Generating Positive Bounding Boxes for Balanced Training of Object Detectors

Kemal Oksuz, Baris Can Cam, Emre Akbas*, Sinan Kalkan*
Department of Computer Engineering
Middle East Technical University, Ankara, Turkey
{kemal.oksuz, can.cam, eakbas, skalkan}@metu.edu.tr

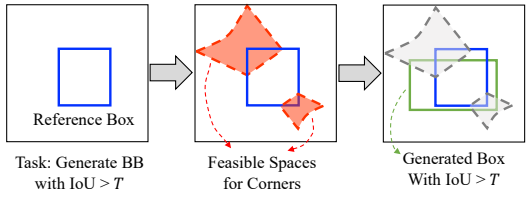
Abstract

Two-stage deep object detectors generate a set of regions-of-interest (RoIs) in the first stage, then, in the second stage, identify objects among the proposed RoIs that sufficiently overlap with a ground truth (GT) box. The second stage is known to suffer from a bias towards RoIs that have low intersection-over-union (IoU) with the associated GT boxes. To address this issue, we first propose a sampling method to generate bounding boxes (BB) that overlap with a given reference box more than a given IoU threshold. Then, we use this BB generation method to develop a positive RoI (pRoI) generator that, for the second stage, produces RoIs following any desired spatial or IoU distribution. We show that our pRoI generator is able to simulate other sampling methods for positive examples such as hard example mining and prime sampling. Using our generator as an analysis tool, we show that (i) IoU imbalance has an adverse effect on performance, (ii) hard positive example mining improves the performance only for certain input IoU distributions, and (iii) the imbalance among the foreground classes has an adverse effect on performance and that it can be alleviated at the batch level. Finally, we train Faster R-CNN using our pRoI generator and, compared to conventional training, obtain better or on-par performance for low IoUs and significant improvements when trained for higher IoUs for Pascal VOC and MS COCO datasets. The code is available at: `https://github.com/kemaloksuz/BoundingBoxGenerator`.

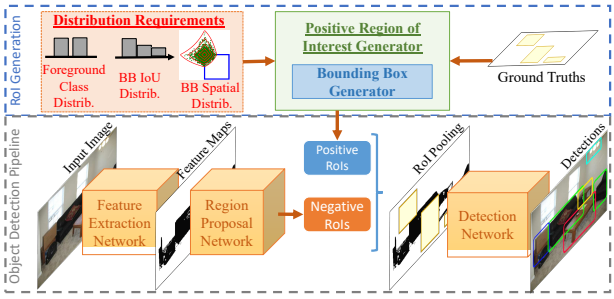
1. Introduction

An important challenge in object detection is class imbalance [2, 17, 20, 27, 31]: even from a single image, an infinite number of negative examples can be sampled, in contrast to only a limited set of positive RoIs (regions-of-interest). Naturally, this leads to significant imbalance between negatives and positives. Class imbalance also exists within foreground classes.

*Equal contribution for senior authorship.



(a) Bounding Box Generation



(b) Generating Bounding Boxes for Training an Object Detector

Figure 1. **(a)** *An illustration of Bounding Box (BB) Generation.* Given a reference box (in blue) and an IoU threshold T, a BB having at least T IoU is generated (drawn in green). **(b)** *An illustration of training an object detector with positive region-of-interests.* Given distribution requirements on foreground classes and BBs, we generate positive RoIs using the BB generator (Fig 1(a)). Negative RoIs are still generated by the region proposal network.

A prominent solution to the foreground-background class imbalance is to have two stages [5, 10, 30]: The first stage estimates regions (i.e., RoIs) that are likely to contain objects, significantly discarding background samples, and the second-stage classifies these regions into objects, and also fine-tunes the coordinates of the bounding boxes. Other solutions generally employ sampling with hard constraints (e.g., online hard example mining [31], Libra RCNN [27]) or soft constraints (e.g., focal loss [20], harmonizing gradients [17]).

The foreground-foreground class imbalance problem, i.e., the imbalance in the number of examples pertaining to different positive classes at the image, dataset or mini-batch levels, has not attracted as much attention. In addition, the IoU distribution of the RoIs generated by the region pro-

posal network (RPN) [30] is imbalanced [1], which biases the BB regressor in favor of the IoU that the distribution is skewed towards. We call this imbalance problem as *IoU distribution imbalance*. Addressing these problems requires a careful analysis of the positive RoIs.

In this paper, we analyze and address foreground-foreground class imbalance and IoU distribution imbalance by actively generating BBs. We first propose the "BB generator", a method that can generate an arbitrary BB overlapping with a reference box with an IoU larger than a given threshold (Figure 1(a)). Using the BB generator, we develop a positive RoI (pRoI) generator that can produce RoIs conforming to desired IoU and relative spatial distributions (Figure 1(b)). Considering that there is a correlation between the hardness of an example and its IoU [27], the pRoI generator can *generate* (rather than sample) not only positive or negative samples, but also samples with any desired property such as hard examples [31] or prime samples [2].

We use our pRoI generator to perform several analyses and improvements. Specifically, we (i) show that IoU and foreground class distributions affect performance, (ii) make a comparative analysis for RPN RoIs and (iii) improve the performance of Faster RCNN for IoU intervals where RPN is not able to generate enough samples.

Finally, we devise an online, foreground-balanced (OFB) sampling method which considers the imbalance among the foreground classes dynamically within a training batch based on multinomial sampling.

Contributions. Overall, our main contributions are as follows:

1. Generators: (i) A BB generator to generate BBs for a given IoU threshold and (ii) a positive RoI generator to generate RoIs with desired foreground class, IoU and relative spatial distributions.

2. Imbalance Problems and Analysis: Using our pRoI generator, we show that IoU distribution and foreground-foreground class imbalance within a training batch affect the performance of the object detectors. We also provide an analysis of RPN RoIs and show that the effect of the hard examples depends on the IoU distribution of the BBs.

