Object Detection via Faster-RCNN in Small COCO Dataset

Luo Dacheng(2009853W-IM20-0058)

May 9, 2022

1 Introduction

This object detection project is based on coco2017 dataset. In order to easy to train the model we use the one percent of sample from the train2017 to fit the models. And the annotation file using the standard version for the link(https://pan.baidu.com/s/10WY7vwI0arrZS5R6E4a4Rw). Because we use the SPADE[10] (a type of GAN[2]) to do the data argumentation. We not only use the train/val2017 and train/val2017 annotations but also use panoptic Train/val2017 annotations to generate fake images help training. This project is based on Faster-RCNN[11] model and we also do some test on RetinaNet[6]. We are using ResNet50[3] and ResNet101 as the backbone of the model and we also try to see the performance of Swin Transformer[8] (which is SOTA model in object detection area) in Faster-RCNN. Some different necks, loss functions, learning rate schedule also been used.

2 Environments and Frameworks

Python = 3.8
PyTorch = 1.9.1
MMDetection = 2.2.0

GPU and CUDA:
NVIDIA V100 CUDA = 10.2
GeForce RTX 2070 SUPER CUDA = 11.2
GeForce RTX 3070 CUDA = 11.3 (Thank Wu Xiaohao for sharing)

3 Data and Code

Dataset and Annotation with SPADE(GAN):
https://drive.google.com/file/d/1l2y-m9Jm–TUJIuXLKpFJkcSFe8GD6or/view?usp=sharing
https://drive.google.com/file/d/1r4-38WgGEmAaw0tnAkUtjQJpV2GDrtoZ/view?usp=sharing
MMDetection configuration file:
https://github.com/DACHENGLUO/object-detection

4 Data preparation and Argumentation

4.1 Data preparation

We prepare:
COCO train/val2017 (From COCO)
COCO train2017 small annotations (From guideline)
COCO val2017 annotations (From COCO)
COCO panoptic train2017 annotations (From COCO, used for SPADE data argumentation)

Build COCO small train2017 dataset according to small annotation file.

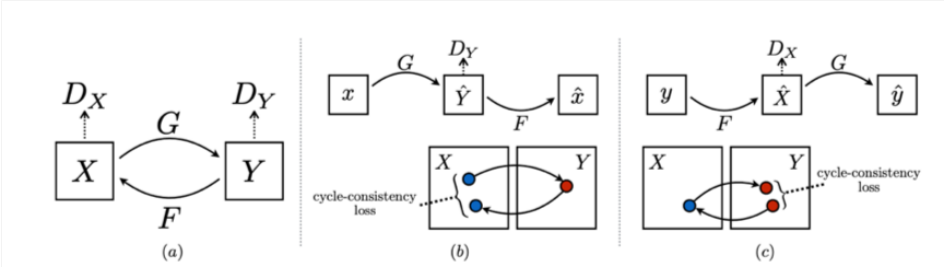

Figure 1: CycleGAN Architecture

.

Figure 2: The images generated by CycleGAN

.

Build COCO small panoptic train2017 dataset according to the small annotation file which is the semantic segmentation masks of the images in small train2017.

4.2 CycleGAN for Argumentation

We use one percent images in the training set of the COCO 2017 Dataset, and the training image is only over 1000 images in the training data. Such small dataset is easy to train compare with original COCO training set, but a few of images is very hard to training a high performance. And according to the requirement, we cannot use other images, so we try to use GAN to generate new image reply on current training images.

The first approach is using the CycleGAN[13] to generate new images that have been style transfer. Like day2night, image2monet. The reason why we use Pix2Pix[4] and CycleGAN technology to generate images is that because we need to do the object detection task after data argumentation and the object position in images generated by Pix2Pix and CycleGAN same as the position in original images, so we need not draw bounding box by our self.Figure 1,2

We can see that the quality of images generated by CycleGAN is pretty good and the position of the objects in the images do not change. However, the changing of the images is confined to texture and color. And we look for more powerful technology.

4.3 SPADE for Argumentation

We use SPADE generate images by semantic segmentation masks of the images from the small training dataset. Because we use the segmentation masks to generate images, so the object position in generated images also should be same with original images theoretically.

