Cornernet&Centernet

12201668 현민주

Cornernet?

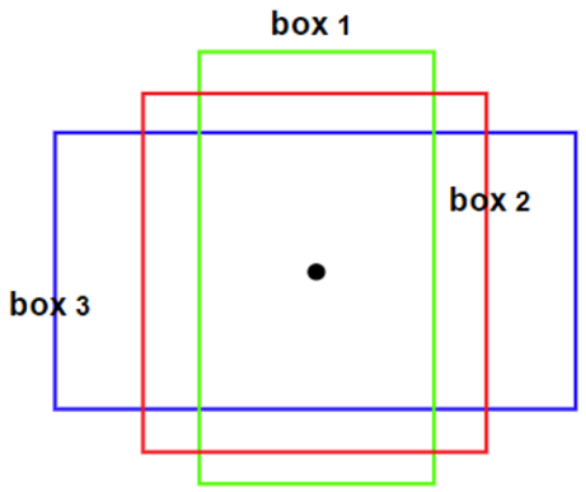
- 2018년에 나온 CornerNet: Detecting Objects as Paired Keypoints(L.Hei) 논문에서 소개된 one-stage detector

- 이전 Detection과 다르게 Anchor box를 사용하지 않는다는 것이 주목할 점

- 추가로 모서리를 지역화(Localize)하기 위해 Corner pooling을 추가함

Simple review : Anchor box

- Anchor box: 특정 높이와 너비의 미리 정의된 경계 상자 세트



- Highly competitive with two-stage detectors

-모든 객체 예측을 한 번에 평가할 수 있다.

-학습 알고리즘을 전문화,특정 물체 감지에 특화.

-겹쳐진 물체 분류에 탁월

- Positive/negative anchor box:

-IoU값이 임계치를 넘으면 positive, 아니면 negative

(나누는 방법)https://joochann.github.io/posts/Learning-from-Noisy-Anchors-for-One-stage-Object-Detection/

Defect of Anchor Box

- Need large set of anchor boxes

DSSD: 40k 이상

RetinaNet: 100k 이상

→huge imbalance between positive & negative anchor box

→학습속도 느려짐

- Many hyperparameters and design choice를 요구

-how many boxed?

-what size?

-what aspect ratios?

-etc..

→more complicated when combined with multiscale architectures

Solution

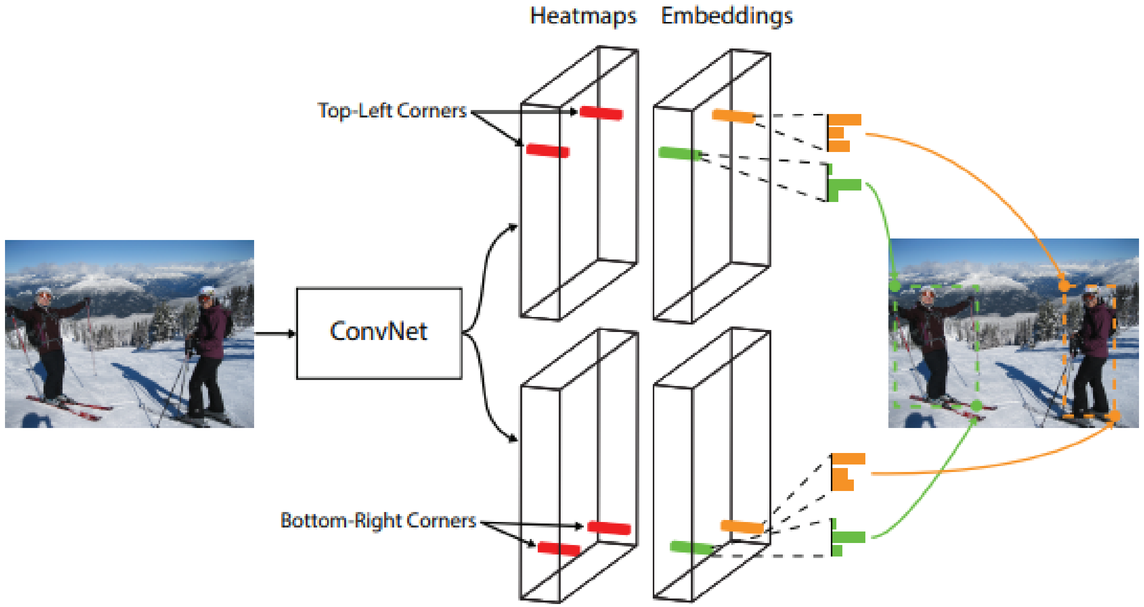
- Single convolution network

-Top-left corners

-bottom-right corners

-embeddings vector for each corners

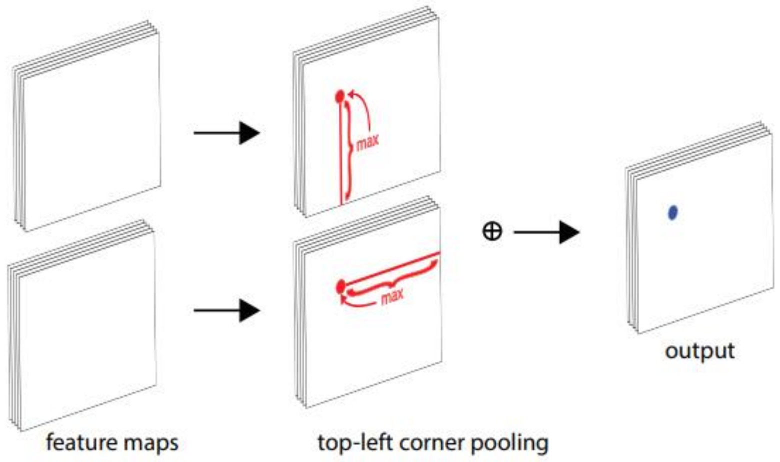
Embedding: serve to group a pair of corners