3. Practical Improvements: We train a detection network using our pRoI generator, which increases the amount and the diversity of the positive examples especially for the larger IoUs, and show that the performance improves compared to the standard training (e.g. for $IoU = 0.8$, $\mathrm{mAP@0.8}$ improves by 10.9% for Pascal VOC). We also train the conventional detection pipeline by using the proposed OFB sampling, and improve the performance.

2. Related Work

Deep Object Detectors: We can group deep object detectors into two: One-stage methods and two-stage methods. While one-stage methods [8, 20, 23, 28, 29] predict the object categories and their BBs directly from anchors, two-stage methods [5, 10, 11, 30] first estimate a set of RoIs from anchors and then predict objects from these RoIs in the second stage. Both approaches use a deep feature extractor [13, 33], optionally followed by steps like feature pyramid networks [9, 15, 19, 22].

Our BB sampling approach is more suitable for the second stage of the two-stage methods since one-stage detectors have structural constraints owing to the fact that each output of a one-stage detector corresponds to a predefined anchor having fixed location, shape and scale. For this reason, an additional module is required to employ our generator. However, having balanced IoU and foreground class distributions are relevant for any object detection pipeline since any object detector needs to deal with BBs even if they are estimated or fixed (in the case of anchors).

Class Imbalance in Object Detection: Following Oksuz et al. [25], we categorize the class imbalance problem for the deep object detectors into two: foreground-background and foreground-foreground class imbalance.

Foreground-background class imbalance has attracted more attention with *hard sampling*, *soft sampling* and *generative approaches*. In hard sampling methods, certain samples are shown more to the network. This can be performed via random sampling [5, 30], or by relying on "sample usefulness" heuristics as in hard-example mining [23, 27, 31] and prime sampling [2]. Hard-example mining methods usually assume that examples with higher loss are more difficult to learn, and therefore, they train a network more with such examples. This approach is adopted for negative samples in SSD [23], while a more systematic approach considering both the positive and negative samples is proposed in online hard example mining (OHEM – [31]). An alternative hardness definition was proposed in Libra R-CNN [27] based on a sample's IoU, and a solution was proposed using hard example mining using BB IoUs without computing the loss for the entire set. A recent interesting method, "prime sampling" [2], asserts that positive samples with higher IoUs are more representative and proposed ranking the positive samples based on its IoU with the ground truth, while still showing that hard example mining for the negative class works well. BB IoU imbalance is addressed by Cascade R-CNN [1] by employing cascaded detectors in such a way that a later-stage detector is trained by a distribution skewed towards higher IoU.

In *soft sampling*, a weight is assigned to each sample rather than performing a discrete (hard) selection of samples. Prominent examples include focal loss [20], which promotes hard examples; prime sampling [2], which assigns more weight to examples with higher IoUs; and finally gradient harmonizing mechanism [17], which assigns lower weights to easy negatives and suppresses the effect of the outliers.

The *generative methods* address imbalance with a different perspective by introducing generated samples. Example approaches include generating hard examples with various deformations and occlusion [32] and generating synthetic examples [12].

Foreground-foreground class imbalance is critical as well. Kuznetsova et al. [16] showed that object detection datasets are highly imbalanced also for foreground classes. The only method to consider the problem at the dataset level handcrafts a similarity measure, and based on the measure clusters the classes to have a more balanced training [26]. In the classification domain where there is no background class, this imbalance is studied more [7, 14] by, e.g., performing class-aware sampling [18]. However, these methods are not directly applicable for two-stage object detectors because the second stage's input is very dynamic since it depends on RoIs estimated by the first stage. Despite this difference, class-aware sampling is said to be adopted by [22], however no comparison is presented for balanced and imbalanced training from the object detection perspective.

Our ideas in this paper are relevant for both foreground-background and foreground-foreground class imbalance. One can generate any number of positive RoIs to address the foreground-background imbalance, and the generated set can also be chosen equally from each class to address the foreground-foreground imbalance. Among the three types of methods mentioned above, we classify our approach as a generative method. Since the end-to-end training pipeline is not disrupted (see Figure 1(b)), any hard sampling method [27, 31] can also be simulated. In addition, we directly address foreground-foreground class imbalance by online foreground balanced (OFB) sampling. Its main difference from the previously proposed class-aware sampling [18] is that while they use a static dataset, our OFB sampling is able to handle the dynamic nature of the RoIs (i.e. the batch depends on the sampled RoIs at each iteration) owing to the proposal network.

3. The Generators

In this section, we describe the methods for generating bounding boxes and balanced positive RoIs.

3.1. Definitions and Notation

Let $B = [x_1, y_1, x_2, y_2]$ denote a ground-truth box with top-left corner $\mathrm{TL}(B) = (x_1, y_1)$ and bottom-right corner $\mathrm{BR}(B) = (x_2, y_2)$ satisfying $x_2 > x_1$ and $y_2 > y_1$. The area of B is simply defined as:

$$\mathrm{A}(B) = (x_2 - x_1) \times (y_2 - y_1), \tag{1}$$

and the area of the intersection between B and $\bar{B}$ is:

$$\begin{aligned} \mathrm{I}(B, \bar{B}) \quad = \quad & (\min(\bar{x_2}, x_2) - \max(\bar{x_1}, x_1)) \times \\ & (\min(\bar{y_2}, y_2) - \max(\bar{y_1}, y_1). \end{aligned} \tag{2}$$

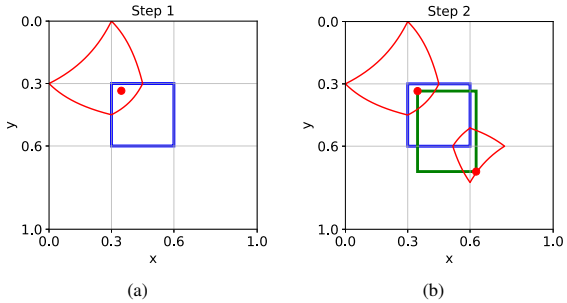


Figure 2. (a,b) Applying Algorithm 1 on the blue BB (B) with $T = 0.5$. Red polygons denote boundaries for top-left and bottom-right points that can be sampled with an IoU larger than $T = 0.5$. Red dots are sampled points, and green box is the generated box ($\bar{B}$) with $IoU = 0.5071$.

