YOLO-Mask: Fast One-Stage Instance Segmentation

Huaiyu Zheng
Northeastern University
zheng.hua@husky.neu.edu

Yaofei Wang
Northeastern University
wang.yaof@husky.neu.edu

Zhiyang Liu
Northeastern University
liu.zhiya@husky.neu.edu

Abstract

We proposed a network combined yolov3 and mask rcnn to do the instance segmentation task. We use darknet as feature map backbone, and yolov3s original approach to get classes and bbox. Then align bbox on the feature maps and go through the mask fcn branch to get mask prediction.

Hopefully our network can be faster than mask rcnn and get high AP point. We train our network on COCO [1] dataset.

1. Introduction

The state-of-art method Mask RCNN [2] is really powerful and has elegant precisions for instance segmentation. However, the slow feature map generation process become a bottleneck when we try to increase the speed. Also, its based on two stages: region proposal network and detection network(classification, detection and mask prediction).

YOLOv3, as one of the trending framework in object detection, enjoys real-time speed with its unified structure and even higher precision with the pyramid of multi-scale feature maps. With similar performance on COCO dataset [1], YOLOv3-416 runs two times faster than RetinaNet-50-500 [3]. Its also observed that with spatial pyramid pooling method, the framework would be more robust to the scale change and occlusion [4]. By adding a mask branch, we incorporate this extreme fast object detection method in our instance segmentation network.

By combining these two framework, we introduce a new method to perform instance segmentation with high precision.

2. Related Works

2.1. Instance Segmentation

A lot of research effort has been made to push instance segmentation accuracy. Some methods first perform semantic segmentation followed by boundary detection [5], pixel clustering [6] [7], or learn an embedding to form instance masks [8] [9] [10] [11]. Mask-RCNN [2] as a representative two-stage instance segmentation approach, first generates candidate region-of-interests (ROIs), then generate mask for ROIs detected. Follow-up works try to improve its accuracy by enriching the FPN features [12] or addressing the incompatibility between a masks confidence score and its localization accuracy [13], which requires subsequent computations that make them unable to obtain real-time speeds (30 fps). In our work, we keep YOLO structure as our bounding box detection part and use only mask branch from Mask-RCNN for mask generation, to keep us away from large computation and save processing time as much as possible.

2.2. Real-time Instance Segmentation

While real-time object detection [14] [15] [16] [3], and semantic segmentation [17] [18] [19] [20] methods exist, few works have focused on real-time instance segmentation. Mask R-CNN [2] remains one of the fastest instance segmentation methods on semantically challenging datasets.

Among some works in real-time instance segmentation, straight to Shapes [21] performs instance segmentation with learned encodings of shapes at 30 fps, but its accuracy is far from that of modern baselines. With a light-weight backbone detector, Box2Pix [22] implements a hand-engineered algorithm to obtain 10.9 fps on Cityscapes [23] and 35 fps on KITTI. However, they report no results on the more challenging and semantically rich COCO dataset, which has 80 classes compared to the 8 of KITTI and Cityscapes. Also, theres a large drop in performance going from a semantically simple dataset (KITTI) to a more complex one (Cityscapes), so the performance on an even more difficult dataset (COCO) is to be questioned.

2.3. Mask RCNN

The proposal-based method is most popular in instance segmentation, which have a strong connection to object detection. Mask R-CNN extends Faster R-CNN [24] by adding a branch for predicting segmentation masks on each Region of Interest (RoI), in parallel with the existing branch for classication and bounding box regression.

They predict an $m \times m$ mask from each RoI using

FCN [25], which allows each layer in the mask branch to maintain the explicit $m \times m$ object spatial layout without collapsing it into a vector representation that lacks spatial dimensions. Also, the FCN requires less parameters compared with fc layers.

Before passing the RoI into the FCN, instead of RoIPool, the Mask RCNN applies RoIAlign to properly align the extracted features with the input. In this way, they avoid any quantization of the RoI boundaries or bins.

2.4. YOLO

YOLO is an end-to-end object detection framework that runs in real time. The first version comes out in 2016, which is quite impressive with its real-time performance. Then we have YOLOv2 and YOLOv3, which takes YOLO to a new level with its even higher precision, especially when detecting small objects.

Aside from a structure called Darknet-53, which originally has 53 layer network trained on ImageNet, YOLOv3 also incorporates residual skip connections, upsampling, and makes detections at three different scales, which helps address the issue of detecting small objects.

