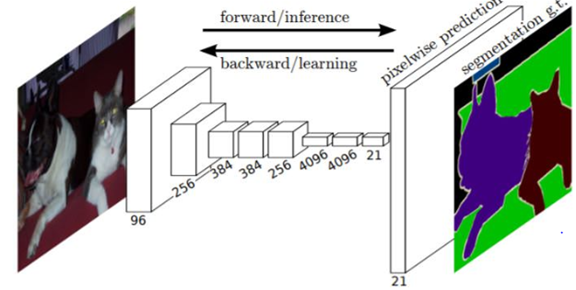
**[Segmentation]**

**FCN vs Deconvolutional network**

FCN이 전체적인 모양에 관심을 갖는 것이라면, Deconvolutional network는 detail에 집중 à 이런 경우는 여러 방법을 섞어 쓰는 것이 효율적(모델 결합)

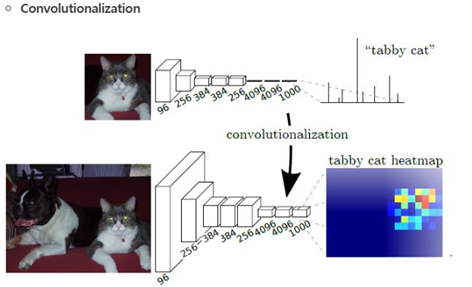
**1.** **FCN 모델**

FCN에서 가장 중요한 부분은 이름에서 나와있듯 Convolution layer만을 사용했다는 것이다. 기본적으로 CNN 모델에서는 모델 뒤쪽에서 Fully Connected layer가 나오는데, FCN에서는 FC Layer 대신 1x1 Convolution layer를 사용했다는 점이다. 이렇게 사용한 이유에 대해 추측을 해보자면, 우선 Fully Connected layer를 사용하기 위해서는 고정된 input size를 가질 수 밖에 없다. 그리고 FC layer를 지나는 순간 각 pixel에 대한 위치정보는 소실된다. 따라서 FCN은 모든 Network를 Convolution layer만 사용함으로써 input size의 제한을 받지 않고, 위치정보를 보존할 수 있게 되었다. (FCN 개발자들은 위치정보가 소실되지 않게 하기 위해서, 또한 어떠한 크기의 입력이미지도 허용하기 위해서 다음과 같이 알고리즘을 발전시켜감)



* Feature를 추출하는 Convolution 단계
* 뽑아낸 future에 대해 pixelwise prediction 단계
* classification을 한뒤 각 원래의 크기로 만들기 위한 Upsampling 단계

이러한 단계를 거친 후 각 pixel에 class따라 색칠을 한뒤 Segmentation 결과를 보여준다.



FCN은 Fully Connected Layer를 사용하지 않고 1x1 Convolution Layer를 사용했다고 했는데, 논문에서 이러한 1x1 Convolution을 Convolutionalization이라 표현했다. 그림 중간의 256크기의 matrix가 4096의 크기로 reshape된 것을 볼 수 있다. 이렇게 reshape을 한 후 여기에 1x1 Convolution을 진행한다. 하지만 이렇게 크기를 줄인다면 output dimension이 줄어들어 원래 크기의 image에 대해 segmentation을 할 수 없게 된다. 따라서 다시 크기를 원래 size로 만들어 줘야 한다. 즉 Upsampling 단계가 필요하다.

* Upsampling

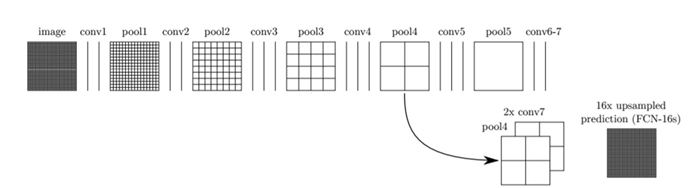
FCN에서 feature extraction을 크기가 줄어 하나의 pixel이 원래 image의 32x32크기를 나타낸다.

