CenterNet review

Title: “CenterNet: Keypoint Triplets for Object Detection”

Authors: Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, Qi Tian

Link: https://arxiv.org/pdf/1904.08189v3.pdf

Tags: Deep Learning, Object detection, CornerNet, Keypoints, Average precision, MSCOCO-dataset, Center pooling

Year: 2019

Code: https://github.com/Duankaiwen/CenterNet.

Summary: CenterNet is a new approach for object detection in Deep Learning, based on detecting and fitting triplet of parameters, like center of object area and corners, which describe the object. It is a one-stage object detection approach and currently it shows pretty good accuracy on commonly used dataset MSCOCO and gives 47% of average precision.

What

As we know, object detection is one of the main problems in CV. Object detection is more challenging and combines these two tasks and draws a bounding box around each object of interest in the image and assigns them a class label. So as an input we have image with one or more objects and the output should be one or more bounding boxes and a class label for each bounding box.

There are two types of object detection algorithms in CV: two-stage and one-stage object detection. First approach consist of two stages: Extract possible objects using a region proposal method (usually Selective search) and second stage is finding correct objects from proposed regions and assign the class for them. The second one is simpler and faster model architecture, although it can sometimes struggle to be flexible enough to adapt to arbitrary tasks.

So CenterNet is framework for one-stage object detection, that has shown pretty good results in comparison with other two-stage approaches. It is similar to ExtremeNet or CornerNet, where the bounding box is now defined by a pair of corner points and the label is defined by the response of the center point. CenterNet is end-to-end differentiable, simpler, faster, and more accurate than corresponding bounding box based detectors.

How

So as was mentioned before, CenterNet describes each object as a center keypoint and a pair of corners. Originally CenterNet was written in PyTorch and shared by the link, provided in the document. The backbone for CenterNet is Hourglass-like model with 52 or 104 layers. After the backbone we can see the main part of CenterNet framework Cascade Corner Pooling and Center Pooling. These two modules output two corner heatmaps and a center keypoint heatmap which are used for computing offsets and embeddings for center and

corners of every object. This approach is similar to CornerNet , where the pair of detected corners and embeddings are used to detect bounding boxes for every potential object. And finally we can use center keypoints to find final bounding boxes. Let's walk through details of the architecture.

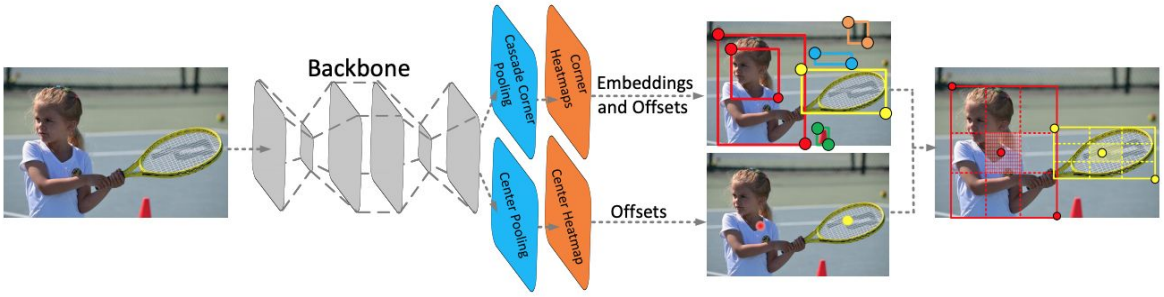


Figure 1. CenterNet architecture

As was mentioned the backbone of the CenterNet is Hourglass model, which is based on the successive steps of pooling and upsampling that are done to produce a final set of predictions and allows for repeated bottom-up, top-down inference. So this backbone pools down to a low resolution, then upsamples and combines features across multiple resolutions. Also it processes spatial information at multiple scales for dense prediction what gives a better representation of features from the image. So the backbone outputs a feature map, and to understand is the the pixel in the map a center keypoint we are looking for maximum in both directions and add them (center pooling), what will be describe below.

