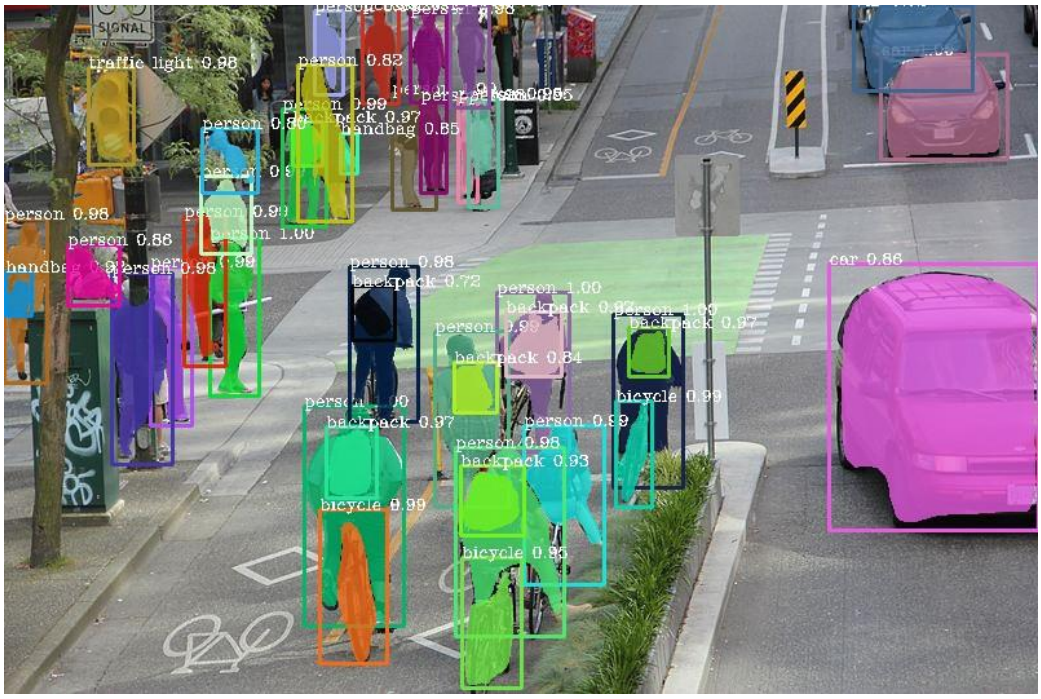
이미지 세그멘테이션 (Image Segmentation)



➔ 이미지 상에서 객체를 탐지할 때 실제 객체의 영역까지 찾는 것

➔ 픽셀 별로 클래스 분류를 하는 것

semantic segmentation은 픽셀 단위로 객체와 배경을 분류하는데 종류까지 구분합니다.
instance segmentation은 픽셀 단위로 객체만 분류합니다. (종류는 구분 x)
panoptic segmentation은 픽셀 단위로 객체와 배경을 분류합니다. (종류는 구분 x)

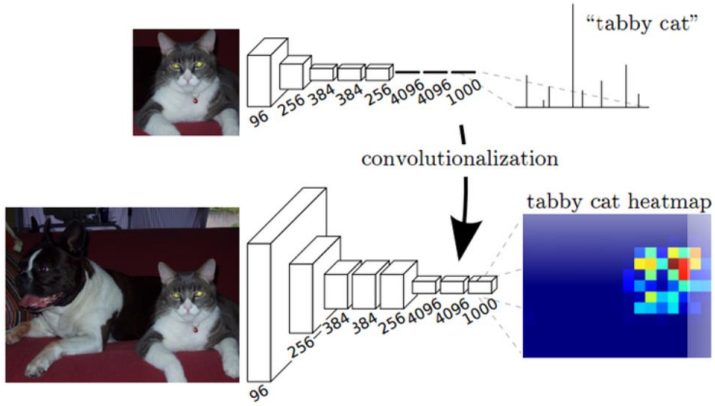
지금부터 딥러닝 기반 Semantic Segmentation에 대해 알아보겠다.

FCN, U-Net, DeepLab(v1~v3+)에 대해 간단히 리뷰할 것이다.

Fully Convolutional Networks for Semantic Segmentation

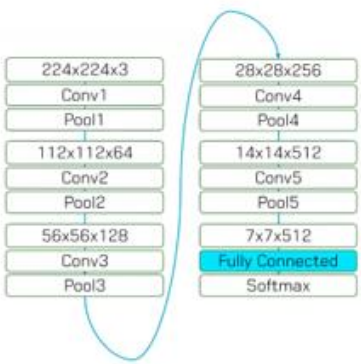
1. Convolutionlization

:모든 layer를 conv layer로 구성하는 방법

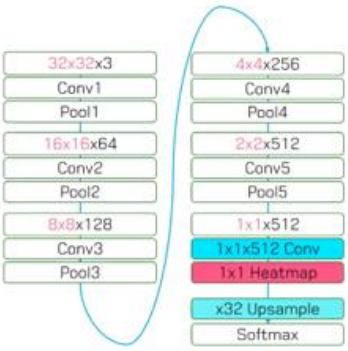


conv layer는 합성곱 연산을 이용해 2d feature를 유지하면서 연산하기 때문에 FC Layer와 다르게 이미지의 위치 정보도 같이 담고 있다.

위처럼 모든 layer를 conv layer로만 구성하면 network에서 위치 정보가 소실되지 않고 유지된다.



