applied sciences

MDPI

Article

BBRefinement: A Universal Scheme to Improve the Precision of Box Object Detectors

Petr Hurtik [1,*], Marek Vajgl [1] and David Hynar [2]

[1] Centre of Excellence IT4Innovations, Institute for Research and Applications of Fuzzy Modeling, 30. Dubna 22, University of Ostrava, 702 00 Ostrava, Czech Republic; marek.vajgl@osu.cz
[2] Varroc Lighting Systems, Suvorovova 195, 742 42 Šenov u Nového Jičína, Czech Republic; dhynar@varroclighting.com
* Correspondence: petr.hurtik@osu.cz; Tel.: +420-553-46-1414

Abstract: We present a conceptually simple yet powerful and general scheme for refining the predictions of bounding boxes produced by an arbitrary object detector. Our approach was trained separately on single objects extracted from ground truth labels. For inference, it can be coupled with an arbitrary object detector to improve its precision. The method, called BBRefinement, uses a mixture of data consisting of the image crop of an object and the object's class and center. Because BBRefinement works in a restricted domain, it does not have to be concerned with multiscale detection, recognition of the object's class, computing confidence, or multiple detections. Thus, the training is much more effective. It results in the ability to improve the performance of SOTA architectures by up to two mAP points on the COCO dataset in the benchmark. The refinement process is fast; it adds 50–80 ms overhead to a standard detector using RTX2080; therefore, it can run in real time on standard hardware. Finally, we show that BBRefinement can also be applied to COCO's ground truth labels to create new, more precise labels. The link to the source code is provided in the contribution.

Keywords: object detection; bounding box refinement; data curation

Citation: Hurtik, P.; Vajgl, M.; Hynar, D. BBRefinement: A Universal Scheme to Improve the Precision of Box Object Detectors. *Appl. Sci.* **2022**, *12*, 3402. https://doi.org/10.3390/app12073402

Academic Editors: Antonio Fernández-Caballero, Byung-Gyu Kim and Hugo Pedro Proença

Received: 23 February 2022
Accepted: 25 March 2022
Published: 27 March 2022

1. Introduction

1.1. Problem Statement

Object detection plays an essential role in computer vision, which has attracted a strong emphasis among researchers, resulting in the continuous development of new and more accurate object detectors. A typical object detector takes an image and produces a set of rectangles, so-called bounding boxes (BB), which define the borders of objects in the input image. The detector quality is measured as an overlap between the detected boxes and ground truth (GT) boxes using intersect over union (IoU). The more descriptive statistic, mean average precision (mAP), uses a set of IoUs with various thresholds, where a threshold distinguishes between the acceptance/rejection of detected boxes. Existing solutions for object detection yield accuracies around 30–55 mAP on the COCO dataset [1]. Such a score allows usage in many real applications, but, on the other hand, there is space for improvement. A combination of the following may achieve such growth: more precise classification, an increase in the rate of true-positive detection, a decrease in false-positive detection, or an improvement in the IoU. There are four reasons why object detection may be difficult. A neural network has to find all objects in an image; the number of objects may vary from zero to hundreds. A neural network has to be sensitive to all possible sizes of an object; the same object class may be tiny or occupy the whole image. A network usually has no a priori information, which should make the detection easier, such as the context of the scene or the number of objects. There is a lack of satisfactory large datasets; therefore, the data distribution is sampled roughly. In this study, we propose BBRefinement, a specialized, one-purpose object detector. It is trained on objects that are cropped from the input images together with augmented information about the object's class and the dimensions of the

bounding box. The purpose is to refine an imprecise bounding box dimension into a precise form. During the training stage, it is trained fully standalone. During the inference stage, it is coupled with an arbitrary object detector and its prediction is refined. The proposed scheme is able to suppress the effect of all four mentioned difficulties, resulting in a higher mAP.

Our contributions to the problem are as follows:

- We propose BBRefinement, a specialized, one-purpose object detector that processes mixture data and can work as an extension to any object detector to improve its accuracy.
- We designed a specialized augmentation technique that combines the augmentation of information about the object's class and bounding box dimensions.
- We define two losses, the first aiming at metrics that involve a single IoU threshold, and the second minimizing metrics that involve multiple IoU thresholds.

The advantages of our proposed method are the following:

- BBRefinement is universal. It can be trained only once and then coupled with an arbitrary object detector that produces the same object classes.
- The proposed scheme is capable of suppressing known difficulties in the object detection area, thus resulting in a higher mAP, mainly for large and medium-sized objects.
- It is fast, and adds only 50 ms overhead per predicted image.
- It can be used to improve the quality of imprecise labels.

1.2. Related Work

The scheme of refinement can be tracked to the origin of two-stage detectors, where R-CNN [2] uses a region proposal algorithm that generates a fixed number of regions. The regions are classified and they are refined by bounding box regressor head. Faster R-CNN [3] replaces the region proposal algorithm with a region proposal network. The same bounding box regressor head can be used iteratively to obtain a more precise detection [4,5]. The effect of iterative refinement may be increased by involving the LSTM module [6]. The aim of refinement can also be anchors; RefineDet [7,8] refines them to obtain customized anchors for each cell. Currently, the top refinement scheme, Cascade R-CNN [9] uses a sequence of bounding box regressor heads to create an n-staged object detector. In Cascade R-CNN, head h_0 takes proposals from the region proposal network and feeds the regressed bounding boxes to network head h_1 and so on. All heads work on the same features extracted from a backbone network. The cascade scheme shows that h_1 depends on the quality of the predictions produced by the h_0 head. If h_0 includes some bias, h_1 balances it. Therefore, all heads have to be trained together (part by part), and if h_0 is retrained, h_1 should be retrained as well.

