Light Corner Based Object Detector with Stacked-ENet Backbones

Qingyu Li and Zhenjiang Miao
Institute of Information Science, Beijing Jiaotong University, Beijing, P.R. China.

Abstract. Object detection algorithms have made great progress in the past few years because of the development of deep learning. From region selection to anchor box regression, the accuracy of the algorithms becoming more and more accurate, but they’re also far away from vision mechanism of human. Corner based is a new approach for object detection, but methods proposed nowadays are in huge wastage of time and resource, leading them not available in vast bulk of jobs. In this paper, we proposed a method which makes corner-based object detector could inference in real-time. We analyze the disadvantages of existing methods, found out the most consuming parts and the solution for improvements, which finally makes the algorithm efficiency and get a result which is competitive with other existing real-time methods.

1. Introduction
Convolutional neural network object detectors have become the state-of-the-art solutions in many challenging tasks. The most commonly used method is anchor box generation, in which many kinds of sizes and ratios are used for detecting different objects. But there some drawbacks for this approach, firstly, we need a huge number of anchor boxes for ensure that boxes could overlap the most objects, which makes that only a few of boxes might fit the ground truth boxes and create a serious imbalance between positives and negatives; on the other hand, this method introduces lots of hyper-parameters and settings, such as amount of the boxes, as well as size and ratio of each box, leading to the difficulty of network designing and performance limitation.

CornerNet [1] provides a new approach which detect an object with a pair of key points instead of anchor box, this approach makes the output of the network simple and the anchor boxes are not need to be eliminated, the keypoints are selected by corner pooling on the heatmap outputted from the hourglass backbone network. But for implementation, the network should be training on 10 Titan X GPUs with a tiny batch-size of 49 and a long time of 244ms for a single-image inference when testing, which indicates its disadvantages. Because of the huge backbone network, the runtime and memory usage for CornerNet [1] is not suitable for time and resource limitation jobs.

In this paper, we proposed a light corner-based object detector with stacked-ENet backbones, in which the original large hourglass network is replace by a stack of the light-weighted ENet [2] networks for heatmap generation.

Furthermore, our methods also show an ability that the network could adjusted by adding or removing several submodules according to the application requirements in which speed or accuracy is concerned more about.
2. Related Works

2.1. Object Detectors

2.1.1. Two-stage object detectors. Two-stage detectors evaluate a small set which contains the regions of interests in the first stage and then classify the content in each of the regions.

R-CNN [3] is the first proposed method in this category, it extracts the regions, then crop the corresponding areas from the original image, resize to a specific scale, and classify each of them by a convolution network. But regions are computed separately and independently, which makes a lot of redundant computations.

Fast R-CNN [4] improve the former method by designing a specific pooling layer which could pool each region from feature map instead of crop on the original image, but it still relay on separate proposal algorithm, that is hard to be end-to-end trained.

Faster R-CNN [5] proposed a region proposal network (RPN) to get rid of the separate low-level proposal algorithm.

2.1.2. One-stage object detectors. One-stage detectors are more commonly used in resource limitation jobs, due to that they remove the region selection steps, and detect the objects in a single network.

YOLO [6] directly predicts bounding boxes from images by a stack of fully-connected layers between the feature maps and the results, and the YOLO9000 [7] improves the method by introducing the anchor boxes.

SSD [8] uses different sizes and ratios of the anchor boxes to slide on the pyramid of the multi-scale feature maps, directly classify and refine each of them. RetinaNet [9], RefineDet [10], DSOD [11] and many other methods extend this thought and get better results.

2.2. Hourglass Shaped Networks and Stacked-hourglass Networks for Backbones

Hourglass-shaped networks are networks combined with down-sampling and up-sampling parts as encoder-decoder structures, used in segmentation jobs commonly.

FCN [12] transforms all the fully connected layers in VGG network to convolutional layers to achieve a fully convolutional structure, and then use a deconvolutional layer for up-sample to a resolution could express the result in pixel level.

U-Net [13] extends this architecture. It contains a series of down-sampling layers followed by same amounts of the up-sampling layers reversed at each scale, make the whole network a shape like letter “U”. The feature maps are transferred and concatenated between last down-sampling layers and first up-sampling layers for resolutions.

But these architectures with huge numbers of parameters and long inference times cannot operate in real-time.

ENet [2] is proposed as a network for fast inference and high accuracy, with controlled network modules and large encoder and small decoder structure, it achieves the balance between runtime and performance.

The hourglass network [14] is a method which introduced firstly for human pose estimation which cascades many hourglass modules. Each hourglass module is a fully convolutional network, down-sampling the input features by a series of convolutional operators and then up-sampling the features back, for captures not only global but also local representations. While multiple modules are stacked, the latter module could reprocess the features extracted from the formers.

DSSD [15] adopts a network similar to a single hourglass module, with a down-sampling and an up-sampling part, it allows the combination of low-level and high-level features.

DocUNet [16] propose a stacked U-Net [13] architecture with intermediate supervision to directly predict the forward mapping from a distorted image to its rectified version.
3. Methods and Network Architectures

3.1. Corner Based Detector and its Disadvantages

As we described above, CornerNet [1] has competitive results in precision.

