platform for classifying image regions within an image. An early application of CNNs to object detection was a combination of region proposals with CNN features given the R-CNN method [4]. Remarkably, combining with pre-trained CNNs allows using small training sets [5], [6] that are very relevant in practice.

Instead of classifying every region using a sliding window, the R-CNN detector only processes those regions that are likely to contain an object. This reduces the computational cost incurred when running a CNN. The region proposals, or bounding boxes, are created by a process called selective search [7]. The image is passed through windows of different sizes, and for each size the selective search method tries to group together adjacent pixels by texture, colour, or intensity to identify objects. Once the proposals are created, the R-CNN detector warps the region to a standard square size and passes it through to a pre-trained CNN. On the final layer of the CNN, a support vector machine (SVM) is added. The SVM classifies whether we have an object, and, if this is so, what object in the bounding box is. Eventually, the box is run through a simple linear regression model to output tighter coordinates to fit true dimensions of the object that has been classified [4].

The R-CNN method has two notable drawbacks: training is expensive in space and time, and the R-CNN detector is slow because it performs a CNN forward pass for each object region proposal, without sharing computation. The Fast R-CNN method removes these drawbacks by its way of sharing the computations over region proposals [8]. A Fast R-CNN architecture has a special pooling layer for regions of interest (RoIs). After a feature map is produced from the input image, a RoI pooling layer extracts a fixed-length feature vector from the feature map for each object region proposal. RoIs from the same image share computation while training. The Fast R-CNN method jointly trains the CNN, classifier, and bounding box regressor in a single model. The SVM classifier is replaced with a softmax layer on the CNN top to output a classification. A linear regression layer is added parallel to the softmax layer to output box coordinates. Whereas the R-CNN method has a CNN for extracting image features, an SVM to classify, and a regressor to tighten bounding boxes, the Fast R-CNN method instead exploits a single network for computing all the three ones.

Despite a clear straightforward move for sparing space and time, RoIs are generated by the fairly slow selective search process. The region proposal step is dramatically sped up through the Faster R-CNN method [9]. A region proposal network (RPN) was introduced in 2015 for sharing full-image convolutional features with the detection network, thus enabling a much faster region proposer. An RPN is a fully convolutional network added prior to the Fast R-CNN detector. The Faster R-CNN makes thereby a region proposal a part of the CNN training and prediction steps. The RPN passes a sliding window over the CNN feature map, outputting potential bounding boxes and their respective scores. Anchor boxes considering certain object aspect ratios and sizes are used for this purpose.

The anchor box has three parameters: a $M \times 2$ matrix $\mathbf{A} = \left(a_{mh}\right)_{M \times 2}$ of minimum anchor box sizes (MABSs), an anchor box pyramid scale factor s, and a number of levels n in an anchor box pyramid. The MABSs are used to build the anchor box pyramid of the RPN [9]. Each row of matrix **A** defines the height and width of an anchor box. In MATLAB, the default value for this parameter uses the minimum size and the median aspect ratio from the bounding boxes for each class in the ground truth data [10]. To remove redundant box sizes, those boxes are kept that have an intersection-over-union (IoU) [4] that is less than or equal to 0.5 ensuring that the minimum number of anchor boxes is used to cover all the object sizes and aspect ratios.

The anchor box pyramid scale factor is used to successively upscale anchor box sizes. By the MATLAB recommendation, this factor is taken from 1 through 2, but the default value is $s = 2$.

Number of levels in an anchor box pyramid is selected for ensuring that the multiscale anchor boxes are comparable in size to the size of objects in the ground truth data. The MATLAB default value of n is selected based on the size of objects within the ground truth data for covering the range of object sizes.

The Faster R-CNN detector is trained in four stages, each of which may have its own training options [9]. At the first stage, an RPN is trained. The RPN is initialized with a pre-trained CNN. Then, a Fast R-CNN network initialized also with the pre-trained CNN is trained using the already trained RPN. At the third stage, the RPN is re-trained using weight sharing with the Fast R-CNN network. Finally, the Fast R-CNN network is re-trained using the updated RPN, where the shared convolutional layers are kept fixed and the unique layers of the Fast R-CNN are fine-tuned.

Setting the anchor box parameters appropriately is very important for training the Faster R-CNN detector successfully. Besides, bounding box overlap ratios (BBORs) [3] for positive training samples (PosTS) and negative training samples (NegTS) must influence the training and detection accuracy. These parameters should not be set by rules of thumb. Hence, a question is how to set those parameters properly, otherwise detectors will not work correctly. Obviously, there are many benchmark datasets and tasks, and each task will have its own appropriate anchor box parameters and BBORs. That is why a hard detection task should be considered. One of such tasks is object detection within monochrome images, where colour contrasts and intensities cannot help in detection. Such tasks are more real in the industry and surveillance practice (except for medical diagnostics) because the images for analysis and object detection are acquired from small webcams that may have either low resolution or just transmit grayscale signals (in some cases, the images are noised due to wireless transmission). An additional obstacle to object detection within monochrome images is that monochromatism often masks objects, which, for instance, may merge with the background [11]. Consequently, an object detection task (ODT) with monochrome images is both a suitable benchmark task for solving a problem of an appropriate Faster R-CNN detector parametrization and a prototype of the factually practiced object detection [3], [11].

III. The Aim and Tasks

Since selecting the anchor box parameters and BBORs is an open question, the aim is to develop a technique of setting them appropriately for a hard ODT. The ODT will be of monochrome images, wherein artificial objects should be defined as much as less detectable for strengthening the role of appropriateness in selecting the parameters and ratios. The criterion of the appropriateness is the Faster R-CNN detector performance, which will be substantiated separately. To achieve the said aim, the following tasks are to be fulfilled:

1. To substantiate a benchmark ODT whose images and objects are monochrome.
2. To select a pre-trained CNN for including it into a Faster R-CNN detector architecture.
3. To select boundaries of the anchor box parameters and BBORs, within which the Faster R-CNN detector will be trained and tested.
4. To run the Faster R-CNN detector through those varied parameters and ratios by registering the detection performance based on properties of IoUs.
5. To analyse the obtained results and decide on the appropriateness of the anchor box parameters and BBORs for the substantiated benchmark ODT.
6. To discuss the technique of setting those parameters and ratios appropriately, and make a conclusion considering an outlook for possible further research or its expansion.

IV. Benchmark ODT

To evaluate performance correctly and consistently, the benchmark image classification problem must be spacious and blanket. An example of such an image classification problem is the ImageNet dataset [12]. However, the ImageNet dataset is very huge and bulky for statistical evaluation. Therefore, a few benchmark datasets will be used.

Let us consider a bunch of monochrome images of suburb house frontal views (Fig. 1). Although the images are pretty simple, they have a lot of small localities like house windows, grass, trees, fence, shadows, and doors, where a darkened object could be masked in. The image resolution is not high, being rather low. The bunch of 120 such images might factually constitute a small training set suitable for any ODT with a few object types to be found within the image.