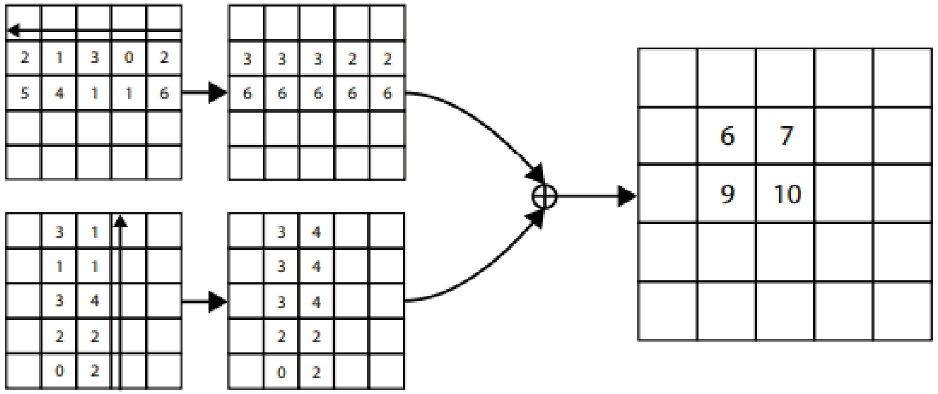
- Simplifies the output of the network
- Eliminate the need for designing anchor boxes

Corner Pooling

- Help a convolutional network better "localize" corners of bounding boxes



Corner Pooling



Corner가 위치한 곳에는 실제로 local feature가 없는 경우가 많다.

따라서 수직, 수평 방향으로 물체의 외곽이 있는지 봐야 해서 사용.

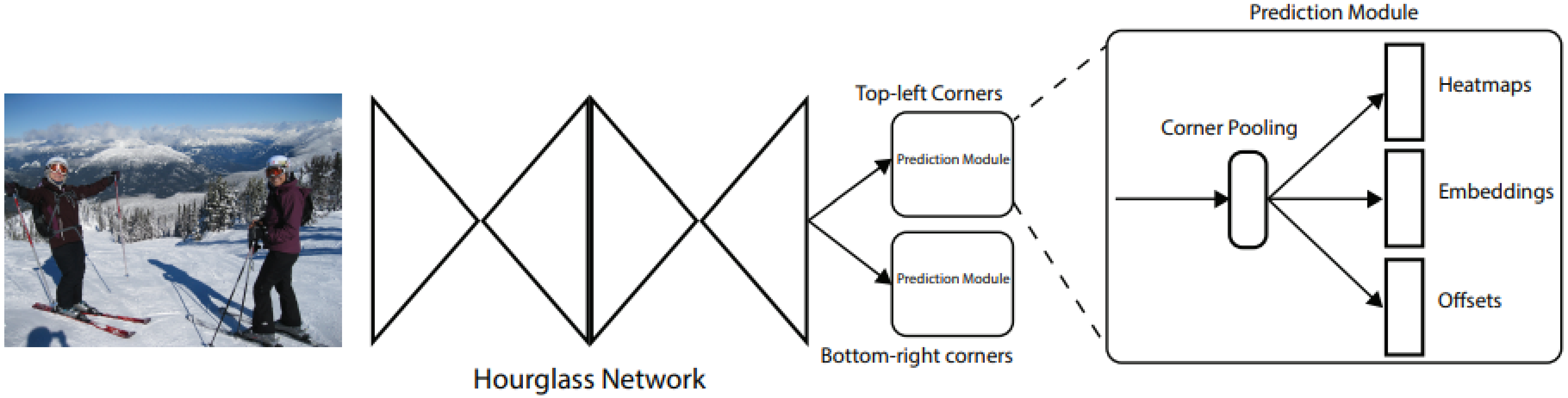
Vertical, Horizontal 두 방향에 대해서 max pooling을 진행

(0, j)와 (i, j) 사이의 값을 pooling 해주고, (i, 0)와 (i, j) 사이의 값을 pooling 해준다.

$$t_{ij} = \begin{cases} \max\left(f_{t_{ij}}, t_{(i+1)j}\right) & \text{if } i < H \\ f_{t_{Hj}} & \text{otherwise} \end{cases} \tag{6}$$

$$l_{ij} = \begin{cases} \max\left(f_{l_{ij}}, l_{i(j+1)}\right) & \text{if } j < W \\ f_{l_{iW}} & \text{otherwise} \end{cases} \tag{7}$$

Detecting Corners



- one for top-left corners and one for bottom-right corners을 예측
-각 heatmap은 C(카테고리의 수) channels를 가짐

