Generating Proposals from Corners in RPN to Detect Bees in Dense Scenes

Yassine Kriouile[1,2][a], Corinne Ancourt[1][b], Katarzyna Wegrzyn-Wolska[1,2][c] and Lamine Bougueroua[2][d]

[1]*Mines ParisTech, PSL University, Centre de Recherche en Informatique, 35 rue Saint Honoré, 77300 Fontainbleau, France*

[2]*EFREI Paris, AliansTIC, 30/32 Avenue de la République, 94800 Villejuif, France*

Keywords: Bees, Object Detection, Faster RCNN, RPN, High Object Density, Corners.

Abstract: Detecting bees in beekeeping is an important task to help beekeepers in their work, such as counting bees, and monitoring their health status. Deep learning techniques could be used to perform this automatic detection. For instance Faster RCNN is a neural network for object detection that is suitable for this kind of tasks. But its accuracy is degraded when it comes to images of bee frames due to the high density of objects. In this paper, we propose to extend the RPN sub-neural network of Faster RCNN to improve detection recall. In addition to detect bees from centers, four branches are added to detect bees from their corners. We constructed a dataset of images and annotated it. We compared this approach to the standard Faster RCNN. It improves the detection accuracy. Code is available at *https://github.com/yassine-kr/RPNCorner*.

1 INTRODUCTION

In recent years, bees suffer from many problems like varroa infestation (Sipos et al., 2021). This problem could cause a negative impact on the health of bees. To overcome this kind of issues, beekeepers should monitor bees and their conditions. Computer vision methods based on deep learning could help beekeepers in bee detection. In fact, there are already advanced neural networks for object detection like Faster RCNN. This network is detecting objects in an image through two stages. First RPN generates proposals then they are classified and enhanced using another sub-network. RPN takes a set of feature maps calculated by a backbone neural network. For each position in the map, it predicts a predefined number of possible objects. To achieve this prediction, a set of bounding boxes called anchors are generated at each feature map position. RPN predicts whether each anchor matches an object or not and gives the offset coordinates to correct the anchor position. The anchors are centered on the position. An anchor corresponds to an object when its IoU (intersection over union) with a ground truth is high. This means that anchors generated at the centers of the objects are more likely classified as objects, while anchors generated at the corners of the objects are more likely classified as non-objects. In dense scenes, there are objects that are partially visible; only the corners are visible. The standard approach could miss these kinds of objects. To overcome this issue, we propose to enhance anchor generation by generating anchors from object corners in addition to object centers. Concretely, four additional types of anchors are used; each one corresponds to an object corner (top left, top right, bottom left, bottom right). At each feature map position, the corner anchor is generated so that the feature map point is located at the anchor corner. We duplicate the RPN prediction components and losses to handle the corner data. To test our approach, we constructed and annotated a specific dataset composed by images of bee frames. To analyze our approach, we made different modifications on our proposed approach and tested them on two databases of different density levels. In this paper, our contribution consists on:

- An extension of standard Faster RCNN architecture to predict objects from corners and improve recall in dense scenes.

[a] https://orcid.org/0000-0002-1139-2789
[b] https://orcid.org/0000-0002-3310-7651
[c] https://orcid.org/0000-0002-9776-3842
[d] https://orcid.org/0000-0002-5322-8231

Kriouile, Y., Ancourt, C., Wegrzyn-Wolska, K. and Bougueroua, L.
Generating Proposals from Corners in RPN to Detect Bees in Dense Scenes.
DOI: 10.5220/0010815000003124
In *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2022) - Volume 5: VISAPP*, pages 339-350
ISBN: 978-989-758-555-5; ISSN: 2184-4321

- A dataset of bee frames annotated with bounding boxes corresponding to bees.

2 RELATED WORK

Our work aims to improve bee detection using deep learning. In this section, we cite some works that are related to our subject. First, we focus on existing methods dealing with bee detection in Section 2.1, then we present the state of the art of neural networks for object detection in Sections 2.2 and 2.3. Section 2.4 presents the dense scene issue, and the existing approach dealing with this situation. Anchors are also an interesting topic as our method is based on the generation of a new set of anchors, related work about anchors is described in Section 2.5. Finally we cite some already proposed approaches based on corner data to detect objects in Section 2.6.

2.1 Bee Detection

There are works treating the problem of detecting bees in images. (Magnier et al., 2018) propose an approach to detect and track bees from videos with white background. This detection is based on background subtraction, ellipse approximation, border detection and color threshold, but its FPS (frame per second) is low. In (Magnier et al., 2019), they suggest a method to track bees by estimating their trajectories through polynomial and interpolation.

(Kulyukin and Mukherjee, 2018) estimate the bee traffic by detecting movements and classifying images based on machine learning methods as CNN, SVM and Random Forest. They published a public database containing annotated images and videos for machine learning.

(Tiwari, 2018) tries to recognize bees and track them from videos using CNN. The recording is managed by BeePi which is a specific hardware dedicated to beehives and makes it possible to collect other kinds of data like sound and temperature.

(Tu et al., 2016) aim to track bee behavior and evaluate the condition of the hive. This is achieved by counting bees at the beehive entrance and estimating their in-out activity using linear regression. Their method does not handle complicated cases such as high bee density and bee occlusion.

(Kulyukin and Reka, 2016) track the traffic of foraging bees using sound and images. After recording videos thanks to BeePi, tracking is performed by pixel separation algorithm and contour detection of binary image.

In (Dembski J., 2020), the authors try to detect bees on video images using three steps: determine the regions of interest ROI for each frame using motion detection, then classify each region whether it contains a bee or not thanks to a convolutional deep neural network, and finally group regions using clustering algorithm.

