Objects as Points

Xingyi Zhou
UT Austin
zhouxy@cs.utexas.edu

Dequan Wang
UC Berkeley
dqwang@cs.berkeley.edu

Philipp Krähenbühl
UT Austin
philkr@cs.utexas.edu

Abstract

Detection identifies objects as axis-aligned boxes in an image. Most successful object detectors enumerate a nearly exhaustive list of potential object locations and classify each. This is wasteful, inefficient, and requires additional post-processing. In this paper, we take a different approach. We model an object as a single point — the center point of its bounding box. Our detector uses keypoint estimation to find center points and regresses to all other object properties, such as size, 3D location, orientation, and even pose. Our center point based approach, CenterNet, is end-to-end differentiable, simpler, faster, and more accurate than corresponding bounding box based detectors. CenterNet achieves the best speed-accuracy trade-off on the MS COCO dataset, with 28.1% AP at 142 FPS, 37.4% AP at 52 FPS, and 45.1% AP with multi-scale testing at 1.4 FPS. We use the same approach to estimate 3D bounding box in the KITTI benchmark and human pose on the COCO keypoint dataset. Our method performs competitively with sophisticated multi-stage methods and runs in real-time.

1. Introduction

Object detection powers many vision tasks like instance segmentation [7, 21, 32], pose estimation [3, 15, 39], tracking [24, 27], and action recognition [5]. It has down-stream applications in surveillance [57], autonomous driving [53], and visual question answering [1]. Current object detectors represent each object through an axis-aligned bounding box that tightly encompasses the object [18, 19, 33, 43, 46]. They then reduce object detection to image classification of an extensive number of potential object bounding boxes. For each bounding box, the classifier determines if the image content is a specific object or background. One-stage detectors [33, 43] slide a complex arrangement of possible bounding boxes, called anchors, over the image and classify them directly without specifying the box content. Two-stage detectors [18, 19, 46] recompute image-features for each potential box, then classify those features. Post-processing, namely non-maxima suppression, then removes duplicated detections for the same instance by computing bounding box IoU. This post-processing is hard to differentiate and train [23], hence most current detectors are not end-to-end trainable. Nonetheless, over the past five years [19], this idea has achieved good empirical success [12, 21, 25, 26, 31, 35, 47, 48, 56, 62, 63]. Sliding window based object detectors are however a bit wasteful, as they need to enumerate all possible object locations and dimensions.

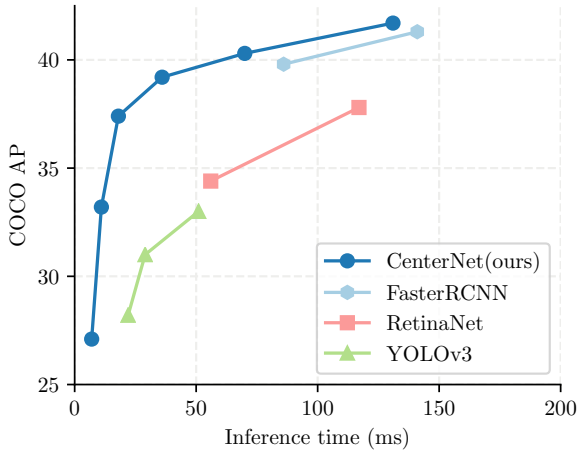


Figure 1: Speed-accuracy trade-off on COCO validation for real-time detectors. The proposed CenterNet outperforms a range of state-of-the-art algorithms.

In this paper, we provide a much simpler and more efficient alternative. We represent objects by a single point at their bounding box center (see Figure 2). Other properties, such as object size, dimension, 3D extent, orientation, and pose are then regressed directly from image features at the center location. Object detection is then a standard keypoint estimation problem [3, 39, 60]. We simply feed the input image to a fully convolutional network [37, 40] that generates a heatmap. Peaks in this heatmap correspond to object centers. Image features at each peak predict the objects bounding box height and weight. The model trains using standard dense supervised learning [39, 60]. Inference is a single network forward-pass, without non-maximal suppression for post-processing.

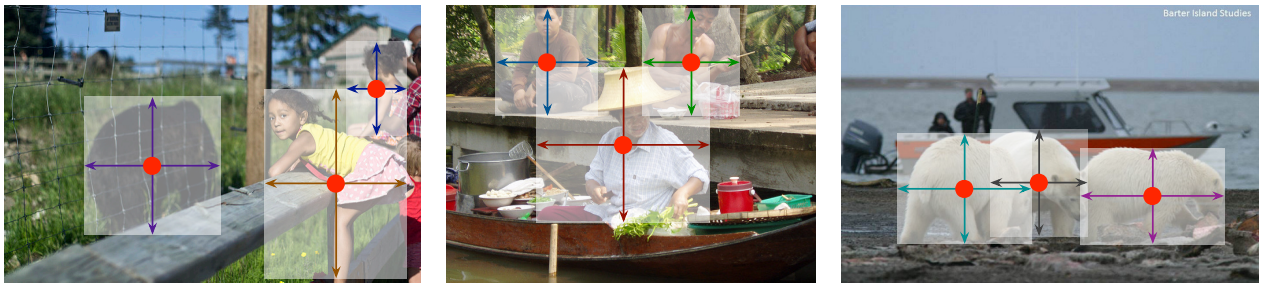


Figure 2: We model an object as the center point of its bounding box. The bounding box size and other object properties are inferred from the keypoint feature at the center. Best viewed in color.

Our method is general and can be extended to other tasks with minor effort. We provide experiments on 3D object detection [17] and multi-person human pose estimation [4], by predicting additional outputs at each center point (see Figure 4). For 3D bounding box estimation, we regress to the object absolute depth, 3D bounding box dimensions, and object orientation [38]. For human pose estimation, we consider the 2D joint locations as offsets from the center and directly regress to them at the center point location.

The simplicity of our method, CenterNet, allows it to run at a very high speed (Figure 1). With a simple Resnet-18 and up-convolutional layers [55], our network runs at 142 FPS with 28.1% COCO bounding box AP. With a carefully designed keypoint detection network, DLA-34 [58], our network achieves 37.4% COCO AP at 52 FPS. Equipped with the state-of-the-art keypoint estimation network, Hourglass-104 [30, 40], and multi-scale testing, our network achieves 45.1% COCO AP at 1.4 FPS. On 3D bounding box estimation and human pose estimation, we perform competitively with state-of-the-art at a higher inference speed. Code is available at https://github.com/xingyizhou/CenterNet.

2. Related work

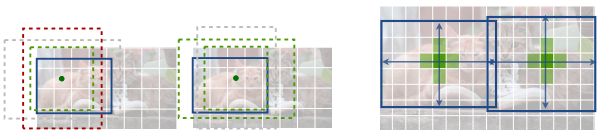
Object detection by region classification. One of the first successful deep object detectors, RCNN [19], enumerates object location from a large set of region candidates [52], crops them, and classifies each using a deep network. Fast-RCNN [18] crops image features instead, to save computation. However, both methods rely on slow low-level region proposal methods.

Object detection with implicit anchors. Faster RCNN [46] generates region proposal within the detection network. It samples fixed-shape bounding boxes (anchors) around a low-resolution image grid and classifies each into "foreground or not". An anchor is labeled foreground with a >0.7 overlap with any ground truth object, background with a <0.3 overlap, or ignored otherwise. Each generated region proposal is again classified [18]. Changing the proposal classifier to a multi-class classification forms the basis of one-stage detectors. Several improvements to one-stage detectors include anchor shape priors [44, 45], different feature resolution [36], and loss re-weighting among different samples [33].