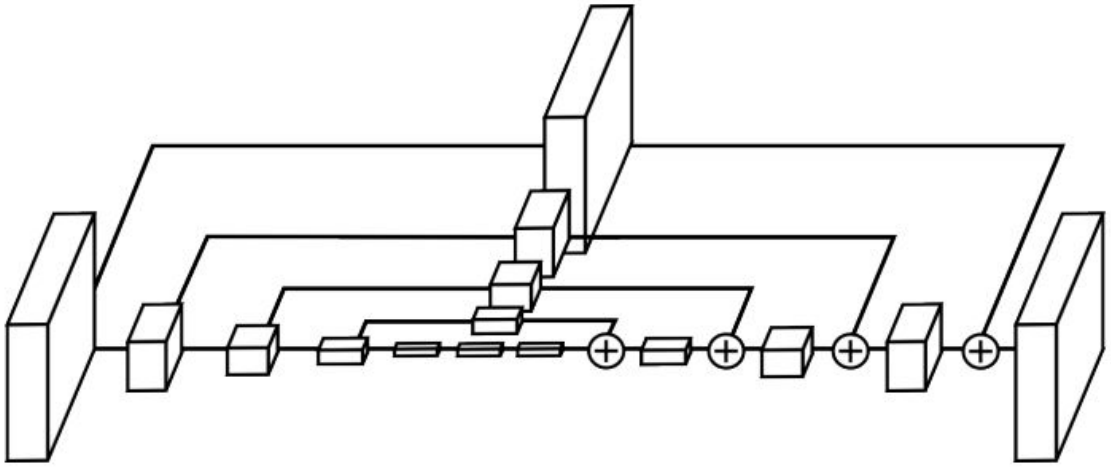


Figure 2. Simple hourglass model

After backbone we can see two customized modules named cascade corner pooling and center pooling, which are the main piece of CenterNet. These two modules give opportunity to enrich collected features from data, also retrieve visual patterns and output more information. First, center pooling module predicts the center keypoints, and in such way to get better patterns about central part of proposal. It is done by taking out the max summed response in both horizontal and vertical directions of the center keypoint on a feature map for

predicting center keypoints. Next is cascade corner pooling, that is improved corner pooling, that takes as input two feature maps and make pooling in two directions and then adds two pools together. So the approach is the same, but actually it pools in two directions (boundary and internal) of objects on a feature map for predicting corners and as result we receive more internal information from image. With cascade corner pooling our corners obtain both the the boundary information and the visual patterns of objects.

These two approaches are used, as, according to the paper, two-directional pooling is more robust to the noise and also is more stable, what impacts the final accuracy of the model. So as a summary of those two modules, we can see, that CenterNet applies center pooling and corner pooling on center feature maps and corner feature maps and as result we receive two corner heatmaps and one corner heatmap.

After that, when we receive our heatmaps, we use them to predict the offsets of the center keypoints and also to predict embeddings and offsets of the corners. Next we are going to find proposals for bounding boxes and for for this task the method from CornerNet is used. So basically we select top-k center keypoints and then remap them to the input image with the help of received offsets. And as soon as we find proposed bounding boxes (pair of corner keypoints), we need to check is this box an object with checking absence or presence of center keypoints of the same class within the central region of bounding box. If there is no center keypoint in the box, then this box will be simply removed.

As result of finding bounding boxes, their score is replaced with average scores of the corners and central keypoint. And finally using the final score we detect objects with specified class from the image.

Results

The evaluation of CenterNet was done using MSCOCO dataset, that is commonly used in the object detection problem accuracy testing. This dataset contains 80 categories and more than 1.5 million of object instances. As a baseline CornerNet was used (another anchor-free approach for object detection).

To test the results and accuracy Average precision was used. Average precision is computed over ten different IoU thresholds (i.e., 0.5 : 0.05 : 0.95) and all categories.