(1)



(2)

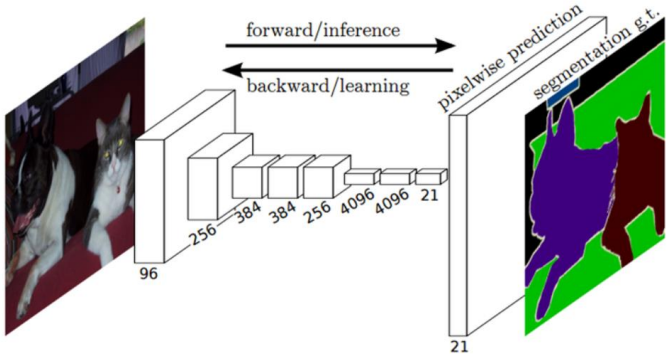
VGG16을 기준으로 추가 설명하면 원래 FC Layer부분이 1x1 Conv로 변경된 것을 볼 수 있는데 FC Layer가 사라짐으로써 어떤 input size가 오더라도 결과를 산출할 수 있다. (2) 예시에서 파란색 부분인 1x1 conv를 사용하기 전까지 결과를 보면 224x224x3이 입력으로 쓰여 7x7x512가 된 것을 볼 수 있다. 만약 32x32x3이 오면 1x1x512이라는 결과가 나올 것이다. 그런데 만약 (1) 예시처럼 파란색 부분이 FC Layer이면 고정된 벡터를 입력으로 받으므로 3차원 feature map의 형태도 고정되어야 될 것이고 결국 input size가 동일해야 한다.

중요. conv layer 를 거치고 나서 얻게 된 마지막 특성맵의 개수는, 훈련된 클래스의 개수와 동일

- 만약, 클래스가 5 개이며, (1, 1) 픽셀에 해당하는 클래스당 확률값들이 강아지 0.45, 고양이 0.94, 나무 0.02, 컴퓨터 0.05, 호랑이 0.21 라면
- 0.94 로 가장 높은 확률을 산출한 고양이 클래스를 (1, 1) 픽셀의 클래스로 예측하는 것

2. Upsampling

:pooling으로 줄어든 이미지를 입력 영상과 같은 크기로 만들어주는 과정



위 네트워크 구조를 보면 feature의 공간의 크기인 spatial size가 작아지는 것을 볼 수 있다. 이는 layer 사이에 존재하는 pooling layer때문인데 pooling은 feature의 size를 줄이는 방법으로 연산량을 줄이고 receptive field를 크게 하는 효과가 있다.

(pooling 후 convolution을 하면 이전과 같은 필터 사이즈여도 기존 인풋 이미지 내 더 큰 영역을 파악하게 되는데 이를 receptive field가 크다고 표현한다.)

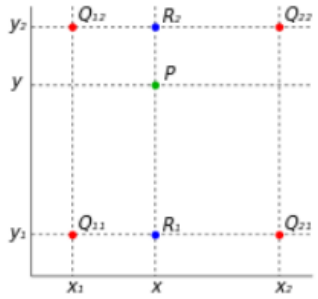
receptive field가 커졌다는 것은 한 픽셀을 계산하기 위해 더 넓은 범위의 픽셀 값들을 고려하여 연산을 했다는 것이다. (-> 한 픽셀이 큰 영역 정보를 요약하게 됨)

semantic segmentation에서 물체를 파악할 때 주변 픽셀만 봐서 판단할 수 있는 물체도 있지만 전체적인 영상을 봐야 판단하기 쉬운 물체도 있기 때문에 receptive field가 큰 것이 유리하다.

하지만 정답 영상(입력 영상과 같은 크기인데 픽셀이 클래스별로 색칠된 영상)과 비교를 통해 학습하기 위해 pooling으로 줄어든 이미지를 입력 영상과 같은 크기로 만들어 줄 필요가 있다. 그래서 upsampling이라는 방법을 사용한다.

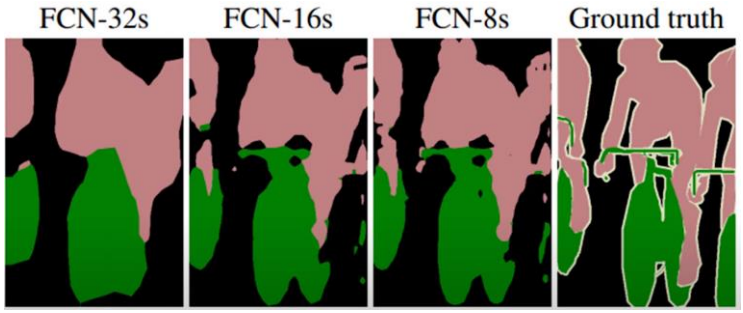
FCN 논문에서는 bilinear interpolation을 이용하여 같은 크기로 만들어준다.

bilinear interpolation은 linear interpolation을 가로 방향, 세로 방향에 대해 두 번한다는 의미이고 interpolation은 주위 픽셀을 이용해서 픽셀 값을 예측하는 기법이다.



bilinear interpolation을 그림으로 설명하자면, P를 예측하기 위해, 주위 Q를 이용하여, R1, R2를 예측하고, 이 예측된 R1, R2를 이용하여 정확한 P의 좌표를 찾는 방식이라고 말할 수 있다.

output feature에 대해 단순히 bilinear interpolation만 이용해 보정하면 물체 간 경계 구분의 구분이 힘들고 대체로 smooth한 예측 결과가 나온다고 한다. -> Skip Connection으로 해결



위 그림을 보면 FCN-32s라고 표시된 부분이 skip connection을 적용하지 않은 결과이다.