여기서 다시 크기를 키우기 위해 32x32의 크기로 바로 만든다면 많은 정보들이 소실되고 정확도 또한 떨어지게 된다. (단순히 upsampling을 시행하면, 특성맵의 크기는 원래 이미지 크기로 복원되고, 그것들로부터 원래 이미지 크기의 segmentation map을 얻게 되지만 여전히 coarse한, 즉 디테일하지 못한 segmentation map을 얻게 된다. 1/32만큼 줄어든 특성맵들을 단숨에 32배만큼 upsampling 했기 때문에 당연히 coarse할 수 밖에 없다. 이렇게 단숨에 32배 upsampling하는 방법을 논문에서는 **FCN-32s**라고 소개하고 있다)

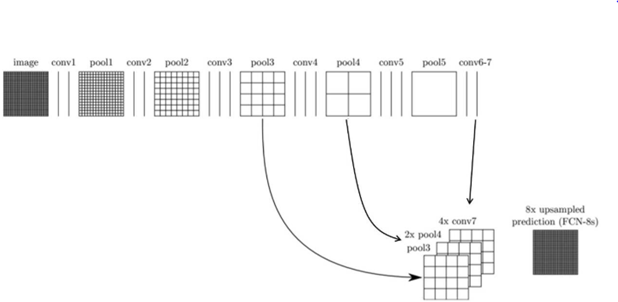
⇒ 따라서 **여기서는 1/32 크기만을 이용하는 것이 아니라 이전 Layer에서의 1/16크기와 1/8에서의 값도 같이 사용한다.** **이전의 layer들의 값을 다른 연산을 거치지 않고 Skip하여 마지막에서 같이 합쳐서 사용하게 된**다. 따라서 이러한 과정을 ‘Skip Layer’ 혹은 ‘Skip Connection’이라 부른다.

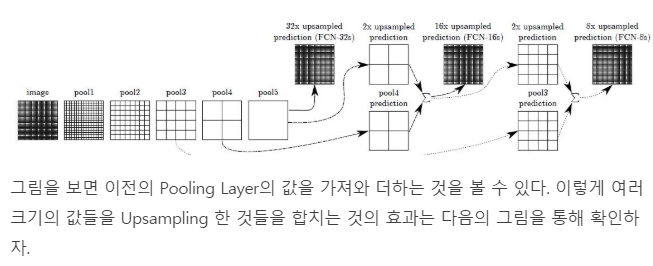
* Skip connection (combining)

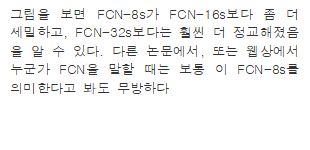
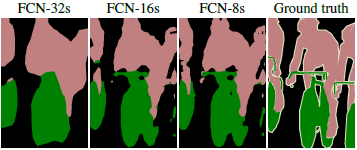
FCN의 개발자들은 좀더 디테일한 segmentation map을 얻기 위해 **skip combining**이라는 기법을 제안한다. 기본적인 생각은 다음과 같다. **컨볼루션과 풀링 단계로 이뤄진 이전 단계의 컨볼루션층들의 특성맵을 참고하여 upsampling을 해주면 좀 더 정확도를 높일 수 있지 않겠냐는 것이다.** 왜냐하면 이전 컨볼루션층들의 특성맵들이 해상도 면에서는 더 낫기 때문이다. 이렇게 바로 전 컨볼루션층의 특성맵(pool4)과 현재 층의 특성맵(conv7)을 2배 upsampling한 것을 더한다. 그 다음 그것(pool + 2x conv7)을 16배 upsampling으로 얻은 특성맵들로 segmentation map을 얻는 방법을 **FCN-16s**라고 부른다

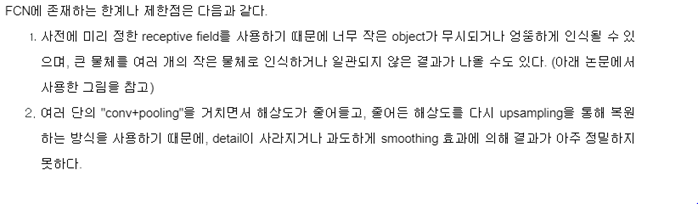


또 더 나아가서 전전 컨볼루션층의 결과도 참고해서 특성맵들을 얻고, 또 그 특성맵들로 segmentation map을 구할 수도 있다. 이 방법은 **FCN-8s**라고 부른다. 좀 더 구체적으로 이야기하면, 먼저 전전 단계의 특성맵(pool3)과 전 단계의 특성맵(pool4)을 2배 upsampling한 것과 현 단계의 특성맵(conv7)을 4배 upsampling 한 것을 모두 더한 다음에 8배 upsampling을 수행하므로 특성맵들을 얻는다. 이것을 모두 종합해서 최종 segmentation map을 산출한다.