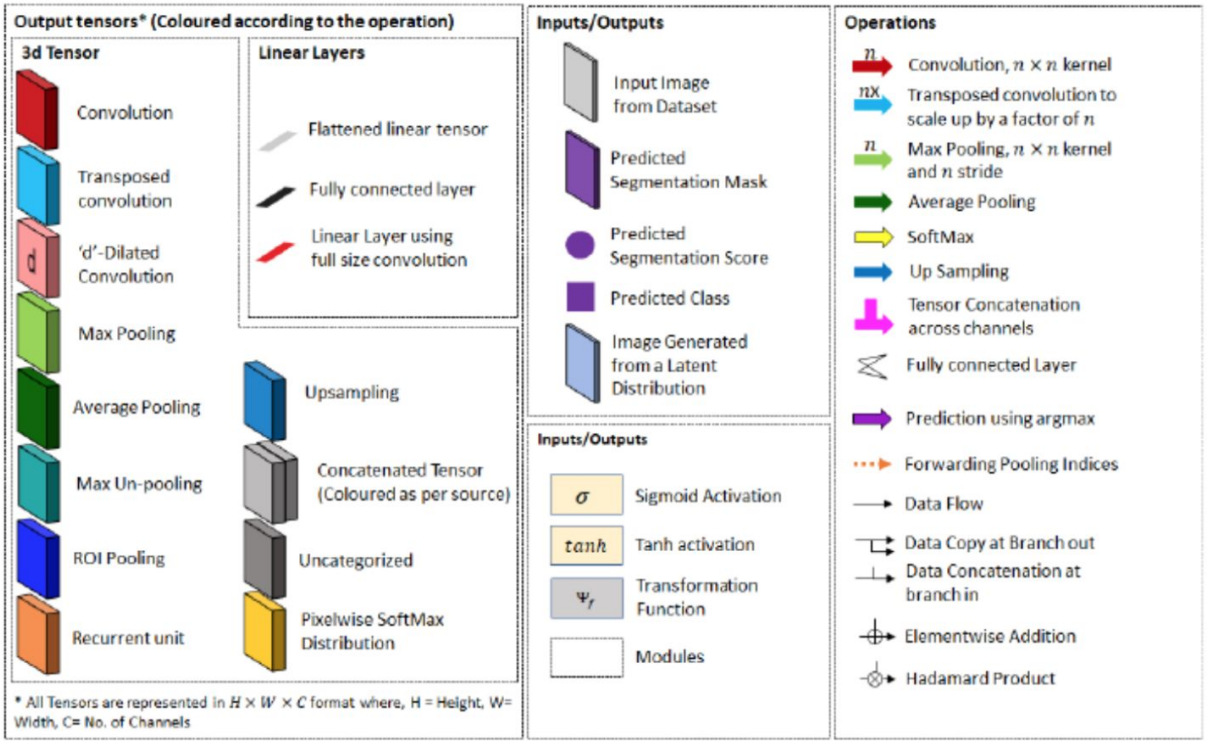
As result, CenterNet gives very good accuracy in comparison with other two-stage object detectors, also the performance is better as we have only one stage, and size of used resources are much lower. The results of accuracy are next:

Method	Backbone	Train input	Test input	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:															
DeNet [40]	ResNet-101 [14]	512×512	512×512	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 [47]	ResNet-101	ori.	ori.	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 [16]	Inception-ResNet-v2 [39]	∼ 1000×600	∼ 1000×600	34.7	55.5	36.7	13.5	38.1	52.0	-	-	-	-	-	-
Faster R-CNN +++ [14]	ResNet-101	∼ 1000×600	∼ 1000×600	34.9	55.7	37.4	15.6	38.7	50.9	-	-	-	-	-	-
Faster R-CNN w/ FPN [23]	ResNet-101	∼ 1000×600	∼ 1000×600	36.2	59.1	39.0	18.2	39.0	48.2	-	-	-	-	-	-
Faster R-CNN w/ TDM [37]	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 [7]	Aligned-Inception-ResNet	∼ 1000×600	∼ 1000×600	37.5	58.0	-	19.4	40.1	52.5	-	-	-	-	-	-
Regionlets [43]	ResNet-101	∼ 1000×600	∼ 1000×600	39.3	59.8	-	21.7	43.7	50.9	-	-	-	-	-	-
Mask R-CNN [12]	ResNeXt-101	∼ 1300×800	∼ 1300×800	39.8	62.3	43.4	22.1	43.2	51.2	-	-	-	-	-	-
Soft-NMS [2]	Aligned-Inception-ResNet	∼ 1300×800	∼ 1300×800	40.9	62.8	-	23.3	43.6	53.3	-	-	-	-	-	-
Fitness R-CNN [41]	ResNet-101	512×512	1024×1024	41.8	60.9	44.9	21.5	45.0	57.5	-	-	-	-	-	-
Cascade R-CNN [4]	ResNet-101	-	-	42.8	62.1	46.3	23.7	45.5	55.2	-	-	-	-	-	-
Grid R-CNN w/ FPN [28]	ResNeXt-101	∼ 1300×800	∼ 1300×800	43.2	63.0	46.6	25.1	46.5	55.2	-	-	-	-	-	-
D-RFCN + SNIP (multi-scale) [38]	DPN-98 [5]	∼ 2000×1200	∼ 2000×1200	45.7	**67.3**	51.1	29.3	48.8	57.1	-	-	-	-	-	-
PANet (multi-scale) [26]	ResNeXt-101	∼ 1400×840	∼ 1400×840	**47.4**	67.2	**51.8**	**30.1**	**51.7**	**60.0**	-	-	-	-	-	-
One-stage:															
YOLOv2 [32]	DarkNet-19	544×544	544×544	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 [34]	DS/64-192-48-1	300×300	300×300	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 [35]	DS/64-192-48-1	320×320	320×320	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 [27]	ResNet-101	513×513	513×513	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 [8]	ResNet-101	513×513	513×513	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) [45]	ResNet-101	512×512	512×512	36.4	57.5	39.5	16.6	39.9	51.4	-	-	-	-	-	-
CornerNet511 (single-scale) [20]	Hourglass-52	511×511	ori.	37.8	53.7	40.1	17.0	39.0	50.5	33.9	52.3	57.0	35.0	59.3	74.7
RetinaNet800 [24]	ResNet-101	800×800	800×800	39.1	59.1	42.3	21.8	42.7	50.2	-	-	-	-	-	-
CornerNet511 (multi-scale) [20]	Hourglass-52	511×511	≤1.5×	39.4	54.9	42.3	18.9	41.2	52.7	35.0	53.5	57.7	36.1	60.1	75.1
CornerNet511 (single-scale) [20]	Hourglass-104	511×511	ori.	40.5	56.5	43.1	19.4	42.7	53.9	35.3	54.3	59.1	37.4	61.9	76.9
RefineDet512 (multi-scale) [45]	ResNet-101	512×512	≤2.25×	41.8	62.9	45.7	25.6	45.1	54.1						
CornerNet511 (multi-scale) [20]	Hourglass-104	511×511	≤1.5×	42.1	57.8	45.3	20.8	44.8	56.7	36.4	55.7	60.0	38.5	62.7	77.4
CenterNet511 (single-scale)	Hourglass-52	511×511	ori.	41.6	59.4	44.2	22.5	43.1	54.1	34.8	55.7	60.1	38.6	63.3	76.9
CenterNet511 (single-scale)	Hourglass-104	511×511	ori.	44.9	62.4	48.1	25.6	47.4	57.4	36.1	58.4	63.3	41.3	67.1	80.2
CenterNet511 (multi-scale)	Hourglass-52	511×511	≤1.8×	43.5	61.3	46.7	25.3	45.3	55.0	36.0	57.2	61.3	41.4	64.0	76.3
CenterNet511 (multi-scale)	Hourglass-104	511×511	≤1.8×	**47.0**	**64.5**	**50.7**	**28.9**	**49.9**	**58.9**	**37.5**	**60.3**	**64.8**	**45.1**	**68.3**	**79.7**

Table 1. Accuracy comparison of CenterNet with commonly used approaches

Mainly performance of CenterNet depends on the backbone, that is used under the hood. As we can see, that in analysis were used two types of Hourglass nets as Hourglass-52 and Hourglass-104 and we can see, that AP is 41.6 and 47. Performance of the model with 52-layer hourglass backbone is in average inference time of 270 ms and 340 ms using a 104-layer hourglass backbone per image.

CNN visualization



So usually in CenterNet there is used Hourglass-like model, that can be used as a backbone (Hourglass-52 with 52 layers or Hourglass-104 with 104 layers). As this is just a type of model, i've tried to understand from the original code from the repo. First of all, want to say, that the code is done with respect to architectural rules, but it is extremely hard to read it and understand the overall picture and the tree, that is build under the hood.

So i'm providing the overall architecture and more details on the specific parts of the CenterNet.