The difference between BBRefinement and the existing refinement schemes is that BBRefinement is trained on the original data standalone, so that it is not integrated into any existing architecture during the training stage; it is independent. That makes BBRefinement universal and able to be applied to various existing image detectors without retraining a detector or BBRefinement. A two-stage object detector like Faster R-CNN refines areas from the first stage using the features extracted from the backbone that are shared across the two stages. Thus, the two stages are directly connected and trained together. The second stage of a particular model cannot be applied to the first stage of a different model. By contrast, BBRefinement is trained standalone, and the input is not the extracted features but the images themselves. This makes it universal and capable of being linked to an arbitrary object detector (even a two-stage detector) to improve its ability (Figure 1).

The novelty lies in such standalone training with the usage of mixture data and the combination with existing models during inference. To our knowledge, it is original and has not been described in the literature so far. According to the results presented later, BBRefinement leads to universal usage and a nice boost of mAP without significantly decreasing the processing speed.

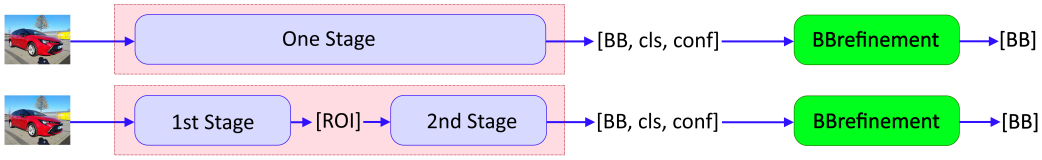

Figure 1. A scheme of coupling BBRefinement with one-stage (top row) and two-stage (bottom row) object detectors.

2. Explaining BBRefinement

The main feature of BBRefinement is a transformation of the problem into a simpler domain where an NN can be trained more efficiently. Compared to a standard object detector, BBRefinement does not search for zero-to-hundred objects because it always detects only a single object and does not produce its confidence. It also does not include the part responsible for classification, so the it does not assess the object's class. The only purpose is to take an image crop of a single object within a normalized scale and generate a more precise bounding box. The training is performed with boxes extracted from a training dataset according to the ground truth labels. The trained model can be coupled with an arbitrary detector to realize the inference. Here, the feeds for BBRefinement are images (crops) and categorical data produced by the linked detector.

2.1. Problem with a Naive Single Object Detector

A neural network is trained to minimize a loss function between its output and the ground truth. If such a network can detect a single BB only, but the input image includes several BBs, then the loss function is minimized when the network produces a BB, that is, the mean of them. Thus, with no guarantee of a single object's presence only, the trivial solution cannot be used. This problem is addressed later by a sliding deformable models/window technique [10], two-stage techniques such as (Fast/Faster) R-CNN [3], single-stage techniques with anchors such as SSD [11] or YOLO [12], anchor-free techniques that are mainly keypoint-based [13], and finally by involving Transformer into the architecture and realizing a bipartite loss [14]. Each such approach affects the architecture of a neural network and is related to a specific model.

To solve the problem of multiple objects in an image, BBRefinement takes as input an image with the information about the object, which should be refined, namely, its class and the coordinates of its center. Note that there may be more objects in a crop because such a bounding box for a non-rectangular object will also involve some background containing other objects. To illustrate the situation, we show crops from the COCO dataset. In Figure 2, we show an easy case where only one object is presented, and in Figure 3 we show a hard case where more overlapping objects are presented. In particular, the COCO dataset includes 1.7M boxes, in which 47% of all boxes have an intersection with a box of the same class, and 84% of the boxes have an intersection with an arbitrary class box.

Figure 2. The figure shows crops that can be refined even with the naive way because the crop includes only one, nicely visible object.

Figure 3. The figure shows crops that cannot be refined by the naive way because a crop includes multiple objects, usually of the same class. Note that precise labeling of such images is a hard task, even for humans.

2.2. The Principle of BBRefinement

Firstly, let us suppose a convolutional neural network $\mathbb{F}$ which is a set of k layers, $\mathbb{F} = \{f_1, f_2, \ldots, f_k\}$. Such a neural network is generally called a backbone and maps an input image iteratively into a feature space. Here, we suppose $e_k : D \subset \mathbb{R}^{n_k} \rightarrow L \subset \mathbb{R}$ to be an embedding of the k-th layer created as $e_k(f_k) = p(f_k)$, where p is a global average pooling or flattening operation. Furthermore, we suppose a fully connected network $\mathbb{G}$ to be a set of j layers, $\mathbb{G} = \{g_1, g_2, \ldots, g_j\}$. According to the motivation, we propose to use mixture data as input to the suggested refinement scheme. The convolution neural network $\mathbb{F}$ processes the input image (crop with fixed resolution) containing an object, and a fully connected neural network $\mathbb{G}$ processes a fixed-size vector that contains information about the class and the center of the object. Both networks are designed to $|e_k(f_k)| = |g_j|$ to be valid, where $|\cdot|$ denotes cardinality. Then, both pieces of information are mixed as $x = e_k(f_k) \cdot g_j$, where $\cdot$ is a dot product (see Figure 4). Such an operation can also be viewed as a kind of attention mechanism [15].

Finally, we connect x to the output layer o consisting of four neurons (to produce (x, y, w, h)) and utilize the sigmoid activation function. The constructed neural network is trained in a fully end-to-end supervised scheme. From a practical point of view, we can use an arbitrary SOTA backbone such as ResNeSt, ResNeXt, or EfficientNet, to mention a few. For BBRefinement, we use EfficientNet [16] because of its easy scalability. In the benchmark section, we present the results for versions B0–B4. According to the version, the input image's resolution is 224^2, 240^2, 260^2, 300^2, and 380^2. The version affects $|e_k(f_k)|$ as well, and it is 1280 (B0 and B1), 1408, 1536, and 1792.

