CenterNet:
Keypoint Triplets for Object Detection

Group 4

2020310243 Bohyun Kim
2019312127 Kangryun Moon
2019314658 Jongeun Park

Outline

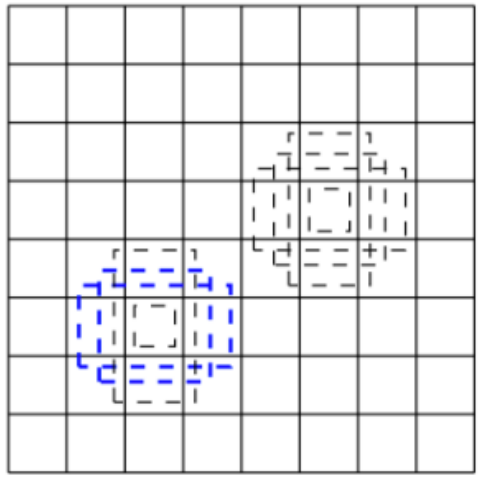
1. Introduction
2. Methods
3. Results & Evaluations
4. Challenges
5. Conclusion
6. Code demo

Part 1

Introduction

Anchor-based approach

a set of rectangles to detect objects



BUT - needs a lot of anchors
- not usually matched with the real boxes

Keypoint-based approach

CornerNet

a pair of corner keypoints to detect objects

BUT - hard to refer to the global information
- generates incorrect bounding boxes

CenterNet

three keypoints

Part 2

Methods

CornerNet

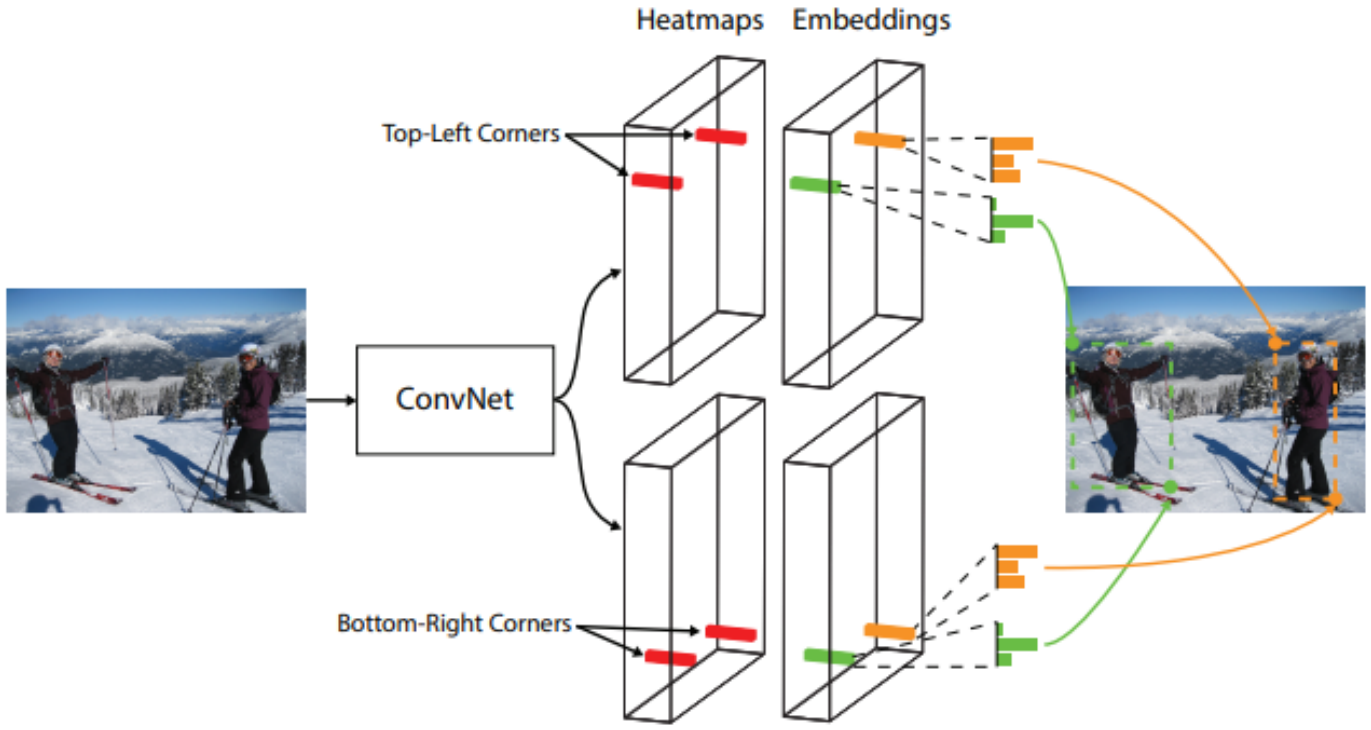
a pair of corner keypoints
- the top-left corner and bottom-right corner of the bounding box

