CoE202 Final Project Report

20180109 김수빈

1. Introduction

Sematic Segmentation 은 주어진 이미지 data에 대해 픽셀들이 어느 클래스에 속해져 있는지 결정하는 것을 의미한다. 이는 일종의 classification 으로, final project 에서는 주어진 10개의 class에 대한 분류를 시행했다. Final Project를 시행하는 동안 UNet, ResNet, PSPNet, DeepLabV3 등에 대한 모델에 대해 dataset에 맞는 parameter, epoch, learning rate을 조절하며 진행하였다. 그 중에서 encoding, decoding layer 4개를 사용한 UNet과 Atrous Spatial Pyramid Pooling을 사용한 모델이 validation set에 대한 mIoU가 약 0.588 로 가장 좋은 성능을 보였다. 이 리포트에서는 직접 작성한 UNet와 DeepLabV3 (논문의 모델과는 다를 수 있다)을 소개할 예정이다.

2. Dataset & Data augmentation

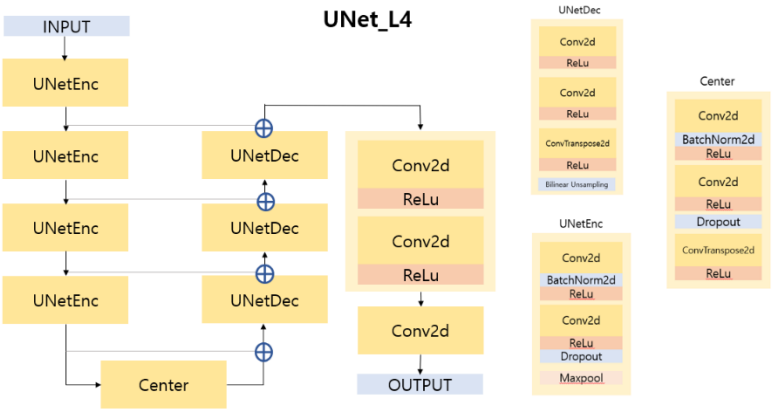
주어진 Dataset에는 1000개의 이미지와 각각에 맞는 라벨이 주어진다. Pretrain process로는 data augmentation을 시도했다. 시도한 augmentation으로는 torchvision.transforms에 내장되어 있는 resize, hflip, vflip을 시도했다. 그러나 Data augmentation을 적용한 training set은 모든 모델에 의해 mIoU를 0.2 이상을 넘지 못했다. 따라서 data augmentation을 코드에서 제외했다. 이는 augment한 data의 픽셀을 augment한 label의 픽셀에 대해 제대로 대응하지 못했거나, augment하는 과정에서 픽셀의 위치변화가 오차를 증가시킨 것으로 추측된다. 시간이 더 있다면 추후에 보다 더 정확한 원인을 밝히고 data augmentation을 많이 써서 더 좋은 성능을 이끌어낸 AlexNet이나 마찬가지로 data augmentation으로 좋은 성능을 냈다고 알려진 VGGNet을 등에 대해 공부해보고자 한다.

3. Model Summary

Semantic Segmentation에 쓰이는 대표적인 검증된 모델들이 이미 존재한다. GoogLeNet, VGGNet, ResNet, PSPNet, AlexNet 등 수많은 모델들이 있다. 그 중에서 sample model로 주어진 UNet과 DeepLabV3의 구조와 유사하게 모델을 짰다. 많은 모델 중 UNet과 DeepLabV3를 선택한 이유는 UNet이 주어진 sample model에서 발전시키기 가장 용이했으며, 또한 DeepLabV3는 UNet의 구조를 응용한 모델로서 성능이 가장 좋은 모델 중 하나로 알려져 있기 때문이다.

3.1 UNet

UNet_L4.py의 UNet의 구조는 [그림1]과 같다.