Based on this notation, $\mathrm{IoU}(B, \bar{B})$ can be easily defined as:

$$\mathrm{IoU}(B, \bar{B}) = \frac{\mathrm{I}(B, \bar{B})}{\mathrm{A}(B) + \mathrm{A}(\bar{B}) - \mathrm{I}(B, \bar{B})}. \tag{3}$$

Finally, we note two useful properties of the IoU function: (**Theorem 1**) $\mathrm{IoU}(B, \bar{B})$ is scale-invariant, and (**Theorem 2**) $\mathrm{IoU}(B, \bar{B})$ is translation-invariant (see Suppl. Mat. for the proofs). These theorems allow us to shift and scale the input BBs to a reference box during BB generation and then shift and scale them back to their original aspect ratio and location.

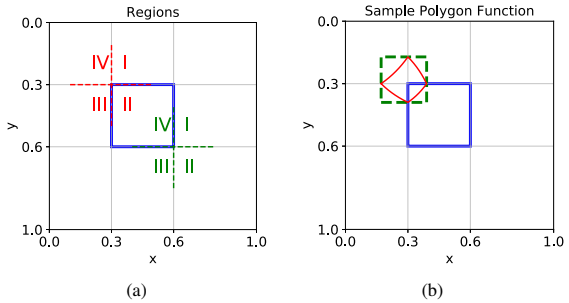
3.2. Bounding Box Generator

Algorithm 1 Bounding Box Generator. See Section 3.2 and the Suppl. Mat. for the definitions of the functions.

1: **procedure** GENERATEBB(B, T)
2: # Step-1: Find top-left corner
3: $TLPoly \leftarrow \mathrm{findTLFeasibleSpace}(B, T)$
4: $\mathrm{TL}(\bar{B}) \leftarrow \mathrm{samplePolygon}(TLPoly)$
5: # Step-2: Find bottom-right corner
6: $BRPoly \leftarrow \mathrm{findBRFeasibleSpace}(B, T, \mathrm{TL}(\bar{B}))$
7: $\mathrm{BR}(\bar{B}) \leftarrow \mathrm{samplePolygon}(BRPoly)$
8: **return** $[\mathrm{TL}(\bar{B}), \mathrm{BR}(\bar{B})]$
9: **end procedure**

Given a reference box B and a threshold T, the goal of the BB generator is to determine a new box $\bar{B} = [\bar{x_1}, \bar{y_1}, \bar{x_2}, \bar{y_2}]$ such that $\mathrm{IoU}(B, \bar{B}) \geq T$. To generate such a box, we propose a 2-step algorithm presented in Algorithm 1 and illustrated in Fig. 2. The first step (lines 3-4) finds the polygon[1] that computes the feasible space for $\mathrm{TL}(\bar{B}) = (\bar{x_1}, \bar{y_1})$, which satisfies the desired IoU, and samples a point in this polygon. The second step (lines 6-7) takes into account the sampled $\mathrm{TL}(\bar{B})$ and, similar to Step 1, determines a feasible space for bottom-right corner, then, samples $\mathrm{BR}(\bar{B})$.

[1]Note that the shape is not strictly a polygon; however, we approximate it as one at regular small intervals, and therefore, we call it a polygon for the sake of simplicity.



(a) (b)

Figure 4. **(a)** The regions around $\mathrm{TL}(B)$ and $\mathrm{BR}(B)$ are splitted into four each. Red and green dashed lines split the top left and bottom right regions respectively. The numbers label the splitted regions.), **(b)** In the execution of the sample polygon function for $T = 0.75$, green dashed box is the enclosing box for the TL space polygon.

This order leads to a non-isotropic distribution with respect to the reference box. To make it isotropic, we can also sample in the reverse order: i.e. sample BR first then TL. We then randomly choose the order, before sampling. Fig. 3 superimposes 1000 generated boxes with $T = 0.6$.

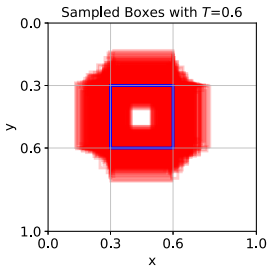


Figure 3. $1K$ generated boxes (shown with red) by Algorithm 1 for reference box drawn in blue (B) and IoU threshold $T = 0.6$.

The following two sections discuss how the feasible space is computed (i.e. $\mathrm{findTLFeasibleSpace}(B, T)$) and how a point can be sampled within a polygon (i.e. $\mathrm{samplePolygon}(TLPoly)$). See the Suppl. Mat. for $\mathrm{BR}(\bar{B})$.

3.2.1 Determining Feasible Space for the Desired IoU

$\mathrm{findTLFeasibleSpace}(B, T)$ is the function determining the feasible set of points that can be the top left point of a box ensuring the desired IoU. In order to find the set of these feasible points (i.e. $\mathrm{TL}(\bar{B})$) that satisfy Eq. 3, we assume that $\mathrm{BR}(\bar{B}) = \mathrm{BR}(B)$ and manipulate Eq. 3, otherwise, some feasible points are excluded in the feasible top left space. Even though $\mathrm{BR}(\bar{B})$ is fixed, there are still two unknown variables $\bar{x_1}$ and $\bar{y_1}$. That's why, we first bound one of these two variables and then find the value of the unbounded variable by moving within the limits of the bounded variable with some precision (we use 0.0001 as precision). Since the definition of the $\mathrm{IoU}(B, \bar{B})$ is different in each of the four regions depicted in Fig. 4(a) due to the $\max$ and $\min$ operations, an equation is to be derived for each region.