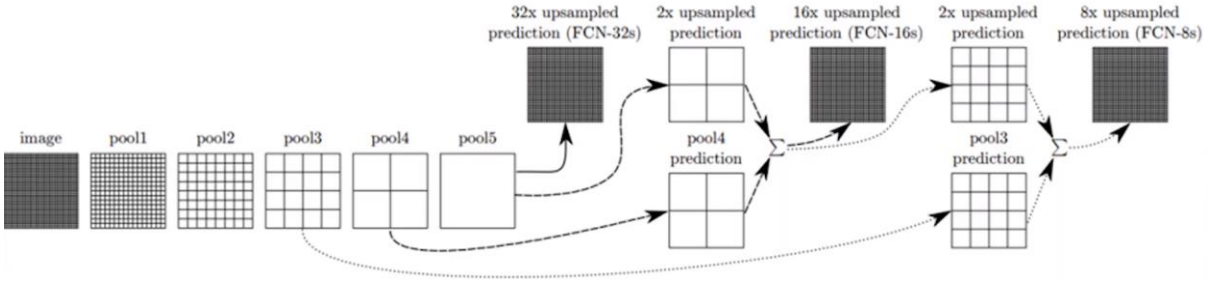
물체 간 경계가 불확실한 것을 볼 수가 있는데 이 이유는 딥러닝(CNN)의 Layer가 output에 가까울수록 abstract한 정보만 가지고 있기 때문에 edge, corner와 같은 세밀한 정보가 부족하기 때문이다.

(output에 가까울수록 receptive field가 크므로 큰 영역의 정보(전체적이면서 추상적인 정보)가 있고 input에 가까울수록 receptive field가 작으므로 작은 영역의 정보(부분적이면서 디테일한 정보)가 있다고 생각하면 된다.)

따라서 디테일한 segmentation map을 얻기 위해, skip connection이라는 기법을 이용한다.

3. Skip connection

: 이전 층의 정보를 이용하기 위해 이전 층의 정보를 연결하는 것



이 그림은 skip connection 의 구조를 나타낸 그림이다.

pool1 부터 pool5 까지는 vgg16 모델에서 각 pooling layer 를 나타낸 것이고 spatial size 는 단계별로 1/2 씩 줄어드는 구조를 가지고 있다.

입력 영상이 224x224x3 이라 하면, pool3 의 feature 는 28x28x512, pool4 는 14x14x512, pool5 는 7x7x512 가 된다.

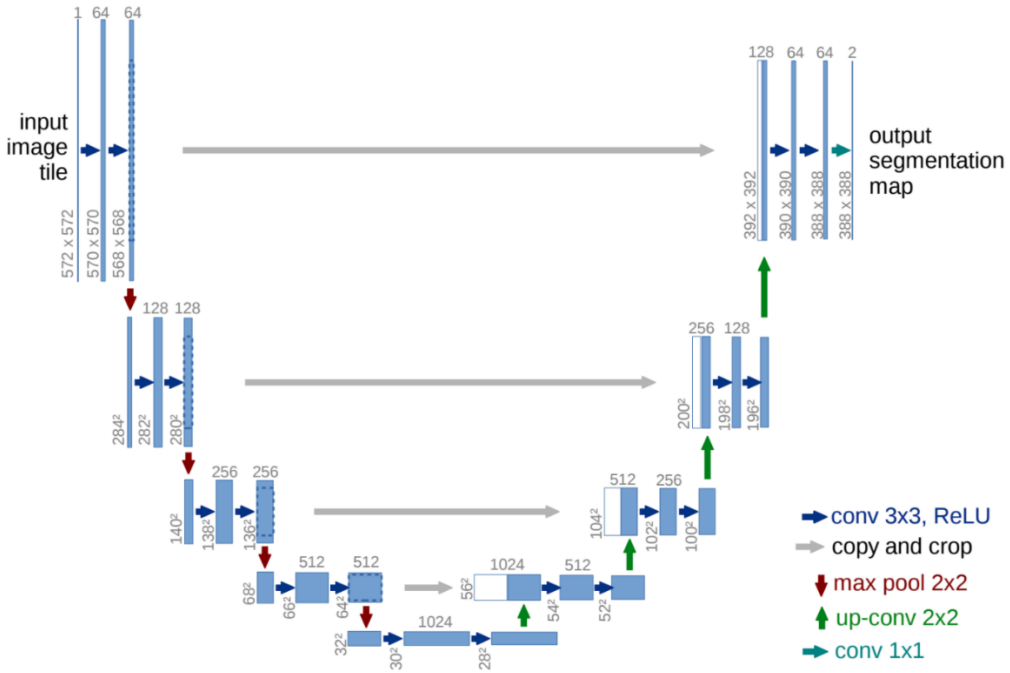
(FCN-8s 에 대한 설명)

pool5 feature(7x7x512)를 bilinear interpolation 으로 2 배 upsampling 한 결과(14x14x512)와 pool4 feature(14x14x512)를 더하여 새로운 feature(14x14x512)를 만들고, 이 feature 를 또다시 2 배 키워 pool3 feature(28x28x512)와 더한다. 이 최종 결과를 8 배 upsampling 해서 (224x224x512)로 만들고 1x1 conv 를 해서 채널 수를 class 개수만큼 줄어주면 된다.

	mean IU VOC2011 test	mean IU VOC2012 test	inference time
R-CNN [12]	47.9	-	-
SDS [16]	52.6	51.6	~ 50 s
FCN-8s	**62.7**	**62.2**	~ **175 ms**

FCN 은 Semantic Segmenation 의 base line 이고 다른 이전의 방법들과 비교하였을 때 월등한 성능을 자랑한다.