Fig. 1. A bunch of 120 monochrome 220×330 images of suburb house frontal views. The images are 8-bit grayscale. A darkened object could be masked in localities like house windows, grass, trees, fence, shadows, and doors.

First, it is sufficient to generate an artificial object to every image. The simplest object is a square occupying from 2 % up to 3 % of the whole image. The object colour is set black for masking it in the dark localities that is expected to strengthen the role of the appropriateness in selecting the anchor box parameters and BBORs. Henceforward, the 40×40 black square is inserted randomly within each of those 120 images, making thus a training set without class "background" (i.e., there are no "pure" images; every image contains an object, which should be detected). Figure 2 presents an assembled image of this training set, where only a few black squares are clearly seen. Although the object to be detected here is elementary, detectability of these objects is not as easy as it may seem. For instance, the squares within the images in Fig. 3, extracted from the training set in Fig. 2, are really well-masked.

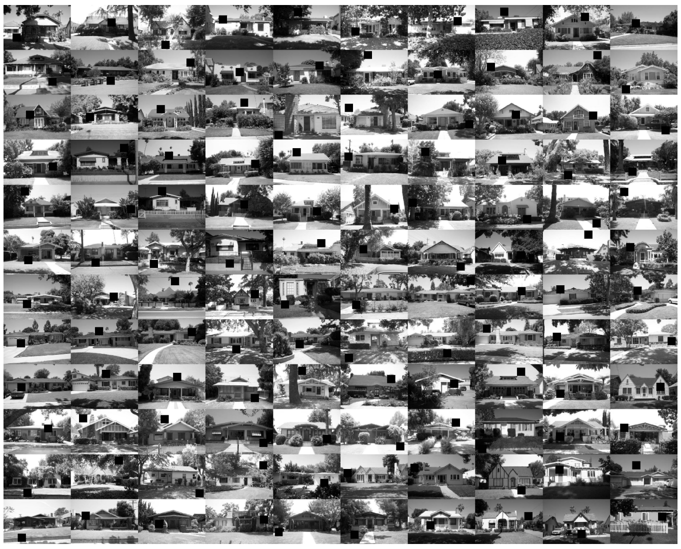


Fig. 2. An assembled image of a training set of 120 monochrome 220×330 images, where every image contains the 40×40 black square to be detected.

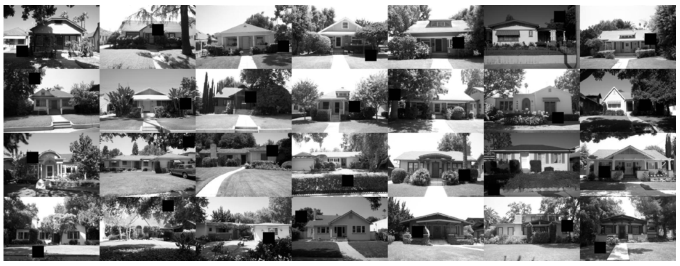


Fig. 3. An assembled image of a subset of the training set shown in Fig. 2, where the black squares are well-masked. Apparently, this is not a unique version of only those 28 "masking" images. A few new images (from Fig. 2) which mask the object might be added. Surely, the set of "masking" images is kind of fuzzy determined by human perception of a definite observer.

A testing set without class "background" is created analogously (Fig. 4). It also contains images that successfully mask the black square (Fig. 5).

For benchmarking with the ODT by the training set in Fig. 2, analogous training and testing sets are created with class "background", where about 20 % of images are kept original, without inserting the black square. There are 25 and 23 images without the black square in the new training and testing sets, respectively (Fig. 6 and Fig. 7).

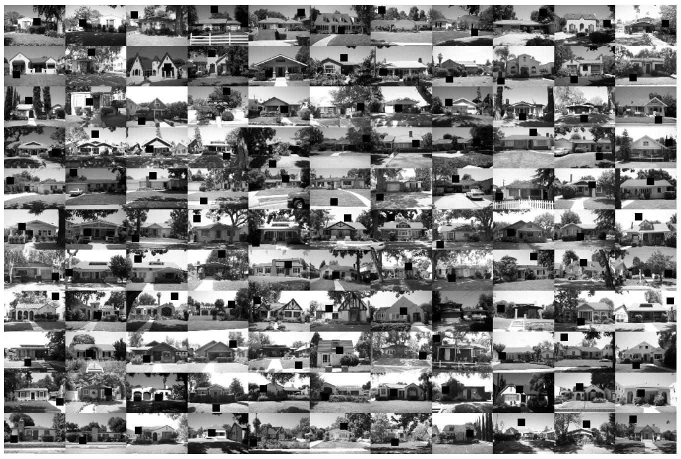


Fig. 4. An assembled image of a testing set of 121 monochrome 220×330 images (they are related to images in Fig. 2 but without repetitions, it is a new set), where every image contains the 40×40 black square to be detected.

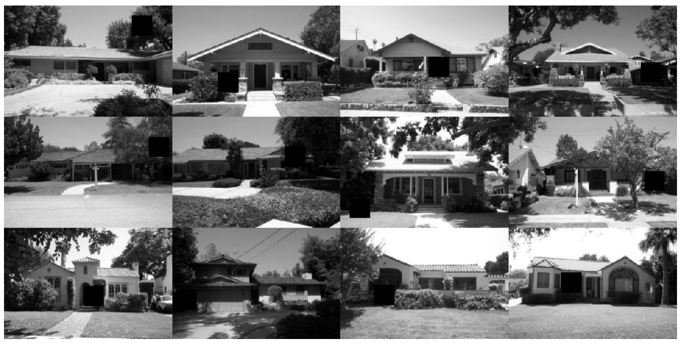


Fig. 5. A subset of 12 images of the testing set in Fig. 4, where the black squares are well-masked due to dark colour and geometrical form similarities. For a layman, it is pretty hard to quickly detect accurately all the squares.

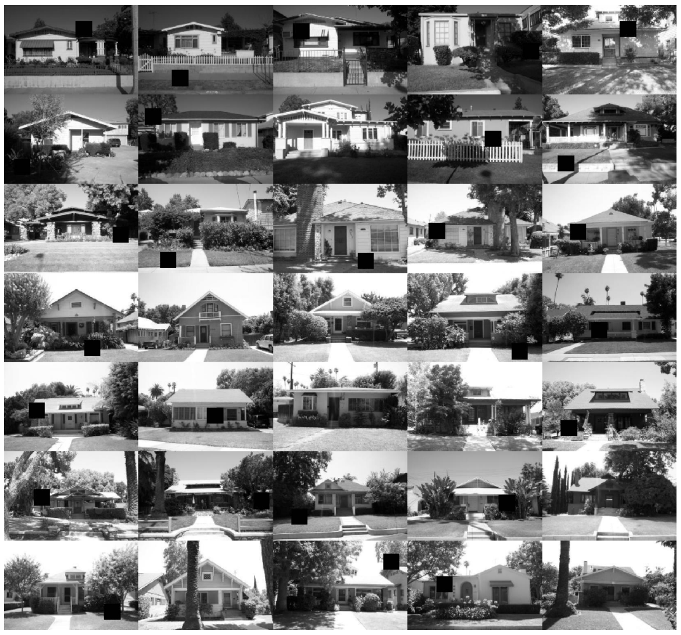


Fig. 6. A subset of 35 images of a new training set of 120 monochrome 220×330 images, where 95 images contain the 40×40 black square to be detected. The rest 25 images in this new set represent class "background".