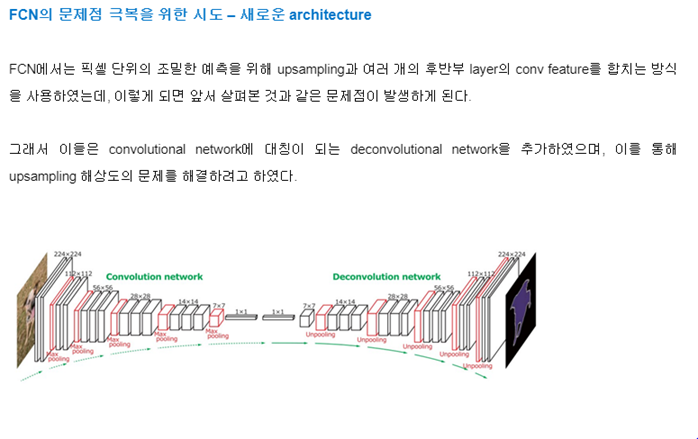
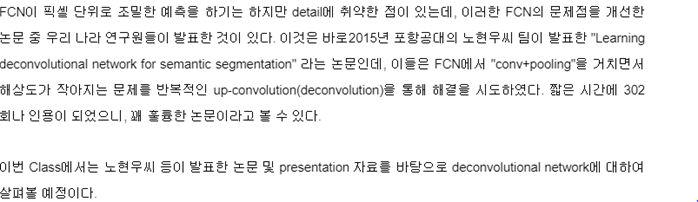


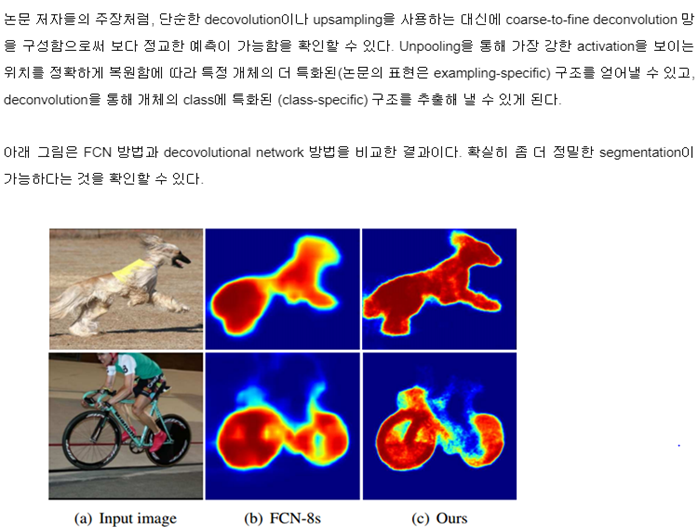


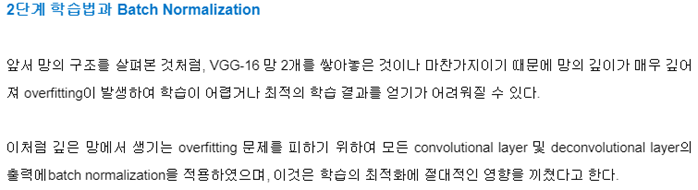


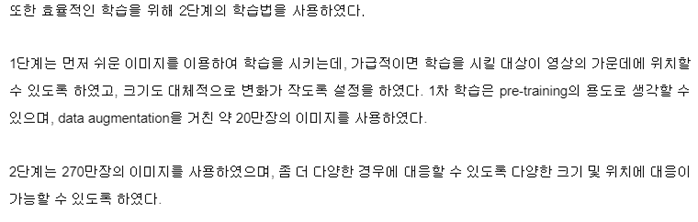
⇒ FCN의 결과를 그대로 사용하는 대신에 “FCN + CRF”를 사용하여 FCN의 결과를 좀 더 정밀하게 튜닝하는 방법을 사용하면 좀 더 정교한 예측이 가능함