Our approach is closely related to anchor-based one-stage approaches [33, 36, 43]. A center point can be seen as a single shape-agnostic anchor (see Figure 3). However, there are a few important differences. First, our CenterNet assigns the "anchor" based solely on location, not box overlap [18]. We have no manual thresholds [18] for foreground and background classification. Second, we only have one positive "anchor" per object, and hence do not need Non-Maximum Suppression (NMS) [2]. We simply extract local peaks in the keypoint heatmap [4, 39]. Third, CenterNet uses a larger output resolution (output stride of 4) compared to traditional object detectors [21, 22] (output stride of 16). This eliminates the need for multiple anchors [47].

Object detection by keypoint estimation. We are not the first to use keypoint estimation for object detection. CornerNet [30] detects two bounding box corners as keypoints, while ExtremeNet [61] detects the top-, left-, bottom-, right-most, and center points of all objects. Both these methods build on the same robust keypoint estimation network as our CenterNet. However, they require a combinatorial grouping stage after keypoint detection, which significantly slows down each algorithm. Our CenterNet, on the other hand, simply extracts a single center point per object without the need for grouping or post-processing.

Monocular 3D object detection. 3D bounding box estimation powers autonomous driving [17]. Deep3Dbox [38] uses a slow-RCNN [19] style framework, by first detecting 2D objects [46] and then feeding each object into a 3D estimation network. 3D RCNN [29] adds an additional head to Faster-RCNN [46] followed by a 3D projection. Deep Manta [6] uses a coarse-to-fine Faster-RCNN [46] trained on many tasks. Our method is similar to a one-stage version of Deep3Dbox [38] or 3DRCNN [29]. As such, CenterNet is much simpler and faster than competing methods.



(a) Standard anchor based detection. Anchors count as positive with an overlap $IoU > 0.7$ to any object, negative with an overlap $IoU < 0.3$, or are ignored otherwise.

(b) Center point based detection. The center pixel is assigned to the object. Nearby points have a reduced negative loss. Object size is regressed.

Figure 3: Different between anchor-based detectors (a) and our center point detector (b). Best viewed on screen.

3. Preliminary

Let $I \in R^{W \times H \times 3}$ be an input image of width W and height H. Our aim is to produce a keypoint heatmap $\hat{Y} \in [0,1]^{\frac{W}{R} \times \frac{H}{R} \times C}$, where R is the output stride and C is the number of keypoint types. Keypoint types include $C = 17$ human joints in human pose estimation [4, 55], or $C = 80$ object categories in object detection [30, 61]. We use the default output stride of $R = 4$ in literature [4, 40, 42]. The output stride downsamples the output prediction by a factor R. A prediction $\hat{Y}_{x,y,c} = 1$ corresponds to a detected keypoint, while $\hat{Y}_{x,y,c} = 0$ is background. We use several different fully-convolutional encoder-decoder networks to predict $\hat{Y}$ from an image I: A stacked hourglass network [30, 40], upconvolutional residual networks (ResNet) [22, 55], and deep layer aggregation (DLA) [58].

We train the keypoint prediction network following Law and Deng [30]. For each ground truth keypoint $p \in \mathcal{R}^2$ of class c, we compute a low-resolution equivalent $\tilde{p} = \lfloor \frac{p}{R} \rfloor$. We then splat all ground truth keypoints onto a heatmap $Y \in [0,1]^{\frac{W}{R} \times \frac{H}{R} \times C}$ using a Gaussian kernel $Y_{xyc} = \exp\left(-\frac{(x-\tilde{p}_x)^2 + (y-\tilde{p}_y)^2}{2\sigma_p^2}\right)$, where σ_p is an object size-adaptive standard deviation [30]. If two Gaussians of the same class overlap, we take the element-wise maximum [4]. The training objective is a penalty-reduced pixel-wise logistic regression with focal loss [33]:

$$L_k = \frac{-1}{N} \sum_{xyc} \begin{cases} (1 - \hat{Y}_{xyc})^{\alpha} \log(\hat{Y}_{xyc}) & \text{if } Y_{xyc} = 1 \\ (1 - Y_{xyc})^{\beta} (\hat{Y}_{xyc})^{\alpha} & \\ \log(1 - \hat{Y}_{xyc}) & \text{otherwise} \end{cases} \quad (1)$$

where α and β are hyper-parameters of the focal loss [33], and N is the number of keypoints in image I. The normalization by N is chosen as to normalize all positive focal loss instances to 1. We use $\alpha = 2$ and $\beta = 4$ in all our experiments, following Law and Deng [30].

To recover the discretization error caused by the output stride, we additionally predict a local offset $\hat{O} \in \mathcal{R}^{\frac{W}{R} \times \frac{H}{R} \times 2}$ for each center point. All classes c share the same offset prediction. The offset is trained with an L1 loss

$$L_{off} = \frac{1}{N} \sum_{p} \left| \hat{O}_{\tilde{p}} - \left(\frac{p}{R} - \tilde{p} \right) \right|. \quad (2)$$

The supervision acts only at keypoints locations $\tilde{p}$, all other locations are ignored.

In the next section, we will show how to extend this keypoint estimator to a general purpose object detector.

4. Objects as Points

Let $(x_1^{(k)}, y_1^{(k)}, x_2^{(k)}, y_2^{(k)})$ be the bounding box of object k with category c_k. Its center point is lies at $p_k = (\frac{x_1^{(k)} + x_2^{(k)}}{2}, \frac{y_1^{(k)} + y_2^{(k)}}{2})$. We use our keypoint estimator $\hat{Y}$ to predict all center points. In addition, we regress to the object size $s_k = (x_2^{(k)} - x_1^{(k)}, y_2^{(k)} - y_1^{(k)})$ for each object k. To limit the computational burden, we use a single size prediction $\hat{S} \in \mathcal{R}^{\frac{W}{R} \times \frac{H}{R} \times 2}$ for all object categories. We use an L1 loss at the center point similar to Objective 2:

$$L_{size} = \frac{1}{N} \sum_{k=1}^{N} \left| \hat{S}_{p_k} - s_k \right|. \quad (3)$$

We do not normalize the scale and directly use the raw pixel coordinates. We instead scale the loss by a constant λ_{size}. The overall training objective is

$$L_{det} = L_k + \lambda_{size} L_{size} + \lambda_{off} L_{off}. \quad (4)$$

We set $\lambda_{size} = 0.1$ and $\lambda_{off} = 1$ in all our experiments unless specified otherwise. We use a single network to predict the keypoints $\hat{Y}$, offset $\hat{O}$, and size $\hat{S}$. The network predicts a total of $C + 4$ outputs at each location. All outputs share a common fully-convolutional backbone network. For each modality, the features of the backbone are then passed through a separate 3×3 convolution, ReLU and another 1×1 convolution. Figure 4 shows an overview of the network output. Section 5 and supplementary material contain additional architectural details.