① producing heatmaps of top-left corners and bottom-right corners

② predicting embeddings & offsets

③ selected top corners

④ object bounding box is created



- Confidence score

likelihood that the output of model is correct

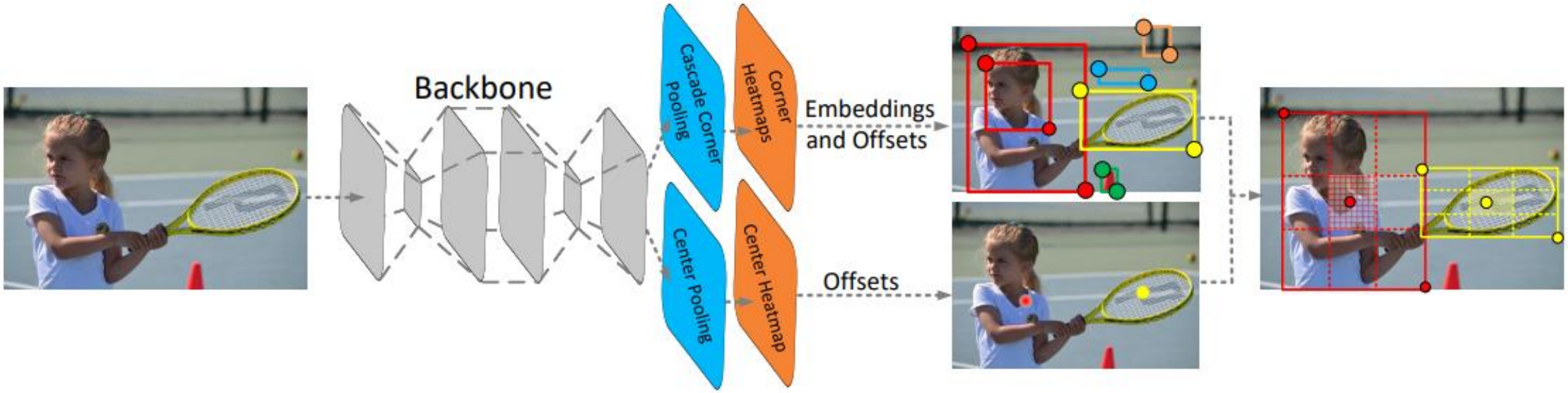
Part 2

Methods

CenterNet

Based on **Convolutional Neural Network(CNN)** with **3(triplet) keypoints**
(a pair of corner keypoints & a center keypoint)

- Architecture of CenterNet -



selecting top center keypoints
→ remapping center keypoints to the input image
→ defining the central region and checking

