Dilated Feature Pyramid Network for image classification

2021741070 이의준

EuiJoon Lee

1. 서 론

희망 진로가 방위산업 분야라 군사 영역에 기여할 수 있는 각 국의 주력 전차를 식별하는 classifier를 만들고 자 한다. 각 국가의 전차의 외형은 매우 유사하기 때문에 특징 추출에 특화된 feature extractor를 찾아야 했다.

FPN(Feature Pyramid Network)이 Object Detection에서 사용되는 Feature extractor임을 알게 되었다.

FPN의 기존 convolution 방식과 down-sampling 방식은 feature map에서의 물체 외곽이 불명확해지는 문제와 dowm-sampling을 진행할 때 해상도가 거칠어 지는 문제가 있다. 이에 반해 Dilation은 feature extract에 있어서 넓은 수용영역을 지녀 해상도가 급격히 떨어지는 것을 막고 물체의 경계를 보존하는 쪽으로 feature map을 만든다. 이에 기존 FPN에 Dilated Convolution을 적용하여 기존 FPN보다 좋은 성능을 가진 Classifier를 만드는 것이 목표이다.

FPN에 FCN(Fully-Contected-Network)를 결합하여 Classifier를 만든 후 Dilation Convolution을 적용하여 Dilated-FPN을 만들어 ResNet, FPN, Dilated-FPN끼리 학습 결과를 비교하려고 한다.

2. 본 론

2.1 프로젝트 환경

2.1.1 Framework

-Workspace: Aanaconda3의 jupyter note book사용(python3)

-Deep Learning framework: pytorch-1.13.1, Cuda-11.7

2.1.2 GPU - NVIDA GeForce RTX 3060 Laptop GPU

-GPU 메모리: 13.7GB

-전용 GPU 메모리: 6.00GB

-공유 GPU 메모리: 7.70GB

--본 프로젝트에서는 전용 GPU 메모리만 사용함—

2.2 Image-Data & Preprocessing

사용한 데이터는 Kaggle의 Battle Tank 데이터셋(256x256)을 사용하였고 여러 Class 중에 “Challenger2”(train set: 670장 / test set: 168장), ”K2(Black Panther)”,(train set: 674장 / test set: 169장) ,“Leopard2” (train set: 677장 / test set: 169장)를 사용하였다. 이미지에 정규화를 거쳐 각 픽셀값들을 0~1 사이 값으로 전환해주었다.

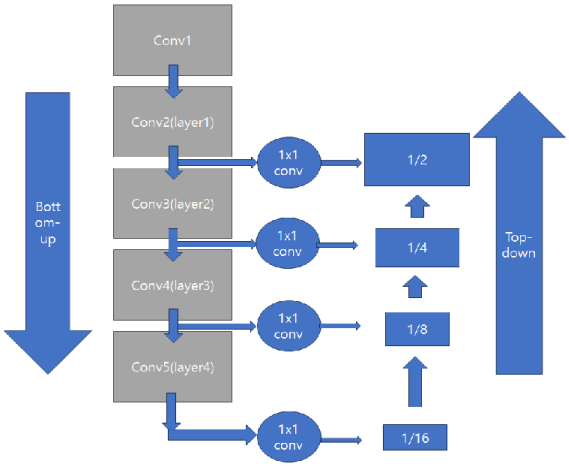
[Fig. 2.2-1] Challenger2 data set

[Fig. 2.2-2] K2(Black Panther) data set

[Fig. 2.2-3] Leopard2 data set

2.3 Dilated Feature Pyramid Network

2.3.1 FPN(Feature Pyramid Network)



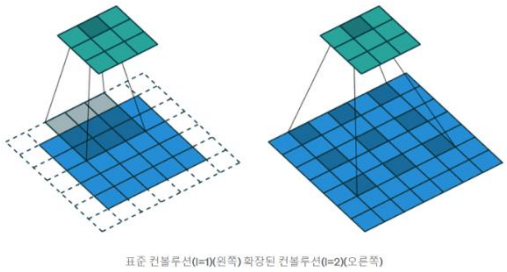
[Fig. 2.3.1] FPN structure

Bottom – Up 방식에서는 Resnet50 구조를 사용하여 이미지 크기를 downscaling한다. 제한된 GPU로 인해 input 채널 수는 [32, 64, 128, 256] (layer1, layer2, layer3, layer4)로 세팅했다. layer1,2,3,4 각각 거쳐 나온 4개의 feature map에 1x1 conv를 취하여 256 채널로 변경한다. 마지막 layer4 결과를 up sampling하여 layer3 결과 feature map과 더한다. up sampling과 feature map 결합을 반복하여 layer1 결과에 더한다. 결과 feature map으로 256 x (원본 이미지 width-size) /2 x (원본 이미지 height-size)/2 가 나온다.