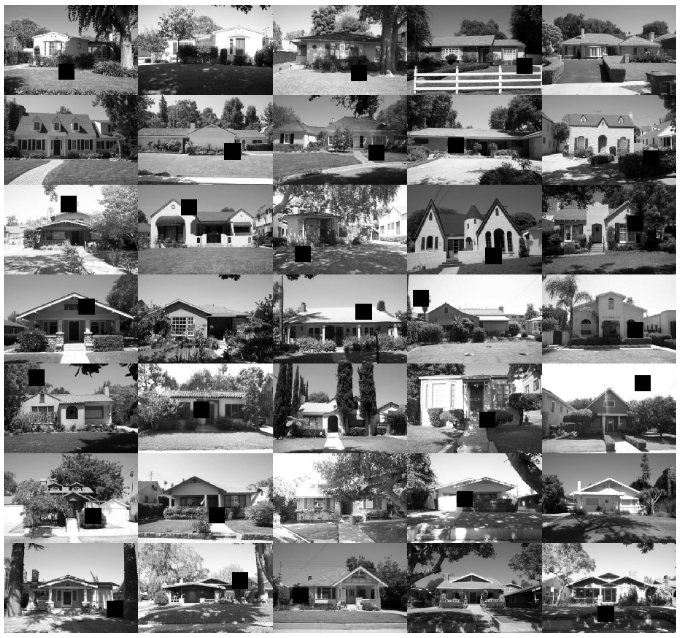


Fig. 7. A subset of 35 images of a new testing set (related to the training set whose subset is shown in Fig. 6) of 121 monochrome 220×330 images, where 98 images contain the 40×40 black square to be detected. The rest 23 images in this new set represent class "background".

The third ODT is created by inserting black rectangles whose width and height vary between 30 and 50 pixels, in all images. Although now the training and testing sets are created without class "background", some training images and testing images truly mask the object (Fig. 8 and Fig. 9, respectively).

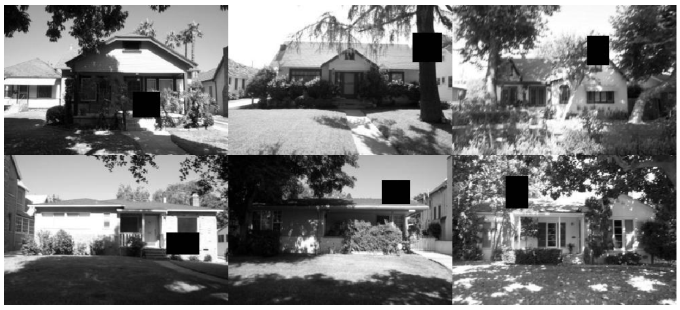


Fig. 8. A tiny subset of the third training set of 120 monochrome 220×330 images containing black rectangles. Some images mask the rectangles as well.

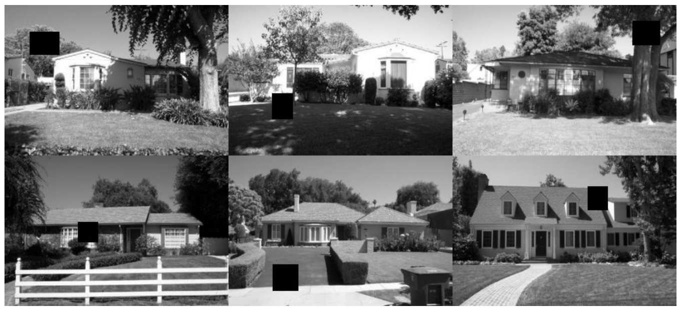


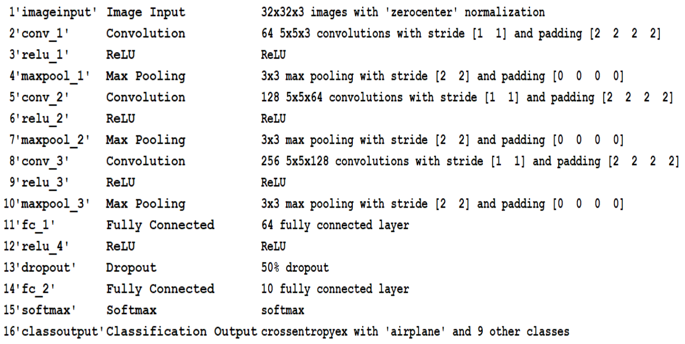
Fig. 9. A tiny subset of the third testing set of 121 monochrome 220×330 images, with a black rectangle in every image. The location of every rectangle is randomly changed, so "masking" images differ from those in Fig. 5 and Fig. 7.

Thus, these three ODTs are pretty diverse, although they are generated from the same monochrome 220×330 images of suburb house frontal views. Owing to the monochromatism and masking the black squares and rectangles, these tasks are hard as well.

V. A Pre-trained CNN for the Faster R-CNN Detector

To deal with small training sets, the pre-trained CNN is used in the sense of the transfer learning workflow [13]. In transfer learning, a CNN trained on a big dataset is used as the starting point to solve a new classification or detection task [14]. This means that the pre-trained CNN has already learned a rich aggregate of image features that are applicable to a wide range of images. This learning is transferable to the new task by fine-tuning the pre-trained CNN. The pre-trained CNN can be successfully fine-tuned by making small adjustments to the weights of the CNN layers. Eventually, the feature representations learned for the original task are slightly adjusted to support the new task. These slight adjustments are executed across small training sets [4], [8], [9], [13], [14].