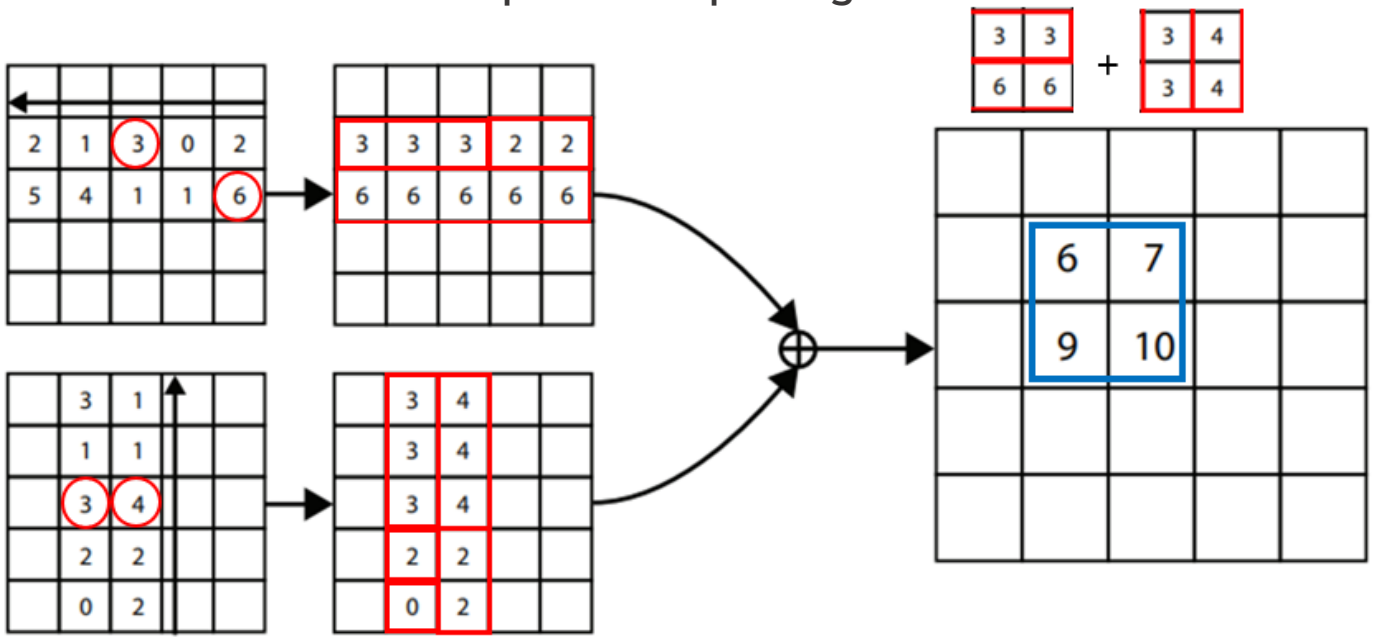
Part 2

Methods

CenterNet

Based upon CornerNet with a center keypoint

- a process of pooling -



Center pooling

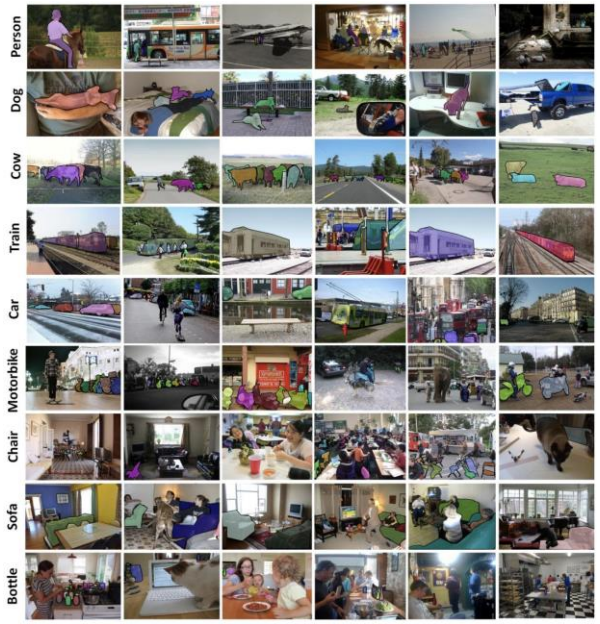
- searching the maximum value in both horizontal and vertical directions

Cascade corner pooling

- a boundary maximum value with an internal maximum value

Part 3

Results & Evaluations



- Dataset : MS-COCO
  - large-scale object detection dataset
  - containing 80 categories and more than 1.5 million object instances
- Evaluation metrics : AP / AR
  - AP: the average precision rate
  - AR: the maximum recall rate
  - FD: the proportion of the incorrect bounding boxes (1 – AP)

- **two-stage approaches**

- extract Region of Interest (RoI), then classify and regress RoIs ex) R-CNN

- **one-stage approaches**

- classify and regress the anchor boxes without RoI extraction process ex) CornerNet