2.2 Two Stage Object Detection

Two stage object detection is an approach of detecting objects using two phases ; generate proposals then classify them. R-CNN is based on a vision algorithm generating proposals which are provided to a convolutional network which performs the classification (Girshick et al., 2014). Fast-RCNN improves object detection by using a ROI pooling layer responsible of extracting features from shared feature maps instead of calculating feature map for each proposal (Girshick, 2015). Faster-RCNN uses the RPN neural network as proposal generator (Ren et al., 2015). This architecture allows to learn predicting object bounding boxes. R-FCN is a two stage object detection neural network built only by convolutional layers, the principle is to use many feature maps, each of which contains information about different object regions (Dai et al., 2016).

2.3 One Stage Object Detection

One stage object detectors are faster and less resource greedy than two stage detectors. SSD and YOLO are the state of art neural networks which detect objects without generating proposals (Liu et al., 2016), (Redmon et al., 2016). The drawback of these approaches is the imbalance between positive and negative examples during training, this leads to limited performance. To overcome this issue, RetinaNet was proposed, their authors suggest to use Focal loss which increases the loss of misclassified object compared to well classified objects (Lin et al., 2017b).

2.4 Dense Scene

In the field of object detection, a dense scene means a scene where the density of objects is very high, in other words it means that the number of objects per unit area is large[1]. In this case, there is a high probability that the image contains two objects that are adjacent or occluding each other. Performing accurate detection using machine learning for computer vision in these conditions could be challenging. There are many works dealing with these issues

[1] https://en.wikipedia.org/wiki/Number_density

in the context of detecting people and pedestrians in crowded scenes, such as (Liu et al., 2019) where authors present a method which predicts whether two overlapping bounding boxes match the same object or two different objects. In (Zhang et al., 2018), the authors try to improve the detection in pedestrian crowded scenes by proposing another way of calculating the loss which aims to improve the compactness of the predicted bounding boxes around the ground-truth, and to improve the prediction of the human body by detecting its parts. The paper (Xi et al., 2020) deals with the detection of human faces in crowded scenes with low resolution by exploring the similarity between detected objects. Finally (Zhang et al., 2019) propose to use two anchors to detect people in a crowd; one for the head and one for the body.

Concerning object detection in a general context; the paper (Gählert et al., 2020) attempts to improve object detection in images through using pixel-base bounding box annotations. (Goldman et al., 2019) detect objects in a dense scene using a new layer of prediction called "Soft-IoU" which predicts IoU of a predicted object bounding box compared to the real object position.

2.5 Anchors

In the state of the art of object detection neural networks, anchors are used to detect objects. They are bounding boxes of different sizes and ratios generated by the neural network. It generates a predefined number of anchors at each position of the feature map. For each of them, a score is predicted, it corresponds to the probability that the anchor contains an object. The learning phase is based on the comparison of anchors with ground-truth bounding boxes. Each anchor is centered on its corresponding feature map position, this matching enables the neural network to predict objects from their centers.

On the other side, there are anchor-free detectors such as (Tian et al., 2019) which do not use anchors. In this approach, for each feature map point, object bounding box coordinates are directly predicted. This approach allows to predict objects from all their pixels.

2.6 Corner Detection

The paper (Qiu et al., 2020) presents an approach called BorderDet which aims to improve the detection of objects in dense scenes by using feature maps of borders in addition to standard feature maps of centers. This way allows not to miss partially hidden objects. But the limit of this method is that it needs a first coarse prediction to extract border points.

(Zhou et al., 2019) suggest to detect the extreme points and the central point of an object. The final detection is obtained by grouping these points. The proposed approach is based on the detection of key points using heat maps and (Law and Deng, 2018). This method needs extreme points annotations.

(Wei et al., 2020) use point-set anchor instead of a rectangular box anchor, this permits to represent objects more accurately. This approach is mainly interesting in the case of image segmentation and pose estimation, where ground-truth annotations are not standard bounding boxes.

(Duan et al., 2020) propose an anchor-free and two stage object detector, based on corner proposals. These proposals are generated by predicting two kinds of corner heat maps; top-left and bottom-right. Corners are extracted from these maps and a fixed number of proposals are then generated.

3 MOTIVATION

In standard Faster RCNN, RPN generates proposals from object centers. In fact, it takes feature maps as input, and generates anchors for each point in each map. An anchor is placed in a way that the point is located at the anchor center. The network predicts for each anchor, objectness (a score corresponding to the probability of the anchor bounding box to be an object) and offsets to correct the anchor position and match the real object. The training phase compares each anchor to ground truth bounding boxes, using IoU metric (Intersection over Union). The anchors with big IoU are considered as objects (ground-truth objectness is equal to 1). The anchors generated at corner regions are likely to be classified as non-objects as their IoUs with ground truth bounding boxes are small. If the image contains objects whose only visible parts are corners, they are more likely to be missed by the network. In the case of bee images, this situation is frequent due to the high density of bees in the bee frame image. The RPN has a direct influence on the recall of Faster-RCNN; if an object is missed during proposal generation, it is almost impossible to detect it by Faster RCNN. We think that this issue can be resolved by generating another kind of anchors, using another way of placing them on feature map points. Each one of this new type of anchors is placed in a way that the point is located at a corner anchor. This way would enable RPN training to consider anchors generated at object corner regions as objects (ground truth objectness is equal to 1) and avoid missing partially hidden bees. Our work is mo-

tivated by these elements:

- In standard anchor-based neural networks for object detection, prediction is based on object center information, corner information is not taken into account.
- Improving anchor generation could be used in all anchor-based neural networks like YOLO.
- Generating anchors influences on detection recall. It is important to generate anchors able to take into account the type of visible region (center or corner).
- Bee detection is an interesting task for beekeeping domain. Improve accuracy of this detection remains an important objective to reach.

4 APPROACH

Faster-RCNN is a two stage neural network. It uses RPN as a proposal generator. This component generates anchors centered on the feature map positions. This way of placing anchors favors objects whose central regions are visible. We propose to generate another kind of anchors in order to predict objects from corners and improve detection recall.