There are several options for defining the loss function ℓ used to train BBRefinement. The first option is to compare each normalized coordinate of the box with the GT label using, e.g., binary cross-entropy (BCE). The second option is to use BCE to compare top-left points and then the Euclidean distance to evaluate the width and height of a box. This approach is used, e.g., in YOLO [17]. The third way is to use the coordinates of all points to determine the boxes' areas and compute IoU, which we also use in BBRefinement. We have two available options for defining the IoU loss function, namely, $\ell_1(\mathbf{b}, \mathbf{b}') = -log(i(\mathbf{b}, \mathbf{b}')/u(\mathbf{b}, \mathbf{b}'))$ for the logarithmized form and $\ell_2(\mathbf{b}, \mathbf{b}') = 1.0 - i(\mathbf{b}, \mathbf{b}')/u(\mathbf{b}, \mathbf{b}')$ for the linear form, where i represents the intersection of two boxes, u their union, $\mathbf{b}$ is the GT box, and $\mathbf{b}'$ predicted box. The logarithmized form is suitable for tasks where the mAP is measured with respect to a single low IoU threshold, such as 0.5, because it pushes bad predictions over this threshold and does not give much attention to good predictions. The linear form improves all predictions equally, which is beneficial for tasks where the mAP is computed for several IoU thresholds, such as $0.5, \ldots, 0.95$, which is the case of the COCO evaluation script; thus, we use the linear form. Note that both forms of the loss function can be based on a more efficient version of the IoU, such as Generalized IoU loss [18], Complete IoU, or Distance IoU [19].

The pipeline for the prediction with BBRefinement is illustrated in Figure 5 and is as follows. A generic object detector takes an input image and detects boxes. Their coordinates are used to take crops from the original input image. The crops are then rescaled to the BBRefinement input resolution. That process has several beneficial consequences. Firstly, larger objects are downscaled, and smaller ones are upscaled to fit the resolution, so that

all objects have the same scale, which is much more effective than training a network for multiscale detection. Second, one image from the dataset yields multiple boxes. In the case of COCO, a standard detector uses 0.2M images (one image as an input), while BBRefinement uses 1.7M images (one box as an input). In addition, a standard detector downscales the input images to a specific resolution to fit GPU memory, so that many pixels are thrown out. BBRefinement does not use non-object parts from the image, but it allows us to use more pixels from the object due to a weaker downsample. Third, due to mixture data usage, BBRefinement obtains information about the object's detected class and center. Although such data may be imprecise, it is a piece of prior information that makes the task more accessible.

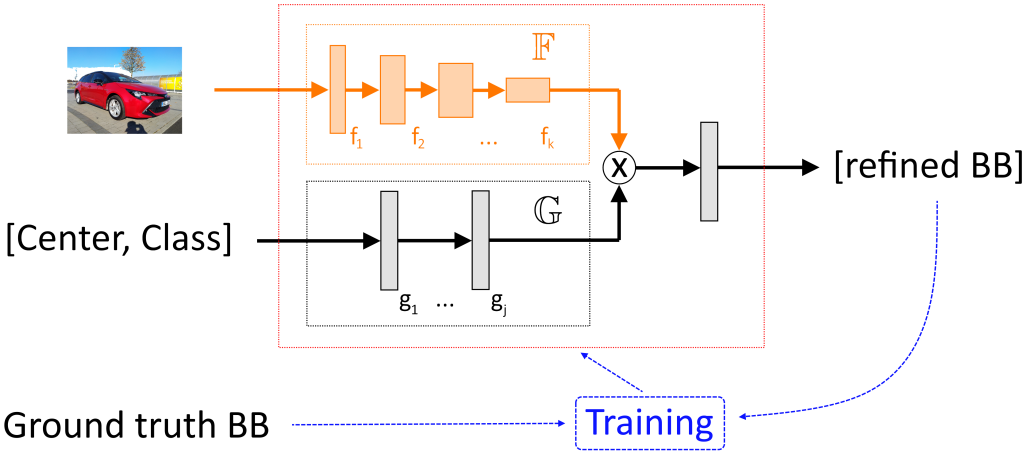

Figure 4. Scheme of the BBRefinement architecture. Orange blocks represent convolutional layers, gray blocks represent fully connected layers. $f_1, ..., f_k$ represent the layers of the sub-network $\mathbb{F}$, $g_1, ..., g_j$ represent layers of the sub-network $\mathbb{G}$.

It is necessary to take into account that BBRefinement is placed on top of an object detector, which may be imprecise. As a result, the data fed into BBRefinement may be ambiguous. Therefore, the crops taken during the inference should not be extracted precisely, but should be surrounded by padding. The same process is used during training, where, in addition, we also distort the center by random shifts. Such augmentation is visualized in Figure 6.

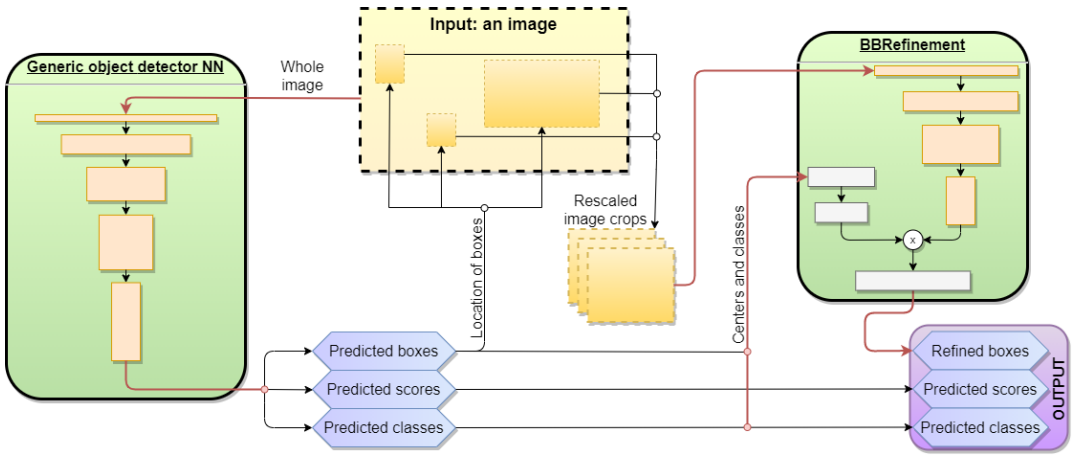

Figure 5. The figure illustrates the proposed prediction pipeline. A generic object detector processes an image, and then the detected boxes are taken from the original image, updated by BBRefinement, and taken as the output predictions. BBRefinement consists of a standard convolutional backbone (e.g., EfficientNet) marked by the orange color and fully connected layers marked by the white color. 'x' represents the dot-product operation. A detailed description of BBRefinement is given in Section 2.2.