The original task used for pre-training a CNN should have a big training set whose images must be very diverse and heterogeneous. A good example of such a task is the CIFAR-10 dataset. The dataset consists of 60,000 colour images of size $32 \times 32 \times 3$, represented as 32×32 matrix in each of the three colour channels. The dataset is divided into 50,000 images intended for training and 10,000 images intended for validating and testing. Although the CIFAR-10 dataset has only 10 image categories (6,000 images per category), its images are heterogeneous and miscellaneous diversely representing such classes as "airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship", "truck" [15], [16]. One of the promising CNN architectures for classifying CIFAR-10 images is of convolutional layers whose number of filters is doubled through the layers [15]. The starting number of filters in the first convolutional layer can be set at 64. To prevent overfitting and improve generalization, a 50 % dropout layer is inserted in-between two fully-connected layers (Fig. 10).



```
 1'imageinput' Image Input           32x32x3 images with 'zerocenter' normalization
 2'conv_1'     Convolution           64 5x5x3 convolutions with stride [1  1] and padding [2  2  2  2]
 3'relu_1'     ReLU                  ReLU
 4'maxpool_1'  Max Pooling           3x3 max pooling with stride [2  2] and padding [0  0  0  0]
 5'conv_2'     Convolution           128 5x5x64 convolutions with stride [1  1] and padding [2  2  2  2]
 6'relu_2'     ReLU                  ReLU
 7'maxpool_2'  Max Pooling           3x3 max pooling with stride [2  2] and padding [0  0  0  0]
 8'conv_3'     Convolution           256 5x5x128 convolutions with stride [1  1] and padding [2  2  2  2]
 9'relu_3'     ReLU                  ReLU
10'maxpool_3'  Max Pooling           3x3 max pooling with stride [2  2] and padding [0  0  0  0]
11'fc_1'       Fully Connected       64 fully connected layer
12'relu_4'     ReLU                  ReLU
13'dropout'    Dropout               50% dropout
14'fc_2'       Fully Connected       10 fully connected layer
15'softmax'    Softmax               softmax
16'classoutput'Classification Output crossentropyex with 'airplane' and 9 other classes
```

Fig. 10. The architecture of a CNN in MATLAB for classifying CIFAR-10 images. An additional fully-connected layer is inserted in-between the convolutional layer of 256 filters and the final convolutional layer of 10 filters (equal to the number of image classes).

After having trained the CNN for 80 epochs, it performs at 84 % accuracy rate (over the CIFAR-10 testing set of 10,000 images). This is sufficient for continuing with the CNN as a pre-trained one to be included into the Faster R-CNN detector architecture.

VI. Boundaries of the Anchor Box Parameters and BBORs

As the object size varies between 30 and 50 pixels, it is sufficient to set $a_{11} = a_{12} = a$ and $a \in \left\{\overline{30, 50}\right\}$ for matrix $\mathbf{A}$ (which here is just a row). Let the step of a be 1. The scale factor is $s \in [1; 2]$ and the number of levels in an anchor box pyramid is $n \in \left\{\overline{1, 8}\right\}$. Let the step of s be 0.1 and the step of n be 1, but these two will be set as default for the first ODT.

The BBOR for NegTS is a value from half-segment $(0; q]$ by $0 < q < 1$. The BBOR for PosTS is a value from half-segment $[r; 1)$ by $q \leqslant r < 1$. For the training set without class "background", $q = 0.001$ and $r \in [0.1; 0.9]$. In general, let the step for the ratios be 0.1, but it will be 0.02 for more accurate evaluation.

VII. Running the Faster R-CNN Detector through the Varied Parameters and Ratios

The Faster R-CNN detector will be trained for 2 epochs with a minibatch size equal to 120. In the first turn, the Faster R- CNN detector is run through $r \in [0.1; 0.9]$ and $a \in \left\{\overline{30, 50}\right\}$ by $q = 0.001$ for the dataset without class "background". Then, after deciding on the best s and n, the detector is run through $q \in [0.1; 0.8]$ and $r \in [q; 0.9]$ with the step equal to 0.1 for the dataset in Fig. 6 and Fig. 7. Finally, the detector is trained by the training set in Fig. 8 and tested over the testing set in Fig. 9 through the same varied parameters.

IoUs summed and normed (IoUs_{SN}) against a (horizontal) and r (vertical) are shown in Fig. 11 for the ODT without class "background". Darker colours and bigger dots correspond to greater IoUs_{SN} without missed detections. Numbers of the detected objects summed and normed (NsDO_{SN}) are shown in Fig. 12 under the same axes and colours.

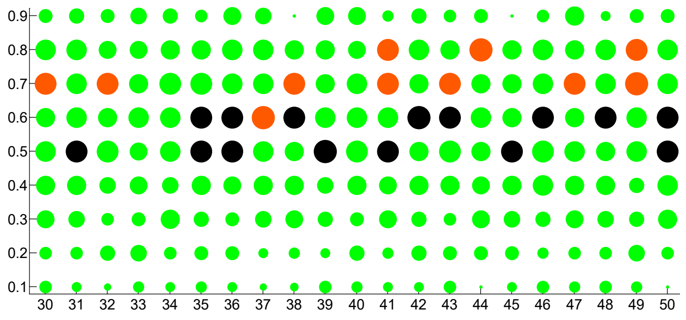


Fig. 11. IoUs_{SN} are top for $r = 0.5$ and $r = 0.6$, too weakly depending on a.

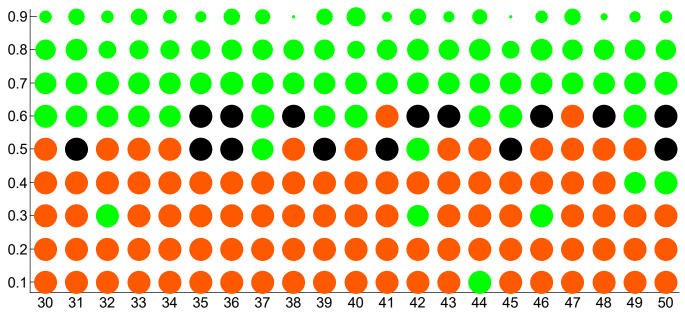


Fig. 12. At the same black dots, NsDO_{SN} are equal to 121 along with highly accurate detection. A lot of detections are missed at $r > 0.7$, where IoUs_{SN} are not top. There are fewer omissions at $r < 0.4$ but the accuracy is far poorer.

As black dots are the best ones, it is clearly seen that the best BBOR for PosTS is at $r \in [0.5; 0.6]$ or close to this interval. The refined IoUs_{SN} are in Fig. 13, where darker colours and bigger dots correspond to greater IoUs_{SN} without missed detections, by rendering dark those dots at which the minimal IoU is greater than 0.4237 (i.e., the best minimal accuracy for the ODT without class "background"). Under the same axes and colours, Fig. 14 presents NsDO_{SN}; Fig. 15 shows those minimal IoUs, and variances of IoUs are shown in Fig. 16 (the same colour convention will be used further). An obvious inference is that an appropriate r is about between 0.49 and 0.51, while the MABS should be set closer at the size of the object to be detected. However, this trend is too unsteady.

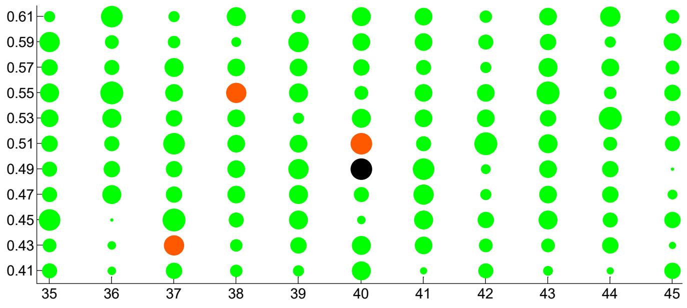


Fig. 13. IoUs_{SN} are top in the middle of the axes. The best accuracy is achieved at $r = 0.49$ ($r = 0.5$ is plausible) and $a = 40$ by no missed detections.