4.1 General Architecture

The proposed neural network is based on Faster RCNN. Figure 1 shows the structure of the network. Our work focuses on RPN, the purpose of this sub-network is to generate region proposals which are then classified and enhanced using another subnetwork. The first step computes feature maps using a pre-trained classification neural network like VGG or Resnet. Then objectness and offsets are predicted. After that the predicted offsets are re-used by anchor generation mechanism to create proposals which are ordered using objectness scores. In the standard approach, proposals are generated from object centers. Our approach consists on generating proposals from object corners. To achieve this objective, we modified the anchor generation mechanism and extended the neural network to predict corner data. Section 4.2 explains the types of generated anchors. Section 4.3 presents the added corner predictors. Section 4.4 describes how generated anchors and prediction data are combined to create proposals. Section 4.5 clarifies the used method for labeling anchors during training. The last section (4.6) cites the used losses for learning.

4.2 Anchor Generation

The aim of the standard anchor generation is to detect objects from their centers. In the case of dense scenes, where usually objects are partially occluded, the central region of the objects could be hidden, therefore some objects could be missed. Anchors are a set of bounding box candidates for predicted objects. They are generated from feature maps. In the case of FPN (Lin et al., 2017a), more than one feature maps are used to detect objects of different sizes. We extend the anchor generation mechanism by adding four anchor generators: top left anchors, top right anchors, bottom left anchors and bottom right anchors. For each generator, K anchors (in our case $K = 3$), of different sizes and ratios, are generated at each position for each feature map. Let s be the dimension of a feature map stride (the image region corresponding to the feature map position), w the anchor width, h the anchor height, and $x1$, $x2$, $y1$ and $y2$ anchor coordinates in the format $X1X2Y1Y2$ ($X1$: x-coordinate of left border, $X2$: x-coordinate of right border, $Y1$: y-coordinate of top border, $Y2$: y-coordinate of bottom border), in the coordinate system whose origin is the center of the stride:

- In the case of central anchor:
  $x1 = -w/2, x2 = w/2, y1 = -h/2, y2 = h/2$
- In the case of top left anchor:
  $x1 = -s/2, x2 = w–s/2, y1 = -s/2, y2 = h–s/2$
- In the case of top right anchor:
  $x1 = -w+s/2, x2 = s/2, y1 = -s/2, y2 = h–s/2$
- In the case of bottom left anchor:
  $x1 = -s/2, x2 = w–s/2, y1 = -h+s/2, y2 = s/2$
- In the case of bottom right anchor:
  $x1 = -w+s/2, x2 = s/2, y1 = -h+s/2, y2 = s/2$

Figure 2 illustrates the five types of generated anchors.

4.3 Objectness and Offset Prediction

Instead of using one layer for predicting objectness and offsets, four other parallel layers are added to predict corner data. In total there are five branches with different weights; (1) the first to generate proposals from object centers, (2) the second to generate proposals from object top left corners, (3) the third to generate proposals from object top right corners, (4) the fourth to generate proposals from object bottom left corners, (5) the fifth to generate proposals from object bottom right corners. Each branch is made up of two layers; the first one takes as input the feature