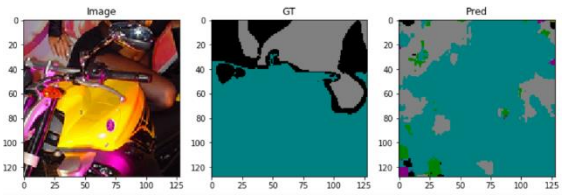
[그림1] UNet_L4의 구조

```
enc1 output:  torch.Size([8, 64, 64, 64])
enc2 output:  torch.Size([8, 128, 32, 32])
enc3 output:  torch.Size([8, 256, 16, 16])
enc4 output:  torch.Size([8, 512, 8, 8])
center output:  torch.Size([8, 512, 8, 8])
dec1 output:  torch.Size([8, 256, 16, 16])
dec2 output:  torch.Size([8, 128, 32, 32])
dec3 output:  torch.Size([8, 64, 64, 64])
dec4 output:  torch.Size([8, 64, 60, 60])
output:  torch.Size([8, 10, 128, 128])
```

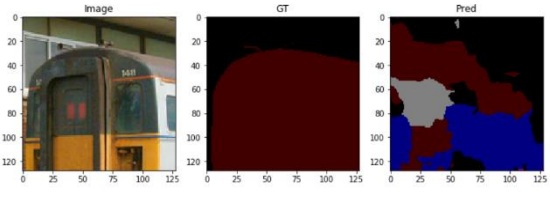
[그림2] UNet_L4 layout shapes

UNet은 전체적으로 autoencoder의 대칭적인 구조를 이루고 있는데, UNetEnc는 encoder layer 이고 UNetDec은 decoder layer이다. 왼쪽에 위치한 encoder layer들이 contracting path의 역할이고 오른쪽에 위치한 decoder layer들이 expanding path역할을 해준다. 이는 contracting path에서 data의 context를 포착하여 이를 expanding path에서 복원된 데이터들과 함께 다음 layer에 넣어주어 픽셀 간에 서로 가지는 관계성을 더욱 잘 파악할 수 있도록 도와준다. 또한, decoder layer 에서는 bilinear unsampling을 해주는데, 이는 linear interpolation을 각각 x축, y축에 대해 적용해주는 방법이다. Bilinear unsampling은 input보다 더 큰 output을 추출해내려고 할 때 값이 비는 pixel들이 있을 때 값을 채워줄 수 있는 방법이다. Decoder layer는 input보다 큰 output을 추출하는 과정이므로 이를 보완해주고자 적용시켜주었다.

Semantic segmentation의 결과가 잘 나오기 위해서는 결국 Classification을 효과적으로 할 수 있어야 하고, 이는 픽셀이 서로 가지는 관계를 통해 파악하는 것이라고 생각했고, 그렇기 때문에 주어진 baseline model에 BatchNorm2d를 각각 UNetEnc 와 Center에 넣어주었다. UNetDec에 batch normalization을 해주지 않은 이유는 expanding path이기 때문에 Batch Normalization으로 인한 오차의 증가가 염려되었기 때문이다. 실제로 UNetDec에 Batch Normalization을 넣었을 때 성능이 떨어지는 것을 확인하였다. 그 외에 Dropout을 추가하여 보다 더 정밀한 모델을 구성하고자 하였다. [그림1]의 구조에서 각각 UNetEnc, UNetDec를 한 layer씩 더 추가한 것이 UNet_L5 모델의 구조이다.