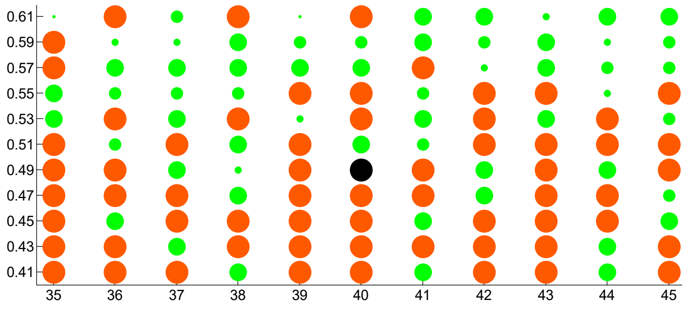


Fig. 14. The axes middle is the best but detection omissions have no trends.

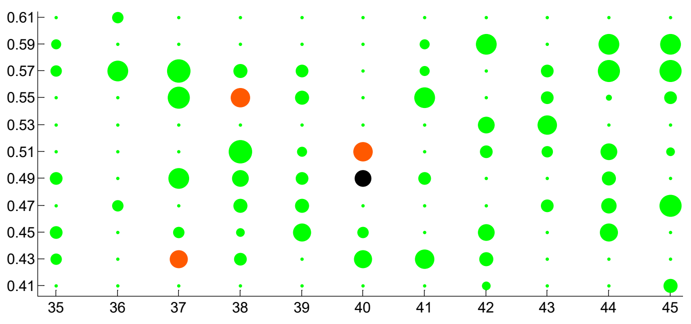


Fig. 15. The axes middle is the best for minimal IoUs but they have no trends.

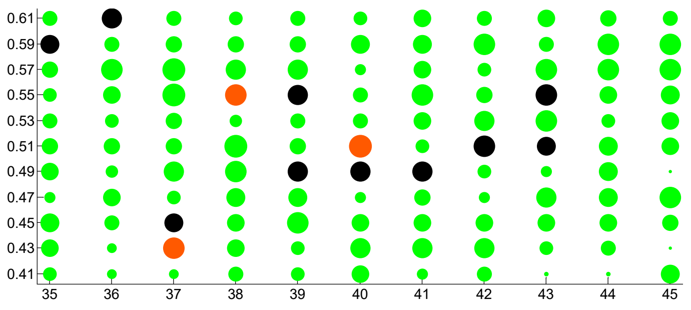


Fig. 16. IoUs are less scattered in the axes middle but some artefacts are seen.

For the ODT whose 25 training (Fig. 6) and 23 testing images (Fig. 7) are of class “background”, we form BBORs by $q=r=0.5$ and MABSs by $a=42$. For such ODTs (where an image may not contain an object), the main characteristics of the detector performance are similar to IoUs$_{\text{SN}}$ and NsDOs$_{\text{SN}}$. Additionally, a number M_{backgrd} of class “background” images (which do not contain any objects) is very important to be considered by a total number N of testing images within their set $\mathscr{I}$ and a set of the detector parameters P. For instance, $P=\{\mathbf{A}, q, r, s, n\}$ in our case. If $f_{\text{object}}(P)$ is a set of correctly found objects (CFOs), $f_{\text{backgrd}}(P)$ is a set of correctly found backgrounds (CFBs), and $u(P, I)$ are IoUs by the respective set I of images with the CFOs, then the standardised indicators of the detector performance are the following:

$$\bar{f}_{\text{object}}(P)=\left|f_{\text{object}}(P)\right| / \left(N-M_{\text{backgrd}}\right), \tag{1}$$

$$\bar{f}_{\text{backgrd}}(P)=\left|f_{\text{backgrd}}(P)\right| / M_{\text{backgrd}}, \tag{2}$$

$$\bar{u}(P)=\sum_{I\in\mathscr{I}} u(P, I) \Big/ \left(N-M_{\text{backgrd}}\right). \tag{3}$$

These indicators are shown against s (horizontal) and n (vertical) in Fig. 17 (CFOs), Fig. 18 (CFBs), and Fig. 19 (IoUs) for the current ODT (its testing subset is in Fig. 7).

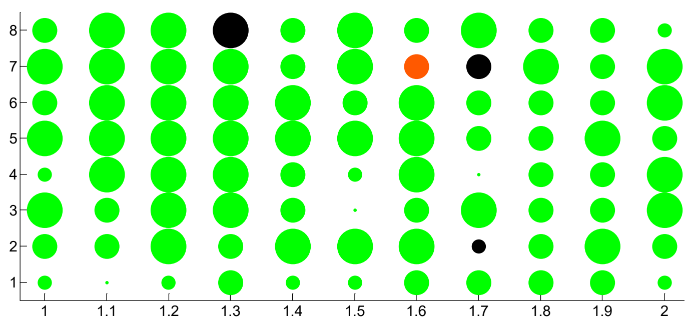


Fig. 17. CFOs by (1) achieve the top rate at higher n but there is no regularity.

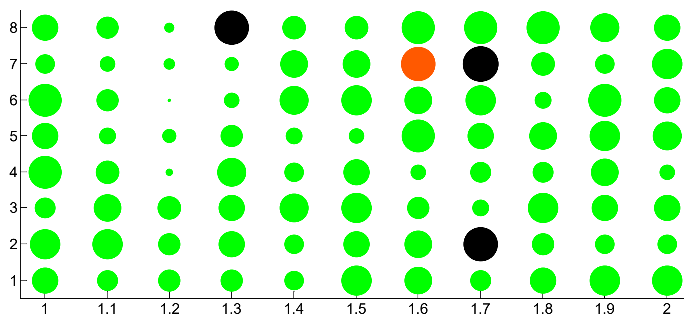


Fig. 18. CFBs by (2) are poorer than CFOs by (1), getting better for $s>1.3$.

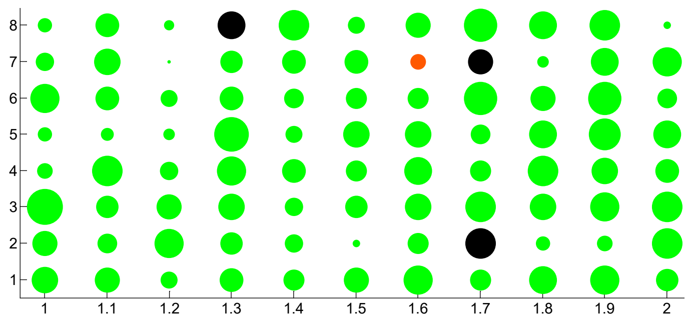


Fig. 19. The rate of IoUs is high but IoUs$_{\text{SN}}$ by (3) are irregularly scattered.