U-Net: Convolutional Networks for Biomedical Image Segmentation



U-Net 을 FCN 기준으로 세 가지가 업그레이드 되었다고 생각하면 된다.

1. Skip Architecture 를 확장했다는 점
2. Up-sampling 기법을 변경했다는 점(bilinear interpolation -> transposed convolution)
3. 속도를 개선했다는 점이다.

업그레이드된 내용을 설명하기 전에 U-Net 의 구조부터 알아보겠다.

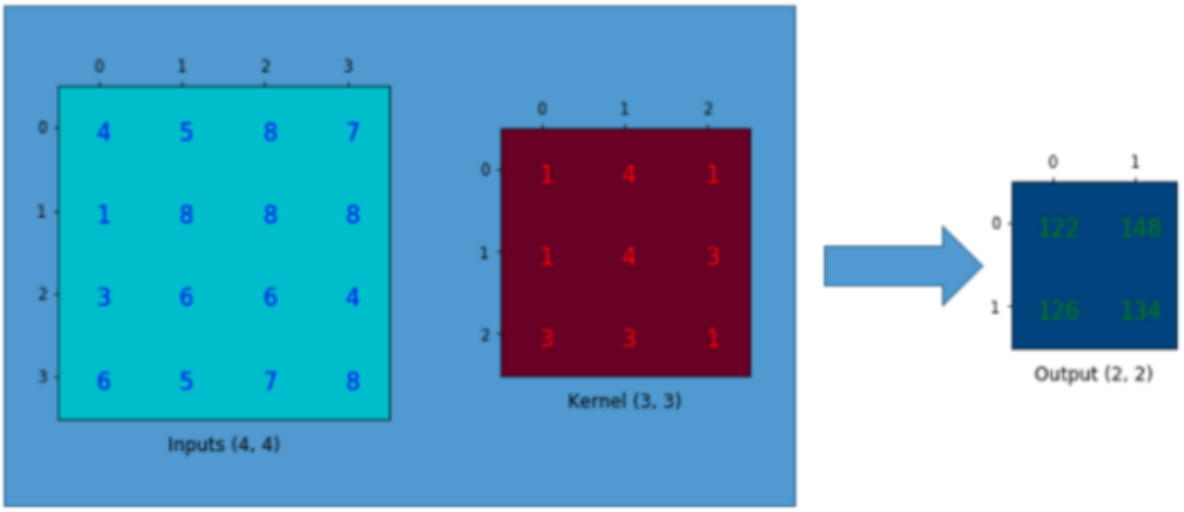
U-Net 의 왼쪽 부분은 Contracting Path 라고 하고 오른쪽 부분은 Expanding Path 라고 한다.

Contracting Path 는 Encoder 의 역할을 수행하는 부분으로 전형적인 Conv+Pool Layer 로 구성이 되고 이미지의 feature 를 파악하는 역할을 한다.

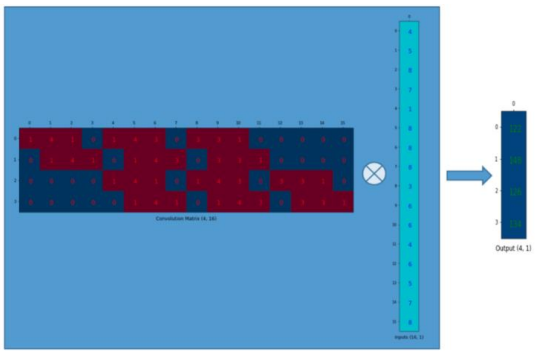
Expanding Path 는 Decoder 의 역할을 수행하는 부분으로 Upsampling+Conv Layer 로 구성이 되고 Contracting Path 에서 줄어든 사이즈를 다시 복원하면서 feature 를 파악하는 역할을 한다.

FCN 처럼 Skip Architecture 가 사용이 되는데, 이전 feature map 이 add 되는 것이 아니라 concatenate 된다는 점에서 차이가 있다.
(add 를 하든 concatenate 를 하든 목적은 같다고 생각한다. 잃어버린 공간 정보 파악 + fine-grained 한 정보 추가)

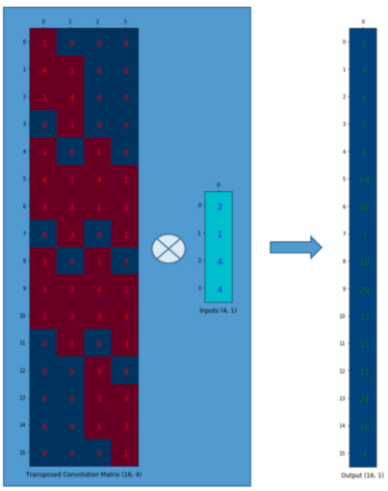
upsampling 을 할 때는 transposed convolution 을 사용한다. Transposed convolution 은 미리 정의된 보간 방법을 사용하지 않으며 학습 가능한 parameter(filter 의 weight)들이 있으며 up-sampling 을 더 최적으로 할 수 있다고 한다.



예를 들어 4x4 영상을 3x3 필터로 합성곱을 연산을 해서 2x2 결과가 나온다면 transposed convolution 은 2x2 영상을 3x3 필터를 이용해서 4x4 로 만드는 것을 의미한다.