**Deconvolutional network**

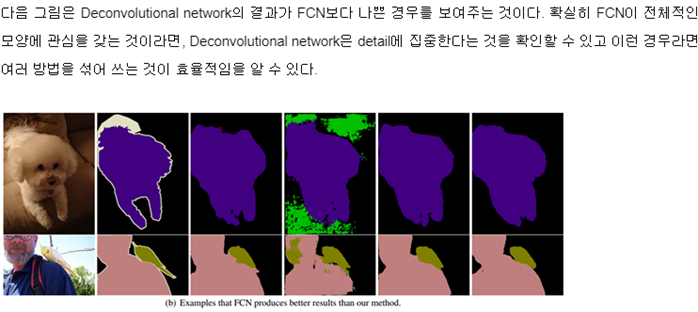
****

****

****

****

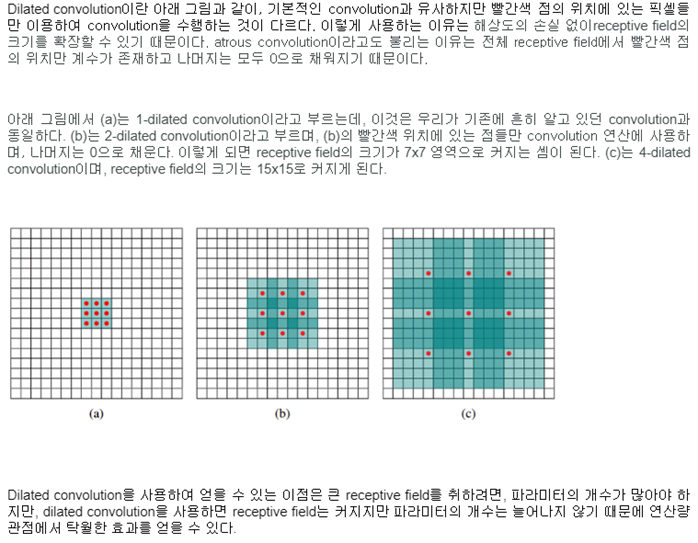
****

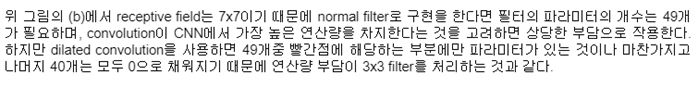
****

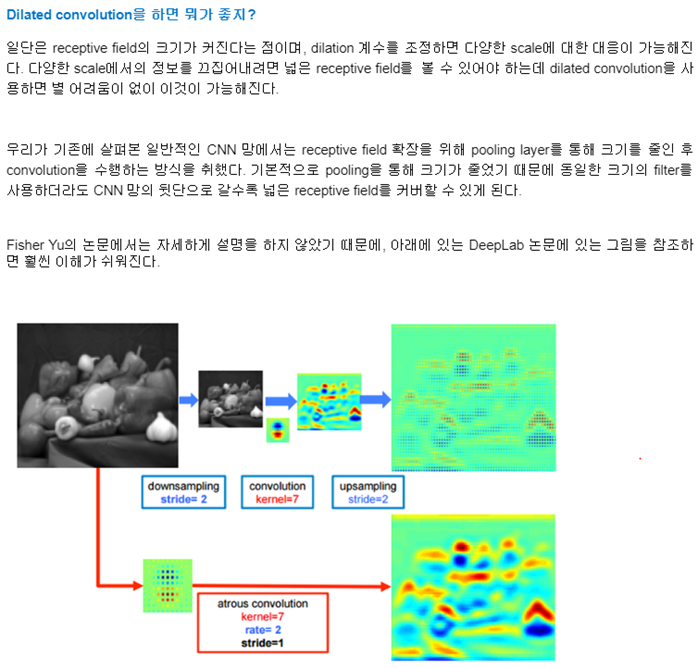
정교함에서는 deconvolutional network이 좋은 결과를 내지만 때로는 좋지 못한 결과도 나오기 때문에 뭐가 절대적인 방법이라고 할 수 없다