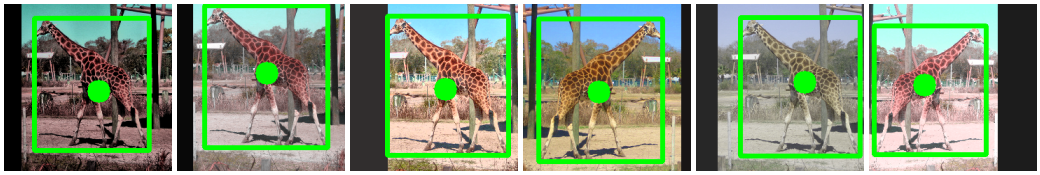

Figure 6. Augmentations of a box. The green box represents the GT label. The original crop is randomly padded. The center's position is slightly distorted (and visualized as a green dot) as we suppose that BBRefinement will be applied to a generic detector's predictions, which can produce such distortion.

3. Benchmark

The training setting: BBRefinement was trained using an RTX2080 graphics card with 11GB VRAM. The resolution of the models corresponds to the default setting of the EfficientNet [16] version, namely, the side size of 224, 240, 260, 300, and 380px for version B0-B4. The batch size was 7–40 according to the version and memory of the graphics card. From the COCO dataset [1], we took train2017 as a training set and val2017 as a testing set. The loss function is the linear IoU, the optimizer is AdaDelta [20] with default learning rate, i.e., $\alpha = 1.0$, and we used the functionality of decreasing the learning rate by a factor 0.5 with patience equal to three. We also experimented with the cyclic LR [21], which converged faster but did not reach the best possible loss. During one time period, all training images were processed, and a single random box was taken from each one of them. Each such box was augmented by random padding, linear/non-linear HSV distortion, CLAHE [22], and flipping. The information about the box center was augmented by distorting the coordinates. The illustration of the augmented box is shown in Figure 6. Models were trained until the loss did not stop decreasing, which took approx 70–90 time periods. For illustration, the heaviest backbone, EfficientNet B4, was trained for nine days.

We selected the most frequently used SOTA networks for the benchmark; see the complete list in Table 1. The models derived from Faster R-CNN [3], Mask R-CNN [23], RetinaNet [24], and Cascade R-CNN [9] are from the Detectron2 framework (https://github.com/facebookresearch/detectron2 accessed on 17 November 2020). For DETR [14], we used the official minimalistic implementation (https://github.com/facebookresearch/detr accessed on 11 November 2020). Models derived from YOLOv3 [17] and SSD [11] are taken from the MMDetection framework (https://github.com/open-mmlab/mmdetection accessed on 15 December 2021). For these networks, we used their reference models trained on COCO train2017 and measured the impact of the BBRefinement.

The detailed results are presented in Table 1. We want to emphasize that BBRefinement improves the mAP of all but Cascade R-CNN models, considering the standard (IoU = 0.50:0.95) setting, while the heavier backbone of BBRefinement usually provides a stronger boost. This is not valid for EfficientNetB4, where we suppose the performance drop is caused by training with a batch size that is too small. There is a hypothesis that training BBRefinement on a more powerful graphics card with a larger VRAM can increase performance; for verification, see Table 2 in [25] or the study in [26]. Furthermore, it roughly holds that the worse the baseline model, the larger the increase in mAP. Considering the objects' size according to the COCO tools (small, medium, or big), the larger the object, the larger the boost that is obtained. There is a hypothesis that strong upscaling of small objects leads to distortion and, therefore, to decreased performance. Thus, searching for a customized backbone, e.g., with switchable atrous convolutions [27], is a reasonable direction for future development. It also shows that datasets consisting of small objects only are a contraindication for BBRefinement. On the other hand, having the dataset with mostly medium and large objects, the involving of BBRefinement will lead to significant improvement.

Table 1. mAP (IoU = 0.50:0.95) performance of original and refined predictions on the COCO dataset. The table shows accuracy in the form of IoU of a generic detector when its official, pre-trained model is used or marked as a baseline. The right part shows IoU accuracy when the same pre-trained model is coupled with BBRefinement. All BBRefinement versions are trained only once, and the same trained version is used for all multiple detectors. For the training of BBRefinement, we use the same split as is common and as has also been used by the authors of generic detectors.