3x3 필터는 4x16 conolution matrix 로 재배치가 가능하고 4x4 는 16x1 로 재배치가 가능하다. 이 두 행렬을 행렬곱해서 나온 결과(4x1)을 2x2 로 재배치하면 convolution 한 결과와 같다.



transposed convolution 을 계산할 때는 convolution matrix 를 transpose(16x4)하고 2x2 영상을 4x1 로 재배치한다. 이것을 행렬곱해서 나온 결과(16x1)를 4x4 로 재배치하면 transposed convolution 한 결과가 된다.

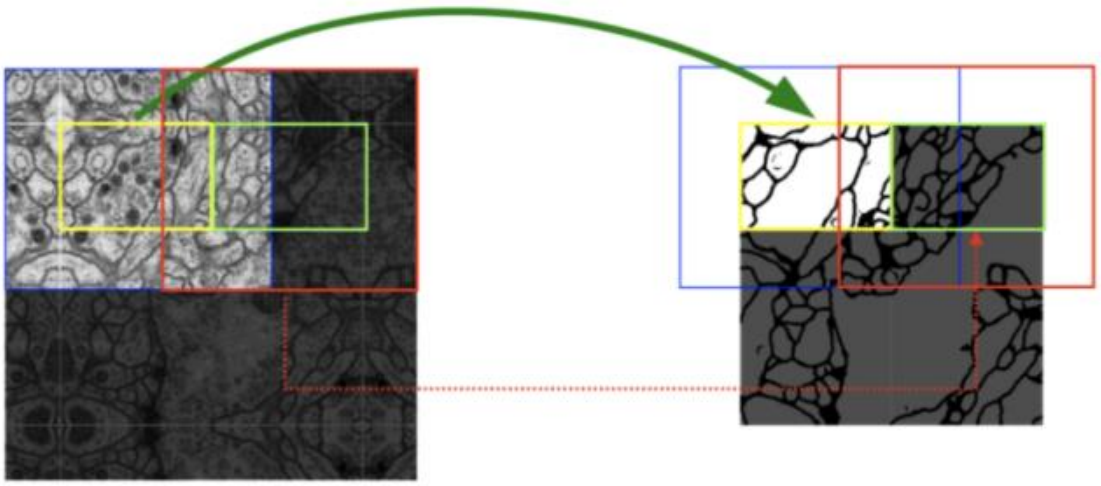
U-Net 의 구조를 보면 굳이 왜 사이즈를 줄였다가 복원하는지 의문이 들 수 있다.

두 가지 이유를 생각할 수 있는데, 첫 번째는 계산복잡성이 크다는 것이다. 만약 pooling 을 사용하지 않고 padding 을 same 으로 계속 처리한다면 상당히 많은 파라미터를 계산하므로 계산 비용을 줄일 필요가 있다.

두 번째는 semantic segmentation 은 receptive field 가 큰 것이 유리하기 때문이다. 예를 들어 버스 창문에 사람이 비추었다고 해서 버스 창문을 사람이라고 인식하면 안 된다. 즉 세그멘테이션은 전체적인 정보를 많이 요구하는 작업이므로 pooling 을 해서 receptive field 를 크게 만들어줘야 한다.

업그레이드 내용 중에서 속도에 대한 이야기를 해보겠다.

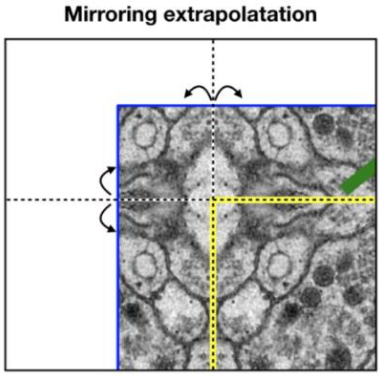
세그멘테이션을 할 때 영상 전체를 입력으로 넣을 수도 있지만 입력 사이즈가 너무 크면 Sliding Window 기법으로 이미지를 타일로 나눠서 입력으로 넣는다고 한다. 하지만 이렇게 입력을 주면 결과로 나온 segmentation map 이 겹치게 된다.



논문에서는 필터를 조금씩 움직여서 입력으로 주는 슬라이딩 윈도우 방식이 아닌,
필터를 크게 움직이는 방식을 사용하였다. 위와 같이 파란 영역의 이미지를 입력하면 노란 영역의 세그멘테이션 결과를 얻게 된다.

이렇게 한 결과, 겹치는 연산이 상당히 줄었고 입력 영상(파란색, 빨간색 박스)은 겹칠지라도 segmention map(노란색, 초록색 박스)은 겹치지 않았다고 한다.

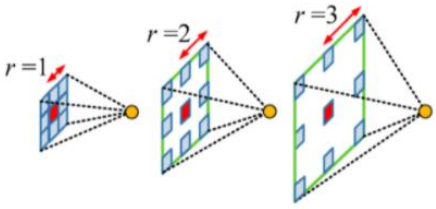
TMI)



이미지 경계 부분 픽셀에 대한 segmentation 을 위해 0 이나 임의의 padding 값을 사용하는 대신 미러링을 이용한 extrapolation(외삽법) 기법을 사용하였다고 한다.