- Each corner, one ground-truth positive locations, and all other locations are negative

- reduce the penalty given to negative locations within a radius of the positive location
→ 어느 정도 근접한 예측을 한 것이기 때문

Corner location Loss(focal loss/heatmap)

$$L_{det} = \frac{-1}{N} \sum_{c=1}^{C} \sum_{i=1}^{H} \sum_{j=1}^{W} \begin{cases} (1 - p_{cij})^{\alpha} \log(p_{cij}) & \text{if } y_{cij} = 1 \\ (1 - y_{cij})^{\beta} (p_{cij})^{\alpha} \log(1 - p_{cij}) & \text{otherwise} \end{cases} \quad (1)$$

- P_{cij}: class c의 location (i, j)에 대한 score
- y_{cij} : ground-truth

Offsets

$$\boldsymbol{o}_k = \left(\frac{x_k}{n} - \left\lfloor \frac{x_k}{n} \right\rfloor, \frac{y_k}{n} - \left\lfloor \frac{y_k}{n} \right\rfloor \right)$$

$$L_{off} = \frac{1}{N} \sum_{k=1}^{N} \text{SmoothL1Loss}(\boldsymbol{o}_k, \hat{\boldsymbol{o}}_k)$$

- o_k: offset
- x_k, y_k: coordinate for corner k

Grouping Corner(Embedding)

- 검출된 Corner들 중 쌍이 맞는 top-left, bottom-right을 embedding을 통해 묶어준다.
- Corner를 기반으로 같은 bbox를 만드는 corner 쌍이면 embedding의 차가 많이 나지 않는 것을 이용하여 grouping
- 같은 group의 corner에는 pull loss, 다른 group에는 push loss를 사용한다.

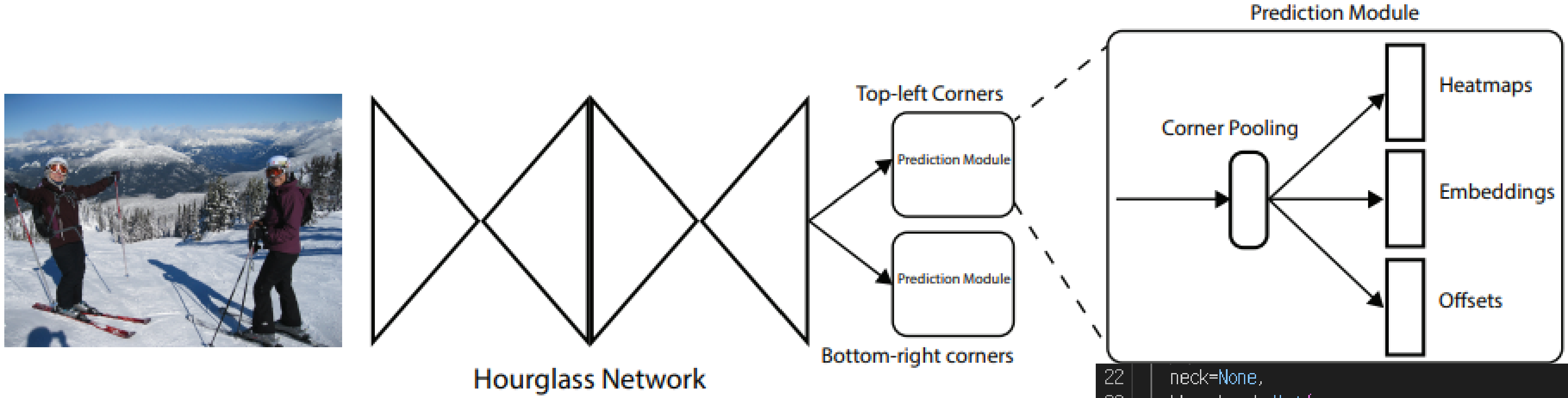
$$L_{pull} = \frac{1}{N}\sum_{k=1}^{N}\left[(e_{t_k} - e_k)^2 + (e_{b_k} - e_k)^2\right],$$

- Corner를 사용하면 두 면의 위치만 알면 돼서 더 쉬움
- Box를 표현하는데 더 효율적인 방법 O(w^2h^2) → O(wh)

$$L_{push} = \frac{1}{N(N-1)}\sum_{k=1}^{N}\sum_{\substack{j=1\\ j\neq k}}^{N}\max\left(0, \Delta - |e_k - e_j|\right),$$

Result

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	-	-	-	-	-	-
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




```python
# model settings
model = dict(
    type='CornerNet',
    data_preprocessor=data_preprocessor,
    backbone=dict(
        type='HourglassNet',
        downsample_times=5,
        num_stacks=2,
        stage_channels=[256, 256, 384, 384, 384, 512],
        stage_blocks=[2, 2, 2, 2, 2, 4],
        norm_cfg=dict(type='BN', requires_grad=True)),
```

```python
    neck=None,
    bbox_head=dict(
        type='CornerHead',
        num_classes=80,
        in_channels=256,
        num_feat_levels=2,
        corner_emb_channels=1,
        loss_heatmap=dict(
            type='GaussianFocalLoss', alpha=2.0, gamma=4.0, loss_weight=1),
        loss_embedding=dict(
            type='AssociativeEmbeddingLoss',
            pull_weight=0.10,
            push_weight=0.10),
        loss_offset=dict(type='SmoothL1Loss', beta=1.0, loss_weight=1)),
```