From points to bounding boxes At inference time, we first extract the peaks in the heatmap for each category independently. We detect all responses whose value is greater or equal to its 8-connected neighbors and keep the top 100 peaks. Let $\hat{\mathcal{P}}_c$ be the set of n detected center points $\hat{\mathcal{P}} = \{(\hat{x}_i, \hat{y}_i)\}_{i=1}^{n}$ of class c. Each keypoint location is given by an integer coordinates (x_i, y_i). We use the keypoint values $\hat{Y}_{x_i y_i c}$ as a measure of its detection confidence, and produce a bounding box at location

$$(\hat{x}_i + \delta\hat{x}_i - \hat{w}_i/2,\ \hat{y}_i + \delta\hat{y}_i - \hat{h}_i/2,$$
$$\hat{x}_i + \delta\hat{x}_i + \hat{w}_i/2,\ \hat{y}_i + \delta\hat{y}_i + \hat{h}_i/2),$$

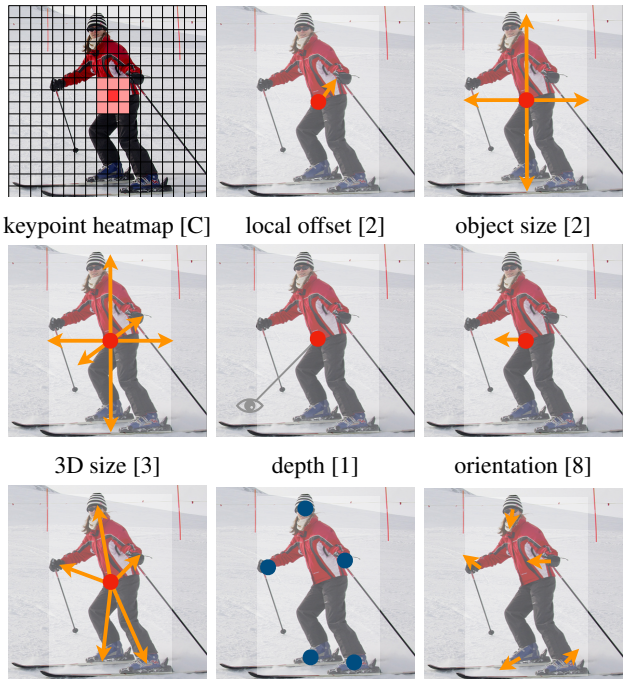


Figure 4: Outputs of our network for different tasks: *top* for object detection, *middle* for 3D object detection, *bottom:* for pose estimation. All modalities are produced from a common backbone, with a different 3×3 and 1×1 output convolutions separated by a ReLU. The number in brackets indicates the output channels. See section 4 for details.

where $(\delta\hat{x}_i, \delta\hat{y}_i) = \hat{O}_{\hat{x}_i,\hat{y}_i}$ is the offset prediction and $(\hat{w}_i, \hat{h}_i) = \hat{S}_{\hat{x}_i,\hat{y}_i}$ is the size prediction. All outputs are produced directly from the keypoint estimation without the need for IoU-based non-maxima suppression (NMS) or other post-processing. The peak keypoint extraction serves as a sufficient NMS alternative and can be implemented efficiently on device using a 3×3 max pooling operation.

4.1. 3D detection

3D detection estimates a three-dimensional bounding box per objects and requires three additional attributes per center point: depth, 3D dimension, and orientation. We add a separate head for each of them. The depth d is a single scalar per center point. However, depth is difficult to regress to directly. We instead use the output transformation of Eigen *et al*. [13] and $d = 1/\sigma(\hat{d}) - 1$, where σ is the sigmoid function. We compute the depth as an additional output channel $\hat{D} \in [0,1]^{\frac{W}{R} \times \frac{H}{R}}$ of our keypoint estimator. It again uses two convolutional layers separated by a ReLU. Unlike previous modalities, it uses the inverse sigmoidal transformation at the output layer. We train the depth estimator using an L1 loss in the original depth domain, after the sigmoidal transformation.

The 3D dimensions of an object are three scalars. We directly regress to their absolute values in meters using a separate head $\hat{\Gamma} \in \mathcal{R}^{\frac{W}{R} \times \frac{H}{R} \times 3}$ and an L1 loss.

Orientation is a single scalar by default. However, it can be hard to regress to. We follow Mousavian *et al*. [38] and represent the orientation as two bins with in-bin regression. Specifically, the orientation is encoded using 8 scalars, with 4 scalars for each bin. For one bin, two scalars are used for softmax classification and the rest two scalar regress to an angle within each bin. Please see the supplementary for details about these losses.

4.2. Human pose estimation

Human pose estimation aims to estimate k 2D human joint locations for every human instance in the image ($k = 17$ for COCO). We considered the pose as a $k \times 2$-dimensional property of the center point, and parametrize each keypoint by an offset to the center point. We directly regress to the joint offsets (in pixels) $\hat{J} \in \mathcal{R}^{\frac{W}{R} \times \frac{H}{R} \times k \times 2}$ with an L1 loss. We ignore the invisible keypoints by masking the loss. This results in a regression-based one-stage multi-person human pose estimator similar to the slow-RCNN version counterparts Toshev *et al*. [51] and Sun *et al*. [49].

To refine the keypoints, we further estimate k human joint heatmaps $\hat{\Phi} \in \mathcal{R}^{\frac{W}{R} \times \frac{H}{R} \times k}$ using standard bottom-up multi-human pose estimation [4,39,41]. We train the human joint heatmap with focal loss and local pixel offset analogous to the center detection discussed in Section. 3.

We then snap our initial predictions to the closest detected keypoint on this heatmap. Here, our center offset acts as a grouping cue, to assign individual keypoint detections to their closest person instance. Specifically, let $(\hat{x}, \hat{y})$ be a detected center point. We first regress to all joint locations $l_j = (\hat{x}, \hat{y}) + \hat{J}_{\hat{x}\hat{y}j}$ for $j \in 1 \ldots k$. We also extract all keypoint locations $L_j = \{\tilde{l}_{ji}\}_{i=1}^{n_j}$ with a confidence > 0.1 for each joint type j from the corresponding heatmap $\hat{\Phi}_{\cdot\cdot j}$. We then assign each regressed location l_j to its closest detected keypoint $\arg\min_{l \in L_j} (l - l_j)^2$ considering only joint detections within the bounding box of the detected object.

5. Implementation details

We experiment with 4 architectures: ResNet-18, ResNet-101 [55], DLA-34 [58], and Hourglass-104 [30]. We modify both ResNets and DLA-34 using deformable convolution layers [12] and use the Hourglass network as is.

Hourglass The stacked Hourglass Network [30, 40] downsamples the input by $4\times$, followed by two sequential hourglass modules. Each hourglass module is a symmetric 5-layer down- and up-convolutional network with skip connections. This network is quite large, but generally yields the best keypoint estimation performance.

	AP			AP_{50}			AP_{75}			Time (ms)			FPS		
	N.A.	F	MS	N.A.	F	MS	N.A.	F	MS	N.A.	F	MS	N.A.	F	MS
Hourglass-104	**40.3**	**42.2**	**45.1**	**59.1**	**61.1**	**63.5**	**44.0**	**46.0**	**49.3**	71	129	672	14	7.8	1.4
DLA-34	37.4	39.2	41.7	55.1	57.0	60.1	40.8	42.7	44.9	19	36	248	52	28	4
ResNet-101	34.6	36.2	39.3	53.0	54.8	58.5	36.9	38.7	42.0	22	40	259	45	25	4
ResNet-18	28.1	30.0	33.2	44.9	47.5	51.5	29.6	31.6	35.1	**7**	**14**	**81**	**142**	**71**	**12**

Table 1: Speed / accuracy trade off for different networks on COCO validation set. We show results without test augmentation (N.A.), flip testing (F), and multi-scale augmentation (MS).