DeepLab v1,2,3,3+

1. atrous convolution(dilated convolution) – v1



$$y[i] = \sum_{k=1}^{K} x[i + r \cdot k]w[k]$$

$$r > 1 : \text{atrous convolution}, \quad r = 1 : \text{standard convolution}$$

기존 conv 와 다르게 필터 내부에 빈 공간을 둔 채로 element-wise 하는 것이다. 얼마나 빈 공간을 둘 지 결정하는 파라미터인 rate r=1 일 경우, 기존 convolution 과 동일하고, r 이 커질 수록, 빈 공간이 넓어지게 된다. 이렇게 하면 기존 convolution 와 동일한 양의 파라미터와 계산량을 유지하면서 receptive field 는 크게 가져갈 수 있다. (실제로는 r 의 값에 따라 input 을 띄엄 띄엄 선택하여 필터와 곱함)

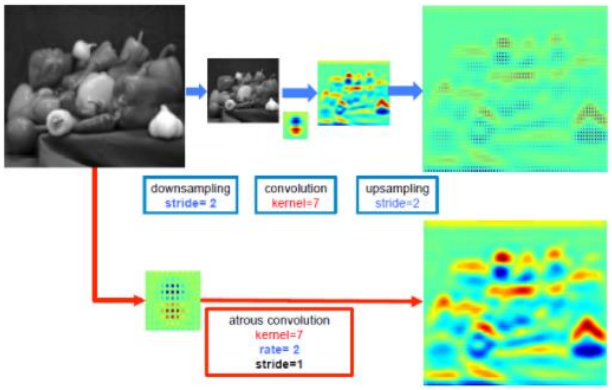


Fig. 3: Illustration of atrous convolution in 2-D. Top row: sparse feature extraction with standard convolution on a low resolution input feature map. Bottom row: Dense feature extraction with atrous convolution with rate $r = 2$, applied on a high resolution input feature map.

위 사진은 단순히 pooling-convolution 하고 upsampling 한 것과 atrous convolution 한 것의 차이를 볼 수 있는데 후자가 receptive field 를 더 크게 가져가면서 공간 정보를 더 많이 이용하므로 더 dense 한 output 이 나왔다.

결국 atrous convolution 을 하면 pooling 을 여러 번 하지 않고도 receptive field 를 크게 가져가는데 이 말은 backbone network 의 마지막에 pooling 을 해서 굳이 downsampling 을 할 필요가 없다는 것을 의미한다. 사실 pooling 을 하면 이미지 전체의 feature 를 요약하는데는 유용하다.

하지만 이러한 feature map 은 spatial information 가 많이 손실되어 있으므로 semantic segmentation 에 불리하다.

따라서 deeplab 은 network 에서 마지막 2 개의 max-pooling layer 를 제거하였고 atrous convolution 을 넣어서 1/8 크기의 feature map 을 추출하였다. 이 feature map 은 bilinear interpolation 을 이용해서 8 배 확대한 후 이용한다.

이 때 backbone 은 resnet-101 을 사용한다.

2. spatial pyramid pooling – v2, v3

feature map 으로부터 여러 개의 rate 가 다른 atrous convolution 을 병렬로 적용한 뒤 이를 다시 합쳐주는 ASPP 기법을 활용하였다.

즉 v1 과 달리 여러 rate 를 이용함으로써 다양한 스케일을 더 잘 처리할 수 있도록 구성된다. (receptive field 가 단순히 큰 것을 넘어서 다양하다고 생각하면 됨)

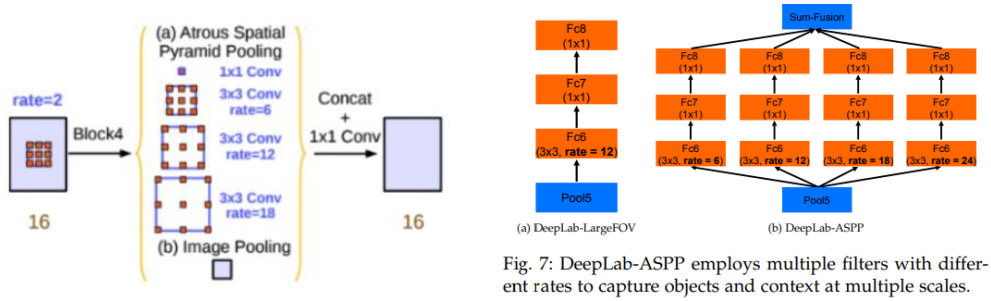
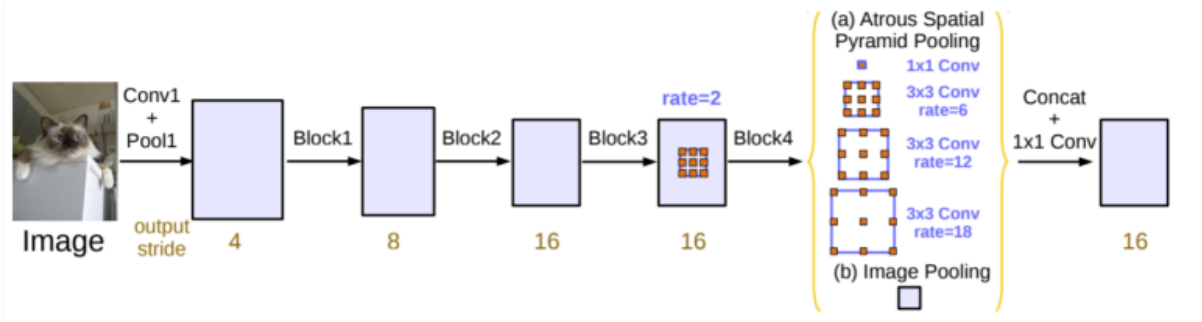
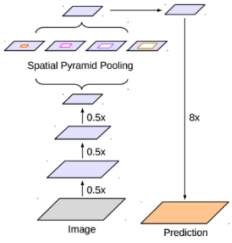


Fig. 7: DeepLab-ASPP employs multiple filters with different rates to capture objects and context at multiple scales.