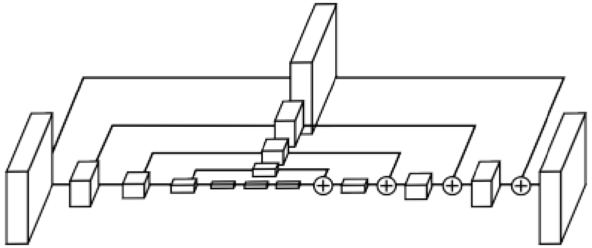


Fig. 3. An illustration of a single "hourglass" module. Each box in the fig
sponds to a residual module as seen in Figure 4. The number of features is c
across the whole hourglass.

```python
def forward(self, x: torch.Tensor) -> List[torch.Tensor]:
    """Forward function."""
    inter_feat = self.stem(x)
    out_feats = []

    for ind in range(self.num_stacks):
        single_hourglass = self.hourglass_modules[ind]
        out_conv = self.out_convs[ind]

        hourglass_feat = single_hourglass(inter_feat)
        out_feat = out_conv(hourglass_feat)
        out_feats.append(out_feat)

        if ind < self.num_stacks - 1:
            inter_feat = self.conv1x1s[ind](
                inter_feat) + self.remap_convs[ind](
                    out_feat)
            inter_feat = self.inters[ind](self.relu(inter_feat))

    return out_feats
```

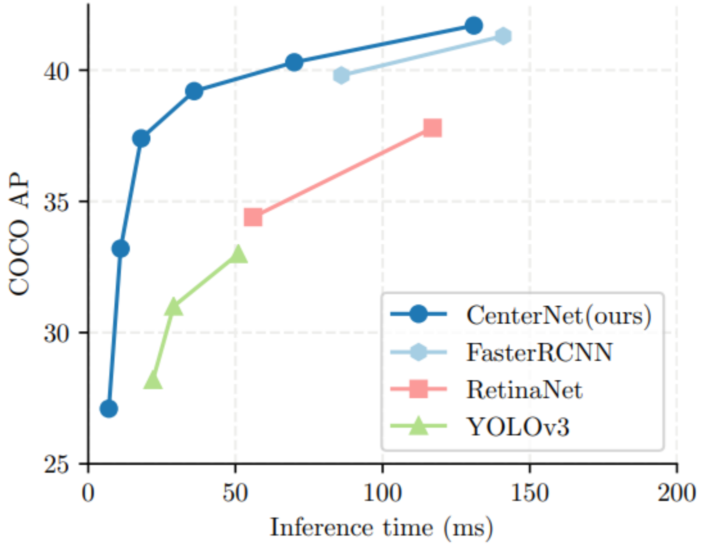
Centernet?

- Cornernet 다음으로 소개된 논문 One-stage detector

- Center를 찾기 위해 keypoint estimation 을 사용

-keypoint estimation: Cornernet에서 corner들을 찾을 때 사용한 방법

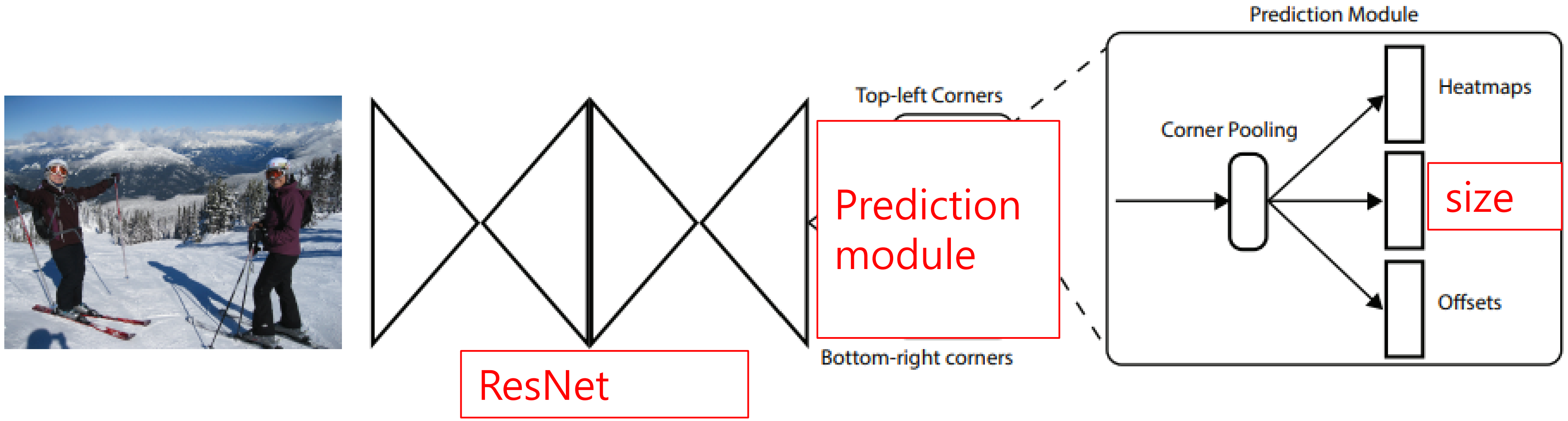
-2개의 지점을 찾아야 했던 cornernet과 달리 "중심점"만 찾으면 됨



- grouping 과정이나 post-processing 과정(ex. NMS)들이 필요 없다.