ResNet Xiao et al. [55] augment a standard residual network [22] with three up-convolutional networks to allow for a higher-resolution output (output stride 4). We first change the channels of the three upsampling layers to $256, 128, 64$, respectively, to save computation. We then add one 3×3 deformable convolutional layer before each up-convolution with channel $256, 128, 64$, respectively. The up-convolutional kernels are initialized as bilinear interpolation. See supplement for a detailed architecture diagram.

DLA Deep Layer Aggregation (DLA) [58] is an image classification network with hierarchical skip connections. We utilize the fully convolutional upsampling version of DLA for dense prediction, which uses iterative deep aggregation to increase feature map resolution symmetrically. We augment the skip connections with deformable convolution [63] from lower layers to the output. Specifically, we replace the original convolution with 3×3 deformable convolution at every upsampling layer. See supplement for a detailed architecture diagram.

We add one 3×3 convolutional layer with 256 channel before each output head. A final 1×1 convolution then produces the desired output. We provide more details in the supplementary material.

Training We train on an input resolution of 512×512. This yields an output resolution of 128×128 for all the models. We use random flip, random scaling (between 0.6 to 1.3), cropping, and color jittering as data augmentation, and use Adam [28] to optimize the overall objective. We use no augmentation to train the 3D estimation branch, as cropping or scaling changes the 3D measurements. For the residual networks and DLA-34, we train with a batch-size of 128 (on 8 GPUs) and learning rate 5e-4 for 140 epochs, with learning rate dropped $10\times$ at 90 and 120 epochs, respectively (following [55]). For Hourglass-104, we follow ExtremeNet [61] and use batch-size 29 (on 5 GPUs, with master GPU batch-size 4) and learning rate 2.5e-4 for 50 epochs with $10\times$ learning rate dropped at the 40 epoch. For detection, we fine-tune the Hourglass-104 from ExtremeNet [61] to save computation. The down-sampling layers of Resnet-101 and DLA-34 are initialized with ImageNet pretrain and the up-sampling layers are randomly initialized. Resnet-101 and DLA-34 train in 2.5 days on 8 TITAN-V GPUs, while Hourglass-104 requires 5 days.

Inference We use three levels of test augmentations: no augmentation, flip augmentation, and flip and multi-scale (0.5, 0.75, 1, 1.25, 1.5). For flip, we average the network outputs before decoding bounding boxes. For multi-scale, we use NMS to merge results. These augmentations yield different speed-accuracy trade-off, as is shown in the next section.

6. Experiments

We evaluate our object detection performance on the MS COCO dataset [34], which contains 118k training images (train2017), 5k validation images (val2017) and 20k hold-out testing images (test-dev). We report average precision over all IOU thresholds (AP), AP at IOU thresholds 0.5(AP_{50}) and 0.75 (AP_{75}). The supplement contains additional experiments on PascalVOC [14].

6.1. Object detection

Table 1 shows our results on COCO validation with different backbones and testing options, while Figure 1 compares CenterNet with other real-time detectors. The running time is tested on our local machine, with Intel Core i7-8086K CPU, Titan Xp GPU, Pytorch 0.4.1, CUDA 9.0, and CUDNN 7.1. We download code and pre-trained models[1][2] to test run time for each model on the same machine.

Hourglass-104 achieves the best accuracy at a relatively good speed, with a 42.2% AP in 7.8 FPS. On this backbone, CenterNet outperforms CornerNet [30] (40.6% AP in 4.1 FPS) and ExtremeNet [61](40.3% AP in 3.1 FPS) in both speed and accuracy. The run time improvement comes from fewer output heads and a simpler box decoding scheme. Better accuracy indicates that center points are easier to detect than corners or extreme points.

Using ResNet-101, we outperform RetinaNet [33] with the same network backbone. We only use deformable convolutions in the upsampling layers, which does not affect RetinaNet. We are more than twice as fast at the same accuracy (CenterNet 34.8%AP in 45 FPS (input 512×512)

[1] https://github.com/facebookresearch/Detectron
[2] https://github.com/pjreddie/darknet

	Backbone	FPS	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
MaskRCNN [21]	ResNeXt-101	**11**	39.8	62.3	43.4	22.1	43.2	51.2
Deform-v2 [63]	ResNet-101	-	46.0	67.9	50.8	27.8	49.1	59.5
SNIPER [48]	DPN-98	*2.5*	46.1	67.0	51.6	29.6	48.9	58.1
PANet [35]	ResNeXt-101	-	47.4	67.2	51.8	30.1	**51.7**	60.0
TridentNet [31]	ResNet-101-DCN	0.7	**48.4**	**69.7**	**53.5**	**31.8**	51.3	**60.3**
YOLOv3 [45]	DarkNet-53	20	33.0	57.9	34.4	18.3	25.4	41.9
RetinaNet [33]	ResNeXt-101-FPN	5.4	40.8	61.1	44.1	24.1	44.2	51.2
RefineDet [59]	ResNet-101	-	36.4 / 41.8	57.5 / 62.9	39.5 / 45.7	16.6 / 25.6	39.9 / 45.1	51.4 / 54.1
CornerNet [30]	Hourglass-104	4.1	40.5 / 42.1	56.5 / 57.8	43.1 / 45.3	19.4 / 20.8	42.7 / 44.8	**53.9** / 56.7
ExtremeNet [61]	Hourglass-104	3.1	40.2 / 43.7	55.5 / 60.5	43.2 / 47.0	20.4 / 24.1	43.2 / 46.9	53.1 / 57.6
FSAF [62]	ResNeXt-101	*2.7*	**42.9** / 44.6	**63.8** / **65.2**	**46.3** / 48.6	**26.6** / **29.7**	**46.2** / **47.1**	52.7 / 54.6
CenterNet-DLA	DLA-34	**28**	39.2 / 41.6	57.1 / 60.3	42.8 / 45.1	19.9 / 21.5	43.0 / 43.9	51.4 / 56.0
CenterNet-HG	Hourglass-104	7.8	42.1 / **45.1**	61.1 / 63.9	45.9 / **49.3**	24.1 / 26.6	45.5 / **47.1**	52.8 / **57.7**

Table 2: State-of-the-art comparison on COCO test-dev. Top: two-stage detectors; bottom: one-stage detectors. We show single-scale / multi-scale testing for most one-stage detectors. Frame-per-second (FPS) were measured on the same machine whenever possible. Italic FPS highlight the cases, where the performance measure was copied from the original publication. A dash indicates methods for which neither code and models, nor public timings were available.

vs. RetinaNet 34.4%AP in 18 FPS (input 500×800)). Our fastest ResNet-18 model also achieves a respectable performance of 28.1% COCO AP at 142 FPS.

DLA-34 gives the best speed/accuracy trade-off. It runs at 52FPS with 37.4%AP. This is more than twice as fast as YOLOv3 [45] and 4.4%AP more accurate. With flip testing, our model is still faster than YOLOv3 [45] and achieves accuracy levels of Faster-RCNN-FPN [46] (CenterNet 39.2% AP in 28 FPS vs Faster-RCNN 39.8% AP in 11 FPS).

State-of-the-art comparison We compare with other state-of-the-art detectors in COCO test-dev in Table 2. With multi-scale evaluation, CenterNet with Hourglass-104 achieves an AP of 45.1%, outperforming all existing one-stage detectors. Sophisticated two-stage detectors [31,35,48,63] are more accurate, but also slower. There is no significant difference between CenterNet and sliding window detectors for different object sizes or IoU thresholds. CenterNet behaves like a regular detector, just faster.