Denoting the minimum and maximum bounds of $\bar{x_1}$ in Region I by x^I_{min} and x^I_{max} respectively, we bound the values in x axis. It is obvious that $x^I_{min} = x_1$ due to the boundary of Region I. To find x^I_{max}, we manipulate Eq. 3 by exploiting that $\bar{y_1} = y_1$ for x^I_{max}, which yields:

$$x^I_{max} = x_2 - (x_2 - x_1) \times T. \tag{4}$$

Having determined the boundaries for $\bar{x_1}$, now we derive a function that determines $\bar{y_1}$ given $\bar{x_1}$. Finally, moving within the bounds yields $\bar{x_1}, \bar{y_1}$ pairs satisfying $\mathrm{IoU}(B, \bar{B}) = T$ when $\mathrm{BR}(\bar{B}) = \mathrm{BR}(B)$. In region I, note that $I(B, \bar{B})$ does not rely on $\bar{y_1}$ (i.e. $I(B, \bar{B}) = (x_2 - \bar{x_1})(y_2 - y_1)$). Bringing these together, $\bar{y_1}$ can be defined as (see Suppl. Mat. for the entire derivation of x^I_{max} and $\bar{y_1}$):

$$\bar{y_1} = y_2 - \frac{\frac{I(B,\bar{B})}{T} + I(B, \bar{B}) - A(B)}{(x_2 - \bar{x_1})}. \tag{5}$$

Here, we only show the derivation steps for Region I and present the equations for all regions in Suppl. Mat. Combining the points in all these regions yields the polygon limiting feasible region with $IoU \geq T$.

3.2.2 Controlling the Relative Spatial Distribution of the Boxes

$\mathrm{samplePolygon}(TLPoly)$ function determines the BB spatial distribution. We follow rejection sampling [3] in such a way that a point is proposed by the proposal distribution until it hits the inside of the polygon. Accordingly, the proposal distribution determines the BB spatial distribution. Fig. 4(b) presents an example for spatial uniform distribution for the top-left space polygon with $T = 0.75$. We sample a point in the rectangle uniformly, which corresponds basically to generating two uniform numbers within a range. If the point is in the polygon, then it is accepted, else a new point is proposed until it is inside the polygon. Note that different proposal distributions lead to different relative spatial distributions for the generated BBs.

3.3. pRoI Generator: Training by Generated BBs

This section provides an application of our BB generator for generating positive RoIs for training a two-stage object detector. By applying our BB generator to the ground-truth boxes, we can generate positive RoIs with desired characteristics. This enables us to (i) analyze how the performance of Faster R-CNN is affected by the properties of the positive RoIs and (ii) improve the performance for IoU intervals where RPN is not able to generate enough samples.

Algorithm 2 Positive RoI Generator. See Section 3.3 and the Suppl. Mat. for the definitions of functions $\mathrm{fgBalancedRoIAlloc}$ and $\mathrm{genRoIs}$.

procedure GENERATEPROI($GTs, \psi_{IoU}, W_{IoU}, RoINum$)
  $perGtRoI = \mathrm{fgBalancedRoIAlloc}(GTs, RoINum)$
  $RoIs = \mathrm{genRoIs}(GTs, perGtRoI, \psi_{IoU}, W_{IoU}, RoINum)$
  return $RoIs$
end procedure

The method, "Positive RoI Generator" (pRoI Generator), described in Algorithm 2, can control several different characteristics of the set of positive RoIs. $\mathrm{fgBalancedRoIAlloc}()$ first divides $RoINum$ by the number of different classes in the given ground truth set, GTs, to determine the allocated box number per class, and then shares this value among each example of the same class equally. As a result, $\mathrm{fgBalancedRoIAlloc}()$ determines the number of boxes to be generated for each ground truth box in GTs. Secondly, given the allocated number of boxes for each ground truth, $\mathrm{genRoIs}()$ iteratively uses BB generator as a subroutine to provide a set of $RoINum$ RoIs. In this step, the IoU distribution requirement is determined by the inputs ψ_{IoU}, the base of the IoU bins and the weight of the each bin denoted by W_{IoU}. W_{IoU} is basically a multinomial distribution over the bins determined by ψ_{IoU}. An important benefit of pRoI generator is that training with the generated RoIs has no impact on the gradient flow for the training process (see Suppl. Mat.). At each training iteration, RPN generates a set of RoIs among which we discard the positive ones and use the positive RoIs generated by the proposed method (Fig. 1(b)). Using our pRoI generator, we can address the imbalance problems regarding RoIs at three different levels:

(1) Foreground-foreground class imbalance, which occurs when a dataset or mini-batch (or batch) contains different numbers of positive examples from different classes. To illustrate on a batch, an image (used as a batch) from PASCAL dataset [6] includes 4 bottles, 2 persons, 2 dining tables and 1 chair. In such a case, having equal number of RoIs per instance may lead the model to be biased in favor of the bottle class while ignoring the chair class. In our pRoI Generator, $\mathrm{fgBalancedRoIAlloc}()$ function allocates the same number of RoIs for each class within the batch.

(2) IoU distribution imbalance, which occurs when the positive RoIs have a skewed IoU distribution (Fig. 5). It has been shown that the hardness of a RoI is related to its IoU [27] and also the regressor overfits to RoIs which has IoU around 0.5 when the distribution of the RPN proposals is concentrated towards 0.5 [1]. Thus, these recent findings imply that the IoU distribution has an important effect on training. As aforementioned, $\mathrm{genRoIs}()$ is able to control the IoU distribution of the BBs.

(3) Relative spatial imbalance, which occurs when the BBs intersect significantly and a diverse set of examples can not be provided to the detection network. This level of imbalance is controlled in our pRoI generator in the subroutine BB generator as discussed in Section 3.2.2.

