Presentation for project 1

Nguyen Van Thanh[1]

[1]VinAI Institude, Hanoi, Vietnam

Abstract

This work conduct experiment for instance segmentation and semantic segmentation. For instance detection, we compare 2 popular framwork: Mask RCNN and YOLOv3. For segmentation we demonstrate semantic segmentation with ResNext. Both task is carried on with single class. Finally, we create new dataset for foldable chair recognition and a baseline to help future research.

1. Introduction

Our contributions include:
• Conduct experiment to compare Mask RCNN and YOLOv3 in both mean average precision and inference time.
• Conduct experiment to evaluate semantic segmentation with Feature Pyramid Network
• Build new dataset which contains 2 classes and provide simple yet strong baseline for new dataset.

2. Instance Detection

This section is compare average precision and inference time between MaskRcnn and YoloV3

2.1. Mask RCNN

Mask RCNN is a frame work for object instance segmentation. This is an improvement from Faster-RCNN [8]. Segmentation is seperated as two stage. First stage is to detect region of interest. Second stage is to classofication and segmetation. Main improvement of this frame work is ROI-Align and mask head which produce high quality mask for instance segmentation. For further detail, please consult [2]

2.2. Yolo V3

YOLOv3 is a framework for object detection. This is improvement from YOLO [6]. YOLO is a single stage object detection. Object bounding box is grounded with predefined anchor which reduce computation cost for bounding box localization. After localize object, additional layers are used for classification task. For further detail, please consult [6] [7]

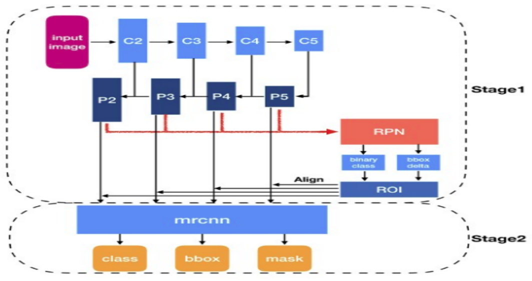

Figure 1: MaskRCNN architecture

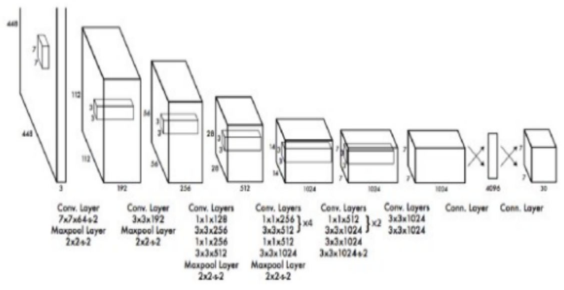
448
448
3
112
112
192
56
56
256
28
28
512
14
14
1024
7
7
1024
7
7
1024
4096
7
7
30
Conv. Layer 7x7x64-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x192 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x128 3x3x256 1x1x256 3x3x512 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x256 3x3x512 }x4 1x1x512 3x3x1024 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x512 3x3x1024 }x2 3x3x1024 3x3x1024-s-2
Conv. Layers 3x3x1024 3x3x1024
Conn. Layer
Conn. Layer

Figure 2: YOLO architecture

3. Semantic Segmentation

We use Squeeze-and-Excitation Networks [4] as backbone. This network is an improve from Resnet [3].This work replace typical residual block with squeeze and excitation block. For further detail please consult [4]

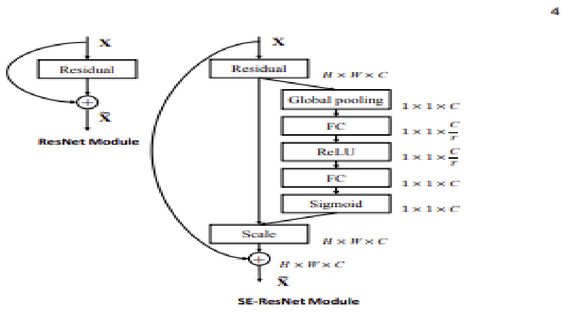

Figure 3: SEnet block

4. Classification

We use the same architecture and hyperparameters as instance segmentation. We don't use simple classification network because they lacks ability to tell where is region of interest. Using traditional classification can mislead the result because instead of focus on object, they might based on enviroment information which reduce ability to generalize of network.

5. Experiments and Results

5.1. Datasets

This dataset is a subset of ADE20K datasets [10]. The only class is used in this work is chair. This dataset contains 2699 images for training and 1524 images for testing. In test set, there are 271 images contain chair while the other doesn't. For classification task, we manually add label (foldable and non foldable) for both train set and test set. If images cann't be reliably labeled, we remove it from dataset. This dataset contains 1426 images contains foldable chair and 1270 images contains non foldable chair in train set. For test set, we have 32 and 239 images for foldable and non foldable class respectively.

5.2. Instance Segmentation

5.2.1 Implementation Detail

MaskRCNN we start from Imagenet [1] pretrained backbone. We warn up learning rate in the first epoch with warn up factor is 0.001 and warn up iteration is 1000. For the next epoch, we use SGD as optimizer with start learning at 0.0005, momentum is 0.9. We fine tune for 30 epochs with batch size of 8.