			BBRefinement, EfficientNet					
Model	**Source**	**Baseline**	**B0**	**B1**	**B2**	**B3**	**B4**	**Boost**
All objects								
Faster R-CNN, ResNet-50 C4 1x	Detectron2	35.7	37.3	37.4	37.7	**37.8**	37.7	+2.1
Faster R-CNN ResNet-101 C4 3x	Detectron2	41.1	41.6	41.6	42.0	**42.1**	42.0	+1.0
Faster R-CNN, ResNeXt-101 FPN 3x	Detectron2	43.0	43.0	43.1	43.4	**43.5**	43.4	+0.5
RetinaNet, ResNet-50 FPN 1x	Detectron2	37.4	38.2	38.3	**38.6**	**38.6**	**38.6**	+1.2
RetinaNet, ResNet-101 FPN 3x	Detectron2	40.4	40.6	40.6	**41.0**	**41.0**	40.9	+0.6
Cascade Mask R-CNN, ResNet-50 FPN 1x	Detectron2	**42.1**	41.1	41.1	41.5	41.5	41.5	−0.6
Mask R-CNN ResNet-50 FPN 1x	Detectron2	38.6	39.6	39.7	40.0	**40.2**	40.0	+1.2
Mask R-CNN ResNeXt-101 FPN 3x	Detectron2	44.3	44.0	44.1	44.4	**44.5**	44.4	+0.2
DETR, ResNet-50	Standalone	34.3	35.7	35.8	36.0	**36.1**	35.9	+1.8
YOLOv3	MMDetection	33.5	34.3	34.4	34.6	**34.7**	34.5	+1.2
SSD 512	MMDetection	29.4	31.2	31.3	31.5	**31.6**	31.4	+2.2
Small objects								
Faster R-CNN, ResNet-50 C4 1x	Detectron2	19.2	19.2	**19.3**	19.1	19.1	18.9	+0.1
Faster R-CNN ResNet-101 C4 3x	Detectron2	22.2	22.1	22.0	22.2	**22.3**	22.0	+0.1
Faster R-CNN, ResNeXt-101 FPN 3x	Detectron2	**27.2**	25.9	25.8	25.7	25.8	25.6	−1.3
RetinaNet, ResNet-50 FPN 1x	Detectron2	**24.0**	22.0	22.0	22.1	22.1	21.8	−1.9
RetinaNet, ResNet-101 FPN 3x	Detectron2	**24.0**	23.4	23.2	23.3	23.6	23.2	−0.4
Cascade Mask R-CNN, ResNet-50 FPN 1x	Detectron2	**24.3**	22.6	22.4	22.5	22.6	22.5	−1.7
Mask R-CNN ResNet-50 FPN 1x	Detectron2	**22.5**	21.9	21.7	21.8	22.0	21.7	−0.5
Mask R-CNN ResNeXt-101 FPN 3x	Detectron2	**27.5**	26.4	26.1	26.2	26.3	26.2	−1.1
DETR, ResNet-50	Standalone	14.3	**16.0**	15.9	15.9	15.9	15.7	+1.7
YOLOv3	MMDetection	19.6	19.7	19.6	19.7	**20.0**	19.5	+0.4
SSD 512	MMDetection	11.7	**12.8**	12.5	12.7	12.7	12.5	+1.1
Medium objects								
Faster R-CNN, ResNet-50 C4 1x	Detectron2	40.9	42.3	42.6	**42.8**	**42.8**	**42.8**	+1.9
Faster R-CNN ResNet-101 C4 3x	Detectron22	45.5	46.2	46.3	**46.6**	**46.6**	**46.6**	+1.1
Faster R-CNN, ResNeXt-101 FPN 3x	Detectron2	46.1	46.5	46.7	**47.0**	**47.0**	46.9	+0.9
RetinaNet, ResNet-50 FPN 1x	Detectron2	41.6	42.8	42.9	**43.2**	**43.2**	43.1	+1.6
RetinaNet, ResNet-101 FPN 3x	Detectron2	44.3	44.8	44.9	**45.3**	45.2	45.0	+1.0
Cascade Mask R-CNN, ResNet-50 FPN 1x	Detectron2	**45.2**	44.5	44.6	45.0	45.1	44.9	−0.1
Mask R-CNN ResNet-50 FPN 1x	Detectron2	42.0	43.3	43.4	**43.8**	**43.8**	43.7	+1.8
Mask R-CNN ResNeXt-101 FPN 3x	Detectron2	47.6	47.9	48.0	**48.4**	48.3	48.3	+0.8
DETR, ResNet-50	Standalone	36.6	38.5	38.6	38.8	**39.0**	38.8	+2.4
YOLOv3	MMDetection	36.4	38.9	39.0	39.3	**39.4**	39.3	+3.0
SSD 512	MMDetection	34.1	37.3	37.4	**37.7**	**37.7**	37.5	+3.6
Large objects								
Faster R-CNN, ResNet-50 C4 1x	Detectron2	48.7	52.4	52.8	53.1	53.1	**53.2**	+4.5
Faster R-CNN ResNet-101 C4 3x	Detectron2	55.9	57.2	57.3	58.0	58.0	**58.1**	+2.2
Faster R-CNN, ResNeXt-101 FPN 3x	Detectron2	54.9	56.3	56.7	57.0	**57.2**	**57.2**	+2.3
RetinaNet, ResNet-50 FPN 1x	Detectron2	48.3	50.3	50.7	**51.5**	51.1	51.2	+3.2
RetinaNet, ResNet-101 FPN 3x	Detectron2	52.2	53.6	53.7	54.2	54.1	**54.4**	+2.2
Cascade Mask R-CNN, ResNet-50 FPN 1x	Detectron2	54.8	54.8	55.0	**55.5**	55.4	55.4	+0.7
Mask R-CNN ResNet-50 FPN 1x	Detectron2	49.9	52.8	53.1	53.4	**53.7**	53.6	+3.8
Mask R-CNN ResNeXt-101 FPN 3x	Detectron2	56.7	57.7	58.4	**58.8**	**58.8**	58.5	+2.1
DETR, ResNet-50	Standalone	51.5	52.1	52.3	**52.7**	**52.7**	52.6	+1.2
YOLOv3	MMDetection	43.6	44.1	44.3	44.7	**44.9**	44.7	+1.3
SSD 512	MMDetection	44.9	47.0	47.2	47.7	**47.8**	47.4	+2.9

4. Discussion

Misleading labels: The object detection task's general issues are incorrect classes, imprecise coordinates of boxes, and missing boxes. BBRefinement, as well as standard object detectors, is vulnerable to the first two issues, but it is resistant to the third issue. If we consider missing labels as illustrated in Figure 7, we will penalize a standard detector during training if the detector produces predictions for such missing labels. This will lead to decreased performance. In the case of BBRefinement, if some label is missing, a cropped image is not produced. Therefore, the missing labels only decrease the training set's size and do not affect BBRefinement's performance.