4. Experimental Setup

Dataset and Implementation Details: We evaluate our generative methods on Faster R-CNN in two different settings: (i) on Pascal VOC 2007 [6] with backbone ResNet-101 following the implementation and training in [34] with batch size 1 image on 1 GPU, and (ii) on MS COCO [21] with backbone ResNet-50 following the implementation and training in [4] with batch size 2 images/GPU on 2 GPUs. During training, 32 positive, and 96 negative RoIs are used from each image in the batch.

Performance Measures: We exhaustively search for the best mean-average-precision (mAP) and mean-optimal-localization-precision-recall (moLRP) error [24] values over epochs and report them. moLRP is a recently introduced metric for object detection, which represents recall, precision and average tightness of the BBs. Note that mAP is a higher-is-better measure, while moLRP is an error metric and thus, it is a lower-is-better measure.

RoI Sources: In addition to RoIs output by RPN, we use the RoIs generated by our pRoI generator, with a given distribution, during the analysis and training. The different distributions are obtained by controlling W_{IoU} (see Suppl. Mat. for the exact configurations of W_{IoU}) in Algorithm 2. Unless otherwise stated, we set $\psi_{IoU} = [0.5, 0.6, 0.7, 0.8, 0.9]$ and $RoINum = 32$. We train these RoI sources with and without foreground balanced sampling in order to see the effects of different imbalance problems on different RoI sources. The results are presented in Table 1.

5. Imbalance Problems and Analysis of RPN RoIs

In this section, using our pRoI generator, we show that IoU and foreground class distributions affect performance, simulate a sampling method and analyze the relative spatial distribution of RPN RoIs.

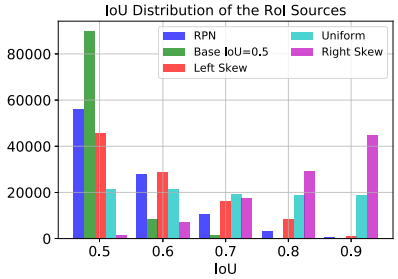


Figure 5. IoU distribution of different RoI Sources. See Suppl. Mat. for the configurations of the RoI sources.

5.1. IoU Distribution Imbalance

Our BB generator method (Algorithm 1) samples boxes for a given IoU threshold, spatially uniformly. It does not impose an upper bound for the IoUs of the sampled boxes. Therefore, in order to analyze the density of the different IoUs for the positive samples, we uniformly generate $100K$ boxes for each IoU distribution type and plot the distribution of the generated boxes in Fig. 5. Note that training a detector with different IoU distributions of positive examples affects the resulting test performance (see Table 1), which implies the effect of IoU distribution imbalance.

From Fig. 5, we observe the following: **(1)** The distribution of the boxes with $baseIoU = 0.5$ is highly bi-

Table 1. Effect of the batch properties for generated positive samples (see Fig. 5 for different RoI sources) on Pascal VOC 2007. We trained each RoI source with balanced foreground-foreground distribution and simulating OHPM. RS, Unif, LS and Base respectively denote pRoI-Right Skew, pRoI-Uniform, pRoI-Left Skew and pRoI-Base IoU=0.5 distributions. FGB refers to foreground balanced generation of RoIs.

RoI Distrib.	$FGB?$	OHPM	moLRP ↓	moLRP (IoU) ↓	moLRP (FP) ↓	moLRP (FN) ↓	mAP@0.5 ↑
RS	No	No	64.6	21.4	18.7	29.8	74.9
	Yes	No	64.5	21.5	18.7	29.5	75.3
	Yes	Yes	60.4	19.5	16.8	**27.2**	77.4
Unif.	No	No	61.3	19.5	17.9	28.5	76.3
	Yes	No	61.1	19.5	17.0	28.8	76.9
	Yes	Yes	**59.9**	19.2	**16.0**	27.6	**77.8**
LS	No	No	60.4	19.1	16.9	28.3	77.0
	Yes	No	60.3	**19.0**	17.3	28.2	77.2
	Yes	Yes	60.7	19.3	17.7	27.8	76.9
Base	No	No	61.5	19.7	17.2	28.8	76.6
	Yes	No	61.4	19.3	16.3	29.4	76.7
	Yes	Yes	61.2	19.7	16.6	28.6	76.7

ased towards 0.5 and includes very low samples with higher IoUs. This implies that the proportion of the boxes with $IoU > 0.9$ is far too low than that of the boxes with $0.6 > IoU > 0.5$ when $T = 0.5$. **(2)** RPN RoIs follow a similar tendency to the sampled boxes with $baseIoU = 0.5$ since the RoIs are based on anchors, which are uniformly distributed with a fixed set of boxes on the image. Thanks to the RPN regressor, the IoU distribution improves compared to the distribution of the sampled boxes with $baseIoU = 0.5$. On the other hand, this bias towards 0.5 is previously argued to make the regressor overfit for smaller IoUs [1]. **(3)** RPN is able to provide hard positive examples inherently; however, the number of prime samples (i.e. examples with larger IoUs) is quite low. This is critical since it is shown that prime sampling performs better than hard positive mining [2].

5.2. Foreground-Foreground Class Imbalance

We observe that, for each RoI source, addressing foreground-foreground imbalance (FGB=Yes) improves performance in terms of both mAP and moLRP, especially for the right skew and uniform cases (see Table 1). Moreover, addressing foreground-foreground class imbalance does not seem to affect the localization error ($\text{moLRP}_{\text{IoU}}$) but improves the classification performance since mAP@0.5, moLRP_{FP} and moLRP_{FN} get better (except for the left-skew case). Therefore, we conclude that foreground-foreground class imbalance can also be alleviated by employing methods in the batch level.