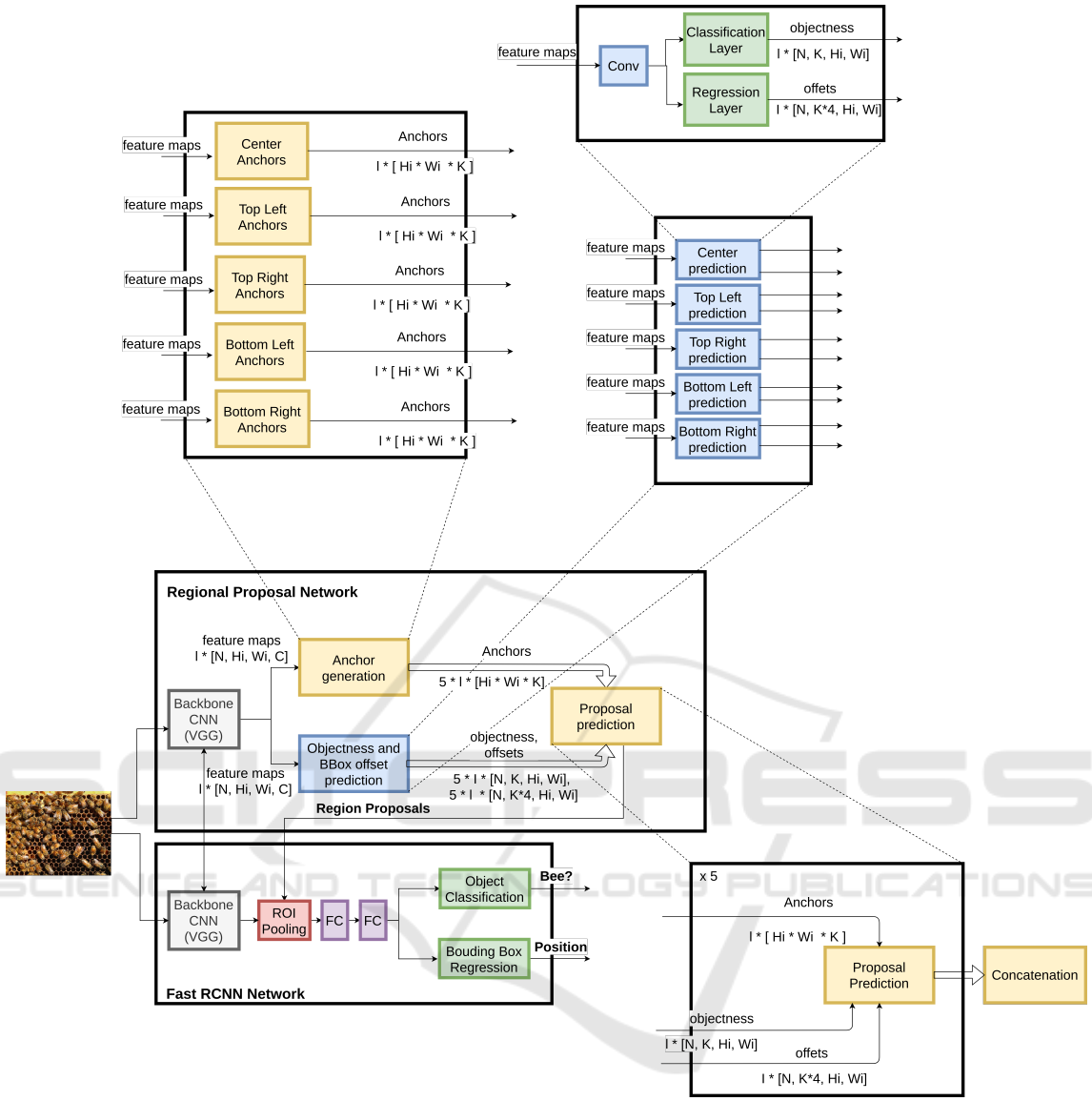


Figure 1: General Architecture of Faster-RCNN based on corner detection. l: number of feature maps, N: number of images, Hi: height of feature map i, Wi: width of feature map i, C: number of channels, K: number of anchors per feature map position.

maps generated by the backbone, and applies a convolution operation. The second layer contains two sub-branches, the first is a convolutional layer to predict objectness at each feature map position, the second is a convolutional layer to predict box offset coordinates to correct the positions of the proposal.

4.4 Generating Proposals

Proposal generation is performed by combining the prediction data and the generated anchors. We duplicate this step four times to generate proposals from corner data. It consists in applying predicted offsets on generated anchors, then a predefined number of best proposals is preserved from each feature map, the NMS (non-maximum suppression) algorithm is applied on each set of feature map proposals to reduce multidetection, and finally a predefined number of best proposals is preserved from all proposals. A proposal is better than the other when its objectness score is higher.

The final set of proposals is simply a union of proposals generated by the five generators.

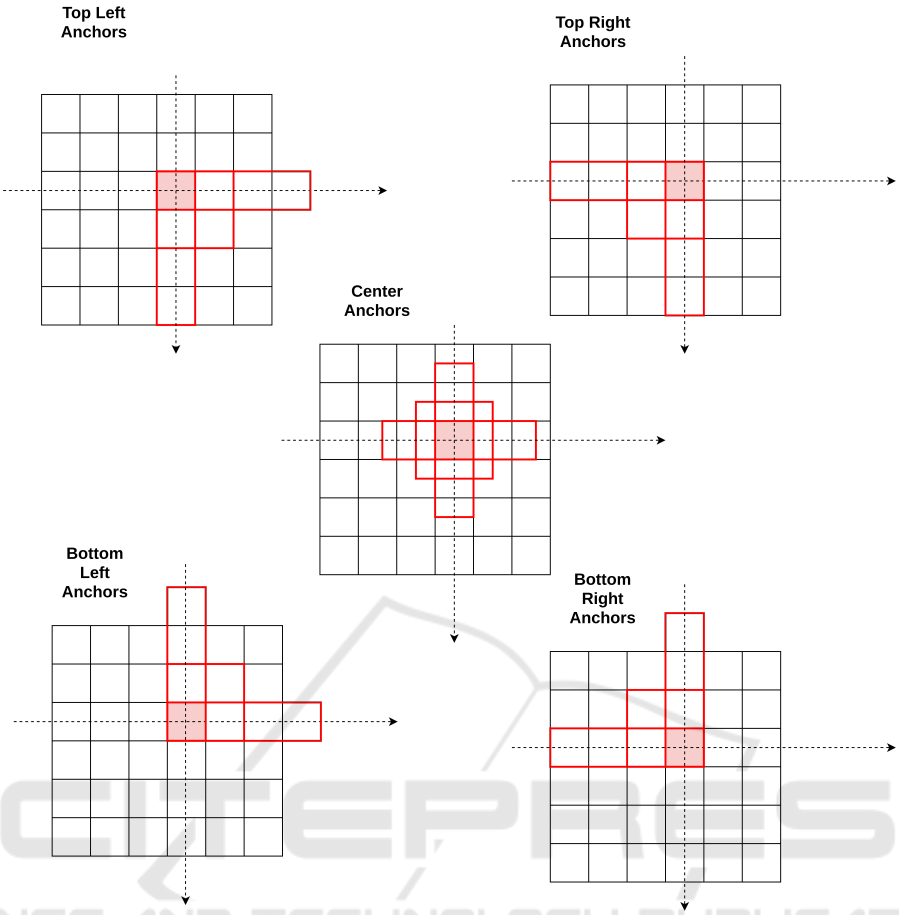


Figure 2: Types of generated anchors from feature maps.

4.5 Anchor Ground Truth Labeling and Offset Assigning

The anchor labeling consists on labeling each anchor with 0 or 1 which corresponds to whether the anchor contains an object or not. Offset assigning is the process of assigning to each anchor the offsets that must be applied to the anchor to reach the correct position. The aim of training is, for each anchor, to predict the objectness and offsets. Our approach uses standard labeling and assigning; for each anchor, the ground truth bounding box with which the anchor has the greatest overlapping score (IoU: Intersection over Union) is mapped to the anchor, if this IoU is greater than a fixed threshold (0.7), the anchor is labeled with 1, if it is less than a fixed threshold (0.3), the anchor is labeled with 0, otherwise it is ignored. The differences between the anchor borders and the borders of the mapped bounding box are the assigned ground truth offsets.