Different with the architecture of pix2pixHD that UNet like model. SPADE using Batch Normal-

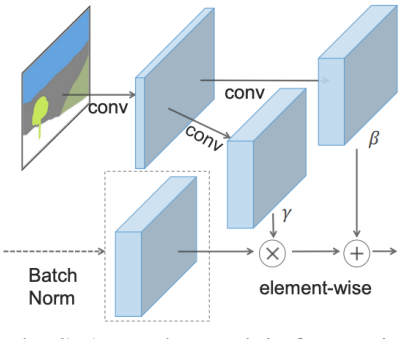

Figure 3: SPADE Architecture

.

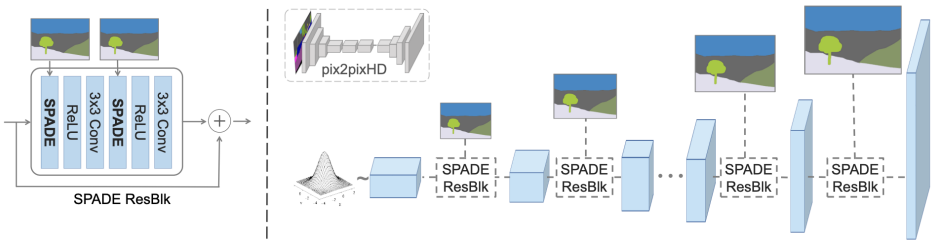

Figure 4: SPADE Architecture

.

ization to normalize the features and using the mean and std of the masks. And using the short block 'SPADE ResBlk'to build the model.Figure3,4

In order to speed up the generate process, I resize the masks to 256*256, and resize the generated images to shape in original images. We can see that the quality of images generated by SPADE is no so well and the face is not so clear. However, the different between the original images and generated images is not limited in texture and color.Figure5

5 Object Detection Models

5.1 Backbone

The backbone of the object detection models are used to extract feature maps from the images. The original faster-RCNN using VGG[12] as the backbone. In recent year, ResNet model is more popular using as the backbone. Because the faster-RCNN is the two stages object detection model and training the two stage model usually speed more time than one stage models, so we using the ResNet50 as the backbone of faster-RCNN and we also compare the performance of the Retina-Net with ResNet101. And in this year, transformers for image recognition[1] show very well performance. The SOTA models in COCO object detection all used swim transformer.Figure 6. So we also test the performance of the faster-RCNN with swim.

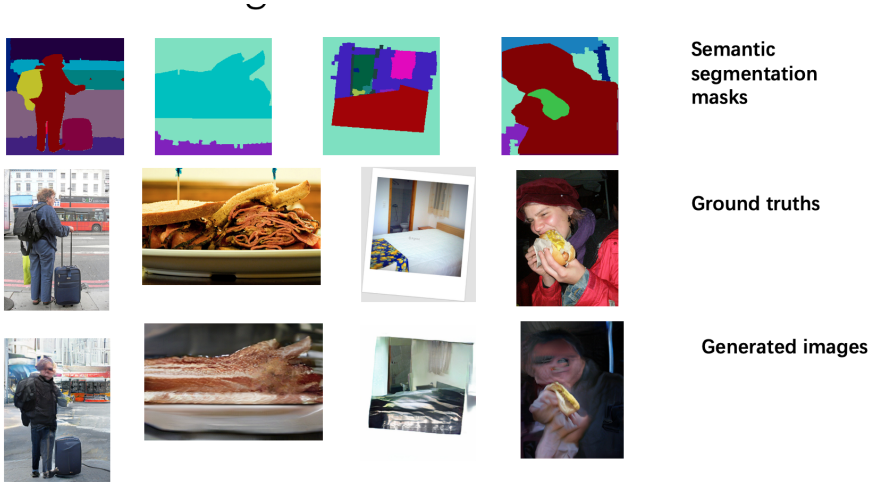

Figure 5: The images generated by SPADE

.