5.3. Effect of Online Hard Positive Mining

Here we demonstrate another useful use-case of our pRoI generator by simulating OHEM [31] on positive examples. OHEM chooses the positive and negative examples with the highest loss values after applying NMS to the examples to preserve example diversity. A recent study [27] showed that the IoU and the hardness of an example are correlated. On the other hand, another study [2] proposed an opposite perspective to the OHEM based on prioritizing "prime samples", i.e. samples with high IoUs. To be more clear, OHEM [31] implies preferring positive examples with IoUs just above 0.5, while prime sampling asserts that the higher the IoU, the better the example. To make an analysis on the positive examples, we simulate OHEM by (i) initially generating 128 BBs by pRoI generator, (ii) applying NMS using loss value of an example, (iii) finally selecting the ones with the larger loss values. We coin this as **online hard positive mining (OHPM)**.

In our experiments, we observe that the effect of the hard examples depends on the IoU distribution of the RoIs and high-quality samples are required during training: In Table 1, when OHPM is applied, uniform and right-skew distributions, which have more difficult examples due to their distribution (Fig. 5), have better performance compared to the left-skew and "Base IoU=0.5" cases. Moreover, while OHPM does not improve the performance of left-skew and "Base IoU=0.5" cases, it is crucial for the right-skew and uniform distributions (Table 1). Therefore, similar to prime sampling [2], we show that examples with higher IoUs are crucial during training, however, we also show that these examples should be supported by hard examples.

5.4. Relative Spatial Imbalance

We now analyze the relative spatial distribution of the RPN RoIs and how they fit within the theoretical IoU boundaries in Fig. 6. To be able to make such an analysis, we selected a reference box with $[x_1, y_1, x_2, y_2] = [0.3, 0.3, 0.6, 0.6]$. At the final epoch of the RPN training, we track positive RPN RoIs with associated ground truths. As discussed in Section 3.2, we scaled and shifted the ground truths to the reference box and applied the same transformations to their associated positive RPN RoIs. Among the positive RPN RoIs, top-left (TL) points of the $2,500$ RoIs are plotted with green dots in Fig. 6. Then, using $\text{findTLFeasibleSpace}()$ function in Algorithm 1, we plot the theoretical limits for the top left points for RoIs with IoUs larger than 0.5, 0.6, 0.7, 0.8 and 0.9.Especially the last two observations may be criti-

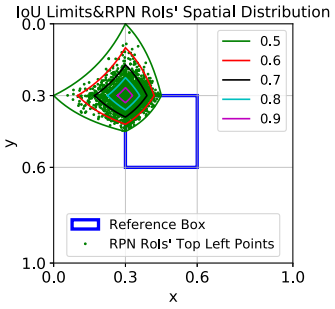


Figure 6. Relative spatial distribution of $2,500$ RPN RoIs TL points and max. IoU limits from $IoU = 0.9$ to 0.5 (in-out direction)

Table 2. Average performance of 3 runs for Faster R-CNN with our OFB sampling on Pascal VOC. Lower is better for moLRP and its components, whereas higher is better for mAP.

Sampling Method	moLRP	moLRP			mAP@0.5 ↑
	moLRP ↓	IoU ↓	FP ↓	FN ↓	
Random	59.4	18.7	16.2	27.7	78.0
OFB	**58.9**	18.7	**15.6**	**27.2**	**78.5**

Table 3. Comparison of different sampling mechanisms on MS COCO using Faster R-CNN. Lower is better for moLRP and its components, whereas higher is better for mAP. mAP stands for COCO-style mAP. R and H denote random and hard sampling respectively, and OFB is our sampling method for positive RoIs. The first block compares among different positive sampling schemes combined with random sampling, while the second block compares their combinations with hard example mining.

Sampling Method				
Positive	Negative	moLRP ↓	mAP ↑	mAP@0.5 ↑
R	R	72.4	34.1	55.2
H	R	75.3	31.0	51.7
OFB	R	**72.1**	**34.7**	**55.8**
R	H	71.9	35.3	54.6
H	H	74.6	31.1	50.0
OFB	H	**70.9**	**35.6**	**55.3**

cal for an object detector since they may result in a positive bias towards specific RoIs and may make the generalization difficult over the entire spatial space. However, the effects of all these observations require experimental or theoretical validation that is not provided in this paper.

Fig. 6 leads to several key findings: **(1)** As expected, as the IoU decreases, the boundaries occupy a larger space around the TL point of the reference box. Hence, the sample space for 0.9 is very small, which makes it more difficult to have distinct RoIs with $IoU > 0.9$. **(2)** We observe that no TL point is outside of the 0.5 boundary, which is a sanity check for the boundaries since a RoI is labeled as positive if it has at least 0.5 IoU with a ground truth. **(3)** The TL points of the RPN RoIs are accumulated around the TL point of the reference box and they are not uniformly distributed within the 0.5 boundary. **(4)** The TL points of the RPN RoIs tend to be inside the reference box more than to be outside. Specifically, RPN RoIs between $x > 0.3, y > 0.3$ and $x < 0.3, y < 0.3$ are 28.2% and 21.0% of the all, respectively.

6. Practical Improvements

In this section, we propose OFB sampling and show the effect of employing pRoI generator for training the second-stage of Faster R-CNN.

6.1. Online Foreground Balanced Sampling

In the conventional training, the set of positive RoIs are limited and they are not generated as in pRoI generator. Motivated from the analysis using pRoI generator on the effect of foreground-foreground class imbalance (see Section 5), we propose an online sampling method to be used in the conventional training pipeline. Denoting the total number of classes in a batch by C and the number of positive RoIs for class c by k_c, each RoI is assigned a probability $1/(Ck_c)$ and the subset of RoIs to train Faster R-CNN is sampled from this multinomial distribution. We call this sampling scheme as Online Foreground Balanced (OFB) Sampling.

In order to see the effect, we train Faster R-CNN with and without OFB sampling and present results in Tables 2 and 3. For the Pascal VOC [6], we observe 0.5% improvement in mAP@0.5 and moLRP, with better performance in precision and recall components of moLRP and no impact on the regression branch. In our experiments with MS COCO (Table 3), we compared our results with hard example mining [23, 31]. Similar to the findings of Cao et al. [2] and our analysis in Section 5, while hard positive mining does not improve performance, our OFB sampling is beneficial for foreground examples. Moreover, the table shows that OFB sampler can be combined with sampling approaches for negative BBs. In any case, similar to our experiments for Pascal VOC, the best performance gain is in mAP@0.5. This suggests that controlling RoIs to balance foreground classes has also a role during training of the object detectors and OFB, an efficient sampling algorithm, can be considered a basic solution for the problem.