Apparently, in Fig. 17 there is contrary to what Fig. 18 and Fig. 19 show. The roughly best s and n can be set at 1.7 and 8, respectively. However, influence of these two on indicators (1)– (3) still seems stochastic-like.

CFOs by (1), CFBs by (2), and IoUs$_{\text{SN}}$ by (3) are shown against q, r, and a in Figs. 20–22. Those 3D graphs show that MABS does not influence the detector performance. Influence of BBOR for NegTS is weak, because faces in the graphs for greater q seem as they are clipped. BBOR for PosTS influences: $r\in(0.6;0.8)$ brings the best performance.

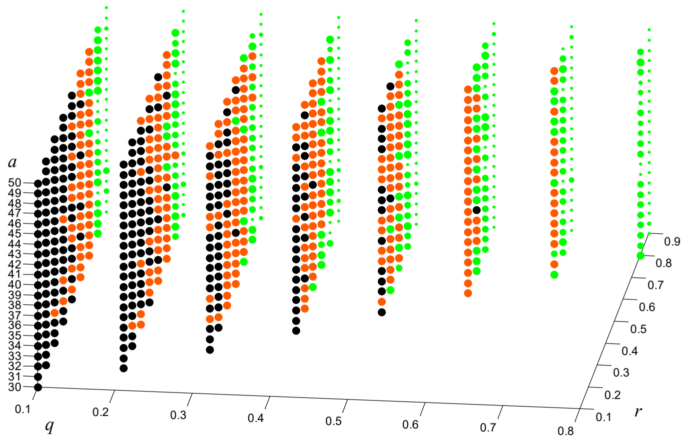


Fig. 20. Ignoring CFBs, CFOs by (1) achieve the top rate at smaller q and r.

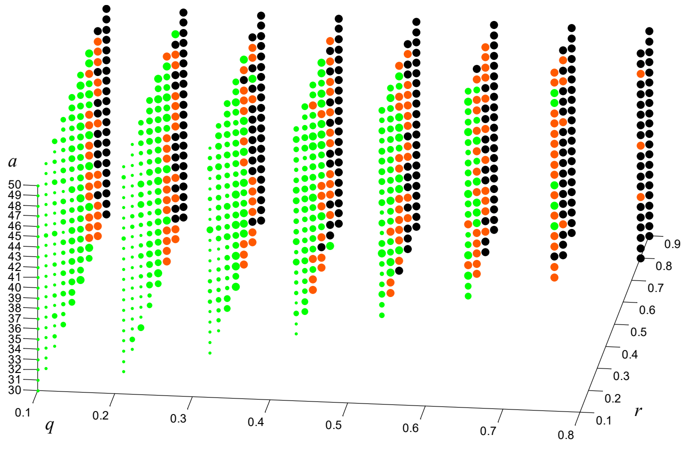


Fig. 21. Ignoring CFOs, CFBs by (2) achieve the top rate at $r > 0.7$ (where $r = 0.9$ is the best for PosTS), independently of MABS and BBOR for NegTS.

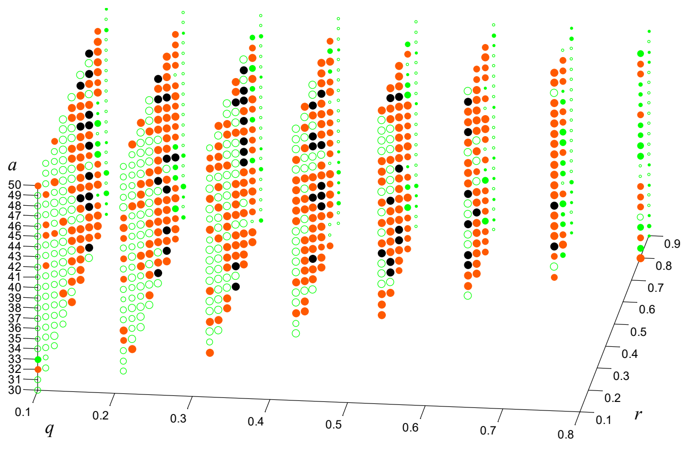


Fig. 22. IoUs$_{SN}$ by (3) are almost independent of MABS achieving their top rate at $q < 0.7$ and $0.6 < r < 0.8$. IoUs$_{SN}$ that have zero IoUs are dot-circled.

After all, how can it be that BBORs for NegTS along with MABS are non-influential? Figures 20–22 do not show it but those dot-circled zero IoUs correspond to poor triplets of q, r, and a. Summed and standardised numbers of such triplets for NegTS show that appropriate values for q are between 0.5 and 0.7 (Fig. 23). A conclusion on MABS is unobvious but setting $a \in \{33, 37\}$ looks reasonable (Fig. 24).

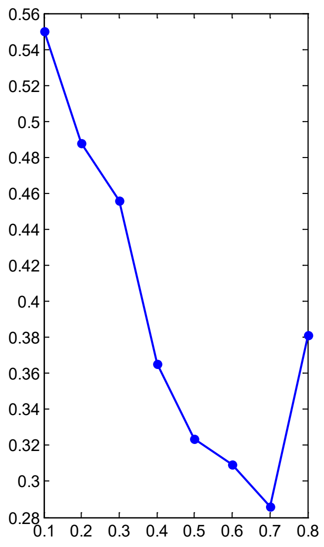


Fig. 23. Relative fails of the detector (recall Fig. 7) against BBOR for NegTS.

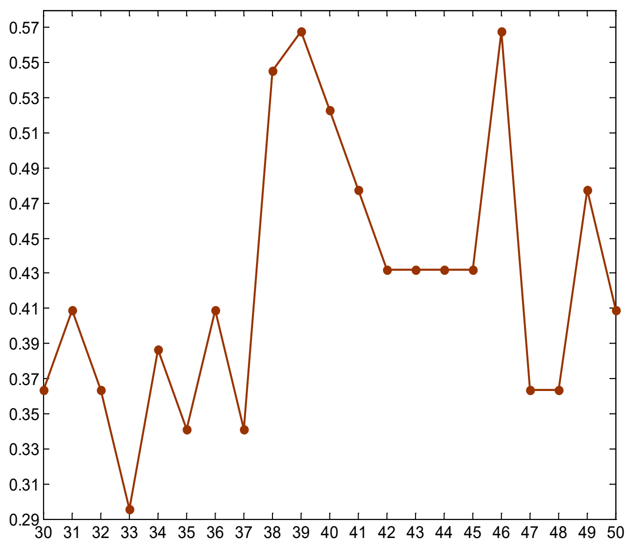


Fig. 24. Relative fails of detector (Fig. 7) against MABS. The polyline appears as not having a distinct minimum, so any conclusion is rough.