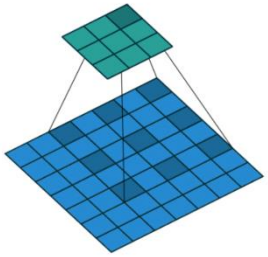
[그림 4] UNet_L5를 이용한 결과 예시



[그림 3] UNet_L4를 이용한 결과 예시

3.2 Atrous Spatial Pyramid Pooling

UNet_L4, UNet_L5의 validation set에 대한 mIoU가 높지 않았다. 이를 보완하기 위해 Atros

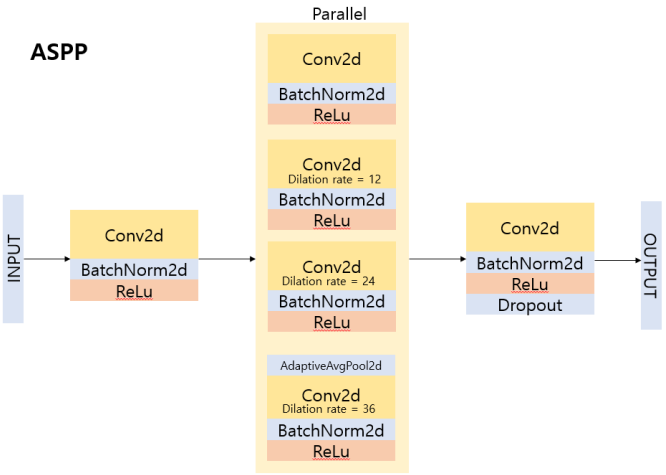


Spatial Pyramid Pooling(ASPP)을 사용한다. Atous layer는 필터 내부에 빈 공간은 둔 채로 계산하는 방법으로, 기존의 convolution layer 와 같은 성능을 보이면서도 보다 더 넓은 영역을 탐색할 수 있다는 것이 장점이다. Semantic segmentation에서는 한

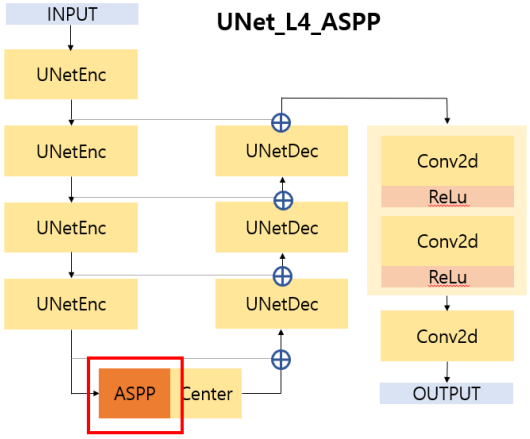
필터가 인식할 수 있는 영역의 넓이가 너무 작아지면 정보를 지나치게 추상화하는 위험성이 있기 때문에 필터가 볼 수 있는 영역이 넓

[그림5] Atrous layer

어진다는 것은 장점이다. 여기서 pyramid pooling이란 빈 공간의 비율이 다른 Atrous layer들을 병렬적으로 계산하여 합쳐주는 기법이다.

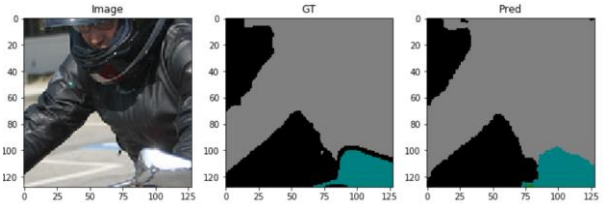


[그림6] ASPP의 구조

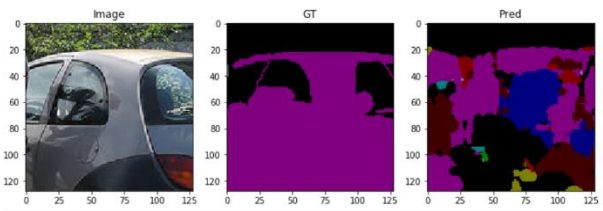


[그림6] UNet_L4_ASPP의 구조

[그림6]에서와 같이 ASPP는 서로 다른 dilation rate를 가진 convolution layer (atrous layer)를 모두 동일한 input에 대해 시행한다. 그 후 각 layer들의 output을 이어 붙여 이를 하나의 convolution layer을 통과시키므로 인해 각 atrous layer에서 뽑은 context를 합쳐준다. 이것을 응용하여 ASPP layer를 UNet_L4와 UNet_L5에 대입하였다. Atrous layer를 center 전에만 넣은 이유는 dilation rate(Atrous layer)를 모든 layer에 적용하면 지나치게 계산량이 많아지게 때문이다. 실제로 **3.3**에 설명할 Deeplab 모델에서 모든 layer에 dilation rate을 넣었지만 training 시간이 지나치게 오래 걸리고, 성능도 UNet_L4_ASPP보다는 안 좋게 나온 것으로 확인되었다.