4.6 RPN Losses

As demonstrated in figure 3, there are two losses in RPN; objectness loss and offset loss. To balance between positive and negative samples, a sampling is performed. The training consists in optimizing the sum of these two losses:

- Objectness loss:

$$L = -\frac{1}{N_{\text{cls}}} \sum_{i=1}^{N_{\text{cls}}} t_i log(p_i) + (1 - t_i) log(1 - p_i) \quad (1)$$

- Offset loss:

$$L = \frac{1}{N_{\text{reg}}} \sum_{i=1}^{N_{\text{reg}}} R(o_i - o_i*) \quad (2)$$

Where R, N_{cls}, N_{reg}, p_i, t_i o_i, o_i* are respectively the smooth L1 loss (Girshick, 2015), the number of anchors to classify, the number of anchors to regress, the predicted objectness, the ground truth label, the predicted offsets, and the ground truth offsets.

For training on different layers, we add similar losses for the predicted corner information. The final loss is an equally weighted sum of the five losses.

5 DATASET

In Section 5.1, we describe the dataset sources from which we retrieve images for training and testing our approach. In Section 5.2 we present tools used for image annotation. Our code was developed using a specific Python library for object detection, and an efficient hardware. Section 5.3 explains these software and hardware features. We chose standard metrics to evaluate our approach, they are cited in Section 5.4.

5.1 Data Source

Our main objective is to improve bee detection using an object detection neural network. Therefore, our image database is only made up of images of bee frames.

The images used for training and testing our approach come from three sources: (1) images downloaded from the Internet, (2) photos taken using camera and phone in an apiary, and (3) images provided by our project partners.

The main preprocessing that we perform on images is cropping to make annotation task easier. In fact there were images with many bees, the complete annotation of such images takes a lot of time and effort.

5.2 Image Annotation

We used two tools to annotate our images. The first is Imagetagger which is a web application for annotating images with bounding boxes [2]. This tool enabled us to work remotely in collaboration with our partners. We installed it and configured it to make it accessible through the Internet. Our project partners used it to help us in annotation task. The second tool is the Matlab application Matlab Image Labler [3]. Through this tool, we annotated some images of our database with bounding boxes, then exported the annotations to matlab workspace. A matlab code was developed to export annotations from Matlab format to Json format.

[2]https://github.com/bit-bots/imagetagger

[3]https://fr.mathworks.com/help/vision/ref/imagelabeler-app.html

5.3 Framework and Environment

Our work is based on the detectron framework, its source code is hosted on this Github repository [4]. The pre-built version used in our work is located on this link [5]. Detectron is a Facebook framework based on Pytorch library, and contains implementations of state of art neural networks for object detection and image segmentation, such as Faster RCNN, Mask RCNN and RetinaNet. We have chosen this framework among others because it contains official implementations of Faster RCNN which is appropriate to our case because of its good performance despite of its slowness. The extendibility and modularity of the framework enabled us to create and integrate our custom subcomponents like anchor generators and RPN predictors.

The training and tests were executed on a machine with these characteristics:

- 32 Go of RAM
- 4 cores, 8 CPUs
- 16 Go Nvidia GPU: it was used for training and testing.

5.4 Metrics

To assess the accuracy of our approach, we used conventional object detection metrics: average recall and average accuracy. Since our approach consists of modifying RPN, we tested the recall of RPN in addition to the accuracy of the whole neural network.

6 EXPERIMENTAL RESULTS

In Section 6.1, we describe the created datasets for training and testing. Then, in Section 6.2 we explain the library parameters that we set for our use case. In Section 6.3, we discuss and compare the accuracy results that we obtained after training and testing the standard approach and our approach.

6.1 Train and Test Data

We created three image sets; (1) the first set was annotated by our partners, it contains arbitrary images coming from our dataset sources. They were not completely annotated; highly occluded bees were generally not annotated, because of the effort required

[4]https://github.com/facebookresearch/detectron2

[5]https://dl.fbaipublicfiles.com/detectron2/wheels/cu110/torch1.7/detectron2-0.3%2Bcu110-cp38-cp38-linux_x86_64.whl

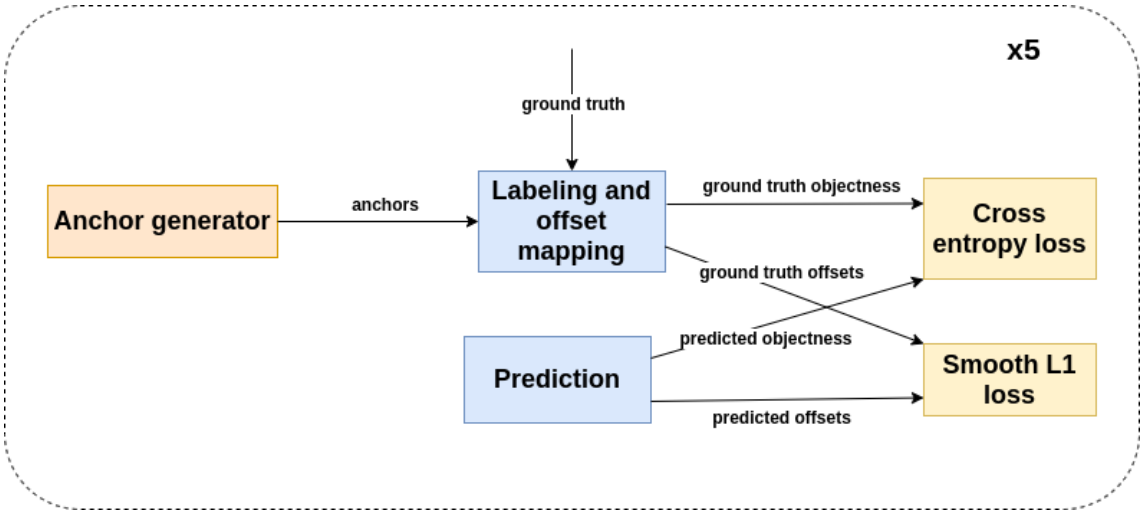


Figure 3: RPN losses. These two losses are duplicated five times to train predictors on center and corner data.

by these annotations. (2) The images of the second set were purposely selected to obtain a database with high object density. Moreover, they were almost entirely annotated. (3) The third set contains images with a lower object density. We took 22 images from our first set with 1086 annotations. We created two test sets: (1) the first one contains 11 images with 535 annotations coming from our first image set, (2) the second contains 25 images with 1691 annotations coming from the second set of images. By using two image sets we aimed at comparing our approach under two situations: images with normal density and images with higher density.