- 중앙 point의 feature값으로 detection뿐 아니라 object size, dimension, 3D extent, orientation, pose등도 regression할 수 있다

Structure


Prediction Module
Top-left Corners
Heatmaps
Corner Pooling
Prediction module
size
ResNet
Offsets
Bottom-right corners

Making box

Keypoints loss function

Loss Function : Focal Loss

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

N is the number of keypoints in image I

$\alpha = 2$ and $\beta = 4$

$\hat{Y}_{x,y,c} = 1$ corresponds to a detected keypoint

$\hat{Y}_{x,y,c} = 0$ is background

Offsets

Loss Function : L1 Loss

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

$$\boldsymbol{o}_k = \left(\frac{x_k}{n} - \left\lfloor \frac{x_k}{n} \right\rfloor , \frac{y_k}{n} - \left\lfloor \frac{y_k}{n} \right\rfloor \right)$$ from CornerNet

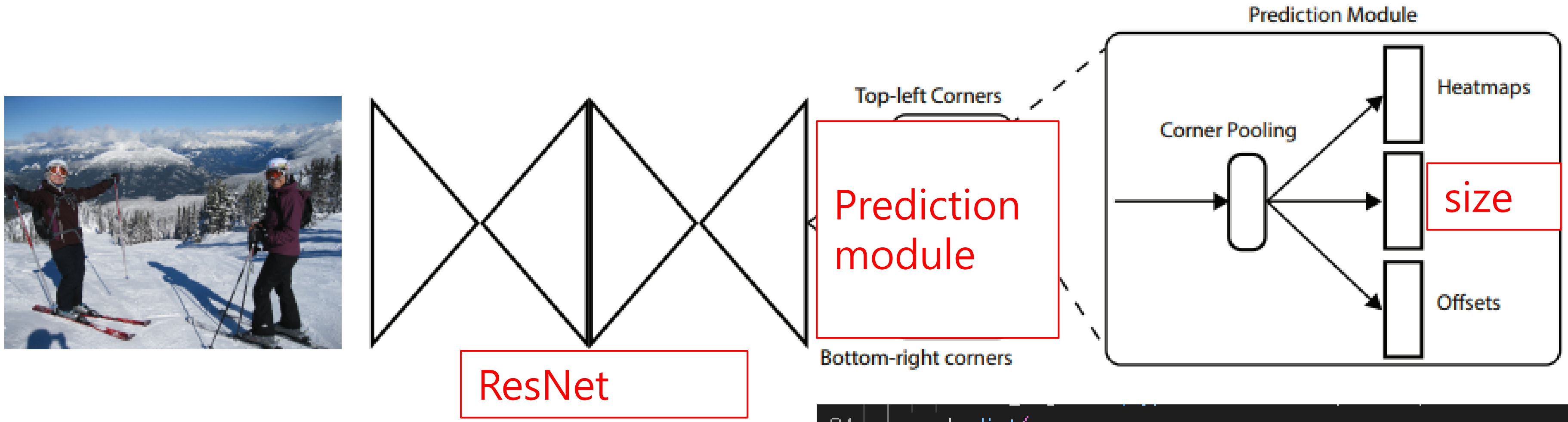
Sizes

Loss Function : L1 Loss

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

$(x_1^{(k)}, y_1^{(k)}, x_2^{(k)}, y_2^{(k)})$ Object k의 bounding box 좌표

$s_k = (x_2^{(k)} - x_1^{(k)}, y_2^{(k)} - y_1^{(k)})$




```python
model = dict(
    type='CenterNet',
    data_preprocessor=dict(
        type='DetDataPreprocessor',
        mean=[123.675, 116.28, 103.53],
        std=[58.395, 57.12, 57.375],
        bgr_to_rgb=True),
    backbone=dict(
        type='ResNet',
        depth=18,
        norm_eval=False,
        norm_cfg=dict(type='BN'),
        init_cfg=dict(type='Pretrained', checkpoint='torchvision://resnet18')),
```

```python
    neck=dict(
        type='CTResNetNeck',
        in_channels=512,
        num_deconv_filters=(256, 128, 64),
        num_deconv_kernels=(4, 4, 4),
        use_dcn=True),
    bbox_head=dict(
        type='CenterNetHead',
        num_classes=80,
        in_channels=64,
        feat_channels=64,
        loss_center_heatmap=dict(type='GaussianFocalLoss', loss_weight=1.0),
        loss_wh=dict(type='L1Loss', loss_weight=0.1),
        loss_offset=dict(type='L1Loss', loss_weight=1.0)),
    train_cfg=None,
    test_cfg=dict(topk=100, local_maximum_kernel=3, max_per_img=100))
```