Refinement of a dataset: We realized an experiment in which we took GT test labels, refined them, and visualized both in an image. Surprisingly, we can claim that BBRefinement can produce more precise labels than the original GT COCO dataset. Figure 8 shows crops from the test set with inpainted boxes: green color marks the GT boxes given by the COCO dataset, and yellow color marks the labels produced by BBRefinement. We selected the images in Figure 8 as such cases, where it was evident that BBRefinement produced more precise boxes. Note that the IoU between the predictions and GT varied here around 0.8. Because the dataset was large, and eight illustrated crops were chosen selectively, we also selected six additional crops as follows. The first one had an index of 1000 in the ordered list of images, the second one 1200, the third one 1300, etc., so that the selection was not affected by our preference. They are illustrated in Figure 9. We can proudly claim that BBRefinement, although not so significant as for the previous cases, still produces in most cases more precise boxes than GT (best seen zoomed-in). In addition, we applied BBRefinement trained on COCO to the Cityscapes dataset. Again, BBRefinement made visually more precise labels than the original Cityscapes labels. This finding leads to three conclusions. First, it is ambiguous to compare high-mAP object detectors because high mAP does not necessarily mark a better detector in the meaning of real-world truth, as the labels are affected by human subjectivity and error. Next, thanks to the high number of boxes, BBRefinement can be trained in such a generalized manner that the labeling error can vanish, so that it can be used for re-labeling a dataset. Finally, there is a hypothesis that IoU between BBRefinement trained on a specific dataset and its GT labels can be used to express the quality of labels. Verification of this hypothesis is a theme for future work.

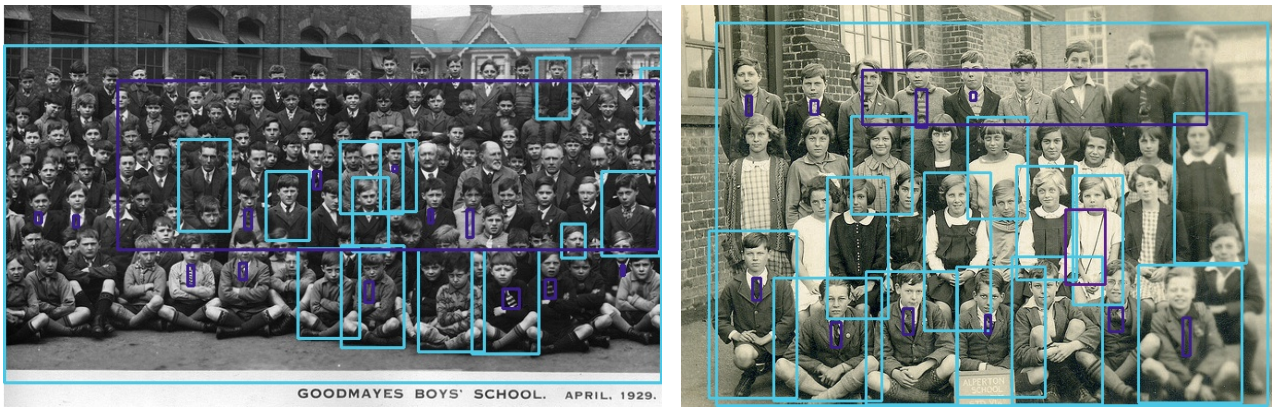

Figure 7. The figure illustrates two images taken from the COCO dataset, where the boxes are inpainted ground truth labels. It is evident that some labels are imprecise and many of the labels are missing. This behavior can be seen mainly in images that include groups, and it is a known issue of the COCO dataset.

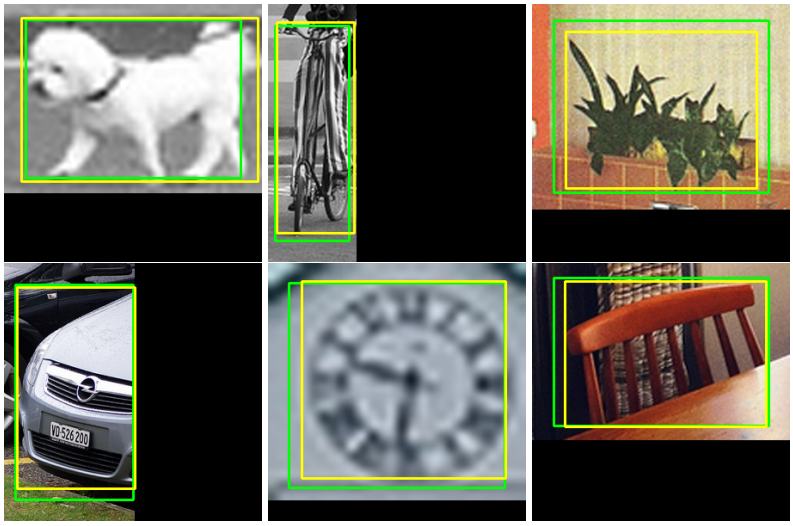

Figure 8. The image illustrated crops with green ground truth and yellow refined inpainted labels. Here, BBRefinement creates labels with significantly higher precision than the ground truth. Best seen zoomed-in.