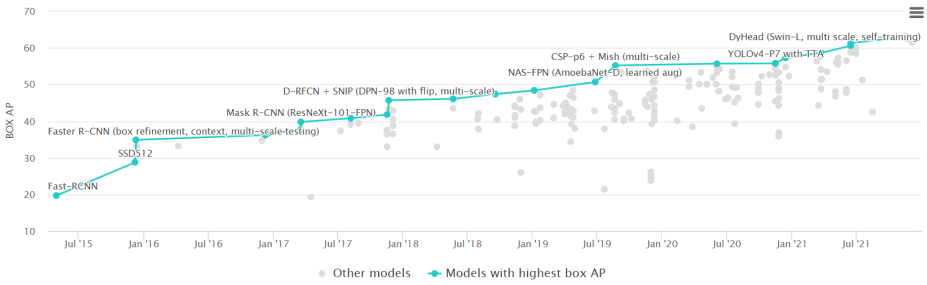

Figure 6: COCO Object Detection

.

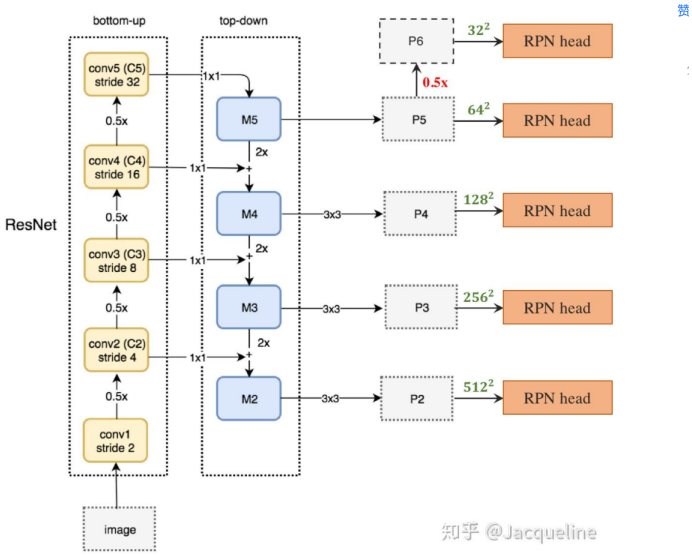

Figure 7: Feature Pyramid Network

.

5.2 Neck

Feature Pyramid Network(FPN[5]).Figure7: Recognizing objects of different sizes is a basic challenge in target detection, and the feature pyramid has always been a basic part of multi-scale target detection. However, due to the large amount of calculation of the feature pyramid, it will slow down the entire detection speed, so most methods In order to detect speed, avoid the use of feature pyramids as much as possible, but only use high-level features to make predictions. Although the high-level features contain rich semantic information, it is difficult to accurately store the position information of the object due to the low resolution. On the contrary, although low-level features have less semantic information, they can accurately contain object location information due to their high resolution. Therefore, if the low-level features and the high-level features can be combined, a target detection system with accurate recognition and positioning can be obtained.

PAFPN[7].Figure8:The proposal of Bottom-up Path Augmentation mainly considers that the shallow features of the network are very important for instance segmentation. It is not difficult to think that the shallow features contain a large number of edge shapes and other features. This is useful for the pixel-level classification task of instance segmentation. Vital role. Therefore, in order to retain more shallow features, the paper introduces Bottom-up Path Augmentation. As we can see from Figure 1, the author uses red and green arrows to indicate how this structure works?

The red arrow indicates that in FPN, because of the bottom-up process, the characteristics of the shallow layer need to pass through dozens or even hundreds of network layers to pass through to the top layer. Of course, this depends on what the BackBone network uses, so it passes through so many. After the layer is transferred, the loss of the characteristic information of the shallow layer will be more serious.

The green arrow table author added a Bottom-up Path Augmentation structure, this structure itself has less than 10 layers, so that the shallow features are connected to P2 through the horizontal in the original FPN, and then passed from P2 to the top layer along the Bottom-up Path Augmentation. , The number of layers passed is less than 10, which can better preserve the shallow feature information. Note that N2 and P2 here represent the same feature map. But N3, N4, N5 and P3, P4, P5 are not the same. In fact, N3, N4, and N5 are the results of the fusion of P3, P4, and P5.