It uses the backbone network to generate a group of feature maps, then predicting two $C$ channels heatmaps which indicates the probability of whether a specific location is the top-left/bottom-right keypoint of a category, two $C$ channels embeddings for distinguish the keypoints pair, and two $C$ channels offsets for location adjusting. At last, select top-left points and bottom-right points with highest heatmap scores, pairing them by belonged categories and embedding distances, bounding boxes are obtained.

But it is far away from commonly used because of its long runtime and huge resources usage and the stacked Hourglass-104 [14] backbone network is the most time and resource consuming and parts.

The original CornerNet [1] has 200.97M parameters and 343.17G flops when inference for 20-classes outputs, much larger than SSD512 [8] (26M parameters, 91G flops) and other light-weight detectors, result to the inference time of 244ms, which is far away from real-time.

3.2. Designing Principles and Details

After research and evaluation, we found that when only considering the corner detector and classifier, 0.9M parameters and 50.85G flops, which provides a condition for improvements. ENet [2] has a high accuracy-density (accuracy per parameter) [17] and the same effects because of its well-designed structure. It has only 4.41G flops for 512x512 input, which makes a possibility for light corner-based object detector coming into use.

Following [14], we add intermediate supervising after each module while training, but in deploying period, the intermediate predictions are not added to the final results as [1].

We use 512x512 images as the original inputs. At the beginning of the network, an initial layer of ENet [2] is deployed for the first down-sampling. In which a non-overlapping 2x2 max pooling and a 15 kernels 3x3 convolution with 2 steps striding are used, which sum up to 16 feature maps of 256x256 after concatenation. Then for each ENet module, we use the layers between “bottleneck1.0” and “bottleneck5.1” layers for features and concatenate the 16 channels 256x256 sized feature maps extracted head-to-tail; besides that, the “fullconv” layers, whose amount of the output channels are the same as the classes, are deployed for intermediate supervising. The pipeline shows in Figure 1. When inferencing, all “fullconv” except for the last module are removed, feature maps from the last “fullconv” layer are used for corner keypoints detection.

As for the loss function, we use the summary of a focal loss [9] as heatmap loss for keypoints classification for each category for top-left and bottom-right corners,

$$L_{cls} = - \sum_{c=0}^{C} \sum_{w=0}^{W} \sum_{h=0}^{H} y_{i}^{c} \cdot (1 - p_{i}^{c})^{a} \cdot log(p_{i}^{c}) + (1 - y_{i}^{c}) \cdot (p_{i}^{c})^{a} \cdot log(1 - p_{i}^{c})$$

(1)
a distance describing inner-class and inter-class for embeddings,
\[
  \ell_{\text{emb}} = \sum_{k=1}^{N} (e_{k}^{\text{topleft}} - e_{k}^{\text{bottomright}})^2 + \sum_{i=1}^{N} \sum_{j=1}^{N} l(i \neq j) \cdot \max(0,1 - \frac{e_{i}^{\text{topleft}} + e_{j}^{\text{bottomright}} - e_{i}^{\text{topleft}} - e_{j}^{\text{bottomright}}}{2})
\]
and smoothL1 [4] loss for offsets. In which there are C classes of objects, size of image is \( W \times H \), and \( N \) boxes in ground-truth.

4. Experiments and Results

The network is implemented by PyTorch 0.4.0. We use 4 Titan X GPUs with entire batch-size of 128. Adam optimizer is deployed for updating the parameters with the direction of loss is converging, for the first 120k iterations, the learning rate is 1e-3, and then reduce it by 0.1 for the following 120k iterations.

In order to comparison with other real-time detectors, we use MS COCO test-dev dataset for evaluation.

We evaluate our method with 1~5 modules of ENets, for different computation and complexity.

The comparison with other recent methods is described in Table 1.

Our method makes it available that corner-based detector could inference in real-time, which is much faster than CornerNet [1] while not damage the accuracy too much. Compared with a mainly baseline method SSD and its variations, which are SSD [8] with 2 types of backbones as well as DSSD [15] and RetinaNet [9], our method shows comprehensive superiorities not only on accuracy but also on efficiency. As for YOLOv2 [7], a slightly decreasing of speed exchanged into a huge performance improvement. RefineDet [10] is an opponent of us, it has a similar performance in both runtime and accuracy (for the setting with almost the same efficiency, we are in the lead by a little advantage), but our method provides several choices for the network structure according to the task limitation.