IoUs$_{SN}$ against a (horizontal) and r (vertical) for the third ODT are shown in Fig. 25, where the range of MABS is thinned twice as its influence is too weak (see analogue in Fig. 11). Once again IoUs$_{SN}$ seem not depend on MABS. NsDOs$_{SN}$ are shown in Fig. 26 with empty circled dots corresponding to performances with zero IoUs. There are two pairs, $\{a = 38, r = 0.5\}$ and $\{a = 44, r = 0.4\}$, at which IoUs$_{SN}$ and NsDOs$_{SN}$ are top, and the minimal IoU is greater than 0.38.

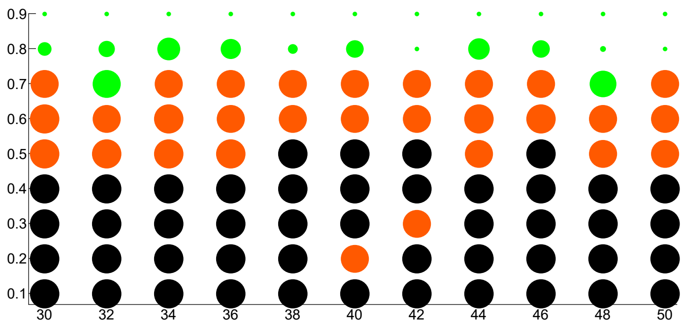


Fig. 25. IoUs$_{SN}$ are better at smaller BBOR for PosTS but a few exclusions exist. No visible influence of MABS. Setting $r > 0.5$ is clearly unacceptable.

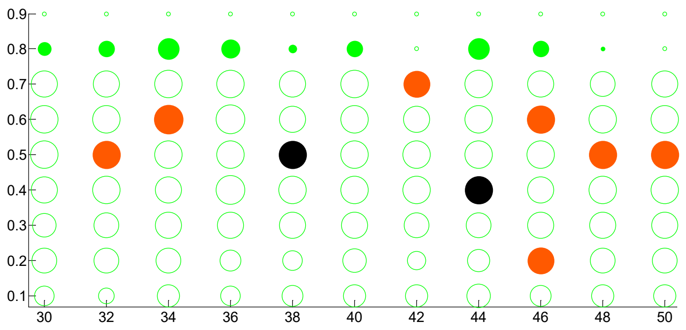


Fig. 26. Top NsDO$_{SN}$ are at $r \in [0.4; 0.5]$ but they are not guaranteed. The minimal IoU is greater than 0.38 in only two cases without detection omissions (black dots). Many zero IoUs, despite great IoUs$_{SN}$ (greater radii).

VIII. Analysis of the Obtained Results and Decision on the Appropriateness

Now, it is obvious that the anchor box parameters are far less influential than BBORs. In particular, MABS is unexpectedly open to be set at any value close to width or height of the object. The anchor box pyramid scale factor is a free choice of a value below 2 (this is the default value in MATLAB) down an aspect ratio (greater than 1) for non-square and non-round shape objects. An appropriate number of levels in an anchor box pyramid for such objects is seemingly between 6 and 10, because $n > 10$ may retard training. For objects, whose shape approaches a square or round, an appropriate number of levels is 2 as setting $n = 1$ is heedless even for ODTs like that with the testing set in Fig. 4.

BBORs strongly influence the detector performance, and their values should be taken more carefully than a value of MABS. Without class "background", BBOR for NegTS does not matter and thus it is set at a negligible minimum (say, 0.001 or so). Referring to graphs in Figs. 11–16 and Fig. 25, Fig. 26, with a purpose of an aggregation, an appropriate BBOR for PosTS is a value from interval (0.4; 0.6). Endpoints 0.4 and 0.6 are hardly ever considerable. However, for objects, whose shape approaches a square or round, an appropriate BBOR for PosTS tends to be greater than for non-square and non-round shape objects. This is ascertained from comparing stripes of black dots in Fig. 11 and Fig. 12 to Fig. 25.

For ODTs containing class "background", i.e., where images do not necessarily contain objects, BBORs for NegTS and PosTS may be identical. A small gap between them may exist but it is unlikely to bring a significant effect. This is ascertained from comparing bunches of black dots in faces in Fig. 20 and Fig. 21. In spite of seeming weak influence of BBORs for NegTS, which is observable in Fig. 20 – Fig. 22, relative fails of the detector decrease as the value of BBOR for NegTS increases up to 0.7 (Fig. 23). Meanwhile, the detector fails least if BBOR for PosTS is set at a value between 0.7 and 0.8 (Fig. 27). The number of such fails disclosed in Fig. 28 against BBORs for NegTS (vertical) and PosTS (horizontal) confirms that. Hence, setting both BBORs at 0.7 or close to that is an appropriate choice to work with class "background".

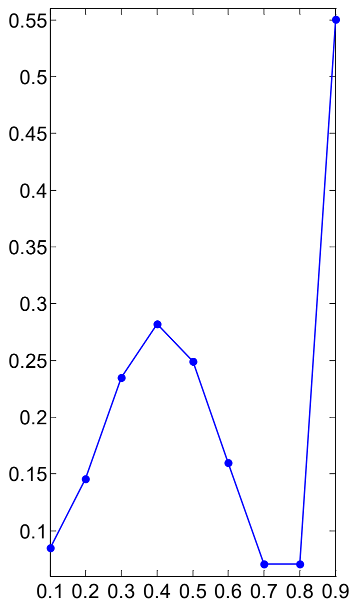


Fig. 27. Relative fails of the detector (recall Fig. 7) against BBOR for PosTS. This graph reminds the graph in Fig. 23 only for $r \geqslant 0.4$. For $r < 0.4$, CFBs by (2) are poor (Fig. 21).

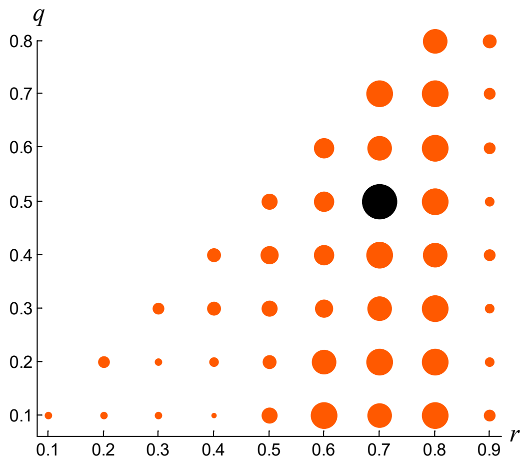


Fig. 28. The number of detector fails inversely proportional to the dot radius. The single black dot is that where we have no fail. Detectors at vertical stripes for $r \in \{0.1, 0.2, 0.3, 0.4, 0.9\}$ perform unacceptably poorly, without exclusions. Every horizontal stripe has its exclusions, at which detectors perform with a few fails.