Part 3

Results & Evaluations

two-stage

one-stage

CornerNet

CenterNet

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

- CenterNet511-104 reports a multi-scale testing AP of 47%, an improvement of **4.9%** over from CornerNet
- CenterNet511-104 improves the AP by **8.1%** on multi scale for small objects
  - 104 means

Using 104 layers as the backbone

- It surpasses other one-stage approach performances
- It is also competitive with the two-stage approaches

Part 3

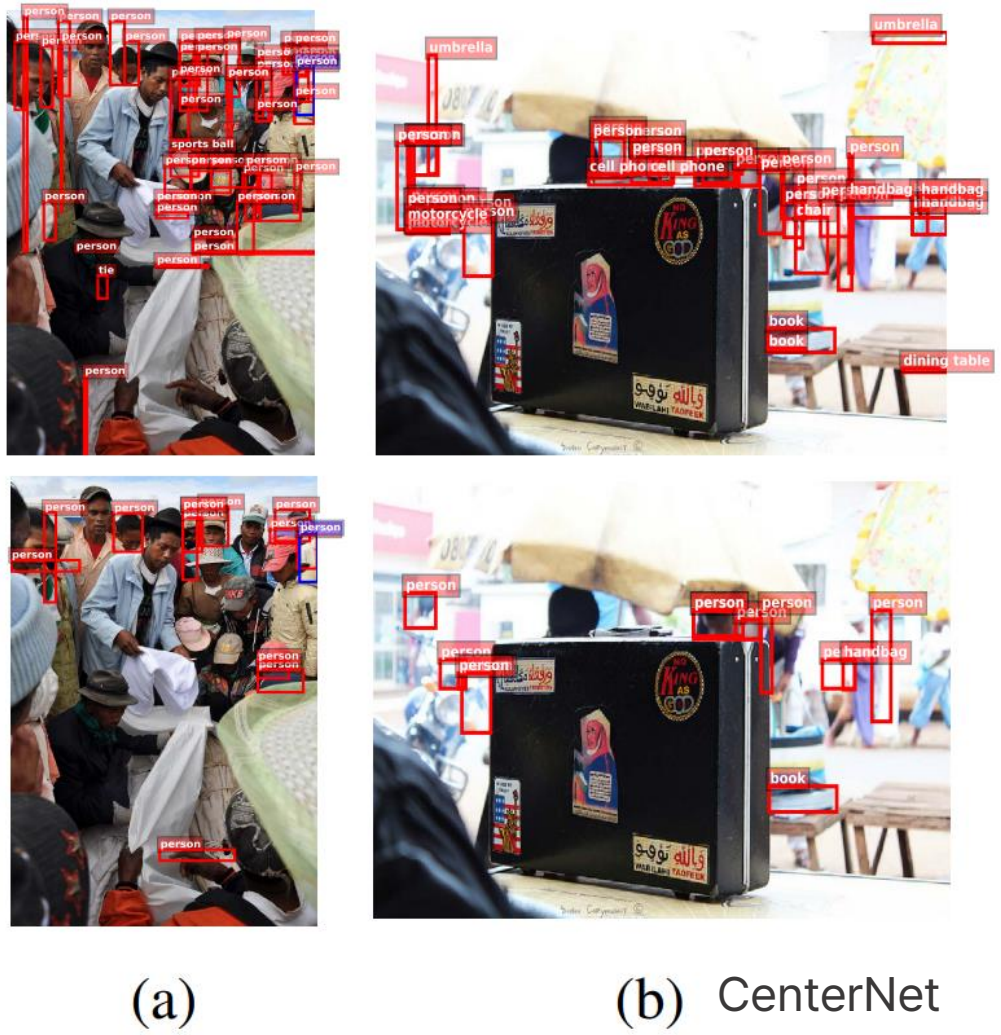
Results & Evaluations

Method	FD	FD_5	FD_{25}	FD_{50}	FD_S	FD_M	FD_L
CornerNet511-52	40.4	35.2	39.4	46.7	62.5	36.9	28.0
CenterNet511-52	**35.1**	**30.7**	**34.2**	**40.8**	**53.0**	**31.3**	**24.4**
CornerNet511-104	37.8	32.7	36.8	43.8	60.3	33.2	25.1
CenterNet511-104	**32.4**	**28.2**	**31.6**	**37.5**	50.7	**27.1**	**23.0**

Table 3: Comparison of false discovery rates (%) of CornerNet and CenterNet on the MS-COCO validation dataset.

