CornerNet: Detecting Objects as Paired Keypoints

Mert Bektas [1]

Abstract

The team developed a new object detection technique with this paper: finding an object's top-left and bottom-right corners. After their evaluations, they saw that their method outperforms all existing one-stage detectors.

1. Introduction

The standard approach for object detection is anchor boxes, which are several boxes that propose candidates for objects' locations. But these boxes have two downsides. First, we usually need many anchor boxes, which causes training to slow down (1). Second, using anchor boxes makes models more complex due to the boxes' amount, size, and aspect ratios. (1)

Instead of anchor boxes, CornerNet applies a single convolutional network(CNN) to predict a heatmap for the top-left and bottom-right corners and an embedding vector for each detected corner. Then, this network predicts similar embedding for these corners. This approach immensely simplifies the network output and makes object detection possible without anchor boxes. The team also introduced the corner pooling approach. This method takes each channel, finds the leftmost and topmost boundaries of the object, and creates a feature map, then takes the maximum values of these maps and adds them together.

2. Related Works

DeNet (2) is a two-stage detector that finds locations for possible corners and combines and classifies them. On the contrary of CornerNet, it does not check if two corners belong to the same object, selects features for classification, and does not use corner pooling.

Point Linking Network (PLN) is a one-stage detector without anchor boxes. It predicts the four corners and the center. Then combines these locations to create a bounding box. In contrast, CornerNet predicts embedding vectors to group corners, while PLN predicts pixel locations to do it and does not use corner pooling like CornerNet.

Mert Bektas [1]Institute of Computer Science, University of Tartu, Tartu, Estonia. Correspondence to: Mert Bektas <mert.bektas@ut.ee>.

3. CornerNet

3.1. Overview

The team used the hourglass network (3) as the backbone network. After the backbone, the top-left and bottom-right corners are getting predicted with the help of pooled features from the hourglass network.

3.2. Detecting Corners

Instead of penalizing wrong predicted corner locations, they reduced the penalty if they were within a radius of the ground-truth location. If these wrong predictions are close enough to the ground truth, they can still produce a box overlapping the ground truth box.

They realized precision might be lost when they remap locations from the heatmaps to the input image. Before remapping the corners, they predicted location offsets to slightly adjust their locations to solve this problem. They predicted one set of top-left and bottom-right corners and applied the smooth L1 Loss at ground truth corners.

3.3. Grouping Corners

Multiple top-left and bottom-right corners may be seen in an image due to numerous objects. Therefore, the model should check if the corners are from the same bounding box. They used the distances between the embeddings to group the corners to overcome this problem.

3.4. Corner Pooling

The team proposed corner pooling to localize the corners better because there is a drawback of not having prior knowledge about corners.

To find if a pixel at location *(i, j)* is a top-left corner, they use feature maps with *(H X W)* dimension and max-pools all feature vectors between *(i, j)* and *(i, H)*. Next, they add the feature vectors before max-pool and after together. Finally, they apply an elementwise max operation.

Likewise, to find bottom-right corners, it max-pools all feature vectors from *(0, j)* to *(i, j)* and from *(i, 0)* to *(i,j)*.