**2.**   **Dilated Convolution / Atrous convolution**

**- 구글 DeepLab이 제안한 방법**

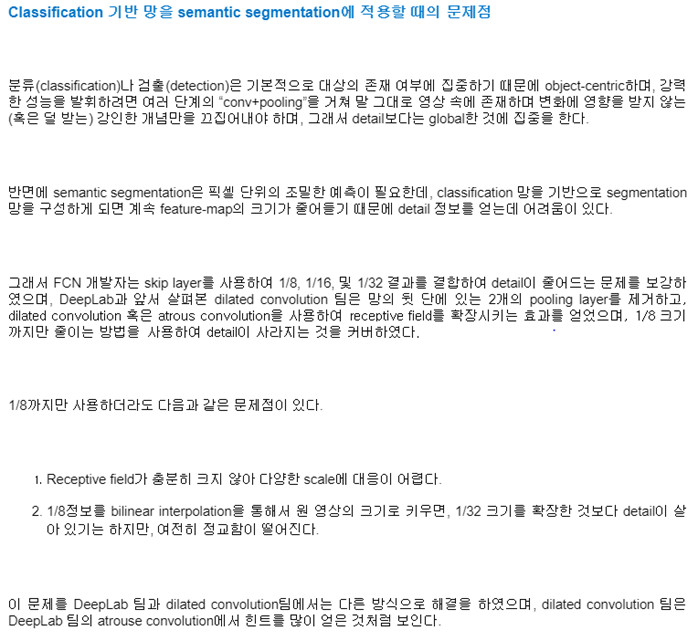
****

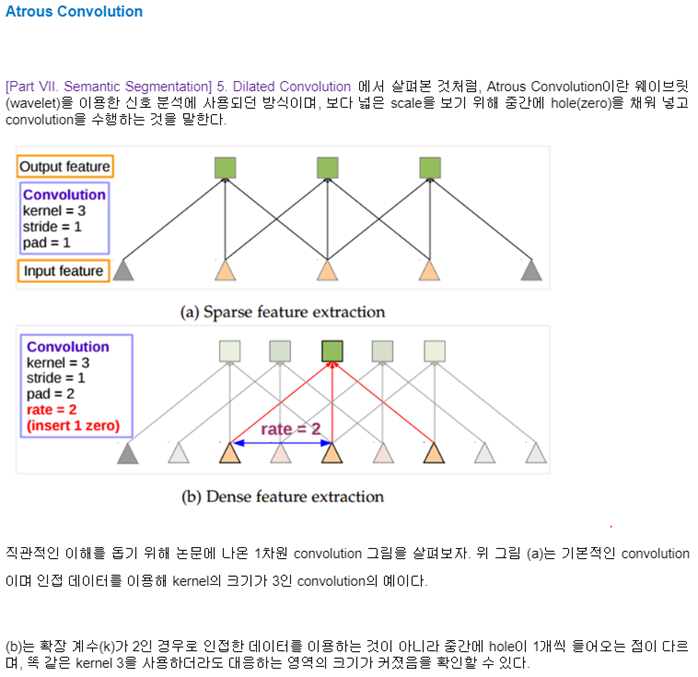
****

****

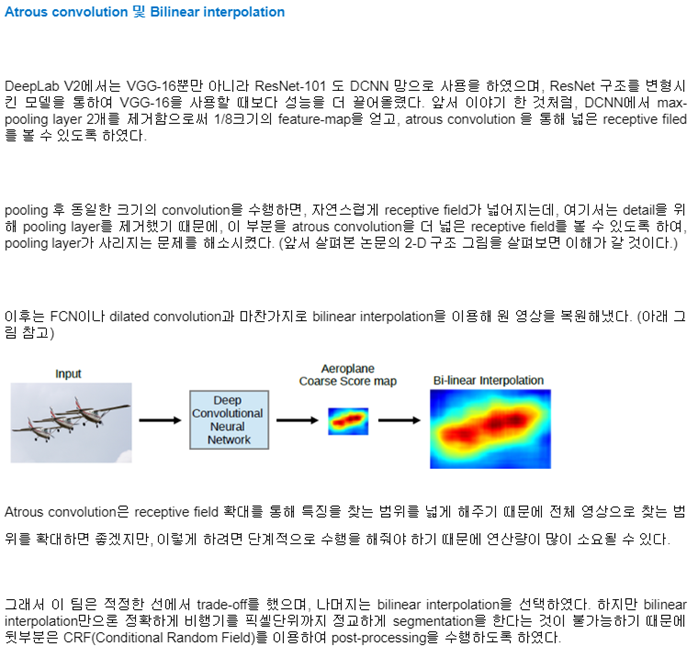
**3. DeepLab V1, V2, V3**

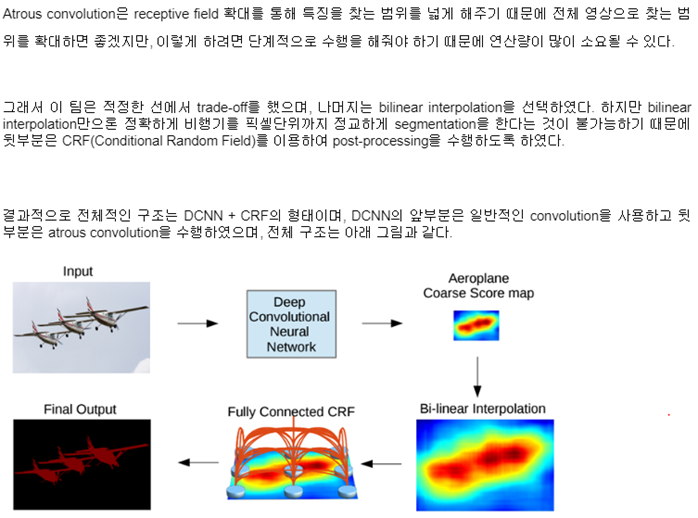
****

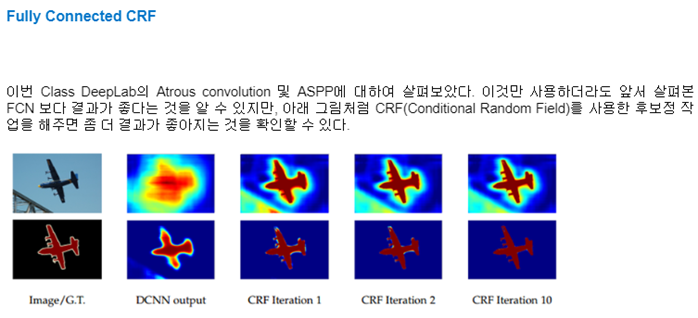
****

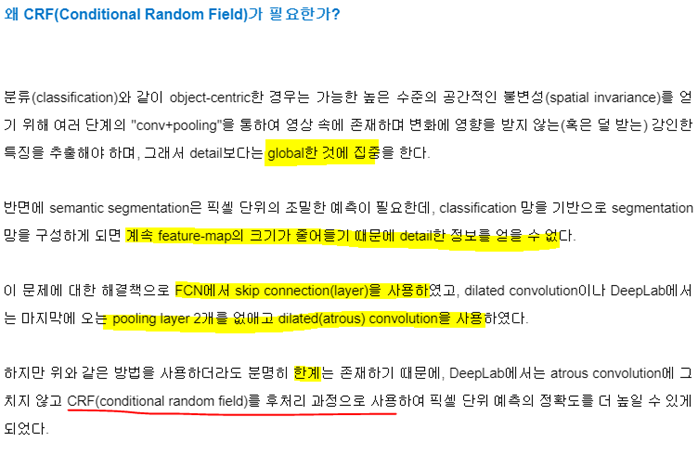
****

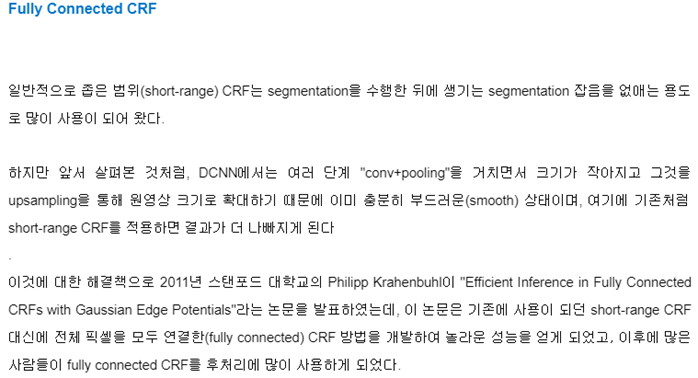
****

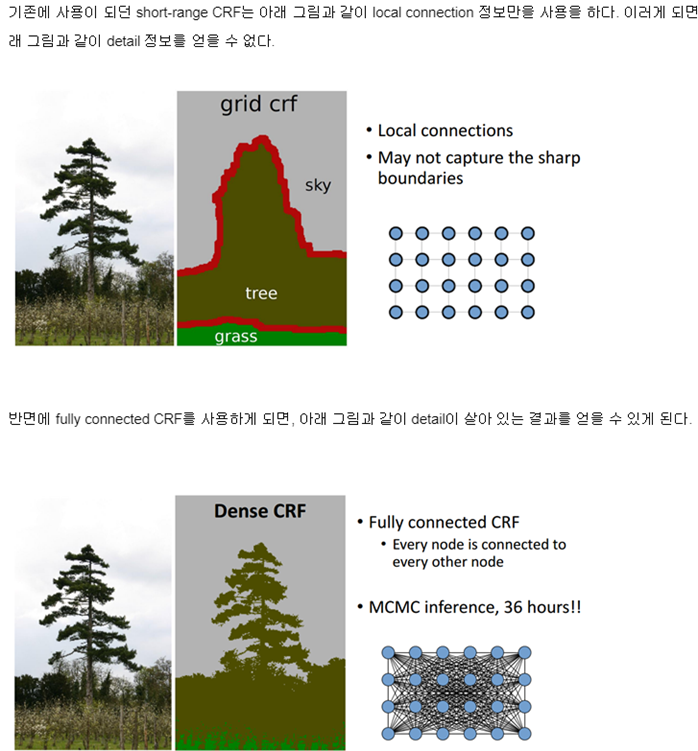
****

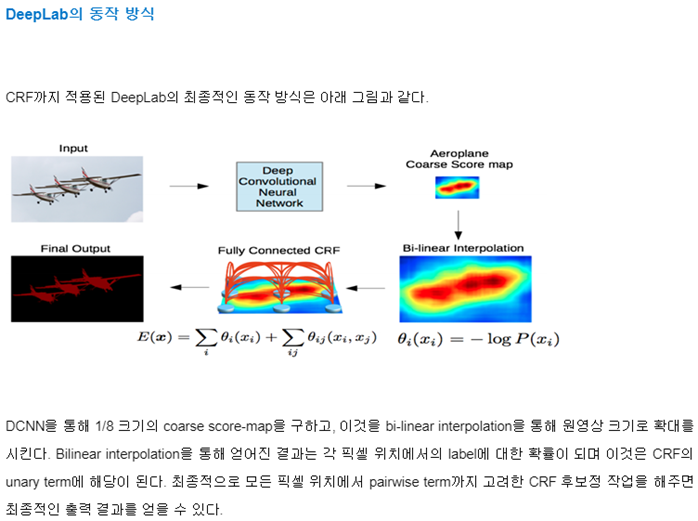
****

****

****

****

****

****

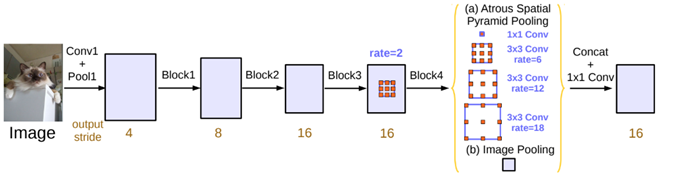
