Pooling Pyramid Network for Object Detection

Pengchong Jin Vivek Rathod Xiangxin Zhu
Google AI Perception
{pengchong, rathodv, xiangxin}@google.com

Abstract

We proposed The ABSTRACT is to be in fully-justified italicized text, at the top of the left-hand column, below the author and affiliation information. Use the word "Abstract" as the title, in 12-point Times, boldface type, centered relative to the column, initially capitalized. The abstract is to be in 10-point, single-spaced type. Leave two blank lines after the Abstract, then begin the main text. Look at previous CVPR abstracts to get a feel for style and length.

1. Introduction

2. Related Work

Multibox [1]
SSD [5]
Faster-RCNN [8]
YOLO [6]
YOLO-v2 [7]
FPN [3]
Survey [2]
RetinaNet [4]

3. Pooling Pyramid Network (PPN)

Our proposed model, called *Pooling Pyramid Network (PPN)*, is a single-stage convolutional object detector, that is designed to be light-weighted, fast to run, while maintains the good detection accuracy. The network architecture is illustrated in Figure 1. There are two major changes to the original SSD [5]: (1) the convolutions between feature maps are replaced with the max pooling operations. (2) the box predictor is shared across feature maps with different scales; In the following sections, we will discuss the rationales behind them and effects of these changes.

3.1. Max Pooling Pyramid

Our goal is to build a multi-scale feature pyramid structure, from which we can make the predictions using the shared box predictor. We achieve this by shrinking down a base feature map from the backbone network serval times using a series of max pooling operations. This is different from SSD where feature maps are built by extracting layers from backbone network and shrinking them using additional convolutions, and FPN where feature maps are built by a top-down pathway with skip connections. We choose max pooling mainly for two reasons. First, using the pooling operations ensures feature maps with different scales live in the same embedding space, which makes training the shared box predictor more effective. In addition, since max pooling does not require any additions and multiplicatons, it is very fast to compute during the inference, therefore, making it suitable for many latency sensitive applications.

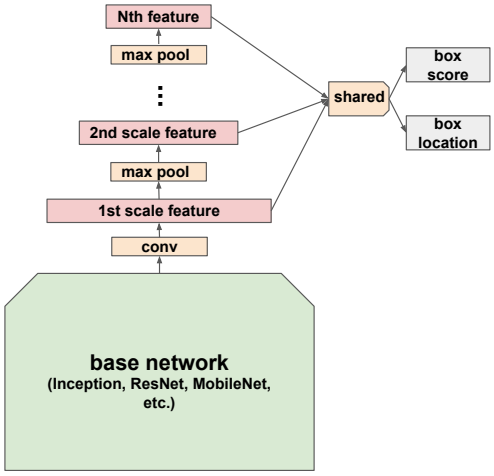

Figure 1. The Pooling Pyramid Network (PPN) architecture.

3.2. Shared Box Predictor

The original SSD uses separate box predictors for feature maps of different scales. While in this design each box predictor is allocated to spend its full capacity on box prediction of one specific scale, one potential problem is miscalibration of the prediction scores across different scales. Because each box predictor is trained independently using only a portion of the groundtruth boxes that it is assigned

to, different box predictors could see very different amount of positive and negative examples during the training. This implicit data imbalance problem could potentially cause the problem that scores from different predictors fall in a very different ranges, which makes them incomparable and difficult to use in the subsequent score-based postprocessing such as non maximum suppression. We design PPN to use the shared box predictor accross feature maps of different scales. As a result, the box predictor sees all the training data during the training and implicitly argument the data when there are imbalance groundtruth boxes with different scales. As a result, this reduces the effect of miscalibration and instable prediction scores.

3.3. Overall Architecture

The final network architecture of our Pooling Pyramid Network (PPN) detector is illustrated in Figure 1. Followed by the backbone network, an optional 1x1 convolution is used to transform the features from the backbone network to a space with desired dimensions. We then apply a series of stride-2 max pooling operations to shrink the feature map down to 1x1. A shared box predictor is applied to feature maps of different scales in order to produce classification scores and location offsets of box predictions. We add one additional shared convolution in the box predictor after pooling operations to prepare the feature to be used for predictions.

4. Experiments

4.1. Comparing SSD and PPN

References

[1] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. In *ECCV*, 2014.

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

[3] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

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

[5] 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.

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *CVPR*, 2016.

[7] J. Redmon and A. Farhadi. YOLO9000: Better, faster, stronger. In *CVPR*, 2017.

[8] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Neural Information Processing Systems (NIPS)*, 2015.

Model	AP	AP50	AP75	FLOPs	number of parameters
MobileNet SSD	1	1	1	1	1
MobileNet PPN	1	1	1	1	1

Table 1. COCO detection: MobileNet SSD vs MobileNet PPN