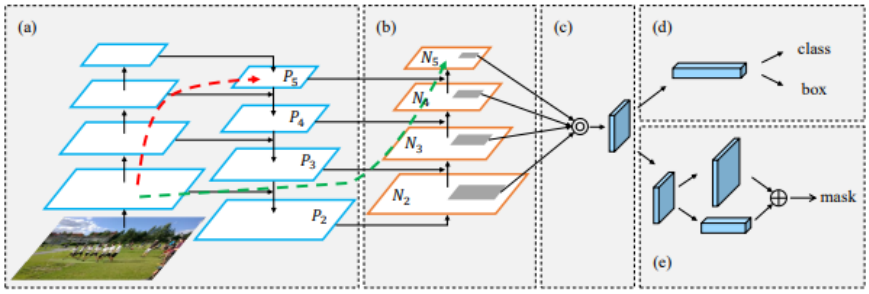

Figure 8: Feature Pyramid Network

.

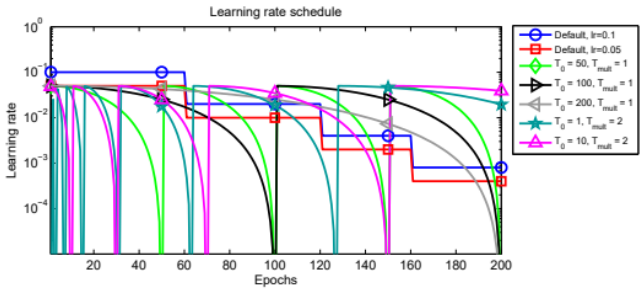

Figure 9: Results

.

5.3 BBox Regression Loss

L1 loss: The derivative of L1 loss to X is constant. Since X represents the difference between the real value and the predicted value, in the later stage of training, X is very small. If the learning rate remains unchanged, the loss function will fluctuate near the stable value and it is difficult to converge to higher accuracy.

GIoU Loss: IoU loss also no perfect.When the prediction frame and the target frame do not intersect, that is, IOU (bbox1, bbox2) = 0, the distance between the two frames cannot be reflected. At this time, the loss function is non differentiable, and IOU loss cannot optimize the non intersection of the two frames.Assuming that the sizes of the prediction box and the target box are determined, as long as the intersection values of the two boxes are determined and their IOU values are the same, the IOU value cannot reflect how the two boxes intersect.

$\text{GIoU} = \text{IoU} - \frac{|C-(A \cup B)|}{|C|}$ GIoU is implemented in the above formula, where C is the circumscribed rectangle of a and B. Subtract the union of a and B by C to get a value, and then subtract this value from IOU of a and B to get the value of GIoU. It can be seen that:The value range of GIoU is [- 1, 1], the maximum value is 1 when the two frames coincide, and the minimum value is - 1 when the two frames are infinite;

Unlike IOU, which only focuses on overlapping areas, GIoU focuses not only on overlapping areas, but also on other non overlapping areas, which can better reflect the coincidence degree of the two.

5.4 Schedule

In own experiment, we find that the we can use more start large learning rate in train Faster-RCNN: (0.01) and we should use the smaller start learning rate in Retina-net to prevent gradient explode. And we also apply warm up in our training process. In order to jump out to the local minimum we apply cosine decay with restart[9]. and the learning rate like:Figure910

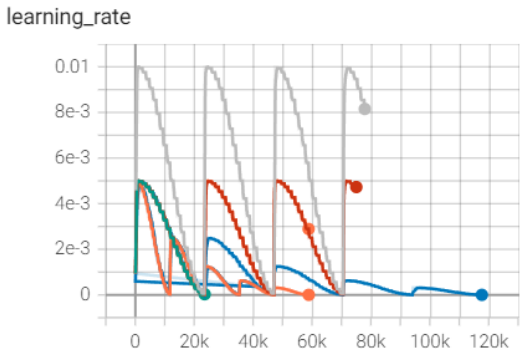

Figure 10: Results

.