6.2. Generating More Samples in Higher IoUs

Our approach can be integrated into an object detector without any hindrance on the gradient paths (see Suppl. Mat.). In this section, we compare a detector trained with our pRoI Generator with a detector trained with the conventional method (i.e. using RPN RoIs) – see Table 4. We use Uniform RoI source with foreground balance and OHPM since it performed the best in Table 1. For $IoU = \Theta$, we randomly sample negative samples from the output of the RPN in the range $[0.1, \Theta]$ and the positive samples are provided by the pRoI generator also using OHPM. To apply OHPM, we first generate $RoINum$ boxes, then select fg many from them. In IoUs $0.6 - 0.8$, for which fewer RoIs are possible than 0.5, we initially train the models for 1 epoch by setting $fg = 32$ and $bg = 96$ and track "Mean RoI #" to see an upper bound for the models to generate RoIs and prevent class imbalance modelwise. In this run, Mean RoI # for IoUs $0.6, 0.7, 0.8$ are $17.26, 7.60, 1.72$ for RPN and $20.0, 11.41, 4.67$ for pRoI-Uniform respectively. Then using $IoU = 0.5$ as an example, we multiply the resulting "Mean RoI #" by 1.5 and set fg approximately to it with $bg = 3 \times fg$ as in the conventional training. This approach makes training more stable and fair especially for the RPN (see Table 4) by balancing foreground and background consistently.

Looking at Table 4 and comparing the methods in the IoUs that they are trained for, we observe the following: **(1)** For $IoU = 0.5$, 0.6 and $IoU = 0.7$ we get comparable

Table 4. Performance Comparison with RPN on PASCAL VOC. $RoINum$ is the input of pRoI generator, fg/bg is the desired fg and bg RoI numbers during training, and Mean RoI # is the actual mean of number of positive RoIs. Note that fg/bg RoI numbers are set differently for pRoI and RPN so that the best performance is achieved for both of these RoI sources in order to provide a fair comparison especially in favor of RPN. We trained the models (except the one with the * mark) for 16 epochs with a learning rate decay at epochs 9 and 14 since our model provides more diverse data than RPN (see in Fig. 6 that the TL points of the RPN RoIs clusters around TL point of B) and there are fewer samples for training in higher IoUs (see Mean RoI # in Table 4)

RoI Source	IoU	$RoINum$	fg/bg	Mean RoI # ↑	moLRP ↓	$\text{moLRP}_{\text{IoU}}$ ↓	moLRP_{FP} ↓	moLRP_{FN} ↓	mAP@IoU ↑
RPN*	0.5	N/A	32/96	**27.12**	59.3	18.7	16.0	**27.7**	**78.0**
pRoI-Uniform	0.5	128	32/96	25.49	**59.2**	**18.4**	**15.5**	28.2	77.1
RPN	0.6	N/A	27/81	16.92	65.4	17.0	**19.4**	31.9	**71.2**
pRoI-Uniform	0.6	128	27/81	**18.28**	65.4	**16.9**	20.8	**31.0**	70.6
RPN	0.7	N/A	9/27	5.39	74.9	**14.7**	**27.2**	42.1	57.3
pRoI-Uniform	0.7	128	18/54	**9.93**	**74.5**	14.9	28.0	**39.8**	**57.5**
RPN	0.8	N/A	2/6	1.08	92.5	13.2	58.8	69.8	21.3
pRoI-Uniform	0.8	64	8/24	**3.92**	**87.7**	**12.1**	**47.8**	**59.3**	**32.2**
RPN	0.9	N/A	2/6	99.5	7.4	94.2	97.1	0.5	0.17
pRoI-Uniform	0.9	32	2/6	**1.62**	**99.3**	**7.3**	**92.4**	**96.0**	**0.9**

Table 5. Effect of $RoINum$ on PASCAL VOC. Speeds are reported on a single Geforce GTX 1080 Ti.

RoI Source	$RoINum$	moLRP ↓	$\text{moLRP}_{\text{IoU}}$ ↓	moLRP_{FP} ↓	moLRP_{FN} ↓	mAP@0.5 ↑	Train Speed ↓	Mean RoI # ↑
pRoI-Uniform	32	60.3	19.3	16.4	27.8	77.5	$0.41s$	14.81
pRoI-Uniform	64	59.7	19.0	16.1	27.4	77.6	$0.58s$	21.32
pRoI-Uniform	128	59.9	19.2	16.0	27.6	77.8	$0.97s$	25.49

results with the conventional training. **(2)** For $IoU = 0.8$, where RPN is not able to generate sufficient samples, the performance increases significantly in terms of both metrics since, at each iteration, generated positive boxes are provided consistently to the second stage. **(3)** Overall, the mean RoI # is approximately four times higher at $IoU = 0.8$; and, $\text{mAP@}0.8$ and moLRP improve by 10.9% and 4.8% respectively. A similar trend is also achieved for $IoU = 0.9$.

In short, these results demonstrate that it is possible to train an object detector using BB generator with comparable results for lower IoUs and significantly better performance for higher IoUs. On par performance for low IoUs can be owing to the fact that there are sufficient amount of samples for these cases to see any imbalance effect.

Effect of $RoINum$: Apart from the input parameters to determine the nature of the RoI source, $RoINum$ is the only new hyperparameter in Algorithm 2. In Table 5, we observe that training improves (mAP increases) when $RoINum$ is increased because we have more positive samples at each iteration. However, more samples mean slower (yet still acceptable) training speed compared to conventional training having $0.23s$ training speed.