6.2 Parameters

We mainly used the default parameter values set by the Detectron library except for some parameters. For training, a pretrained Faster-RCNN model with FPN architecture was used. This model uses a pretrained ResNet-50 backbone trained on ImageNet dataset. The model was trained on COCO dataset. As long as transfer learning is used, 8000 iterations of training on bee images was enough to achieve state of art results. The number of preserved proposals before and after NMS algorithm were also fixed through two specific Detectron parameters. We changed the default values because the relevance of our approach depends on that parameters.

6.3 Result Analysis

Table 1 shows the results of the tests for the standard approach against the proposed approach. Recall and precision are significantly improved by the corner approach. The recall evaluates the number of detected objects compared to ground truth ones. The precision corresponds to the proportion of correct detections among all detections.

To compare the standard approach to the corner approach, the number of RPN output proposals must be the same. The number of proposals preserved before NMS is 200. When FPN is used, 200 best proposals are retrieved from each feature map. There are 5 feature maps, it means that 1000 proposals are preserved. After NMS, in the standard case, 1000 best proposals are preserved, it means that all the proposals are preserved. In the case of our approach, 200 best proposals are preserved from each center/corner prediction after NMS, before merging the five sets to obtain 1000 proposals. These parameter values were chosen because of the constraint of the inference. In fact NMS is a greedy algorithm, it requires time and resources. Providing a big number of proposals ttopo NMS is not very convenient if quick detection is required.

The results demonstrate that our approach detects more objects than the standard. This is due to the predicted corner data which enables the neural network to detect bees from corners.

7 DISCUSSION

To understand our results, and explore our approach advantages and limits, we have analyzed some aspects of them. First, we verified the accuracy of the center predictor alone, this is explained in Section 7.1. In Section 7.2 we talk about the influence of the number of the preserved proposals on the neural network

Table 1: Results showing accuracy metrics of standard approach and our proposed approach. RPN AR@1000: RPN average recall on 1000 first detections using 0.5 IoU, Faster RCNN AR: Faster RCNN average recall on 100 first detections using 0.5 IoU, Faster RCNN AR: Faster RCNN average precision on 100 first detections using 0.5:0.95 IoU. Number of proposals before RPN NMS is 200, number of porposals after RPN NMS is 1000.

Metric / Approach		RPN AR@1000	Faster RCNN AR	Faster RCNN AP
	Metric type	Recall	Recall	Precision
Standard Approach	Normal density	76.45	76.45	46.03
	High density	56.06	53.87	27.6
Corner Approach	Normal density	86.73	82.8	52.17
	High density	66.88	55.82	28.54

Table 2: Results showing accuracy metrics of our approach when some modifications are applied: (1) One loss instead of five losses, (2) One branch for detecting corner data instead of four branches, (3) Using shifted convolution in corner branches instead of normal convolution. RPN AR@1000: RPN average recall on 1000 first detections using 0.5 IoU, Faster RCNN AR: Faster RCNN average recall on 100 first detections using 0.5 IoU, Faster RCNN AR: Faster RCNN average precision on 100 first detections using 0.5:0.95 IoU. Number of proposals before RPN NMS is 200, number of porposals after RPN NMS is 1000.

Metric / Approach		RPN AR@1000	Faster RCNN AR	Faster RCNN AP
	Metric type	Recall	Recall	Precision
One loss	Normal density	85.79	81.31	51.95
	High density	65.58	52.27	28.35
One branch	Normal density	82.82	82.61	48.71
	High density	64.7	59.43	30.02
Shifted convolution	Normal density	84.67	81.87	51.94
	High density	63.87	54.35	29.02

accuracy. In Section 7.3 we focus on the possibility of merging the proposals before NMS. On the other hand, we explored using one loss instead of five losses and one branch instead of five branches as explained in Section 7.4 and Section 7.5. In Section 7.6, the new convolution method that we used to enhance corner data representation is described. Finally, in Section 7.7, we analyze the results that we obtained when we trained our neural network on bee images with low density.

7.1 Accuracy of Center Data

When we compared the accuracy of the standard neural network and the network based on our approach using only center prediction, the standard gives better results. Indeed, it is more difficult for the training to find backbone parameters that satisfy all the objective losses. Nevertheless, the results demonstrated that the proposed approach detects corner data which compensates the reduced accuracy of center data prediction.

7.2 Number of Preserved Proposals before NMS

We noticed that the accuracy of our approach is not good when the number of preserved proposals from each feature map exceeds 200. It remains a limitation of this approach that have to be addressed in the future. But the results show that corner data prediction is a promising way to detect objects in dense scenes.

7.3 Proposal Concatenation

The proposed approach is based on the concatenation of the proposals after NMS. Merging these proposals before NMS algorithm is another possibility: in this case, the predicted offsets for each branch (center/corner) are applied to the anchors to obtain proposals, these bounding boxes are concatenated in a set, then they are ordered by their objectness score, and NMS is applied to reduce multi-detection. The advantage of this method is that it executes the NMS algorithm once instead of five times. But its results were not satisfactory. Furthermore, in this alternative approach, NMS should be applied to a greater number of proposals which require a greater use of resources.