IX. Discussion and Conclusion

We have considered detecting a single object and the case when an image does not contain it. What if there were two or three or more objects? What would influence of BBORs and MABSs then be? Would they influence in a similar way compared to the considered cases? These questions are an obvious extension of the question raised in the beginning. Answering them requires far broader research involving more heterogeneous and bigger datasets.

The present research, even based on simply black squares and rectangles within one-origin ODTs, turned out to have much confusing results. Generally, influence of BBORs on the Faster R-CNN detector performance is some fuzzy. Influence of the anchor box parameters is fuzzier. Nonetheless, appropriateness in setting those parameters and ratios exists. First, a range of object sizes should be learned, although BBORs seem not to depend on the range. Second, an anticipated difference exists between detecting just ever-present objects and revealing sometimes that an image does not contain any objects. Thus, it should be learned whether class "background" is present. If it is so, BBORs are tried both at 0.7; if not, BBOR for PosTS is tried at a less value but greater than 0.4. MABS is better to try at a lesser value from the range (recall Fig. 24). The anchor box pyramid scale factor and the number of levels are better to try at 2 and 8, respectively. Subsequently, these parameters may be corrected if IoUs or one of indicators (1)– (3) is poor.

The main result of the experimentation is that anchor box parameters and BBORs are impossible to set at their best values at one stroke. They instead are corrected at a few stages. The first stage, the fuzziest one, has been described. Due to the said fuzziness, this is still a rough technique that might be called a start-off. It can probably be expanded by trying to detect more objects in an image, but without losing generality. The expansion is expected to confirm that way of the appropriateness for adjusting the anchor box parameters and BBORs, rather than ascertain appropriate values for them.

Acknowledgment

The research has technically been supported by the Faculty of Navigation and Naval Weapons at the Polish Naval Academy, Gdynia, Poland.

References

[1] R. Klette, *Concise Computer Vision: An Introduction into Theory and Algorithms*. Springer, 2014. https://doi.org/10.1007/978-1-4471-6320-6

[2] A. Balasubramanian, S. Kamate, and N. Yilmazer, "Utilization of robust video processing techniques to aid efficient object detection and tracking," *Procedia Computer Science*, vol. 36, pp. 579–586, 2014. https://doi.org/10.1016/j.procs.2014.09.057

[3] V. V. Romanuke, "Parametrization of the optical flow car tracker within MATLAB Computer Vision System Toolbox for visual statistical surveillance of one-direction road traffic," *Radio Electronics, Computer Science, Control*, no. 3, pp. 40–48, 2015. https://doi.org/10.15588/1607-3274-2015-3-5

[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2014),* June 23–28, pp. 580–587, 2014. https://doi.org/10.1109/cvpr.2014.81

[5] E. R. Davies, *Computer Vision: Principles, Algorithms, Applications, Learning*. Academic Press, 2018. https://doi.org/10.1016/B978-0-12-809284-2.00021-6

[6] X. Cheng, J. Lu, J. Feng, B. Yuan, and J. Zhou, "Scene recognition with objectness," *Pattern Recognition*, vol. 74, pp. 474–487, 2018. https://doi.org/10.1016/j.patcog.2017.09.025

[7] J. R. Uijlings, K. E. Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, 2013. https://doi.org/10.1007/s11263-013-0620-5

[8] R. Girshick, "Fast R-CNN," *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV 2015),* December 7–13, pp. 1440–1448, 2015. https://doi.org/10.1109/iccv.2015.169

[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017. https://doi.org/10.1109/TPAMI.2016.2577031

[10] Y. Zhang, Y. Bai, M. Ding, Y. Li, and B. Ghanem, "Weakly-supervised object detection via mining pseudo ground truth bounding-boxes," *Pattern Recognition*, vol. 84, pp. 68–81, 2018. https://doi.org/10.1016/j.patcog.2018.07.005

[11] K. M. Adal, D. Sidibé, S. Ali, E. Chaum, T. P. Karnowski, and F. Mériaudeau, "Automated detection of microaneurysms using scale-adapted blob analysis and semi-supervised learning," *Computer Methods and Programs in Biomedicine*, vol. 114, no. 1, pp. 1–10, 2014. https://doi.org/10.1016/j.cmpb.2013.12.009

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 1, pp. 1097–1105, 2012.

[13] L. H. S. Vogado, R. M. S. Veras, F. H. D. Araujo, R. R. V. Silva, and K. R. T. Aires, "Leukemia diagnosis in blood slides using transfer learning in CNNs and SVM for classification," *Engineering Applications of Artificial Intelligence*, vol. 72, pp. 415–422, 2018. https://doi.org/10.1016/j.engappai.2018.04.024

[14] D. Han, Q. Liu, and W. Fan, "A new image classification method using CNN transfer learning and web data augmentation," *Expert Systems with Applications*, vol. 95, pp. 43–56, 2018. https://doi.org/10.1016/j.eswa.2017.11.028

[15] V. V. Romanuke, "Appropriateness of DropOut layers and allocation of their 0.5 rates across convolutional neural networks for CIFAR-10, EEACL26, and NORB datasets," *Applied Computer Systems*, vol. 22, pp. 54–63, 2017. https://doi.org/10.1515/acss-2017-0018

[16] V. V. Romanuke, "Appropriate number of standard 2×2 max pooling layers and their allocation in convolutional neural networks for diverse and heterogeneous datasets," *Information Technology and Management Science*, vol. 20, pp. 12–19, 2017. https://doi.org/10.1515/itms-2017-0002

Vadim Romanuke graduated from the Technological University of Podillya (Ukraine) in 2001. In 2006, he received the Degree of Candidate of Technical Sciences in Mathematical Modelling and Computational Methods. The Candidate Dissertation suggested a way of increasing interference noise immunity of data transferred over radio systems. The degree of Doctor of Technical Sciences in Mathematical Modelling and Computational Methods was received in 2014. The Dissertation to be granted the degree of Doctor of Science solved a problem of increasing efficiency of identification of models for multistage technical control and run-in under multivariate uncertainties of their parameters and relationships. In 2016, he received the status of Full Professor. He is a Professor of the Faculty of Navigation and Naval Weapons at the Polish Naval Academy. His current research interests concern decision making, game theory, statistical approximation, and control engineering based on statistical correspondence. He has 323 published scientific articles, one monograph, one tutorial, three methodical guidelines in functional analysis, Master Thesis development in mathematical and computer modelling, conflict-controlled systems. Before January 2018, Vadim Romanuke was the scientific supervisor of a Ukrainian budget grant work concerning minimization of water heat transfer and consumption. He also leads a branch of fitting statistical approximators at the Center of Parallel Computations in Khmelnitskiy, Ukraine.
Address for correspondence: 69 Śmidowicza Street, Gdynia, Poland, 81–127.
E-mail: romanukevadimv@gmail.com
ORCID iD: https://orcid.org/0000-0003-3543-3087