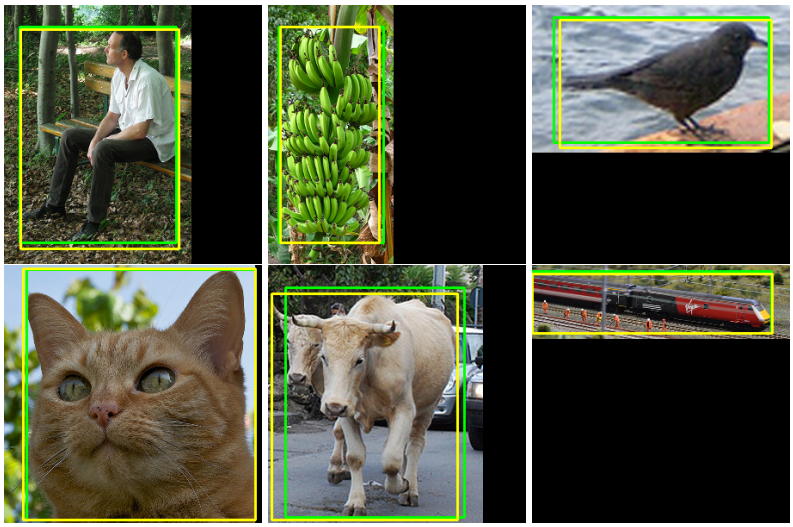

Figure 9. The image illustrated crops with green ground truth and yellow refined inpainted labels. The crops were selected uniformly according to their index to show general cases. Here, BBRefinement provides slightly higher precision than ground truth. Best seen zoomed-in.

5. Ablation Study

Comparing with a naive refinement without mixture data: We trained BBRefinement with EfficientNetB1 with the same setting as was used in Section 3, but without mixture data. This means that only visual information represented by image crops was available during training and inference. During the inference, we coupled it with 'Faster R-CNN, ResNet-50 C4 1x'. It achieved mAP performance on all/small/medium/large areas of 36.6/19.3/40.8/50.4, which is better than the baseline performance of 35.7/19.2/40.9/48.7 but worse than the full couple with the mixture data, which yielded 37.4/19.3/42.6/52.8. This finding confirms the meaningfulness of the proposed scheme.

Influence of the accuracy of center and class: The performance of BBRefinement is affected by the accuracy of the object detector used. Therefore, we realized an experiment in which GT data were distorted and fed into BBRefinement, and measured IoU between the refined and GT boxes. As we demonstrate in Figure 10, we distorted the position of the center and the correct class separately. The distortion of center $\mathbf{c}$ was realized as $\mathbf{c} = (c_x + d_x, c_y + d_y)$, where $d_x, d_y \sim \mathcal{U}(-d, d)$, and by d we mean the maximum distortion.

For class distortion, we replaced n% of correct classes with random incorrect classes. BBRefinement is robust for incorrect classes, but sensitive to center position distortion.

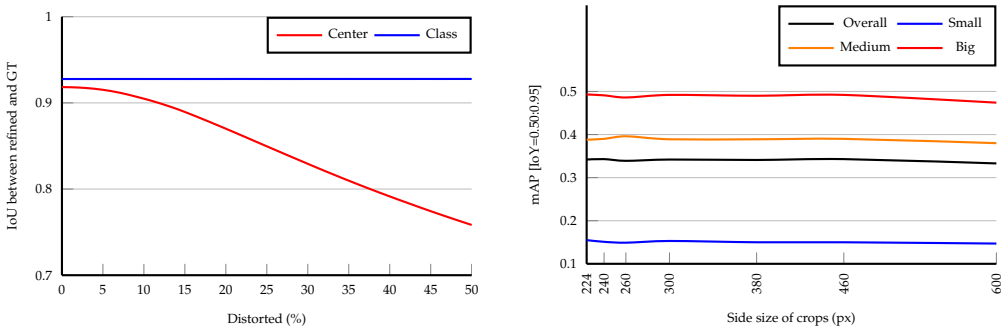

Figure 10. Ablation study. **Left**: influence of distortion center coordinates or incorrect class. **Right**: influence of crop size.

Influence of crop size: We selected BBRefinement with the EfficientNetB2 backbone and trained it for various crop sizes to reveal the impact. To converge faster, we weakened the setting compared to the 'full experiment', namely, 3000 steps per time period, batch size 8, and patience 1 in reducing the learning rate. During the inference, it was coupled with 'Faster R-CNN, ResNet-50 C4 1x'. According to the graph in Figure 10, the model was stable and the crop size had a minor impact for all sizes except 600, where there was a decrease in performance. For the extreme case of crop size 600, almost all objects, including the large ones, were upsampled, which distorted them.

Speed of inference: BBRefinement's inference time consists of two parts, the preparation of the crops and the inference itself. While the first part depends on the CPU, the second part relies only on the GPU power. We measured the processing time of BBRefinement with EfficientNetB1 and EfficientNetB3 backbones. For both cases, the non-optimized preparation of crops on CPU cost 32 ms per whole image, where the image could include multiple crops. The time on GPU was 23 ms for B1 and 44 ms for B3. BBRefinement predicted all boxes of a single image in one batch, which kept the time small. The aggregation of both CPU+GPU times means that BBRefinement ran 18FPS for the B1 backbone and 13FPS for the B3 backbone. The processing speed could be further increased by parallelizing the crops' preparation and optimizing the model, e.g., by fusing a batch normalization layer with a convolution layer into a single layer.

6. Concluding Remarks

We discussed the difficulties of the object detection problem, and showed that the difficulties could be suppressed by a refinement stage which we proposed to be coupled with an already trained standard object detector during the inference. We proposed the refinement to be a single object detector that works over crops extracted from an input image according to the information produced by a standard object detector. To solve the problem when one extracted crop includes several objects, we proposed using mixture data where the image information was complemented with information about the object's class and center, which helped the network to refine the desired object.

In the benchmark section, we demonstrated that BBRefinement is capable of increasing the mAP of 10 SOTA networks out of the 11 tested. The added overhead was 76 ms for BBRefinementB3 and for processing a single input image with multiple crops. Finally, we showed that our scheme, BBRefinement, is able to produce predictions that are in most cases more precise than the ground truth labels of a dataset.