Preliminary Results on MS COCO: In order to back up our claims, we also conducted an experiment on MS COCO dataset using $IoU = 0.8$ with Faster R-CNN. Compared to the baseline achieving $\text{moLRP} = 95.1$ and $\text{mAP@}0.8 = 13.2$, using pRoI generator the model has $\text{moLRP} = 93.7$ and $\text{mAP@}0.8 = 15.3$. These results suggest that our model is able to generate more diverse examples than the baseline in larger IoUs.

7. Conclusion

In this paper, we proposed a BB generator and a positive RoI generator. We showed that generated RoIs can be used both as an analysis tool (owing to its controllable nature) and a training method for the two-stage object detectors.

We showed that there is a bias in the RPN RoIs' IoU and spatial distribution with respect to the IoU boundaries that are physically possible and analyzed the IoU distributions of RPN and other RoI sources.

Using our BB generator, we developed a pRoI generator that can generate RoIs overlapping with a GT box with a desired IoU or relative spatial distribution. Then, we trained Faster R-CNN's second-stage with the RoIs generated according to different distributions. We showed that, by producing more samples than RPN, we can achieve better or comparable performance to Faster R-CNN. Moreover, our results reconciliated two conflicting recent studies [2, 31] that both high-IoU and hard RoIs can have positive effect on the training if the IoU distribution is appropriate.

Our ideas can be used for analyzing the anchors of a one-stage detector (as well as those of a two-stage detector) in order to design a better anchor set. Furthermore, other applications, e.g. tracking, that require spatially distributed BBs with certain properties can also exploit our approach.

Acknowledgments

This work was partially supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) through a project titled "Object Detection in Videos with Deep Neural Networks" (grant number 117E054). Kemal Öksüz is supported by the TÜBİTAK 2211-A National Scholarship Programme for Ph.D. students. The numerical calculations reported in this paper were performed at TUBITAK ULAKBIM High Performance and Grid Computing Center (TRUBA), and Roketsan Missiles Inc. sources.

References

[1] Z. Cai and N. Vasconcelos. Cascade R-CNN: Delving into high quality object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[2] Y. Cao, K. Chen, C. C. Loy, and D. Lin. Prime Sample Attention in Object Detection. *arXiv*, 1904.04821, 2019.

[3] A. T. Cemgil. *A Tutorial Introduction to Monte Carlo methods, Markov Chain Monte Carlo and Particle Filtering*. Academic Press Library in Signal Processing, 2013.

[4] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu, Z. Zhang, D. Cheng, C. Zhu, T. Cheng, Q. Zhao, B. Li, X. Lu, R. Zhu, Y. Wu, J. Dai, J. Wang, J. Shi, W. Ouyang, C. C. Loy, and D. Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv*, 1906.07155, 2019.

[5] J. Dai, Y. Li, K. He, and J. Sun. R-FCN: Object detection via region-based fully convolutional networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[6] 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*, 88(2):303–338, 2010.

[7] A. Fernndez, S. G. M. Galar, R. Prati, and B. K. F. Herrera. *Learning from Imbalanced Data Sets*. Springer, 2018.

[8] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg. DSSD: Deconvolutional single shot detector. *arXiv*, 1701.06659, 2017.

[9] G. Ghiasi, T. Lin, R. Pang, and Q. V. Le. NAS-FPN: learning scalable feature pyramid architecture for object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[10] R. Girshick. Fast R-CNN. In *The IEEE International Conference on Computer Vision (ICCV)*, 2015.

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[14] J. M. Johnson and T. M. Khoshgoftaar. Survey on deep learning with class imbalance. *Journal of Big Data*, 6(27), 2019.

[15] T. Kong, F. Sun, W. Huang, and H. Liu. Deep feature pyramid reconfiguration for object detection. In *The European Conference on Computer Vision (ECCV)*, 2018.

[16] A. Kuznetsova, H. Rom, N. Alldrin, J. R. R. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, T. Duerig, and V. Ferrari. The open images dataset V4: unified image classification, object detection, and visual relationship detection at scale. *arXiv*, 1811.00982, 2018.

[17] B. Li, Y. Liu, and X. Wang. Gradient harmonized single-stage detector. In *AAAI Conference on Artificial Intelligence*, 2019.

[18] S. Li, L. Zhouche, and H. Qingming. Relay Backpropagation for Effective Learning of Deep Convolutional Neural Networks. In *European Conference on Computer Vision (ECCV)*, 2016.

[19] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature pyramid networks for object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[20] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

[21] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision (ECCV)*, 2014.

[22] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia. Path aggregation network for instance segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[23] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg. SSD: single shot multibox detector. In *European Conference on Computer Vision (ECCV)*, 2016.

[24] K. Oksuz, B. C. Cam, E. Akbas, and S. Kalkan. Localization recall precision (LRP): A new performance metric for object detection. In *European Conference on Computer Vision (ECCV)*, 2018.

[25] K. Oksuz, B. C. Cam, S. Kalkan, and E. Akbas. Imbalance Problems in Object Detection: A Review. *arXiv*, 1909.00169, 2019.

[26] W. Ouyang, X. Wang, C. Zhang, and X. Yang. Factors in finetuning deep model for object detection with long-tail distribution. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[27] J. Pang, K. Chen, J. Shi, H. Feng, W. Ouyang, and D. Lin. Libra R-CNN: Towards balanced learning for object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[28] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[29] J. Redmon and A. Farhadi. YOLO9000: Better, faster, stronger. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[30] 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*, 39(6):1137–1149, 2017.

[31] A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[32] X. Wang, A. Shrivastava, and A. Gupta. A-fast-rcnn: Hard positive generation via adversary for object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[33] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. *arXiv*, 1611.05431, 2016.

[34] J. Yang, J. Lu, D. Batra, and D. Parikh. A faster pytorch implementation of faster r-cnn. *https://github.com/jwyang/faster-rcnn.pytorch*, Last Accessed: 24 April 2019.