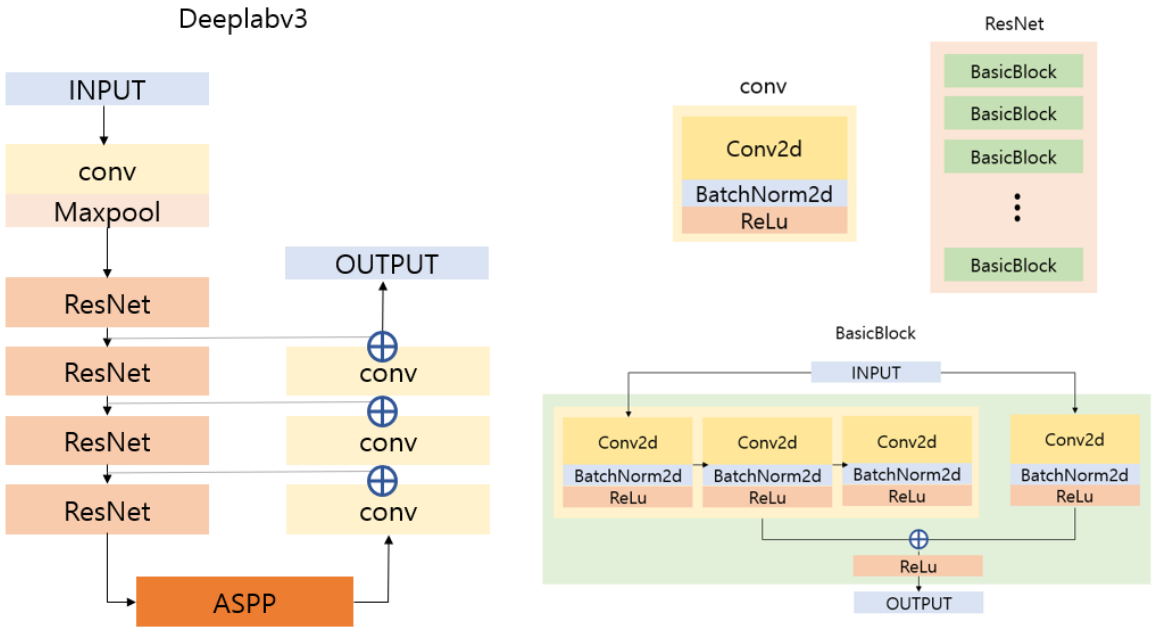
[그림7] UNet_L4_ASPP를 이용한 결과 예시



[그림8] UNet_L5_ASPP를 이용한 결과 예시

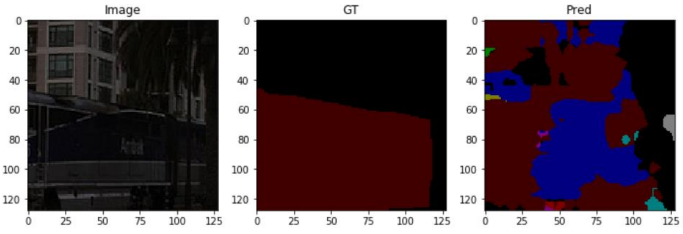
3.3 DeepLabV3

Deeplab 은 V1, V2, V3, V3+ 까지 총 네 번에 걸쳐 수정된 모델이 발표되었다. 그 중에서 deeplabv3.py에서 모델링한 DeeplabV3은 UNet과 비슷하게 contracting path(encoder), expanding path(decoder) 두 부분으로 나뉘고, encoder부분은 Atrous convolution을 적용한 ResNet, center layer는 ASPP, decoder 부분은 interpolating과 convolution layer를 사용하였다. DeepLabV3 모델을 implement할 때는 UNet_L4에서 더 발전한 구조로서 다양한 dilation rate과 convolution layer의 개수를 증가시키고자 하였다. 그러나 layer의 개수를 증가시키는 것은 overfitting을 유발할 가능성이 높았고, overfitting을 최대한 batch normalization과 dropout으로 조절하려고 했지만 효과적이지 않았던 것 같다.



[그림9] deeplabv3의 구조

Deeplabv3은 UNet 모델과 마찬가지로 contracting path과 expanding path로 나뉘고, contracting path에서의 output을 expanding path의 output에 다시 넣어주었다. 각 ResNet layer(encoder layer)는 rate을 각각 다르게 주었다.