7.4 One Loss vs Five Losses

Instead of using one loss for each corner or center, only one loss could be used. In fact, this can be implemented by combining corner and center predicted data in a single layer. The five anchor generators are replaced by one generator which generates the five types of anchors. The labeling and mapping step is performed on all the generated anchors. As demon-

Table 3: Results showing accuracy metrics of standard approach and our proposed approach when training on dataset with low object density. RPN AR@1000: RPN average recall on 1000 first detections using 0.5 IoU, Faster RCNN AR: Faster RCNN average recall on 100 first detections using 0.5 IoU, Faster RCNN AR: Faster RCNN average precision on 100 first detections using 0.5:0.95 IoU.

Metric / Approach		RPN AR@1000	Faster RCNN AR	Faster RCNN AP
	Metric type	Recall	Recall	Precision
Standard Approach	Normal density	62.99	45.61	23.01
	High density	49.67	21.94	10.81
Corner Approach	Normal density	65.42	55.33	27.69
	High density	53.93	31.52	14.27

strated by Table 2 this approach gives lower results than the one based on five different losses.

7.5 One Branch for Corner Data

A similar approach was also tested. Instead of using four different branches for the corner data, one branch is used. This method has the advantage of reducing the number of parameters to learn. Despite of the constraint of representing different types of corners with the same set of parameters, the approach results are promising as shown in Table 2.

7.6 Shifted Convolution

Our approach is based on predicting objectness and offsets from object corners. Whether or not a feature map region corresponds to a corner depends on the region content and its surroundings. Regarding the corners, the prediction is more influenced by the region containing the object. This region is not encircling the corner as it is the case for object center, but it is shifted. For instance, for a top left corner, instead of using a surrounding centered on the corner point, this surrounding should be shifted to right and bottom. Therefore instead of convolving the points of standard surrounding, the convolution is made on this new surrounding which contains more relevant corner data. The results of this approach are presented by Table 2.

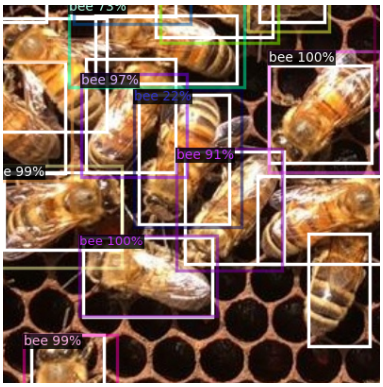
7.7 Sparse Images as Training Set

To check the relevance of our approach, we trained the neural network on a simple database made up of 22 images with only 424 annotations. In fact these images contain sparse bees. The objective was to verify whether the approach could detect partially visible objects using only visible parts which are mainly bee corners. So we used a sparse image set to prevent the network to learn hidden objects by considering them as small. The results are shown in Table 3. The limits of those results are that the accuracy is very low compared to the values of state of the art. In fact it is certainly due to the limited number of annotations used for training. But the advantage of that case, is that there is no limit to the number of considered proposals before NMS. More investigation should be conducted in the future to understand the relation between the training annotations and the performance of the approach. Figure 4 shows that our approach detects some occluded bees while the normal approach could not predict them.

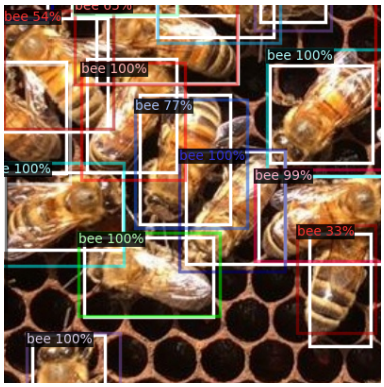
8 CONCLUSION

In this paper, we propose a new approach to detect objects in dense scenes. It is an extension of the standard Faster RCNN. In addition to predicting whether the positions of the feature map correspond to the object centers, we predict whether they correspond to the object corners. This is achieved through generating a new set of anchors and predicting their corresponding objectness and offset data. Our approach aims to improve object detection in the case of bee frame images. Therefore, we built and annotated a specific dataset. We executed the training and testing to evaluate our approach. When the number of the preserved proposals before NMS is less than 200, our approach performs better than the standard. So it can be used in situations where computation resources are limited. This case can occur when there are constraints to obtain images. Moreover, our proposed neural network can be used in other object detection contexts. Since our approach is only to extend anchor generation mechanism, it is not specific to Faster RCNN, but it can be exploited in other anchor based neural networks like YOLO. We hope that our approach can help beekeepers to monitor their beehives more accurately, and that our work can contribute to the advancement in computer vision research.

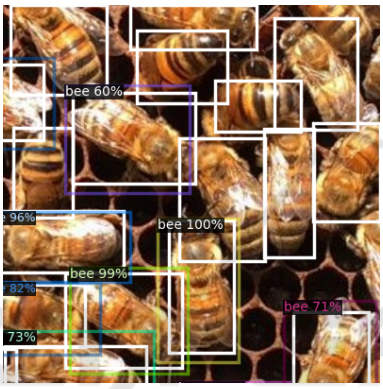
Nevertheless, there are some limitations that must be addressed in the future. How to improve recall independently to the number of preserved proposals should be treated in more depth. On the other hand,



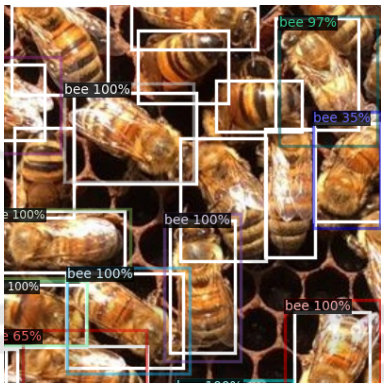
(a) Detected bees in the first image using standard approach.