YOLOv3 For fair comparision, we start from [5]. We start with learning rate of 0.0001 and decay rate of 0.0005. For other hyperparamers, we use the same as MaskRCNN.

We don't finetune hyperparameter which would mislead results and cause unfair comparision.

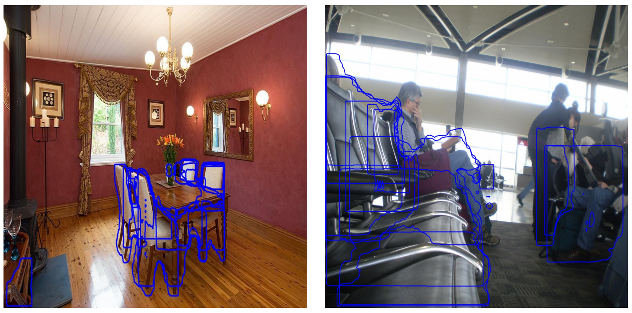

Figure 4: Example of MaskRCNN

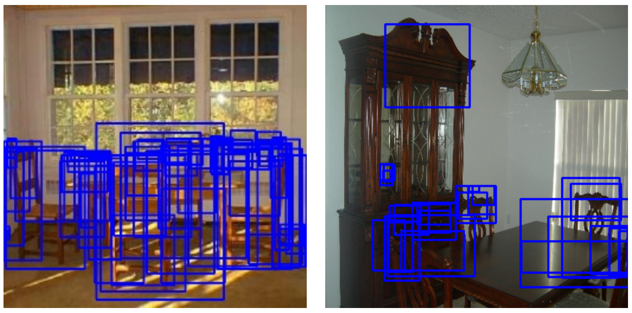

Figure 5: Example of YOLOv3

5.2.2 Resuls

In this work, we focus on averate precision and inference time.

Method	TT	TM	mAP	IT
MaskRCNN	31	16.4	0.556	0.7214
YOLOv3	38	8.3	0.44	0.0559

TT: Train time (min/epoch) . TM: Train memory(Gigabyte): IT: Inference time(sec/images)

In table 1, we report our result for our both method. By comparing MaskRCNN with YOLOv3, we observe that while MaskRCNN outperforms YOLOv3 in mAP and training time, it takes longer time to inference. Faster training time might caused by subtly in implementation because we don't optimize YOLOv3 yet. Mask RCNN also comsume more memory.

5.3. Semantic Segmentation

5.3.1 Implementation detail

We adopt implementation [9] with SE Resnet backbone pretrained on Imagenet [1]. We use Adam as optimizer and learing rate is 0.0001. We train with batch size of 16 for 40 epochs. We use horizontal flip, shift scale rotate, random contrast as augmentation method. Default parameter for these augmentation can be found in source code.

5.3.2 Results

Method	mAP	IOU
SENet	0.078	0.7666

From above result, we see for single class semantics segmentation, our model works quite well. We aren't able to convert MaskRCNN result to mIOU yet.a

5.4. Classification

The implementation is the same as MaskRCNN for instance segmentation.

5.4.1 Results

We provide our result for baseline as an example for future research.

AP	$AP_{0.5}$	$AP_{0.75}$	AP_s	AP_m	AP_l
0.173	0.286	0.187	0.071	0.149	0.241

AR	$AR_{0.5}$	$AR_{0.75}$	AR_s	AR_m	AR_l
0.128	0.430	0.500	0.293	0.424	0.666

6. Conclusion

In this work, we compare MaskRCNN and YOLOv3 for instance detection. The result show that MaskRCNN provide better result with higher cost. We also conduct experiment FPN with SE-Resnet backbone for semantic segmentation. Due to contraints, we haven't finish similar experiment with MaskRCNN for Finally, we provide new dataset and a a baseline for 2 class recognition with MaskRCNN.
To facilitate future research, we release source code at: https://github.com/NguyenVanThanhHust/ComputerVisionClass.git. To help to reproduce result and trained model, please visit:/vinai/thanhnv57/ComputerVisionClass/. For instruction to run, please consult README.MD in github. All experiments are conducted in tessera:quangbd/dc-pytorch:1.4.0-python3.7-cuda10.0-cudnn7-ubuntu16.04.

References

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009. 2

[2] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn, 2018. 1

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015. 1

[4] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu. Squeeze-and-excitation networks, 2019. 1

[5] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár. Microsoft coco: Common objects in context, 2015. 2

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection, 2016. 1

[7] J. Redmon and A. Farhadi. Yolov3: An incremental improvement, 2018. 1

[8] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016. 1

[9] P. Yakubovskiy. Segmentation models pytorch. https://github.com/qubvel/segmentation_models.pytorch, 2020. 2

[10] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Semantic understanding of scenes through the ade20k dataset. *arXiv preprint arXiv:1608.05442*, 2016. 2