- CenterNet decreases FD rate by exploring central regions
- FD rates for small bounding boxes decreased by **9.6%** for CenterNet511-104

CenterNet avoids a lot of incorrect bounding boxes, especially for small ones



(a) (b) CenterNet

Part 4

Challenges

1. Setting the programming environment

- modify codes, and update libraries
- subdivide a process of unzipping images

Own Code: Unzip the train2014.zip file to trainval2014 folder.

```
%cd /content/gdrive/MyDrive/CenterNet_master_2/data/coco/images/trainval2014/
!unzip  -j /content/gdrive/MyDrive/CenterNet_master_2/data/coco/images/train2014.zip -x __MACOSX/*
```

Own Code: Unzip the val2014.zip file to trainval2014 folder.

```
%cd /content/gdrive/MyDrive/CenterNet_master_2/data/coco/images/trainval2014/
!unzip -j /content/gdrive/MyDrive/CenterNet_master_2/data/coco/images/val2014.zip -x __MACOSX/*
```

Own Code: Unzip the test2017.zip file to testdev2017 folder.

```
%cd /content/gdrive/MyDrive/CenterNet_master_2/data/coco/images/testdev2017/
!unzip -j /content/gdrive/MyDrive/CenterNet_master_2/data/coco/images/test2017.zip -x __MACOSX/*
```

Part 4

Challenges

2. Excessive training time

- Reduce a batch size from 48 to 4
- Reduce max iterations to 10000
- Shorten training time to 6 hours

3. Understanding concepts in the paper

- Especially read a paper about CornerNet to accumulate background knowledge
- H. Law and J. Deng. Cornernet: Detecting objects as paired keypoints

Part 5

Conclusions

1	CenterNet is improved from CornerNet by using center pooling and cascade corner pooling
2	[limitation] For large objects, cascade corner pooling did not show any improvement in precision rate
3	CenterNet resolves one-stage model's limitations

Code demo

by colab

Additional Explanation For Q&A

Part 6

Additional Explanation for Q&A

Q1. The meaning of 'triplet' keypoints

> Compared to 2(=pair) keypoints that CornerNet used, CenterNet looks 3(=triplet) keypoints to detect bounding boxes. This was explained in presentation as "adding one more keypoint in the central region".

Q2. Evaluation of trained model, training time for 1 epoch, and use of pre-trained model

> The model our group trained with batch size=4, max iteration=10000 showed AP/AR rates of 1~8% which was much less than results in paper. Detailed numbers are provided in **Appendix** or code demo notebook "Testing" cell.

(# of iterations in 1 epoch) = (data size) / (batch size) = 123,000 / 4 = 30,750
epoch = (maximum iterations) / (# of iterations in 1 epoch) = 10,000 / 30,750 =~ 1/3 epoch (6 hours)

It would have taken about 18 hours to train 1 epoch. But due to limitation of runtime in Colab Pro, we continuously reduced our iteration until 10,000. (In the paper, authors trained 480,000 / (123,000 / 48) =~ 187 epochs)

Because of low accuracy we decided to use pretrained model. At first, we could not get the expected values, but by using pre-trained model provided from Github, we could get the expected result, almost same accuracy as the paper.