Our work is based on YOLOv3, and explores several morphing structures like YOLOv3-tiny and YOLOv3-spp. The latter one, YOLOv3-spp is especially robust to objects in different scales with its multiple effective field-of-views. Based on the YOLOv3 part, we add a mask branch to turn the object detection problem into an instance segmentation task.

3. YOLO-Mask Model

Inspired from mask rcnn to build a multi-task learning using multiple fully connected layer.Our goal is to add a mask branch to an existing one-stage object detection model in the same vein as Mask R-CNN does to Faster R-CNN. We want to add another branch on yolo to do the instance segmentation task.

A good feature map backbone is base for tasks like classifier, bbox perdition and instance segmentation. In mask rcnn, the author proposed resnext and pyramid feature network to give fully convolutional network better feature maps.the better backbone alone improves AP by 4 point. And YOLOv3 purposed darknet59 which has deeper networks. It has multi scale layer structure like feature pyramid network. Compare with mask rcnn, YOLOv3 can provide more accurate bbox proposal and higher speed. Hopefully more accuracy bbox can give us higher AP point on instance segmentation task.

3.1. Feature Map

We use darknet59 which proposed in yolov3 as our backbone network to extract feature. The darknet59 framework looks like fig 1.

Darknet59 is a deeper network than previous version of yolo. It contains 106 fully convolutional layers. This deeper network does slower the algorithm, but it gives us better feature maps. It also has multi scale layer structure like feature pyramid network.Darknet59 introduced the idea of ResidualNet, which inclues 3x3 convolution kernel, 1x1 convolution kernel. The newer architecture boasts of residual skip connections, and up sampling. The most salient feature of darknet59 is that it makes detections at three different scales which is similar to the mask rcnns pyramid feature network. By upsampling the lower layer and adding it with previous layer, the darknet59 has the down-to-top and top-to-down structure, which can have better performance on detecting different size object in multiple scales.

It can extract feature in the image faster than ResNet but with high quality feature maps. The multi-scale feature map is merged element-wide by adding. In this way, we can get more high-level semantic information from the later layers, and also get fine-grained information from the pervious layer.

3.2. Bbox prediction and classification

For the bounding box prediction and classification part, we follow the algorithm in yolov3. the output bbox and class label are generated by applying a 1×1 kernel on a feature map. The detection is done by applying 1×1 detection kernels on feature maps of three different sizes at three different scales in the network. The size of the smallest feature map is 13×13, which can detect big object in the image. The scale 2 is 26×26 size feature map, which used to detect medium size object. The last scale size is 52×52, which can detect small objects in the picture very well.

The shape of the detection kernel is $1\times1\times$(B$\times$(5 + C)). Here B is the number of bounding boxes a cell on the feature map can predict, 5 is for the 4 bounding box attributes and one object confidence, and C is the number of classes. In YOLO v3 trained on COCO, B = 3 and C = 80, so the kernel size is $1\times1\times255$. We predict 3 boxes per scale. Applying convolution layers on each scales feature map shrinks it to the size of predition feature map which is $N \times N \times [B \times (4 + 1 + 80)]$.

The first detection is made by the 82nd layer. For the first 81 layers, the image is down sampled by the network, such that the 81st layer has a stride of 32. If we have an image of 416×416, the resultant feature map would be of size 13×13. One detection is made here using the 1×1 detection kernel, giving us a detection feature map of $13\times13\times255$.

Then, the feature map from layer 79 is subjected to a few convolutional layers before being up sampled by 2x to dimensions of 26×26. This feature map is then depth concatenated with the feature map from layer 61. Then the combined feature maps is again subjected a few 1×1 convolutional layers to fuse the features from the earlier layer