결과 feature map을 256 x 1 x 1로 flatten 처리한 후 Linear(input=256, output=3)을 연결하였다.

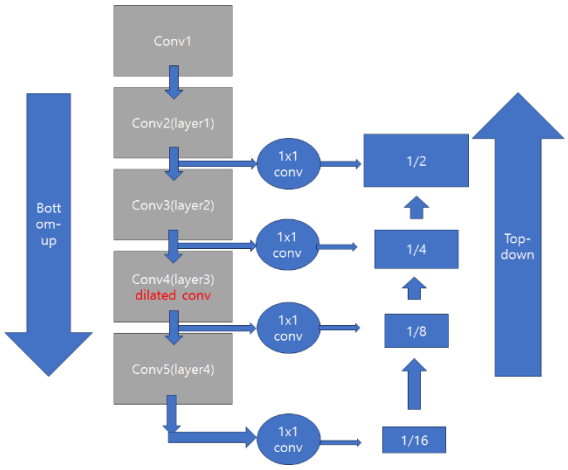
2.3.2 DFPN(Dilated Feature Pyramid Network)

Dilated convolution은 기존 convolution과는 다르게 conv 연산 시 dilation rate에 따라 커널 픽셀 사이 간격을 rate만큼 띄워서 conv 연산을 진행하는 방식이다.



[Fig. 2.3.2-1] Dilated Convoultion

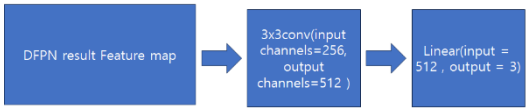
기존 FPN 구조에서 layer3의 convolution 기법에 dilation 을 적용하였다. 기존 Dilated convolution Network에는 ResNet의 conv4, conv5 단계에 stride=0으로 세팅한 후 convolution에 dilation rate: 2 , dilation rate: 4 를 적용하였다. 본 프로젝트의 DFPN에는 conv4(layer3) 단계에 dilation rate: 2인 dilated convolution와 padding rate:2, 첫번째 Bottle-neck block에 stride = 2 를 적용하였다.



[Fig. 2.3.2-2] DFPN basic structure

2.3.3 DFPN_A(Dilated Feature Pyramid Network_A)

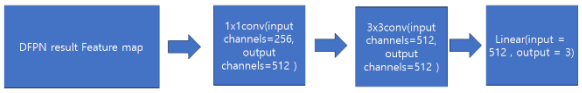
기존 DFPN 결과에 3x3Conv(input channels = 256, output channels = 512, padding=1, stride=1)를 취한 뒤 Linear(input = 512, output =3)을 취한 방식이다.



[Fig. 2.3.3] DFPN_A structure

2.3.4 DFPN_B(Dilated Feature Pyramid Network_B)

기존 DFPN 결과에 1x1Conv(input channels = 256, output channels = 512, padding=1, stride=1)를 취한 뒤 3x3Conv(input channels = 512, output channels = 512, padding=1, stride=1) + Linear(input = 512, output =3)을 취한 방식이다.