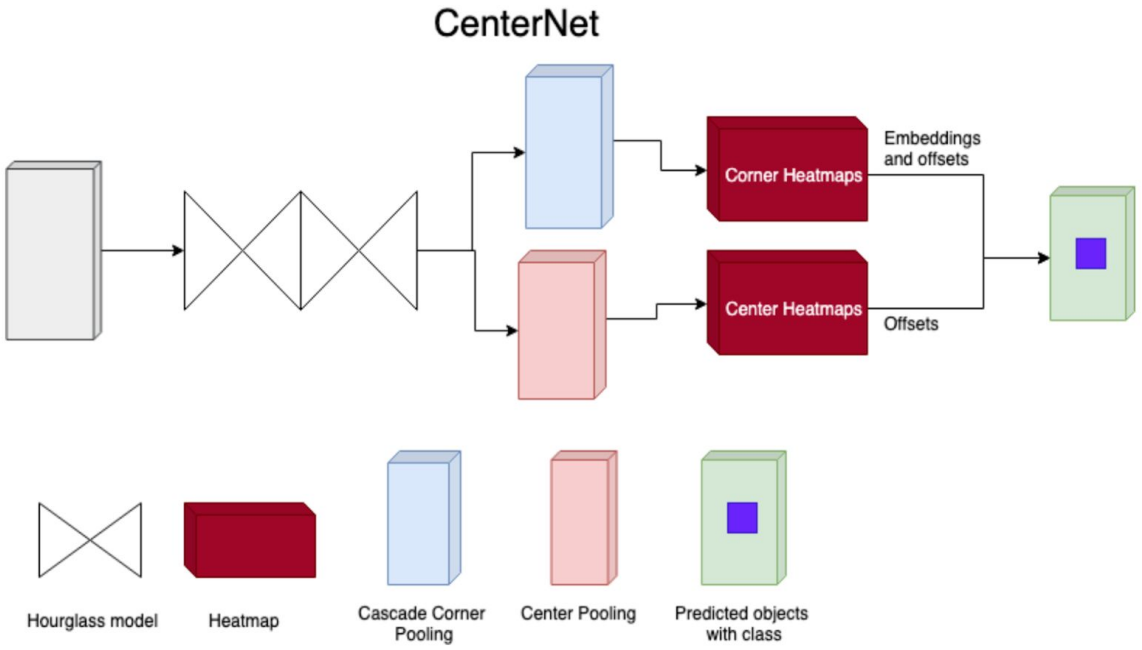


Figure 3. General diagram of CenterNet architecture

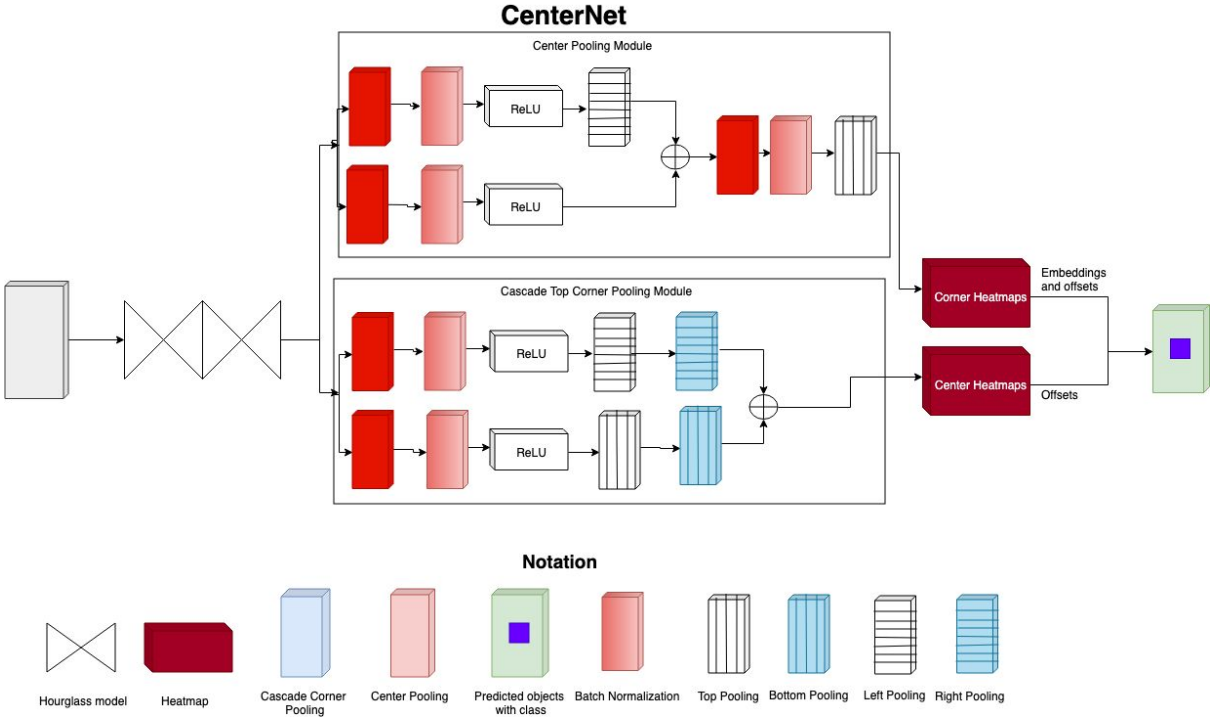


Figure 4. Extended CenterNet architecture diagram