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.450
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.625
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.486
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.260
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.492
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.592
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.361
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.579
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.626
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.415
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.677
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.785
```

CenterNet511 (single-scale)

AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	AR_1	AR_{10}	AR_{100}	AR_S	AR_M	AR_L
44.9	62.4	48.1	25.6	47.4	57.4	36.1	58.4	63.3	41.3	67.1	80.2

Figure 1, 2. Evaluation of pre-trained model and result of paper from Table 2

Part 6

Additional Explanation for Q&A

Q3. Inference about the reason of big performance improvement & relatively long training time

> They could increase the performance by looking inside the boundary boxes, while applying advanced ways of pooling that can effectively reduce the rich information.

Information at boundaries(CornerNet) tends not to give enough information for bounding boxes, since it is rare that the object fits to the square-shaped bounding boxes. By exploring central regions(CenterNet), model could reach more valuable data that is helpful for detecting objects.

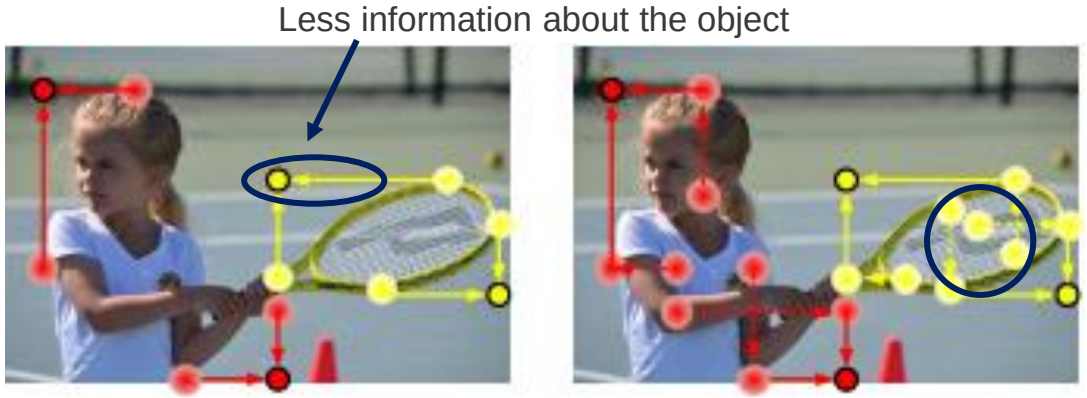


Figure 3, 4. Comparison of Corner pooling and Cascade corner pooling

Meanwhile, giving more information to the model does not always lead to positive results, since it requires increased time of training or a hardware that can provide such calculations. But authors could partially resolve this problem by applying advanced pooling method. (Center pooling, Cascade corner pooling)

And this additional internal information might be the reason the authors took many days to train the model with GPUs having the highest performance. Reducing abundant information in corners can be one way to improve CenterNet model with lower costs.

Part 7

Appendices

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.012
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.023
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.011
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.010
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.017
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.014
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.034
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.058
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.064
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.024
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.061
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.087
```

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.450
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.625
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.486
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.260
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.492
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.592
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.361
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.579
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.626
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.415
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.677
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.785
```

Figure A, B. Comparison of evaluation between trained model with reduced hyper-parameters and pre-trained model

Group management

Bohyun Kim (Team Leader)
Main Understanding paper concepts
Sub Code implementation

Kangryun Moon
Main Code implementation and code tweaking
Sub Understanding paper concepts

Jongeun Park
Main Code implementation and code tweaking
Running train/test procedure

Figure C. Adjusting hyper-parameters in configuration file

CenterNet-104.json

```
{
    "system": {
        "dataset": "MSCOCO",
        "batch_size": 4,
        "sampling_function": "kp_detection",

        "train_split": "trainval",
        "val_split": "minival",

        "learning_rate": 0.00025,
        "decay_rate": 10,

        "val_iter": 500,

        "opt_algo": "adam",
        "prefetch_size": 6,

        "max_iter": 10000,
        "stepsize": 450000,
        "snapshot": 5000,

        "chunk_sizes": [4],

        "data_dir": "./data"
```

CenterNet:
Keypoint Triplets for Object Detection

Thank you ☺

Although we tried to submit a zip file, it cannot be uploaded on iCampus because of its large size (about 28 GB). So, we attach links of our notebook and files including dataset.

1. Our Notebook: Group4_fullandbasic_01.ipynb

https://colab.research.google.com/drive/1Zmy04POuYYeywyu3z70fQAGV8NcK0HpM#scrollTo=gTeqnZAaGFYh

2. files: CenterNet_master_2.zip

https://drive.google.com/drive/folders/1TG2wvci5-AMPZmAwGqddREcNgH3yOXal

used dataset: val2014.zip, test2017.zip, train2014.zip

https://cocodataset.org/#download