ASPP 는 v2 에서 제안되었고 v3 에서 ResNet 과 같이 사용됨

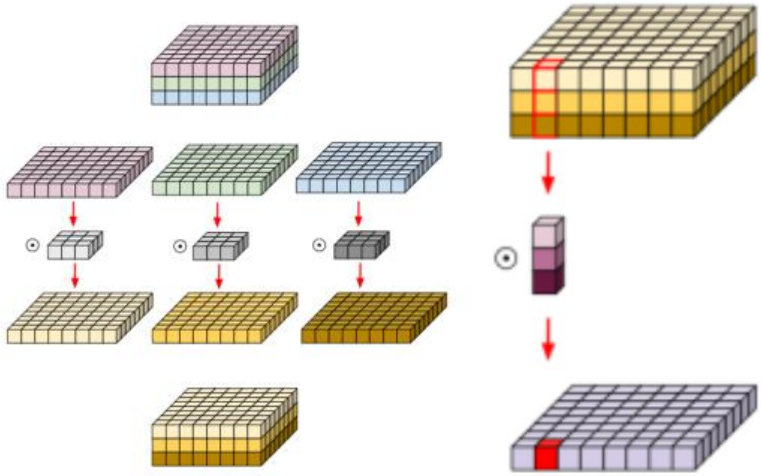


ResNet 에서 feature map 을 추출 -> feature map 의 크기를 조정하기 위해서 backbone 의 마지막 layer 에서 atrous convolution 을 수행 -> ASPP network 는 backbone 에서 추출된 feature 에 더해짐 -> ASPP network 의 출력은 1x1 convolution 을 통해서 뒷단으로 전달



뒷단으로 전달된 feature map 은 upsampling 하고 prediction 을 한다.

3. Deepthwise Seperable Convolution – v3+



입력 영상의 채널을 모두 분리시킨 뒤, 채널 수가 1 인 여러 개의 필터들로 convolution 연산하고 다시 합치는 과정을 deepthwise convolution 이라고 한다.

deepthwise convolution 결과를 1x1xc 크기의 필터로 convolution 연산하는 과정을 deepthwise seperable convolution 이라고 한다.

이렇게 하면 기존 Convolution 과 유사한 성능을 보이면서 사용하는 파라미터 수와 연산량을 획기적으로 줄일 수 있다.

ex) 입력 영상: 8x8x3, 필터: 3x3x3 (16 개)

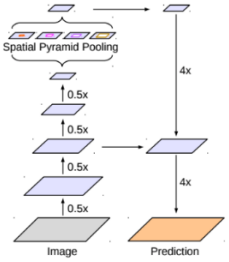
➔ convolution: 3x3x3x16 = 432 개의 파라미터 학습 [bias 까지 따지면 (3x3x3+1)x16)]

➔ deepthwise seperable convolution: 3x3x3 + 1x1x3x16 = 75 개의 파라미터 학습

(필터 수를 고려할 때는 seperable convolution 에서만)

위의 Seperable Convolution 을 ASPP 에 적용, Inception 에 적용 (Xception) -> V3+

4. Encoder-Decoder 구조 – v3+



Bilinear Upsampling 을 Simplified U-Net style decoder 으로 변경하였음

(skip architecture 추가)

➔ fine-grained 한 정보 추가되면서 경계를 더 잘 파악함

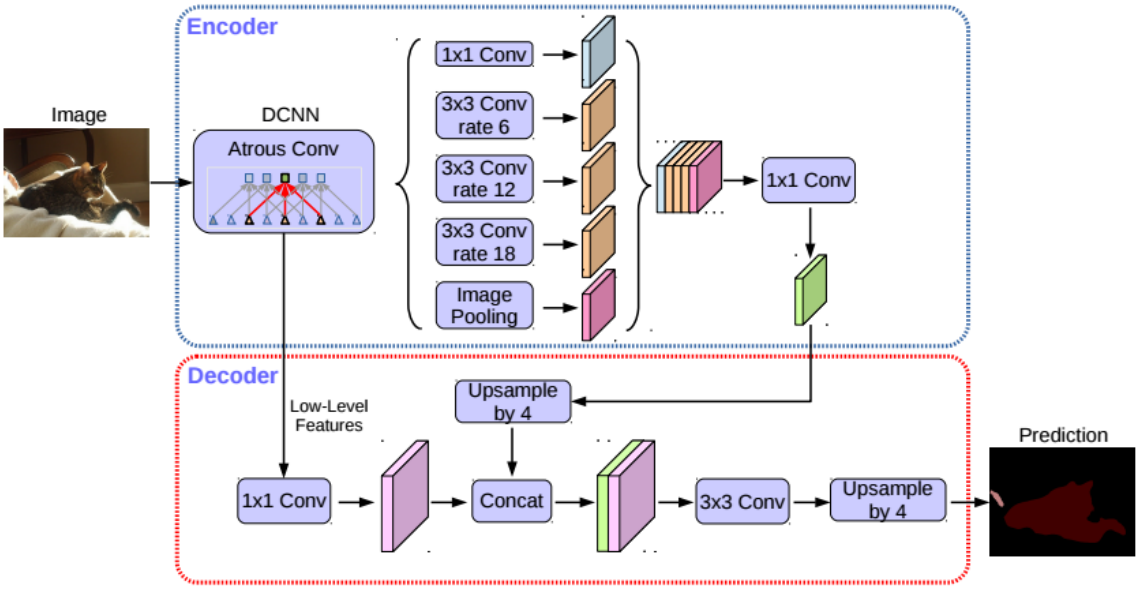


Fig. 2. Our proposed DeepLabv3+ extends DeepLabv3 by employing a encoder-decoder structure. The encoder module encodes multi-scale contextual information by applying atrous convolution at multiple scales, while the simple yet effective decoder module refines the segmentation results along object boundaries.