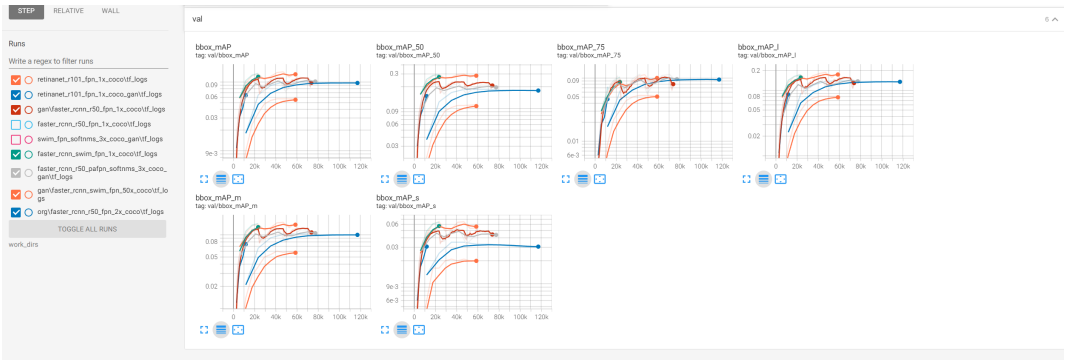

Figure 11: Results

.

6 Results and Conclusion

6.1 Results

We show the results with all the test— see Table 1
The validation mAP in each model.Figure11
The validation mAP in each model.Figure12

6.2 Conclusion

We can see that the data after GAN argumentation show very well performance in training models. and the two stage Faster-RCNN of performance better than the Retina-Net, however the Retina-Net is faster than Faster-RCNN because that just have one stage. And the swin transformer is better than the traditional convolutional networks: ResNet. Some slight change in the neck and loss can help model be great, but this help is limited in such small dataset compare with the data argumentation.

GAN	Model	Schedule	(AP)[IoU=0.50:0.95]	(AP)[IoU=0.50]	(AP)[IoU=0.75]
False	RetinaR101FPNL1lossNMS	100(epochs) Cos	0.057	0.11	0.056
True	RetinaR101FPNL1lossNMS	100(epochs) Cos	0.097	0.177	0.094
False	FasterR50FPNL1lossNMS	50(epochs) Cos	0.091	0.199	0.075
True	FasterR50FPNL1lossNMS	50(epochs) Cos	0.114	0.228	0.1021
True	FasterR50PAFPNIoUSNMS	50(epochs) Cos	0.117	0.222	0.114
True	FasterSwinFPNL1lossNMS	50(epochs) Cos	**0.141**	**0.288**	**0.113**

Table 1: Results

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=1000 ] =
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=1000 ] =
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] =
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] =
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] =
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=300 ] = 0
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=1000 ] =
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] =
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] =
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] =
```

2022-01-08 21:59:30,092 - mmdet - INFO -

category	AP	category	AP	category	AP
person	0.301	bicycle	0.101	car	0.195
motorcycle	0.175	airplane	0.223	bus	0.313
train	0.278	truck	0.085	boat	0.056
traffic light	0.112	fire hydrant	0.329	stop sign	0.388
parking meter	0.149	bench	0.049	bird	0.097
cat	0.269	dog	0.208	horse	0.215
sheep	0.139	cow	0.157	elephant	0.287
bear	0.341	zebra	0.301	giraffe	0.332
backpack	0.040	umbrella	0.093	handbag	0.012
tie	0.043	suitcase	0.042	frisbee	0.213
skis	0.031	snowboard	0.039	sports ball	0.259
kite	0.151	baseball bat	0.061	baseball glove	0.123
skateboard	0.112	surfboard	0.058	tennis racket	0.164
bottle	0.108	wine glass	0.077	cup	0.154
fork	0.010	knife	0.004	spoon	0.004
bowl	0.132	banana	0.043	apple	0.022
sandwich	0.075	orange	0.097	broccoli	0.042
carrot	0.037	hot dog	0.080	pizza	0.217
donut	0.165	cake	0.086	chair	0.057
couch	0.111	potted plant	0.057	bed	0.191
dining table	0.113	toilet	0.211	tv	0.258
laptop	0.217	mouse	0.300	remote	0.043
keyboard	0.169	cell phone	0.097	microwave	0.154
oven	0.087	toaster	0.022	sink	0.109
refrigerator	0.146	book	0.037	clock	0.296
vase	0.070	scissors	0.000	teddy bear	0.135
hair drier	0.000	toothbrush	0.007	None	None

Figure 12: Results

References

[1] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[2] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[4] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, 2017.

[5] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[6] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[7] 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.

[8] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021.

[9] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[10] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2337–2346, 2019.

[11] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015.

[12] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[13] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.