As the refinement process is partially independent of the detector, this approach opens a new direction of research. Original research which is focused on increasing accuracy by proposing new architectures, etc., can now be complemented with independent research on refinement networks. The final system, which can be deployed to real production in

various competitions (such as Kaggle or Signate), may consist of a combination of the best algorithms from both types of research. Future work remains to be completed to search for a BBRefinement backbone with switchable atrous convolutions or to further improve processing speed.

We used the scheme of refinement in Signate's competition on object detection and tracking (https://signate.jp/competitions/256/leaderboard, accessed on 20 December 2021), where our solution based on Poly-YOLO [28] + BBRefinement ended as a runner-up—notwithstanding the fact that the international rivals used more powerful architectures such as RetinaNet or CenterTrack. In this competition, BBRefinement improved Poly-YOLO's mAP from 57.8 to 59.8 and proved its usefulness.

The code of BBRefinement is available at https://gitlab.com/irafm-ai/bb-refinement, accessed on 2 March 2022.

Author Contributions: Conceptualization, P.H., M.V. and D.H.; data curation, D.H.; formal analysis, P.H., M.V. and D.H.; methodology, P.H. and D.H.; software, M.V. and P.H.; validation, P.H., M.V. and D.H.; visualization, M.V.; writing, P.H. and M.V. All authors have read and agreed to the published version of the manuscript.

Funding: The work is supported by ERDF/ESF "Centre for the development of Artificial Intelligence Methods for the Automotive Industry of the region" (No. CZ.02.1.01/0.0/0.0/17_049/0008414).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The source codes for implementing the presented approach are freely available at: https://gitlab.com/irafm-ai/bb-refinement, accessed on 2 March 2022.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

BB	Bounding Box
FPS	Frames per Second
GT	Ground Truth
IoU	Intersection over Union
mAP	Mean Average Precision
SOTA	State Of The Art

References

1. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 740–755.
2. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
3. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; pp. 91–99.
4. Gidaris, S.; Komodakis, N. Object detection via a multi-region and semantic segmentation-aware cnn model. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1134–1142.
5. Li, J.; Liang, X.; Li, J.; Wei, Y.; Xu, T.; Feng, J.; Yan, S. Multistage object detection with group recursive learning. *IEEE Trans. Multimed.* **2017**, *20*, 1645–1655. [CrossRef]
6. Gong, J.; Zhao, Z.; Li, N. Improving Multi-stage Object Detection via Iterative Proposal Refinement. In Proceedings of the BMVC, Cardiff, UK, 9–12 September 2019; p. 223.
7. Zhang, S.; Wen, L.; Bian, X.; Lei, Z.; Li, S.Z. Single-shot refinement neural network for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4203–4212.
8. Zhang, S.; Wen, L.; Lei, Z.; Li, S.Z. RefineDet++: Single-Shot Refinement Neural Network for Object Detection. *IEEE Trans. Circuits Syst. Video Technol.* **2020**, *31*, 674–687. [CrossRef]

9. Cai, Z.; Vasconcelos, N. Cascade r-cnn: Delving into high quality object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 6154–6162.
10. Felzenszwalb, P.F.; Girshick, R.B.; McAllester, D.; Ramanan, D. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.* **2009**, *32*, 1627–1645. [CrossRef] [PubMed]
11. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single shot multibox detector. In *European Conference on Computer Vision*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 21–37.
12. Redmon, J.; Farhadi, A. YOLO9000: Better, faster, stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 7263–7271.
13. Law, H.; Deng, J. Cornernet: Detecting objects as paired keypoints. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 734–750.
14. Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; Zagoruyko, S. End-to-End Object Detection with Transformers. *arXiv* **2020**, arXiv:2005.12872.
15. Hu, J.; Shen, L.; Sun, G. Squeeze-and-excitation networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 7132–7141.
16. Tan, M.; Le, Q.V. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv* **2019**, arXiv:1905.11946.
17. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
18. Rezatofighi, H.; Tsoi, N.; Gwak, J.; Sadeghian, A.; Reid, I.; Savarese, S. Generalized intersection over union: A metric and a loss for bounding box regression. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 658–666.
19. Zheng, Z.; Wang, P.; Liu, W.; Li, J.; Ye, R.; Ren, D. Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression. In Proceedings of the The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20), New York, New York, USA, 7–12 February 2020; pp. 12993–13000.
20. Zeiler, M.D. Adadelta: An adaptive learning rate method. *arXiv* **2012**, arXiv:1212.5701.
21. Smith, L.N. Cyclical learning rates for training neural networks. In Proceedings of the 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), Santa Rosa, CA, USA, 24–31 March 2017; pp. 464–472.
22. Pizer, S.M.; Johnston, R.E.; Ericksen, J.P.; Yankaskas, B.C.; Muller, K.E. Contrast-limited adaptive histogram equalization: Speed and effectiveness. In *Proceedings of the First Conference on Visualization in Biomedical Computing*; IEEE Computer Society: Washington, DC, USA, 1990; pp. 337–338.
23. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2961–2969.
24. Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2980–2988.
25. Brock, A.; De, S.; Smith, S.L. Characterizing signal propagation to close the performance gap in unnormalized ResNets. *arXiv* **2021**, arXiv:2101.08692.
26. Radiuk, P.M. Impact of training set batch size on the performance of convolutional neural networks for diverse datasets. *Inf. Technol. Manag. Sci.* **2017**, *20*, 20–24. [CrossRef]
27. Qiao, S.; Chen, L.C.; Yuille, A. DetectoRS: Detecting Objects with Recursive Feature Pyramid and Switchable Atrous Convolution. *arXiv* **2020**, arXiv:2006.02334.
28. Hurtik, P.; Molek, V.; Hula, J.; Vajgl, M.; Vlasanek, P.; Nejezchleba, T. Poly-YOLO: Higher speed, more precise detection and instance segmentation for YOLOv3. *arXiv* **2020**, arXiv:2005.13243.