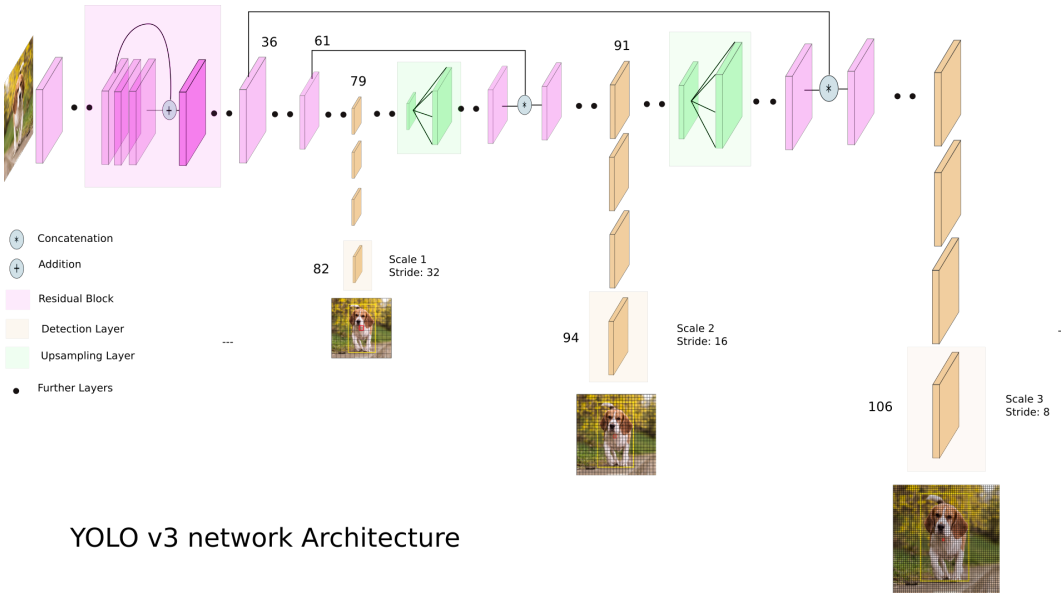

Figure 1. YOLOv3 network Architecture

(61). Then, the second detection is made by the 94th layer, yielding a detection feature map of 26×26×255.

A similar procedure is followed again, where the feature map from layer 91 is subjected to few convolutional layers before being depth concatenated with a feature map from layer 36. Like before, a few 1×1 convolutional layers follow to fuse the information from the previous layer (36). We make the final of the 3 at 106th layer, yielding feature map of size 52×52×255.

In bbox prediction, we use 9 anchor boxes. Three for each scale. Then, arrange the anchors is descending order of a dimension. Assign the three biggest anchors for the first scale , the next three for the second scale, and the last three for the third. The number of bbox prediction is $3 \times (13 \times 13 + 26 \times 26 + 52 \times 52) = 10,648$. Then we do the NMS to select the bbox. After selecting the bbox, we rescale the bbox to original image size as the bbox proposal for mask branch prediction.

3.3. RoI Align

Unlike the RoI pooling layer, RoI Align does not adjust the input proposal from RPN to fit the feature map correctly. It simply takes the object proposal and divides it into a certain number of bins. In each bin, a certain number of points are sampled and value at those points is determined using the bilinear interpolation.

For example, in the figure 2 (taken from the Mask RCNN paper), the 5×5 feature map from the last convolutional layer is shown with dotted lines. Lets consider that the rectangular proposal is overlaid on the feature map at a certain position and is divided into four bins. Now you cannot simply apply the max pooling on each bin because of the misalignment of the region proposal on feature map. So four points are sampled in each bin and values at those points are determined using the bilinear interpolation; four nearest values on the feature map can be used for that purpose. So in each cell, the maximum value among the four is chosen. you get four values from the four cells in this example. So you get the 2×2feature map which corresponds to the region proposal.

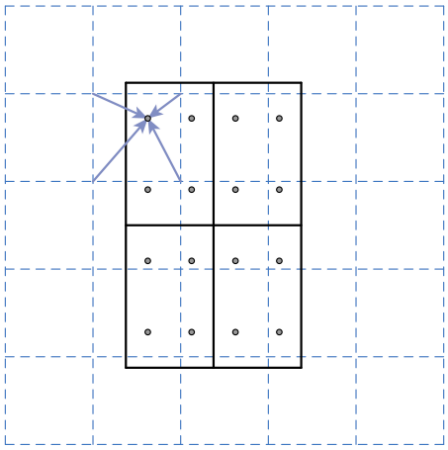

Figure 2. RoIAlign

3.4. FCN mask branch

And we add a fully connected branch on the layer before upsampling which used as our mask prediction feature map.

We can know which layer we should use based on Darknet. We align predicted bbox on that feature map layer. Then put that feature map into mask branch to get mask prediction. The structure of mask branch also followed the design of mask rcnn shows in fig 3. RoIAlign generates a feature map with size of 14×14. Then make a deconvolution to increase the feature map to 28×28, and take sigmoid for each classes. The output is a binary mask with size of 28×28. At last, transform the 28×28 mask using bilinear interpolation to the size of the box in the original image.