Method	Backbone	AP	AP^{50}	AP^{75}	AP^s	AP^m	AP^l	AR^1	AR^{10}	AR^{100}	AR^s	AR^m	AR^l
Two-stage detectors													
DeNet (Tychsen-Smith and Petersson, 2017a)	ResNet-101	33.8	53.4	36.1	12.3	36.1	50.8	29.6	42.6	43.5	19.2	46.9	64.3
CoupleNet (Zhu et al., 2017)	ResNet-101	34.4	54.8	37.2	13.4	38.1	50.8	30.0	45.0	46.4	20.7	53.1	68.5
Faster R-CNN by G-RMI (Huang et al., 2017)	Inception-ResNet-v2 (Szegedy et al., 2017)	34.7	55.5	36.7	13.5	38.1	52.0	-	-	-	-	-	-
Faster R-CNN+++ (He et al., 2016)	ResNet-101	34.9	55.7	37.4	15.6	38.7	50.9	-	-	-	-	-	-
Faster R-CNN w/ FPN (Lin et al., 2016)	ResNet-101	36.2	59.1	39.0	18.2	39.0	48.2	-	-	-	-	-	-
Faster R-CNN w/ TDM (Shrivastava et al., 2016)	Inception-ResNet-v2	36.8	57.7	39.2	16.2	39.8	52.1	31.6	49.3	51.9	28.1	56.6	71.1
D-FCN (Dai et al., 2017)	Aligned-Inception-ResNet	37.5	58.0	-	19.4	40.1	52.5	-	-	-	-	-	-
Regionlets (Xu et al., 2017)	ResNet-101	39.3	59.8	-	21.7	43.7	50.9	-	-	-	-	-	-
Mask R-CNN (He et al., 2017)	ResNeXt-101	39.8	62.3	43.4	22.1	43.2	51.2	-	-	-	-	-	-
Soft-NMS (Bodla et al., 2017)	Aligned-Inception-ResNet	40.9	62.8	-	23.3	43.6	53.3	-	-	-	-	-	-
LH R-CNN (Li et al., 2017)	ResNet-101	41.5	-	-	25.2	45.3	53.1	-	-	-	-	-	-
Fitness-NMS (Tychsen-Smith and Petersson, 2017b)	ResNet-101	41.8	60.9	44.9	21.5	45.0	57.5	-	-	-	-	-	-
Cascade R-CNN (Cai and Vasconcelos, 2017)	ResNet-101	42.8	62.1	46.3	23.7	45.5	55.2	-	-	-	-	-	-
D-RFCN + SNIP (Singh and Davis, 2017)	DPN-98 (Chen et al., 2017)	45.7	67.3	51.1	29.3	48.8	57.1	-	-	-	-	-	-
One-stage detectors													
YOLOv2 (Redmon and Farhadi, 2016)	DarkNet-19	21.6	44.0	19.2	5.0	22.4	35.5	20.7	31.6	33.3	9.8	36.5	54.4
DSOD300 (Shen et al., 2017a)	DS/64-192-48-1	29.3	47.3	30.6	9.4	31.5	47.0	27.3	40.7	43.0	16.7	47.1	65.0
GRP-DSOD320 (Shen et al., 2017b)	DS/64-192-48-1	30.0	47.9	31.8	10.9	33.6	46.3	28.0	42.1	44.5	18.8	49.1	65.0
SSD513 (Liu et al., 2016)	ResNet-101	31.2	50.4	33.3	10.2	34.5	49.8	28.3	42.1	44.4	17.6	49.2	65.8
DSSD513 (Fu et al., 2017)	ResNet-101	33.2	53.3	35.2	13.0	35.4	51.1	28.9	43.5	46.2	21.8	49.1	66.4
RefineDet512 (single scale) (Zhang et al., 2017)	ResNet-101	36.4	57.5	39.5	16.6	39.9	51.4	-	-	-	-	-	-
RetinaNet800 (Lin et al., 2017)	ResNet-101	39.1	59.1	42.3	21.8	42.7	50.2	-	-	-	-	-	-
RefineDet512 (multi scale) (Zhang et al., 2017)	ResNet-101	41.8	62.9	45.7	25.6	45.1	54.1	-	-	-	-	-	-
CornerNet511 (single scale)	Hourglass-104	40.6	56.4	43.2	19.1	42.8	54.3	35.3	54.7	59.4	37.4	62.4	77.2
CornerNet511 (multi scale)	Hourglass-104	42.2	57.8	45.2	20.7	44.8	56.6	36.6	55.9	60.3	39.5	63.2	77.3

Table 1. CornerNet versus others on MS COCO test-dev. CornerNet outperforms all one-stage detectors and achieves results competitive to two-stage detectors.

Then, applies the same operations as finding top-left corners.

3.5. Hourglass Network

This module takes the input and downsamples its features after a series of convolutional and max pooling layers. Then, it upsamples it by using upsampling and convolution layers. The goal of this module is to catch both global and local features. It is also possible to use multiple hourglasses, as the team used two in this architecture. Finally, they stated that CornerNet only uses the features from the network's final layer to make predictions, unlike other cutting-edge detectors.

4. Experiments

4.1. Training Details

They did not use any pretraining on any external dataset, and they used the default settings of PyTorch. They used data augmentation methods to reduce overfitting, such as random horizontal flipping, scaling, cropping, and color jittering. In addition, they applied PCA (4) to the input image and used the Adam optimization function (5) for loss.

4.2. Testing Details

They applied non-maximal suppression (NMS) by a 3x3 max pooling layer on the corner heatmaps. After this, they picked the top 100 top-left and bottom-right corners from these heatmaps. Then, they calculated the L1 distance between the top-left and bottom-right corners. Finally, they dropped the corner pairs that had distances greater than 0.5 and used average scores of distances as the detection scores.

4.3. MS COCO

The team used the MS COCO dataset, which has 140k images. This dataset uses average precisions (APs) as the primary metric for evaluation.

4.4. Comparisons with state-of-the-art detectors

CornerNet gets an AP of 42.2% with multiscale evaluation, which is the best among available one-stage approaches and competitive with two-stage methods. Results can be seen from the Table 1

5. Conclusion

The paper introduced CornerNet, a novel object recognition method that recognizes pairings of corners in bounding boxes. In addition, the team tested CornerNet on MS COCO and presented competitive results.

References

[1] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollar, P. ((2017). Focal loss for dense object detection. arXiv preprint arXiv:1708.02002

[2] Tychsen-Smith, L. and Petersson, L. ; (2017a). Denet: Scalable real-time object detection with directed sparse sampling. arXiv preprint arXiv:1703.10295

[3] Newell, A., Yang, K., and Deng, J (2016). Stacked hourglass networks for human pose estimation. In European Conference on Computer Vision, pages 483–499. Springer.

[4] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105.

[5] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980