****

## **Dilated/Atrous Convolution 을 사용하는 방법**

Deeplab 이 제시하는 방법으로, 신호가 소멸되는것을 제어하고 다양한 크기의 특징을 익히는 방법을 제시합니다. Atrous Convolution 은 dilation rate (확장 비율)이라는 새로운 변수를 사용합니다. 이 비율은 커널에서 사용할 값들 사이에 얼마 만큼의 공간을 넣어줄 것인지에 대한 것입니다. 커널 안에 공간이 없는 일반적인 convolution 은 확장비율은 1로 정의합니다. 3 x 3 의 크기의 커널이 2의 확장비율을 갖는다면 실제로는 5 x 5 커널의 시야를 갖게 됩니다. 위와 같은 방법은 동일한 계산 비용으로 보다 넓은 시야를 갖을 수 있게 합니다.

Deeplab V3는 ImageNet에서 학습된 ResNet을 기본적인 특징 추출기로 사용합니다. ResNet의 마지막 블럭에서는 여러가지 확장비율을 사용한 Atrous Convolution을 사용해서 다양한 크기의 특징들을 뽑아낼 수 있도록 합니다.

또한 이전 Deeplab 버젼에서 소개되었던 Atrous Spatial Pyramid Pooling (ASPP)을 사용합니다. 새로운 아이디어라기 보다는, 좋은 성능을 보였던 모델들의 특징들을 섞어놓은 모델입니다. 다양한 확장비율을 가진 커널을 병렬적으로 사용한 convolution 입니다.

****

**Deeplab v1,v2,v3**

DeepLab v2는 v1과 Atrous Convolution과 Fully Connected CRF(Conditional Random Field)를 사용한다는 점에서 비슷하지만 v2에서는 Atrous spatial pyramid pooling을 사용해서 Multiple sclae에 대응하는 방법이 개선되었다.

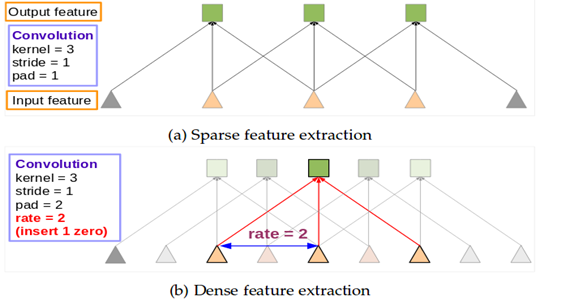
Deep Lab. v2에서는 위의 세 가지 문제를 다음을 사용해서 해결하려 한다

* Reduced feature resolution
* Atrous Convolution
* Reduced Local accuracy
* CRF(Conditional Random Field)
* Existence of objects at multiple scale
* Atrous Spatial pyramid pooling (ASPP)

위 3가지 기술들에 대해 하나씩 살펴보자.