[그림10] deeplabv3를 이용한 결과 예시

3.4 limitation

DeepLabV3 모델은 mIoU를 높이기 위해서 최대한 모사해서 짰지만, 모든 과정의 parameters를 꼼꼼히 체크하지 않았다는 아쉬움이 남는다. 또한, 주어진 데이터의 크기 [(batch size), 3, 128, 128]에 맞추기 위해 parameter를 조정해주었는데, convolution layer에 적용하는 kernel의 크기, stride의 크기에 대한 가짓수가 너무나 많기 때문에 수많은 layer들의 parameter를 모두 미세하게 조정하기 힘들었다. 조금 더 parameter를 바꿔가면서 training할 수 있었다면 더 좋은 결과가 나왔을 것이라는 아쉬움이 남는다. 그러나 DeepLabV3는 모델 training 하는데 걸리는 시간이 한 번 epoch 하는 데 4분씩 걸릴 정도로 지나치게 길다. 그렇기 때문에 시간적, 물리적(GPU)의 자원으로 인해 여러 번 돌리기 힘들었다. 마찬가지로 UNet_L4_ASPP와 UNet_L5_ASPP 에서도 dilation rate을 조금 더 다양하게 변화시키고 싶었지만, 시간적, 물리적 자원의 부족으로 인해 parameter를 다양하게 실험하지 못하였다. Batch size가 8을 초과하면 CUDA OUT OF MEMORY error 가 뜨기 때문에, Batch size를 8에서 4 사이로 조절해가며 모델을 train 했다.

4. Results

	train_loss	valid_loss	train_mIoU	valid_mIoU	best_mIoU	test_mIoU
UNet_L4	5.473	45.9812	0.8594	0.2003	0.2067	0.192
UNet_L5	0.2714	88.3198	0.8854	0.175	0.1974	0.1858
Unet_L4_ASPP	3.7775	5.2179	0.8704	0.7469	0.7947	0.5884
Unet_L5_ASPP	1.0172	46.6497	0.8814	0.2163	0.2362	0.1933
Deeplabv3	151.6544	22.094	0.2035	0.1079	0.1885	0.1745

[표1] UNet_L4, UNet_L5, UNet_L4_ASPP, UNet_L5_ASPP, Deeplabv3에 대한 mIoU와 Loss

전체적으로 UNet_L4 계열의 모델이 좋은 성능을 보인다는 것을 알 수 있다. UNet_L4보다 layer 개수를 많이 쓴 UNet_L5과 Deeplabv3의 성능이 떨어지는 것은 layer와 모델 복잡도가 증가할 수록 overfitting이 발생하기 때문일 것으로 추정된다. 또한, 성능이 낮은 모델들의 output 이미지([그림4], [그림8], [그림10])를 ground truth과 비교해보면 ground truth 보다 더 많은 class들을 인식했다는 것을 알 수 있고, 역시 overfitting 때문인 것으로 추정된다. 이를 방지하기 위해 Batch normalization, Dropout, learning rate, momentum 등을 이용하여 Overfitting을 방지하려고 했지만 한계가 존재했다. 보다 더 정밀하게 parameter를 조절하고 정교하게 layer를 짠다면 좋은 성능을 기대할 수 있을 것이다.

또한, UNet_L4 보다 UNet_L4_ASPP가 성능이 좋고, UNet_L5 보다 UNet_L5_ASPP가 성능이 좋다는 것을 알 수 있다. 이는 ASPP의 dilation convolution layer를 통해 더 넓은 영역에서 context를 파악하고 누적한 것이 classification에 효과적이었음을 의미한다.

결론적으로 가장 좋은 성능을 보인 것은 UNet_L4_ASPP이다.

5. Reference

-Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." *International Conference on Medical image computing and computer-assisted intervention*. Springer, Cham, 2015.

- Chen, Liang-Chieh, et al. "Semantic image segmentation with deep convolutional nets and fully connected crfs." *arXiv preprint arXiv:1412.7062* (2014).

- Chen, Liang-Chieh, et al. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs." *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017): 834-848.

- Chen, Liang-Chieh, et al. "Rethinking atrous convolution for semantic image segmentation." *arXiv preprint arXiv:1706.05587* (2017).

-Chen, Liang-Chieh, et al. "Rethinking atrous convolution for semantic image segmentation. arXiv 2017." *arXiv preprint arXiv:1706.05587* (2019).