6.1.1 Additional experiments

In unlucky circumstances, two different objects might share the same center, if they perfectly align. In this scenario, CenterNet would only detect one of them. We start by studying how often this happens in practice and put it in relation to missing detections of competing methods.

Center point collision In the COCO training set, there are 614 pairs of objects that collide onto the same center point at stride 4. There are 860001 objects in total, hence CenterNet is unable to predict $< 0.1\%$ of objects due to collisions in center points. This is much less than slow- or fast-RCNN miss due to imperfect region proposals [52] ($\sim 2\%$), and fewer than anchor-based methods miss due to insufficient anchor placement [46] (20.0% for Faster-RCNN with 15 anchors at 0.5 IOU threshold). In addition, 715 pairs of objects have bounding box IoU > 0.7 and would be assigned to two anchors, hence a center-based assignment causes fewer collisions.

NMS To verify that IoU based NMS is not needed for CenterNet, we ran it as a post-processing step on our predictions. For DLA-34 (flip-test), the AP improves from 39.2% to 39.7%. For Hourglass-104, the AP stays at 42.2%. Given the minor impact, we do not use it.

Next, we ablate the new hyperparameters of our model. All the experiments are done on DLA-34.

Training and Testing resolution During training, we fix the input resolution to 512×512. During testing, we follow CornerNet [30] to keep the original image resolution and zero-pad the input to the maximum stride of the network. For ResNet and DLA, we pad the image with up to 32 pixels, for HourglassNet, we use 128 pixels. As is shown in Table. 3a, keeping the original resolution is slightly better than fixing test resolution. Training and testing in a lower resolution (384×384) runs 1.7 times faster but drops 3AP.

Regression loss We compare a vanilla L1 loss to a Smooth L1 [18] for size regression. Our experiments in Table 3c show that L1 is considerably better than Smooth L1.

Resolution	AP	AP_{50}	AP_{75}	Time
Original	**36.3**	54.0	**39.6**	19
512	36.2	**54.3**	38.7	16
384	33.2	50.5	35.0	**11**

(a) Testing resolution: Lager resolutions perform better but run slower.

λ_{size}	AP	AP_{50}	AP_{75}
0.2	33.5	49.9	36.2
0.1	**36.3**	54.0	**39.6**
0.02	35.4	**54.6**	37.9

(b) Size regression weight. $\lambda_{size} \leq 0.1$ yields good results.

Loss	AP	AP_{50}	AP_{75}
l1	**36.3**	**54.0**	**39.6**
smooth l1	33.9	50.9	36.8

(c) Regression loss. L1 loss works better than Smooth L1.

Epoch	AP	AP_{50}	AP_{75}
140	36.3	54.0	39.6
230	**37.4**	**55.1**	**40.8**

(d) Training schedule. Longer performs better.

Table 3: Ablation of design choices on COCO validation set. The results are shown in COCO AP, time in milliseconds.

It yields a better accuracy at fine-scale, which the COCO evaluation metric is sensitive to. This is independently observed in keypoint regression [49, 50].

Bounding box size weight We analyze the sensitivity of our approach to the loss weight λ_{size}. Table 3b shows 0.1 gives a good result. For larger values, the AP degrades significantly, due to the scale of the loss ranging from 0 to output size w/R or h/R, instead of 0 to 1. However, the value does not degrade significantly for lower weights.

Training schedule By default, we train the keypoint estimation network for 140 epochs with a learning rate drop at 90 epochs. If we double the training epochs before dropping the learning rate, the performance further increases by 1.1 AP (Table 3d), at the cost of a much longer training schedule. To save computational resources (and polar bears), we use 140 epochs in ablation experiments, but stick with 230 epochs for DLA when comparing to other methods.

Finally, we tried a multiple "anchor" version of CenterNet by regressing to more than one object size. The experiments did not yield any success. See supplement.

6.2. 3D detection

We perform 3D bounding box estimation experiments on KITTI dataset [17], which contains carefully annotated 3D bounding box for vehicles in a driving scenario. KITTI contains 7841 training images and we follow standard training and validation splits in literature [10, 54]. The evaluation metric is the average precision for cars at 11 recalls (0.0 to 1.0 with 0.1 increment) at IOU threshold 0.5, as in object detection [14]. We evaluate IOUs based on 2D bounding box (AP), orientation (AOP), and Bird-eye-view bounding box (BEV AP). We keep the original image resolution and pad to 1280×384 for both training and testing. The training converges in 70 epochs, with learning rate dropped at the 45 and 60 epoch, respectively. We use the DLA-34 backbone and set the loss weight for depth, orientation, and dimension to 1. All other hyper-parameters are the same as the detection experiments.

Since the number of recall thresholds is quite small, the validation AP fluctuates by up to 10% AP. We thus train 5 models and report the average with standard deviation.

We compare with slow-RCNN based Deep3DBox [38] and Faster-RCNN based method Mono3D [9], on their specific validation split. As is shown in Table 4, our method performs on-par with its counterparts in AP and AOS and does slightly better in BEV. Our CenterNet is two orders of magnitude faster than both methods.

6.3. Pose estimation

Finally, we evaluate CenterNet on human pose estimation in the MS COCO dataset [34]. We evaluate keypoint AP, which is similar to bounding box AP but replaces the bounding box IoU with object keypoint similarity. We test and compare with other methods on COCO test-dev.

We experiment with DLA-34 and Hourglass-104, both fine-tuned from center point detection. DLA-34 converges in 320 epochs (about 3 days on 8GPUs) and Hourglass-104 converges in 150 epochs (8 days on 5 GPUs). All additional loss weights are set to 1. All other hyper-parameters are the same as object detection.

The results are shown in Table 5. Direct regression to keypoints performs reasonably, but not at state-of-the-art. It struggles particularly in high IoU regimes. Projecting our output to the closest joint detection improves the results throughout, and performs competitively with state-of-the-art multi-person pose estimators [4, 21, 39, 41]. This verifies that CenterNet is general, easy to adapt to a new task.

Figure 5 shows qualitative examples on all tasks.

	AP			AOS			BEV AP		
	Easy	Mode	Hard	Easy	Mode	Hard	Easy	Mode	Hard
Deep3DBox [38]	98.8	97.2	81.2	98.6	96.7	80.5	30.0	23.7	18.8
Ours	90.2±1.2	80.4±1.4	71.1±1.6	85.3±1.7	75.0±1.6	66.2±1.8	31.4±3.7	26.5±1.6	23.8±2.9
Mono3D [9]	95.8	90.0	80.6	93.7	87.6	78.0	30.5	22.4	19.1
Ours	97.1±0.3	87.9±0.1	79.3±0.1	93.4±0.7	83.9±0.5	75.3±0.4	31.5±2.0	29.7±0.7	28.1±4.6

Table 4: KITTI evaluation. We show 2D bounding box AP, average orientation score (AOS), and bird eye view (BEV) AP on different validation splits. Higher is better.