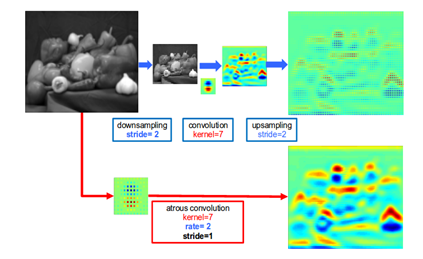
Atrous Convolution Classification 문제와는 달리 Image를 Pixel별로 구분해야 하는 Segmentaion에서는 CNN의 Layer가 깊어질수록 Feature의 크기가 작아지는 특징이 단점으로 작용했다. 따라서 이러한 Reduced Feature Resolution 문제를 Atrous Convolution을 사용해 해결했다.

Atrous Convolution이란 의미적으로 구멍 뚫린 Convolution이라 해석하면 될 것이다. 아래 논문의 그림을 보자.



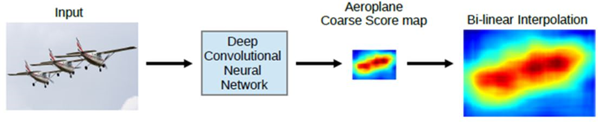
그림의 위쪽은 일반적인 Convolution과정이고 아래는 Atrous Convolution을 의미한다

중간중간 Hole(중간중간을 0으로 만듬)을 만들어서 좀 더 먼 쪽과 Convolution을 진행한다. 이런 과정을 통해 모델은 좀더 Dense한 Feature를 학습한다.

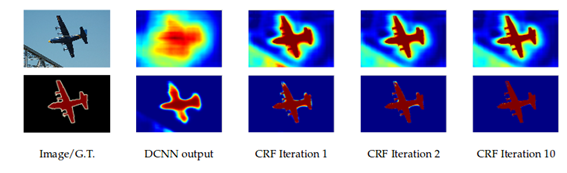


이제 큰 그림을 줄이면서 각 pixel에 대한 feature를 얻었으니 다시 Segmentation을 위해 다시 image를 늘리는 과정을 해야 한다. 아래 그림과 같이 Bilinear Interpolation만을 수행하면 원래 Image의 Segment를 정확히 얻지 못하는 것을 볼 수 있다.

* Conditional random field



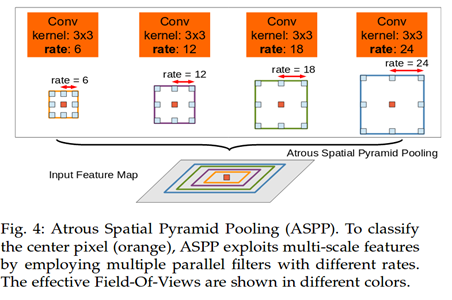
따라서 Conditinal Ramdom Field를 사용해서 더욱 정확한 Segment를 얻는다. 아래의 그림은 CRF를 반복할 수록 더욱 정확한 Segment를 얻는 과정을 보여준다.



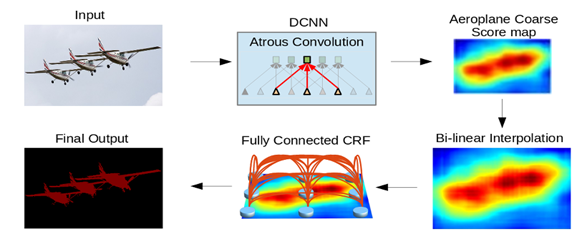
* Atrous spatial pyramid pooling(ASPP)

DeepLab v1에서는 크기가 다른 물체들이 있을 떄 이를 잘 Segmentation하지 못하는 문제가 있었는데 v2에서는 ASPP를 사용해서 이를 해결했다.

ASPP란 특정 레이어 대해서 Atrous Convolution을 여러 rate를 이용해서 진행한 후 결과들을 합치는 방법이다. 아래의 그림을 보자.



위의 3가지 방법들을 합친 DeepLab v2의 전반적인 진행과정은 다음 그림과 같다.



실제 현업 프로젝트 진행 과정

<https://tech.socarcorp.kr/data/2020/02/13/car-damage-segmentation-model.html#index6>

기반모델 / 분석모델 고르기

<https://devkor.tistory.com/entry/%EB%94%A5%EB%9F%AC%EB%8B%9D%EC%9D%84-%ED%86%B5%ED%95%9C-Image-Segmentation-%EC%9E%85%EB%AC%B8>

구현 깃허브 - keras 내장모델 활용

<https://github.com/divamgupta/image-segmentation-keras>