1x1 conv 하면서 class 수를 채널 수로 조정

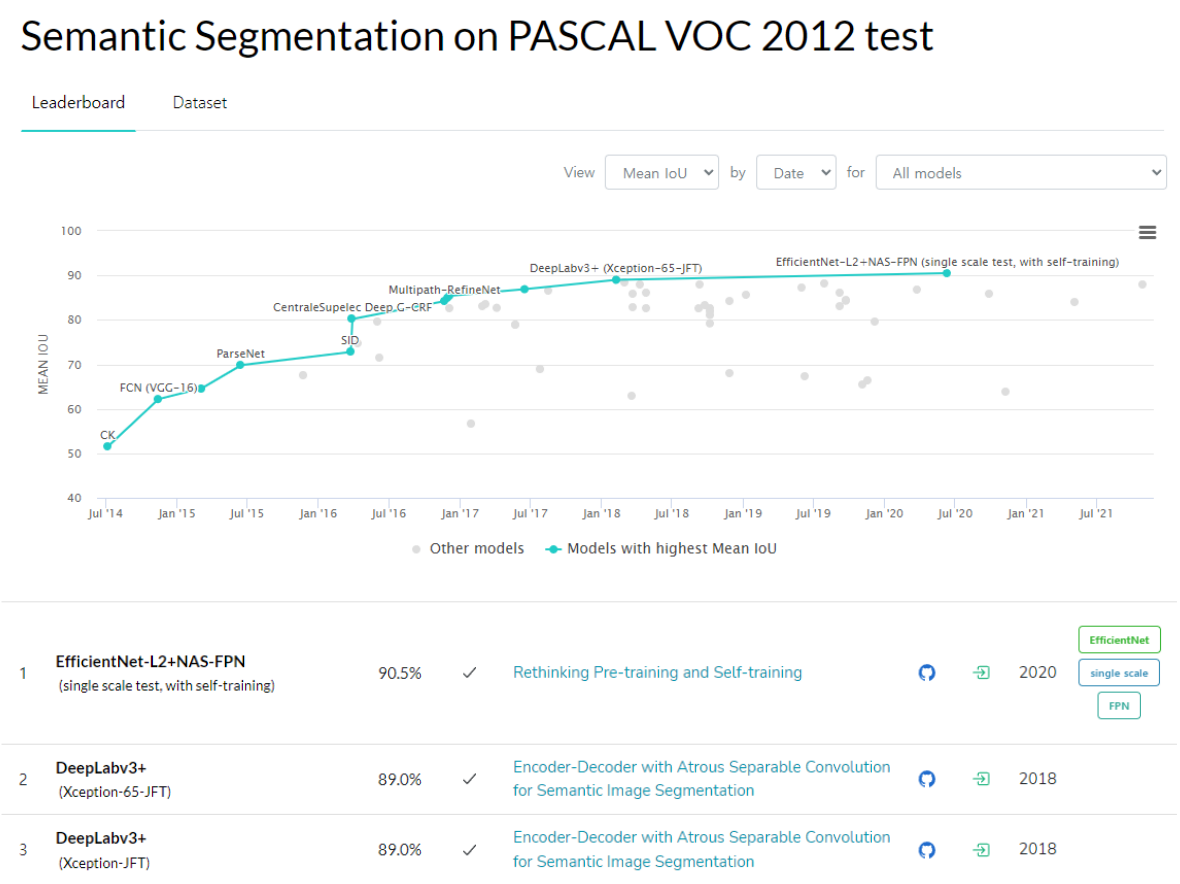
최종 3x3 conv 하면서 concat 한 feature 에 대해 공간적인 정보 파악 + 늘은 채널 수를 class 수로 재조정

Method	mIOU
Deep Layer Cascade (LC) [82]	82.7
TuSimple [77]	83.1
Large_Kernel_Matters [60]	83.6
Multipath-RefineNet [58]	84.2
ResNet-38_MS_COCO [83]	84.9
PSPNet [24]	85.4
IDW-CNN [84]	86.3
CASIA_IVA_SDN [63]	86.6
DIS [85]	86.8
DeepLabv3 [23]	85.7
DeepLabv3-JFT [23]	86.9
DeepLabv3+ (Xception)	87.8
DeepLabv3+ (Xception-JFT)	89.0

Table 6. PASCAL VOC 2012 *test* set results with top-performing models.

DeepLab v3+가 발표 당시(2018 년) SOTA 성능을 보여준다.

[현재 2022 년 기준 VOC 2012 Test]



1	**EfficientNet-L2+NAS-FPN** (single scale test, with self-training)	90.5%	✓	Rethinking Pre-training and Self-training			2020	EfficientNet, single scale, FPN
2	**DeepLabv3+** (Xception-65-JFT)	89.0%	✓	Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation			2018	
3	**DeepLabv3+** (Xception-JFT)	89.0%	✓	Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation			2018	

현재까지도 최상위권을 차지하고 있다.

https://paperswithcode.com/sota/semantic-segmentation-on-pascal-voc-2012