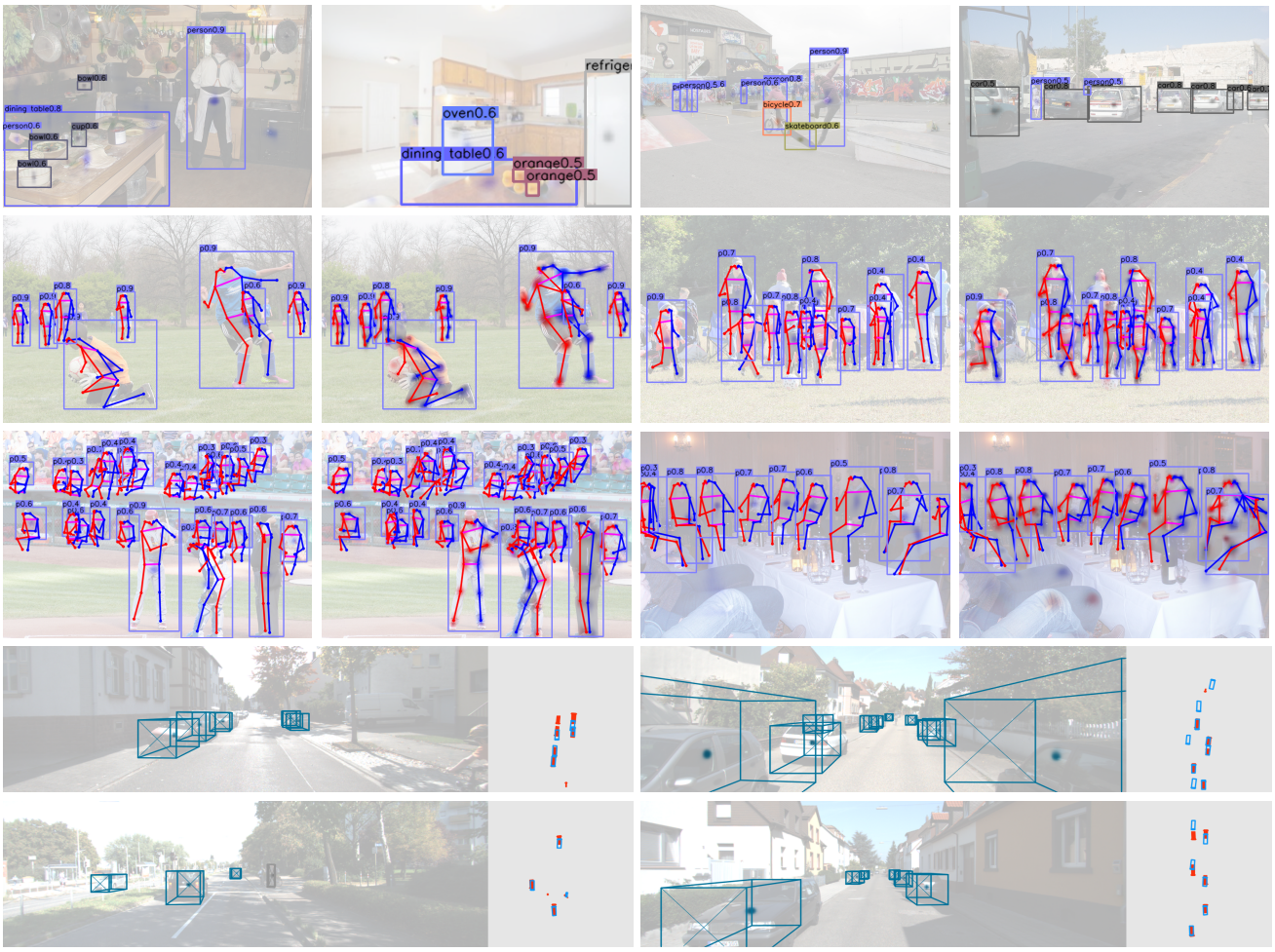


Figure 5: Qualitative results. All images were picked thematically without considering our algorithms performance. *First row:* object detection on COCO validation. *Second and third row:* Human pose estimation on COCO validation. For each pair, we show the results of center offset regression (left) and heatmap matching (right). *fourth and fifth row:* 3D bounding box estimation on KITTI validation. We show projected bounding box (left) and bird eye view map (right). The ground truth detections are shown in solid red solid box. The center heatmap and 3D boxes are shown overlaid on the original image.

	AP^{kp}	AP^{kp}_{50}	AP^{kp}_{75}	AP^{kp}_{M}	AP^{kp}_{L}
CMU-Pose [4]	61.8	84.9	67.5	58.0	70.4
Pose-AE [39]	62.8	84.6	69.2	57.5	70.6
Mask-RCNN [21]	63.1	87.3	68.7	57.8	71.4
PersonLab [41]	66.5	85.5	71.3	62.3	70.0
DLA-reg	51.7	81.4	55.2	44.6	63.0
HG-reg	55.0	83.5	59.7	49.4	64.0
DLA-jd	57.9	84.7	63.1	52.5	67.4
HG-jd	63.0	86.8	69.6	58.9	70.4

Table 5: Keypoint detection on COCO test-dev. -reg/ -jd are for direct center-out offset regression and matching regression to the closest joint detection, respectively. The results are shown in COCO keypoint AP. Higher is better.

7. Conclusion

In summary, we present a new representation for objects: as points. Our CenterNet object detector builds on successful keypoint estimation networks, finds object centers, and regresses to their size. The algorithm is simple, fast, accurate, and end-to-end differentiable without any NMS post-processing. The idea is general and has broad applications beyond simple two-dimensional detection. CenterNet can estimate a range of additional object properties, such as pose, 3D orientation, depth and extent, in one single forward pass. Our initial experiments are encouraging and open up a new direction for real-time object recognition and related tasks.

References

[1] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Lawrence Zitnick, and D. Parikh. Vqa: Visual question answering. In *ICCV*, 2015.

[2] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis. Soft-nmsimproving object detection with one line of code. In *ICCV*, 2017.

[3] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. In *arXiv preprint arXiv:1812.08008*, 2018.

[4] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, 2017.

[5] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017.

[6] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teuliere, and T. Chateau. Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image. In *CVPR*, 2017.

[7] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 2018.

[8] X. Chen and A. Gupta. An implementation of faster rcnn with study for region sampling. *arXiv preprint arXiv:1702.02138*, 2017.

[9] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun. Monocular 3d object detection for autonomous driving. In *CVPR*, 2016.

[10] X. Chen, K. Kundu, Y. Zhu, A. G. Berneshawi, H. Ma, S. Fidler, and R. Urtasun. 3d object proposals for accurate object class detection. In *NIPS*, 2015.

[11] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. In *NIPS*, 2016.

[12] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. In *ICCV*, 2017.

[13] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NIPS*, 2014.

[14] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results.

[15] H.-S. Fang, S. Xie, Y.-W. Tai, and C. Lu. RMPE: Regional multi-person pose estimation. In *ICCV*, 2017.

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

[17] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012.

[18] R. Girshick. Fast r-cnn. In *ICCV*, 2015.

[19] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[20] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[21] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *ICCV*, 2017.

[22] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[23] J. H. Hosang, R. Benenson, and B. Schiele. Learning non-maximum suppression. In *CVPR*, 2017.

[24] H.-N. Hu, Q.-Z. Cai, D. Wang, J. Lin, M. Sun, P. Krähenbühl, T. Darrell, and F. Yu. Joint monocular 3d vehicle detection and tracking. *arXiv preprint arXiv:1811.10742*, 2018.

[25] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *CVPR*, 2017.

[26] B. Jiang, R. Luo, J. Mao, T. Xiao, and Y. Jiang. Acquisition of localization confidence for accurate object detection. In *ECCV*, 2018.

[27] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *TPAMI*, 2012.

[28] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *ICLR*, 2014.

[29] A. Kundu, Y. Li, and J. M. Rehg. 3d-rcnn: Instance-level 3d object reconstruction via render-and-compare. In *CVPR*, 2018.