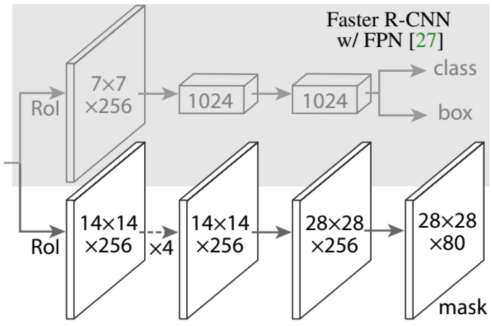

Figure 3. Mask branch prediction

4. Experiment

We first try to train our network end-to-end, but as mask branch is highly depended on the bbox proposed by backbone network, the training tends to be very unstable. So we choose to train our network in 2 steps. In the first stage, we only train our darknet to predict bbox and labels. After the 200+ epoch, we can get about 0.57 AP for bbox. Then we froze the weight of darknet. So our mask branch can get a good RoI proposal. When we train the mask branch, the batch size must set as 1. Because the CropAndResize function can only run on single batch.

When we run the experiment, we found that the mask size 28x28 maybe too large for the darknet. Because the smallest feature map only has 13×13. So we set stride in last layer as 4, so we can get a 7×7 mask as out prediction. The speed of our network reached our goal. One forward for darknet only takes about 0.002s per image and the mask branch takes only 0.008s per image.

5. Conclusion

We achieve high speed with this one-stage instance segmentation framework. As for mask branch, it reaches 125 fps on COCO2014 validation dataset, while the detector part could generate bounding boxes at the speed of 0.002 per image. The processing time is reasonable for a real-time instance segmentation. However, we fall behind state-of-the-art instance segmentation methods in overall performance, especially the accuracy term. This mainly caused by errors in detection part, which is Darknet here.

References

[1] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[2] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[3] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement, 2018.

[4] Kwang-Ju Kim, Pyong-Kun Kim, Yun-Su Chung, and Doo-Hyun Choi. Performance enhancement of yolov3 by adding prediction layers with spatial pyramid pooling for vehicle detection. pages 1–6, 11 2018.

[5] Alexander Kirillov, Evgeny Levinkov, Bjoern Andres, Bogdan Savchynskyy, and Carsten Rother. Instancecut: from edges to instances with multicut. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5008–5017, 2017.

[6] Min Bai and Raquel Urtasun. Deep watershed transform for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5221–5229, 2017.

[7] Xiaodan Liang, Liang Lin, Yunchao Wei, Xiaohui Shen, Jianchao Yang, and Shuicheng Yan. Proposal-free network for instance-level object segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2978–2991, 2018.

[8] Alejandro Newell, Zhiao Huang, and Jia Deng. Associative embedding: End-to-end learning for joint detection and grouping. In *Advances in Neural Information Processing Systems*, pages 2277–2287, 2017.

[9] Adam W Harley, Konstantinos G Derpanis, and Iasonas Kokkinos. Segmentation-aware convolutional networks using local attention masks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5038–5047, 2017.

[10] Bert De Brabandere, Davy Neven, and Luc Van Gool. Semantic instance segmentation with a discriminative loss function. *arXiv preprint arXiv:1708.02551*, 2017.

[11] Alireza Fathi, Zbigniew Wojna, Vivek Rathod, Peng Wang, Hyun Oh Song, Sergio Guadarrama, and Kevin P Murphy. Semantic instance segmentation via deep metric learning. *arXiv preprint arXiv:1703.10277*, 2017.

[12] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8759–8768, 2018.

[13] Zhaojin Huang, Lichao Huang, Yongchao Gong, Chang Huang, and Xinggang Wang. Mask scoring r-cnn, 2019.

[14] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[15] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[16] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[17] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation, 2015.

[18] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation, 2016.

[19] Nikita Dvornik, Konstantin Shmelkov, Julien Mairal, and Cordelia Schmid. Blitznet: A real-time deep network for scene understanding, 2017.

[20] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. Icnet for real-time semantic segmentation on high-resolution images, 2017.

[21] Saumya Jetley, Michael Sapienza, Stuart Golodetz, and Philip H. S. Torr. Straight to shapes: Real-time detection of encoded shapes, 2016.

[22] Jonas Uhrig, Eike Rehder, Bjrn Frhlich, Uwe Franke, and Thomas Brox. Box2pix: Single-shot instance segmentation by assigning pixels to object boxes. pages 292–299, 06 2018.

[23] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding, 2016.

[24] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[25] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.