Result

	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.

```
[4]
# We download the pre-trained checkpoints for inference and finetuning.
!mkdir ./checkpoints
!mim download mmdet --config cornernet_hourglass104_8xb6-210e-mstest_coco --dest ./checkpoints
#!mim download mmdet --config centernet_r18-dcnv2_8xb16-crop512-140e_coco --dest ./checkpoints
#!mim download mmdet --config yolov3_mobilenetv2_8xb24-320-300e_coco --dest ./checkpoints
```

```
processing cornernet_hourglass104_8xb6-210e-mstest_coco...
downloading ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 767.6/767.6 MiB 73.8 MB/s eta 0:00:00
Successfully downloaded cornernet_hourglass104_mstest_8x6_210e_coco_20200825_150618-79b44c30.pth to /content/mmdetection/checkpoints
Successfully dumped cornernet_hourglass104_8xb6-210e-mstest_coco.py to /content/mmdetection/checkpoints
```

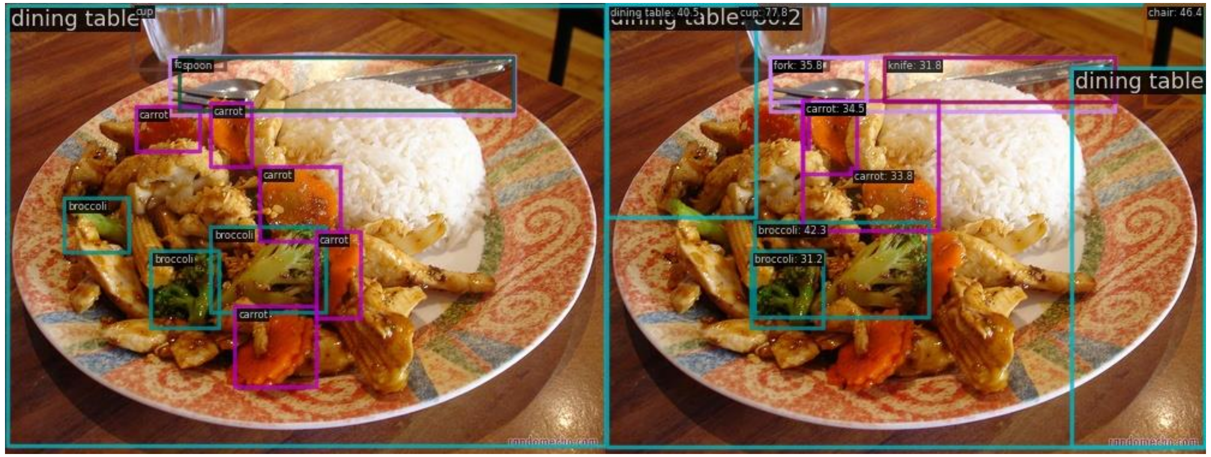
```
[5]
from mmdet.apis import DetInferencer

# Choose to use a config
model_name = 'cornernet_hourglass104_8xb6-210e-mstest_coco'
#model_name = 'centernet_r18-dcnv2_8xb16-crop512-140e_coco'
# model_name = 'yolov3_mobilenetv2_8xb24-320-300e_coco'
# Setup a checkpoint file to load
checkpoint = './checkpoints/cornernet_hourglass104_mstest_8x6_210e_coco_20200825_150618-79b44c30.pth'
#checkpoint = './checkpoints/centernet_resnet18_dcnv2_140e_coco_20210702_155131-c8cd631f.pth'
# checkpoint = './checkpoints/yolov3_mobilenetv2_320_300e_coco_20210719_215349-d18dff72.pth'

# Set the device to be used for evaluation
device = 'cuda:0'

# Initialize the DetInferencer
inferencer = DetInferencer(model_name, checkpoint, device)

# Use the detector to do inference
img = './demo/demo.jpg'
result = inferencer(img, out_dir='./output')


```

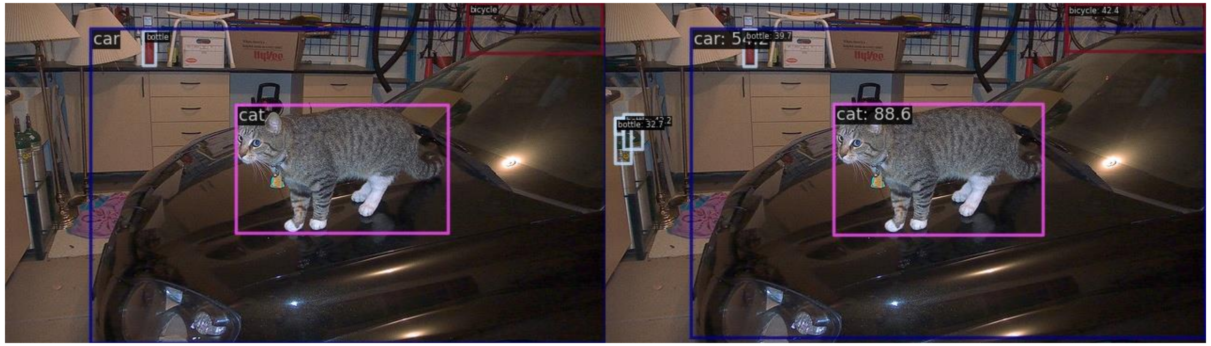
```
Loads checkpoint by local backend from path: ./checkpoints/cornernet_hourglass104_mstest_8x6_210e_coco_20200825_150618-79b44c30.pth
11/13 15:19:23 - mmengine - WARNING - Failed to search registry with scope "mmdet" in the "function" registry tree. As a workaround, the current "function" registry in "
Inference ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
/usr/local/lib/python3.10/dist-packages/mmengine/visualization/visualizer.py:196: UserWarning: Failed to add <class 'mmengine.visualization.vis_backend.LocalVisBackend'>
  warnings.warn(f'Failed to add {vis_backend.__class__}, '
```

```
[17]  # Download the data and unzip it
      !python tools/misc/download_dataset.py --dataset-name coco2017_study --save-dir data/coco --unzip --delete
```

```
[29]  !python tools/test.py ₩
          configs/cornernet/cornernet_hourglass104_8xb6-210e-mstest_coco.py ₩
          checkpoints/cornernet_hourglass104_mstest_8x6_210e_coco_20200825_150618-79b44c30.pth ₩
          --show
```

```
11/13 17:07:46 - mmengine - INFO - Epoch(test) [4950/5000]    eta: 0:00:27  time: 0.5695  data_time: 0.3128  memory: 1121
11/13 17:08:12 - mmengine - INFO - Epoch(test) [5000/5000]    eta: 0:00:00  time: 0.5187  data_time: 0.2792  memory: 1121
11/13 17:08:25 - mmengine - INFO - Evaluating bbox...
Loading and preparing results...
DONE (t=1.75s)
creating index...
index created!
Running per image evaluation...
Evaluate annotation type *bbox*
DONE (t=51.27s).
Accumulating evaluation results...
DONE (t=19.60s).
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.405
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=1000 ] = 0.560
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=1000 ] = 0.433
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.210
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.435
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = 0.553
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.589
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=300 ] = 0.589
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=1000 ] = 0.589
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.381
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.632
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = 0.759
```