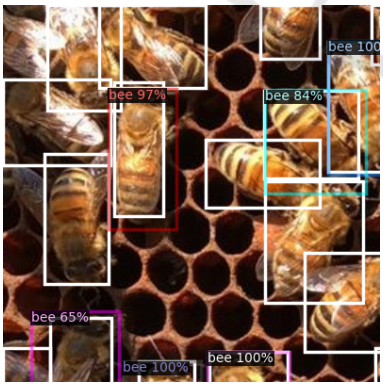
(b) Detected bees in the first image using corner approach.



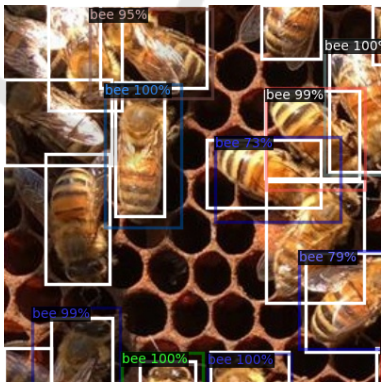
(c) Detected bees in the second image using standard approach.



(d) Detected bees in the second image using corner approach.



(e) Detected bees in the third image using standard approach.



(f) Detected bees in the third image using corner approach.

Figure 4: The detected bees in three images using standard and proposed approaches. (a) and (b) relate to the same image. (c) and (d) relate to the same image. (e) and (f) relate to the same image.

our approach should be tested in other contexts and neural networks.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the Ministry of Agriculture and Food which funds PNAPI through CASDAR (the special appropriation account "Agriculture and Rural Development") under project number 18 ART 1831 as well as the support and help of Alexandre Dangléant, ITSAP (Technical and Scientific Institute of Beekeeping and Pollination).

REFERENCES

Dai, J., Li, Y., He, K., and Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. In *International Conference on Neural Information Processing Systems*. Curran Associates Inc.

Dembski J., S. J. (2020). Weighted clustering for bees detection on video images. In *Computational Science*. Springer.

Duan, K., Xie, L., Qi, H., Bai, S., Huang, Q., and Tia, Q. (2020). Corner proposal network for anchor-free, two-stage object detection. In *European Conference on Computer Vision – ECCV*.

Gählert, N., Hanselmann, N., Franke, U., and Denzler, J. (2020). Visibility guided nms: Efficient boosting of amodal object detection in crowded traffic scenes.

Girshick, R. (2015). Fast r-cnn. In *2015 IEEE International Conference on Computer Vision (ICCV)*.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation.

Goldman, E., Herzig, R., Eisenschtat, A., Goldberger, J., and Hassner, T. (2019). Precise detection in densely packed scenes. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kulyukin, V. and Mukherjee, S. (2018). On video analysis of omnidirectional bee traffic: Counting bee motions with motion detection and image classification. *Applied Sciences*.

Kulyukin, V. A. and Reka, S. K. (2016). Toward sustainable electronic beehive monitoring: Algorithms for omnidirectional bee counting from images and harmonic analysis of buzzing signals.

Law, H. and Deng, J. (2018). Cornernet: Detecting objects as paired keypoints. In *European Conference on Computer Vision (ECCV)*.

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

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017b). Focal loss for dense object detection. In *IEEE International Conference on Computer Vision (ICCV)*.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. *Lecture Notes in Computer Science*.

Liu, Y., Liu, L., Rezatofighi, H., Do, T.-T., Shi, Q., and Reid, I. (2019). Learning pairwise relationship for multi-object detection in crowded scenes.

Magnier, B., Ekszterowicz, G., Laurent, J., Rival, M., and Pfister, F. (2018). Bee hive traffic monitoring by tracking bee flight paths. In *Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. SCITEPRESS - Science and Technology Publications.

Magnier, B., Gabbay, E., Bougamale, F., Moradi, B., Pfister, F., and Slangen, P. R. (2019). Multiple honey bees tracking and trajectory modeling. In *Multimodal Sensing: Technologies and Applications*. SPIE.

Qiu, H., Ma, Y., Li, Z., Liu, S., and Sun, J. (2020). Borderdet: Border feature for dense object detection. In *European Conference on Computer Vision – ECCV*.

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

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc.

Sipos, T., Donkó, T., Jócsák, I., and Keszthelyi, S. (2021). Study of morphological features in pre-imaginal honey bee impaired by varroa destructor by means of computer tomography. *Insects*.

Tian, Z., Shen, C., Chen, H., and He, T. (2019). Fcos: Fully convolutional one-stage object detection. In *IEEE/CVF International Conference on Computer Vision (ICCV)*.

Tiwari, A. (2018). *A deep learning approach to recognizing bees in video analysis of bee traffic*. PhD thesis, Utah state university.

Tu, G. J., Hansen, M. K., Kryger, P., and Ahrendt, P. (2016). Automatic behaviour analysis system for honeybees using computer vision. *Computers and Electronics in Agriculture*.

Wei, F., Sun, X., Li, H., Wang, J., and Lin, S. (2020). Point-set anchors for object detection, instance segmentation and pose estimation. In *European Conference on Computer Vision – ECCV*.

Xi, Y., Zheng, J., He, X., Jia, W., Li, H., Xie, Y., Feng, M., and Li, X. (2020). Beyond context: Exploring semantic similarity for small object detection in crowded scenes. *Pattern Recognition Letters*.

Zhang, K., Xiong, F., Sun, P., Hu, L., Li, B., and Yu, G. (2019). Double anchor r-cnn for human detection in a crowd.

Zhang, S., Wen, L., Bian, X., Lei, Z., and Li, S. Z. (2018). Occlusion-aware r-cnn: Detecting pedestrians in a crowd. In *Computer Vision – ECCV*. Springer International Publishing.

Zhou, X., Zhuo, J., and Krähenbühl, P. (2019). Bottom-up object detection by grouping extreme and center points. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.