[30] H. Law and J. Deng. Cornernet: Detecting objects as paired keypoints. In *ECCV*, 2018.

[31] Y. Li, Y. Chen, N. Wang, and Z. Zhang. Scale-aware trident networks for object detection. *arXiv preprint arXiv:1901.01892*, 2019.

[32] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei. Fully convolutional instance-aware semantic segmentation. In *CVPR*, 2017.

[33] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *ICCV*, 2017.

[34] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[35] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia. Path aggregation network for instance segmentation. In *CVPR*, 2018.

[36] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016.

[37] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[38] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka. 3d bounding box estimation using deep learning and geometry. In *CVPR*, 2017.

[39] A. Newell, Z. Huang, and J. Deng. Associative embedding: End-to-end learning for joint detection and grouping. In *NIPS*, 2017.

[40] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016.

[41] G. Papandreou, T. Zhu, L.-C. Chen, S. Gidaris, J. Tompson, and K. Murphy. Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model. *ECCV*, 2018.

[42] G. Papandreou, T. Zhu, N. Kanazawa, A. Toshev, J. Tompson, C. Bregler, and K. Murphy. Towards accurate multi-person pose estimation in the wild. In *CVPR*, 2017.

[43] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016.

[44] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. *CVPR*, 2017.

[45] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[46] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[47] B. Singh and L. S. Davis. An analysis of scale invariance in object detection–snip. In *CVPR*, 2018.

[48] B. Singh, M. Najibi, and L. S. Davis. SNIPER: Efficient multi-scale training. *NIPS*, 2018.

[49] X. Sun, J. Shang, S. Liang, and Y. Wei. Compositional human pose regression. In *ICCV*, 2017.

[50] X. Sun, B. Xiao, F. Wei, S. Liang, and Y. Wei. Integral human pose regression. In *ECCV*, 2018.

[51] A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. In *CVPR*, 2014.

[52] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *IJCV*, 2013.

[53] D. Wang, C. Devin, Q.-Z. Cai, F. Yu, and T. Darrell. Deep object centric policies for autonomous driving. In *ICRA*, 2019.

[54] Y. Xiang, W. Choi, Y. Lin, and S. Savarese. Subcategory-aware convolutional neural networks for object proposals and detection. In *WACV*, 2017.

[55] B. Xiao, H. Wu, and Y. Wei. Simple baselines for human pose estimation and tracking. In *ECCV*, 2018.

[56] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

[57] J. Xu, R. Zhao, F. Zhu, H. Wang, and W. Ouyang. Attention-aware compositional network for person re-identification. In *CVPR*, 2018.

[58] F. Yu, D. Wang, E. Shelhamer, and T. Darrell. Deep layer aggregation. In *CVPR*, 2018.

[59] S. Zhang, L. Wen, X. Bian, Z. Lei, and S. Z. Li. Single-shot refinement neural network for object detection. In *CVPR*, 2018.

[60] X. Zhou, A. Karpur, L. Luo, and Q. Huang. Starmap for category-agnostic keypoint and viewpoint estimation. In *ECCV*, 2018.

[61] X. Zhou, J. Zhuo, and P. Krähenbühl. Bottom-up object detection by grouping extreme and center points. In *CVPR*, 2019.

[62] C. Zhu, Y. He, and M. Savvides. Feature selective anchor-free module for single-shot object detection. *arXiv preprint arXiv:1903.00621*, 2019.

[63] X. Zhu, H. Hu, S. Lin, and J. Dai. Deformable convnets v2: More deformable, better results. *arXiv preprint arXiv:1811.11168*, 2018.

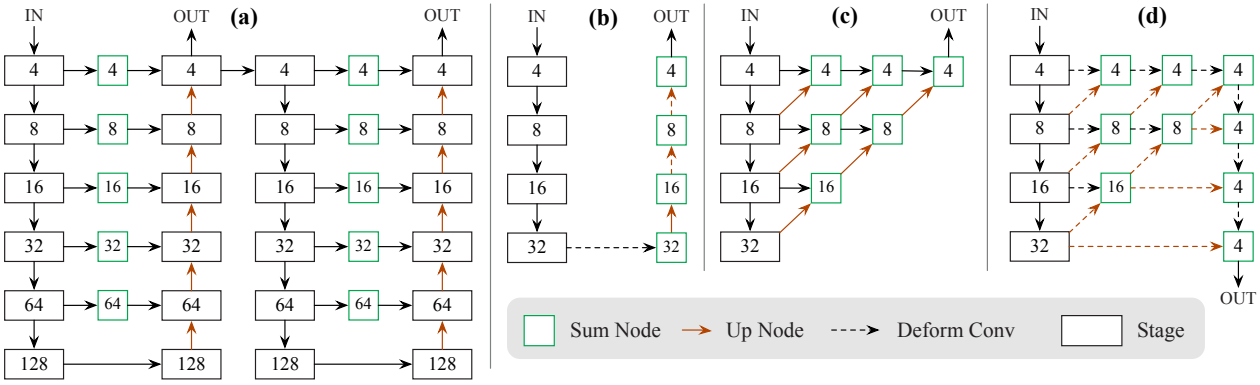


Figure 6: Model diagrams. The numbers in the boxes represent the stride to the image. (a): Hourglass Network [30]. We use it as is in CornerNet [30]. (b): ResNet with transpose convolutions [55]. We add one 3×3 deformable convolutional layer [63] before each up-sampling layer. Specifically, we first use deformable convolution to change the channels and then use transposed convolution to upsample the feature map (such two steps are shown separately in $32 \rightarrow 16$. We show these two steps together as a dashed arrow for $16 \rightarrow 8$ and $8 \rightarrow 4$). (c): The original DLA-34 [58] for semantic segmentation. (d): Our modified DLA-34. We add more skip connections from the bottom layers and upgrade every convolutional layer in upsampling stages to deformable convolutional layer.

Appendix A: Model Architecture

See figure. 6 for diagrams of the architectures.

Appendix B: 3D BBox Estimation Details

Our network outputs maps for depths $\hat{D} \in R^{\frac{W}{R} \times \frac{H}{R}}$, 3d dimensions $\hat{\Gamma} \in R^{\frac{W}{R} \times \frac{H}{R} \times 3}$, and orientation encoding $\hat{A} \in R^{\frac{W}{R} \times \frac{H}{R} \times 8}$. For each object instance k, we extract the output values from the three output maps at the ground truth center point location: $\hat{d}_k \in R$, $\hat{\gamma}_k \in R^3$, $\hat{\alpha}_k \in R^8$. The depth is trained with L1 loss after converting the output to the absolute depth domain:

$$L_{dep} = \frac{1}{N} \sum_{k=1}^{N} |\frac{1}{\sigma(\hat{d}_k)} - 1 - d_k| \tag{5}$$

where d_k is the groud truth absolute depth (in meter). Similarly, the 3D dimension is trained with L1 Loss in absolute metric:

$$L_{dim} = \frac{1}{N} \sum_{k=1}^{N} |\hat{\gamma}_k - \gamma_k| \tag{6}$$

where γ_k is the object height, width, and length in meter.