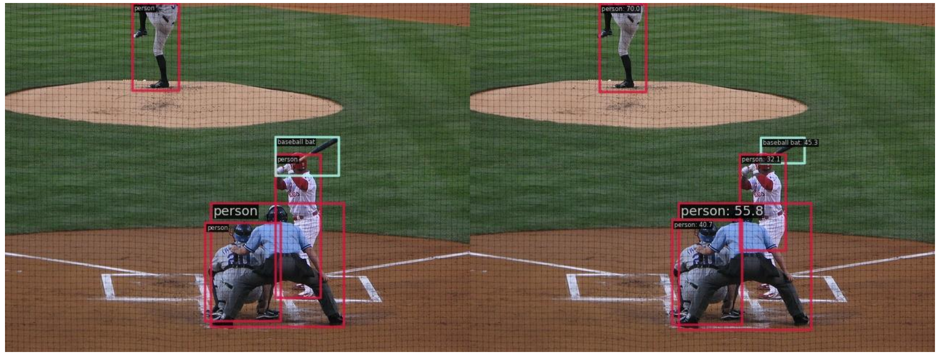


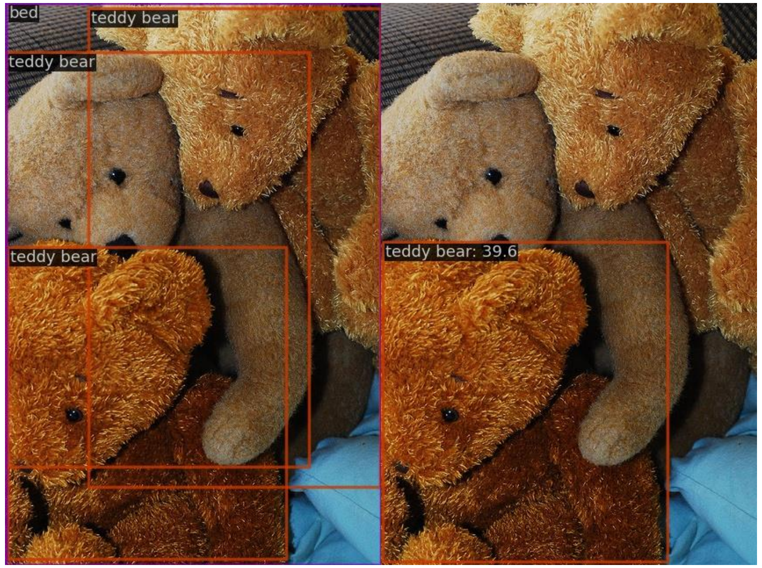


```
1 !python tools/test.py \
2     configs/centernet/centernet_r18-dcnv2_8xb16-crop512-140e_coco.py \
3     checkpoints/centernet_resnet18_dcnv2_140e_coco_20210702_155131-c8cd631f.pth \
4     --show-dir centernet_r18-dcnv2_8xb16-crop512-140e_results
```