As shown in Figure 2, we can infer that the methods we proposed achieve a better balance and trade-off performance between speed and accuracy.

| Method         | Backbone      | AP  | AP50 | AP75 | APs | Am | API | FPS |
|----------------|---------------|-----|------|------|-----|----|-----|-----|
| YOLOv2[7]      | DarkNet-19    | 21.6| 44.0 | 19.2 | 5.0 | 22.4| 35.5| 40  |
| SSDS512[8]     | VGG-16        | 28.8| 48.5 | 30.3 | 10.9| 31.8| 43.5| 19  |
| SSDS513[8]     | ResNet-101    | 31.2| 50.4 | 33.3 | 10.2| 34.5| 49.8| 6.8 |
| DSSD513[15]    | ResNet-101    | 33.2| 53.3 | 35.2 | 13.0| 35.4| 51.1| 5.5 |
| RetinaNet500[9]| ResNet-101    | 34.4| 53.1 | 36.8 | 14.7| 38.5| 49.1| 11  |
| RefineDet512[10]| ResNet-101  | 36.4| 57.5 | 39.5 | 16.6| 39.9| 51.4| 24.1|
| CornerNet[1]   | Hourglass-104 | 40.5| 56.5 | 43.1 | 19.4| 42.7| 53.9| 4.1 |
| Ours(1x)       | 1 x ENet      | 35.7| 53.3 | 39.3 | 16.2| 39.6| 50.6| 25.3|
| Ours(2x)       | 2 x ENet      | 36.6| 54.7 | 39.9 | 16.8| 40.2| 51.3| 24  |
| Ours(3x)       | 3 x ENet      | 37.4| 55.4 | 40.4 | 17.3| 40.7| 51.9| 22.6|
| Ours(4x)       | 4 x ENet      | 38.1| 55.9 | 40.8 | 17.7| 41.1| 52.3| 21.3|
| Ours(5x)       | 5 x ENet      | 38.5| 56.1 | 41.1 | 18.1| 41.4| 52.7| 19.9|
5. Conclusion
In this paper, we proposed a method which makes corner-based object detector could inference in real-time. We analyze the disadvantages of existing methods, found out the most consuming parts and the solution for improvements, which finally makes the algorithm efficiency and get a result which is competitive with other existing methods by experiments and comparison on MS COCO dataset.

Acknowledgements
This work is supported by the NSFC 61672089, 61273274, 61572064, and National Key Technology R&D Program of China 2012BAH01F03.

References
[1] Law, Hei, and Jia Deng. 2018. “CornerNet: Detecting Objects as Paired Keypoints.” European Conference on Computer Vision, 765–81.
[2] Paszke, Adam, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. 2017. “ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation.” ArXiv Preprint ArXiv:1606.02147.
[3] Girshick, Ross B., Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation.” In CVPR ’14 Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, 580–87.
[4] Girshick, Ross B. 2015. “Fast R-CNN.” In 2015 IEEE International Conference on Computer Vision (ICCV), 1440–48.
[5] Ren, Shaoqing, Kaiming He, Ross B. Girshick, and Jian Sun. 2015. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.” In NIPS’15 Proceedings of the
28th International Conference on Neural Information Processing Systems - Volume 1, 2015:91–99.

[6] Redmon, Joseph, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. 2016. “You Only Look Once: Unified, Real-Time Object Detection.” In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 779–88.

[7] Redmon, Joseph, and Ali Farhadi. 2017. “YOLO9000: Better, Faster, Stronger.” In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 6517–25.

[8] Liu, Wei, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. 2016. “SSD: Single Shot MultiBox Detector.” European Conference on Computer Vision, 21–37.

[9] Lin, Tsung-Yi, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. 2017. “Focal Loss for Dense Object Detection.” In 2017 IEEE International Conference on Computer Vision (ICCV), 2999–3007.

[10] Zhang, Shifeng, Longyin Wen, Xiao Bian, Zhen Lei, and Stan Z. Li. 2018. “Single-Shot Refinement Neural Network for Object Detection.” In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 4203–12.

[11] Shen, Zhiqiang, Zhuang Liu, Jianguo Li, Yu-Gang Jiang, Yurong Chen, and Xiangyang Xue. 2017. “DSOD: Learning Deeply Supervised Object Detectors from Scratch.” In 2017 IEEE International Conference on Computer Vision (ICCV), 1937–45.

[12] Long, Jonathan, Evan Shelhamer, and Trevor Darrell. 2015. “Fully Convolutional Networks for Semantic Segmentation.” In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 3431–40.

[13] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. 2015. “U-Net: Convolutional Networks for Biomedical Image Segmentation.” Medical Image Computing and Computer Assisted Intervention, 234–41.

[14] Newell, Alejandro, Kaiyu Yang, and Jia Deng. 2016. “Stacked Hourglass Networks for Human Pose Estimation.” European Conference on Computer Vision, 483–99.

[15] Fu, Cheng-Yang, Wei Liu, Ananth Ranga, Ambrish Tyagi, and Alexander C. Berg. 2017. “DSSD: Deconvolutional Single Shot Detector.” ArXiv Preprint ArXiv:1701.06659.

[16] Ma, Ke, Zhixin Shu, Xue Bai, Jue Wang, and Dimitris Samaras. 2018. “DocUNet: Document Image Unwarping via a Stacked U-Net.” In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 4700–4709.

[17] Canziani, Alfredo, Adam Paszke, and Eugenio Culurciello. 2017. “An Analysis of Deep Neural Network Models for Practical Applications.” ArXiv Preprint ArXiv:1605.07678.