The orientation θ is a single scalar by default. Following Mousavian *et al.* [24, 38], We use an 8-scalar encoding to ease learning. The 8 scalars are divided into two groups, each for an angular bin. One bin is for angles in $B_1 = [-\frac{7\pi}{6}, \frac{\pi}{6}]$ and the other is for angles in $B_2 = [-\frac{\pi}{6}, \frac{7\pi}{6}]$. Thus we have 4 scalars for each bin. Within each bin, 2 of the scalars $b_i \in R^2$ are used for softmax classification (if the orientation falls into to this bin i). And the rest 2 scalars $a_i \in R^2$ are for the sin and cos value of in-bin offset (to the bin center m_i). I.e., $\hat{\alpha} = [\hat{b}_1, \hat{a}_1, \hat{b}_2, \hat{a}_2]$ The classification are trained with softmax and the angular values are trained with L1 loss:

$$L_{ori} = \frac{1}{N} \sum_{k=1}^{N} \sum_{i=1}^{2} (softmax(\hat{b}_i, c_i) + c_i |\hat{a}_i - a_i|) \tag{7}$$

where $c_i = \mathbb{1}(\theta \in B_i)$, $a_i = (\sin(\theta - m_i), \cos(\theta - m_i))$. $\mathbb{1}$ is the indicator function. The predicted orientation θ is decoded from the 8-scalar encoding by

$$\hat{\theta} = arctan2(\hat{a}_{j1}, \hat{a}_{j2}) + m_j \tag{8}$$

where j is the bin index which has a larger classification score.

Appendix C: Collision Experiment Details

We analysis the annotations of COCO training set to show how often the collision cases happen. COCO training set (train 2017) contains $N = 118287$ images and $M = 860001$ objects (with $M_S = 356340$ small objects, $M_M = 295163$ medium objects, and $M_L = 208498$ large objects) in $C = 80$ categories. Let the i-th bounding box of image k of category c be $bb^{(kci)} = (x_1^{(kci)}, y_1^{(kci)}, x_2^{(kci)}, y_2^{(kci)})$, its center after the $4\times$ stride is $p^{kci} = (\lfloor \frac{1}{4} \cdot \frac{x_1^{(kci)} + x_2^{(kci)}}{2} \rfloor, \lfloor \frac{1}{4} \cdot \frac{y_1^{(kci)} + y_2^{(kci)}}{2} \rfloor)$. And Let $n^{(kc)}$ be the number of object of category c in image k. The number of center point collisions is calculated by:

$$N_{center} = \sum_{k=1}^{N} \sum_{c=1}^{C} \sum_{i=1}^{n^{(kc)}} \sum_{j=i+1}^{n^{(kc)}} \mathbb{1}(p^{kci} = p^{kcj}) \tag{9}$$

We get $N_{center} = 614$ on the dataset.

Similarly, we calculate the IoU based collision by

$$N_{IoU@t} = \sum_{k=1}^{N} \sum_{c=1}^{C} \sum_{i=1}^{n^{(kc)}} \sum_{j=i+1}^{n^{(kc)}} \mathbb{1}(IoU(bb^{(kci)}, bb^{(kcj)}) > t) \tag{10}$$

This gives $N_{IoU@0.7} = 715$ and $N_{IoU@0.5} = 5179$.

Missed objects in anchor based detector. RetinaNet [33] assigns anchors to a ground truth bounding box if they have > 0.5 IoU. In the case that a ground truth bounding box has not been covered by any anchor with IoU > 0.5, the anchor with the largest IoU will be assigned to it. We calculate how often this forced assignment happens. We use 15 anchors (5 size: 32, 64, 128, 256, 512, and 3 aspect-ratio: 0.5, 1, 2, as is in RetinaNet [33]) at stride $S = 16$. For each image, after resizing it as its shorter edge to be 800 [33], we place these anchors at positions $\{(S/2 + i \times S, S/2 + j \times S)\}$, where $i \in [0, \lfloor \frac{(W-S/2)}{S} \rfloor]$ and $j \in [0, \lfloor \frac{(H-S/2)}{S} \rfloor]$. W, H are the image weight and height (the smaller one is equal to 800). This results in a set of anchors $\mathcal{A}$. $|\mathcal{A}| = 15 \times \lfloor \frac{(W-S/2)}{S} + 1 \rfloor \times \lfloor \frac{(H-S/2)}{S} + 1 \rfloor$. We calculate the number of the forced assignments by:

$$N_{anchor} = \sum_{k=1}^{N} \sum_{i=1}^{n^{(k)}} \mathbb{1}((\max_{A \in \mathcal{A}} IoU(bb^{(k \cdot i)}, A)) < 0.5) \tag{11}$$

RenitaNet requires $N_{anchor} = 170220$ forced assignments: 125831 for small objects (35.3% of all small objects), 18505 for medium objects (6.3% of all medium objects), and 25884 for large objects (12.4% of all large objects).

Appendix D: Experiments on PascalVOC

Pascal VOC [14] is a popular small object detection dataset. We train on VOC 2007 and VOC 2012 trainval sets, and test on VOC 2007 test set. It contains 16551 training images and 4962 testing images of 20 categories. The evaluation metric is mean average precision (mAP) at IOU threshold 0.5.

We experiment with our modified ResNet-18, ResNet-101, and DLA-34 (See main paper Section. 5) in two training resolution: 384×384 and 512×512. For all networks, we train 70 epochs with learning rate dropped $10\times$ at 45 and 60 epochs, respectively. We use batchsize 32 and learning rate $1.25e$-4 following the linear learning rate rule [20]. It takes one GPU 7 hours/ 10 hours to train in 384×384 for ResNet-101 and DLA-34, respectively. And for 512×512, the training takes the same time in two GPUs. Flip augmentation is used in testing. All other hyper-parameters are the same as the COCO experiments. We do not use Hourglass-104 [30] because it fails to converge in a reasonable time (2 days) when trained from scratch.

	Resolution	mAP@0.5	FPS
Faster RCNN [46]	600×1000	76.4	5
Faster RCNN* [8]	600×1000	79.8	5
R-FCN [11]	600×1000	80.5	9
Yolov2 [44]	544×544	78.6	40
SSD [16]	513×513	78.9	19
DSSD [16]	513×513	81.5	5.5
RefineDet [59]	512×512	81.8	24
CenterNet-Res18	384×384	72.6	142
CenterNet-Res18	512×512	75.7	100
CenterNet-Res101	384×384	77.6	45
CenterNet-Res101	512×512	78.7	30
CenterNet-DLA	384×384	79.3	50
CenterNet-DLA	512×512	80.7	33

Table 6: Experimental results on Pascal VOC 2007 test. The results are shown in mAP@0.5. Flip test is used for CenterNet. The FPSs for other methods are copied from the original publications.

	AP	AP_{50}	AP_{75}
	36.3	54.0	39.6
w/ gt size	41.9	56.6	45.4
w/ gt heatmap	54.2	82.6	58.1
w/ gt heatmap+size	83.1	97.9	90.1
w/ gt hm.+size+offset	99.5	99.7	99.6

Table 7: Error analysis on COCO validation. We show COCO AP(%) after replacing each network prediction with its ground truth.

The results are shown in Table. 6. Our best CenterNet-DLA model performs competitively with top-tier methods, and keeps a real-time speed.

Appendix E: Error Analysis

We perform an error analysis by replacing each output head with its ground truth. For the center point heatmap, we use the rendered Gaussian ground truth heatmap. For the bounding box size, we use the nearest ground truth size for each detection.

The results in Table 7 show that improving both size map leads to a modest performance gain, while the center map gains are much larger. If only the keypoint offset is not predicted, the maximum AP reaches 83.1. The entire pipeline on ground truth misses about 0.5% of objects, due to discretization and estimation errors in the Gaussian heatmap rendering.