[Fig. 2.3.4] DFPN_B structure

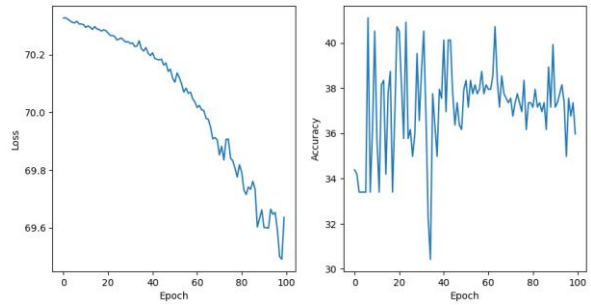
3. 실험 결과

[Table 1] model parameter

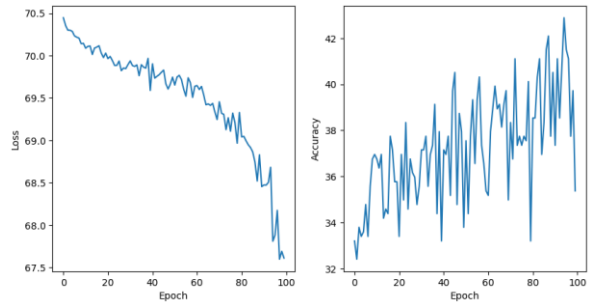
ResNet50	Total params	5,905,667
	Trainable params	5,905,667
	Non-trainable params	0
FPN	Total params	8,152,355
	Trainable params	8,152,355
	Non-trainable params	0
Dilated FPN	Total params	8,156,195
	Trainable params	8,156,195
	Non-trainable params	0
Dilated FPN_A	Total params	9,337,123
	Trainable params	9,337,123
	Non-trainable params	0
Dilated FPN_B	Total params	10,648,355
	Trainable params	10,648,355
	Non-trainable params	0

[Table 2] epochs = 100, batch size = 32

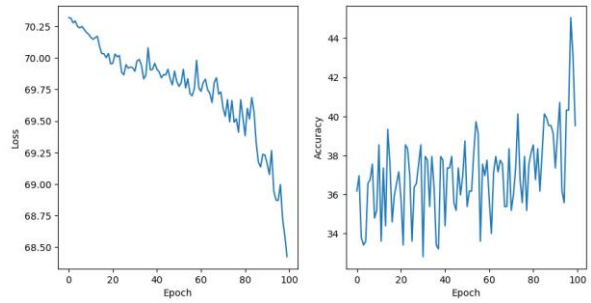
ResNet50	loss	69.664
	test accuarcy	38 %
FPN	loss	67.888
	test accuarcy	43 %
Dilated FPN	loss	67.810
	test accuarcy	42 %
Dilated FPN_A	loss	68.724
	test accuarcy	45 %
Dilated FPN_B	loss	69.421
	test accuarcy	38 %



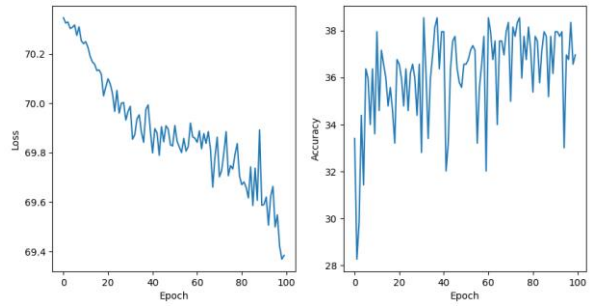
[Fig. 3-1] ResNet50 result



[Fig. 3-2] Dilated FPN result



[Fig. 3-3] Dilated FPN_A result



[Fig. 3-4] Dilated FPN_B result

4. 결 론

google Colab에서 동일 조건의 학습 결과 와는 다르게 학습 결과가 좋지는 못하다. GPU의 성능 문제가 있는 것 같다. 우선 ResNet50과 FPN은 구조적 차이 때문에 파라미터 수의 차이가 클 수밖에 없다. ResNet50 구조를 사용한 FPN의 test 결과가 ResNet50보다 좋게 나왔음을 확인할 수 있었다. 기존 FPN에 Dilation convolution을 적용한 Dilated_FPN에서 버전 A가 Dilated FPN과 파라미터 차이가 비슷하며 test 결과가 3% 더 좋게 나왔다. 오히려 1 by 1 conv을 취한 B의 성능이 상당히 안 좋게 나왔다. Dilation convolution을 stride를 대체하는 기능은 빼고 feature extract 기능면에서 기존 convolution net보다 좋은 성능을 보였음을 확인할 수 있었다.

References

[1] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[2] F. Yu, V. Koltun and T. Funkhouser, "Dilated Residual Networks," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 636-644, doi: 10.1109/CVPR.2017.75.

[3] T. -Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan and S. Belongie, "Feature Pyramid Networks for Object Detection," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 936-944, doi: 10.1109/CVPR.2017.106.

[github]

[1] https://github.com/kuangliu/pytorch-fpn.git

[2] https://github.com/inhovation97/Image_classification_pipeline_

Project.git

[3] https://github.com/kentaroy47/faster-rcnn.pytorch_resnet50.git

[Fig. 2.3.2-1]
https://miro.medium.com/v2/resize:fit:1100/1*btockft7dtKyzwXqfq70_w.gif