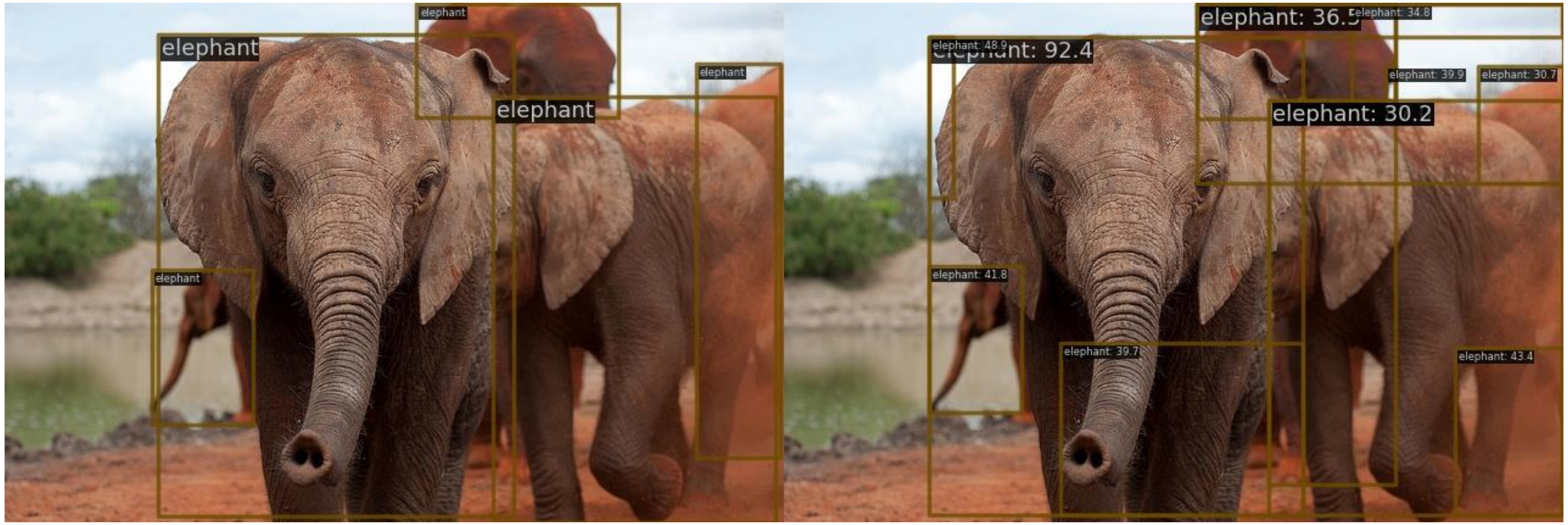
```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.295
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=1000 ] = 0.461
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=1000 ] = 0.314
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.102
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.329
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = 0.467
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.451
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=300 ] = 0.451
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=1000 ] = 0.451
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.191
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.496
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = 0.696
```

	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**

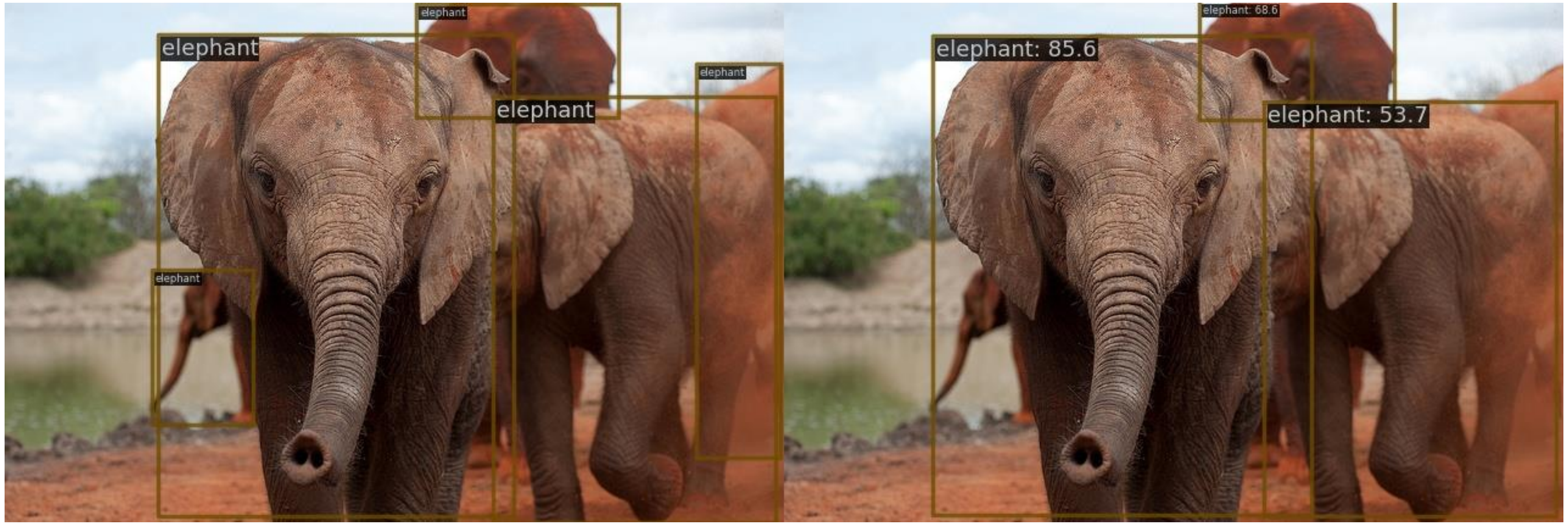




Cornernet



Centernet



참고자료

- Conernet

-https://arxiv.org/abs/1808.01244
-https://talktato.tistory.com/19

- Anchor box

-https://wikidocs.net/173914

- Centernet

-https://seongkyun.github.io/papers/2019/10/28/centernet/
-https://nuggy875.tistory.com/34