Fully Convolutional Networks for Semantic Segmentation

Abstract
Convolutionalization
Deconvolution
    Interpolation
    Backwards convolution
    Skip Architecture
정리
Reference

Abstract

FCN은 Semantic Segmentation문제를 Convolution 연산을 이용해 해결하고자 제안된 딥러닝 모델입니다. 결론적으로 Image Classification Task를 위해 제안된 모델의 Head(classification layer)에 Convolution layer로 변경해 Sementatic Segmentation Task로의 Shifting을 성공적으로 했습니다.

이러한 Image Classification to Sementic Segmentatin로의 shifting은 크게 다음의 세 과정으로 표현할 수 있습니다.

- Convolutionalization
- Deconvolution (upsampling)
- Skip architecture

Convolutionalization

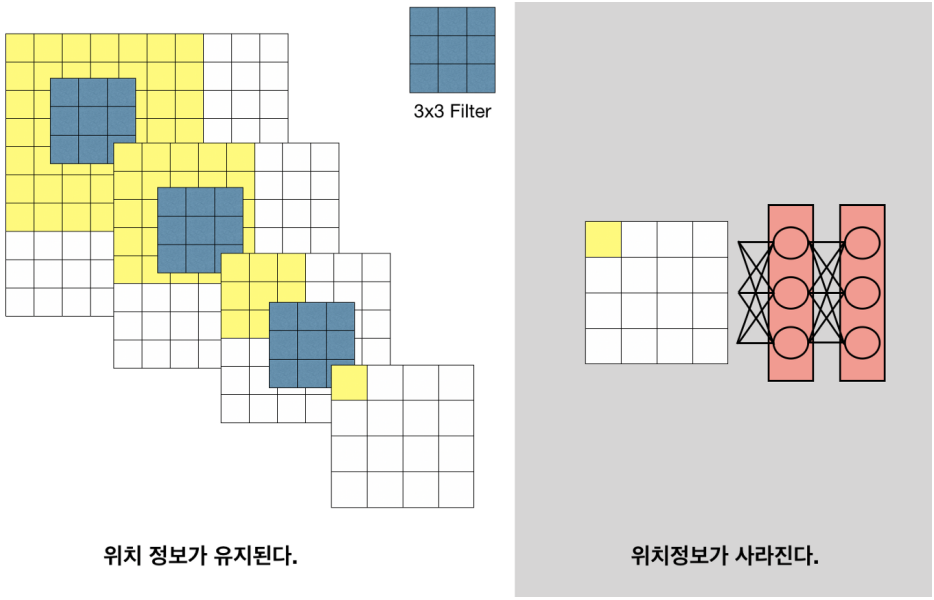
Convolutionalization(컨볼루션화)이라는 표현의 의미를 이해하기 위해서는 기존의 이미지 분류 모델들을 먼저 살펴볼 필요가 있습니다.

Image classification 모델들은 기본적으로 내부부 구조와 관계없이 모델의 근본적인 목표를 위해 출력층이 Fully-connected(FC) layer로 구성되어 있습니다. 이러한 구성은 네트워크의 입력층에서 중간부분까지 ConvNet을 이용하여 영상의 Feature들을 추출하고 해당 특징들을 출력층 부분에서 FC layer를 이용해 이미지를 분류하기 위함이였습니다.

하지만 Semantic Segmentation관점에서는 fc layer가 갖는 한계점이 있습니다.

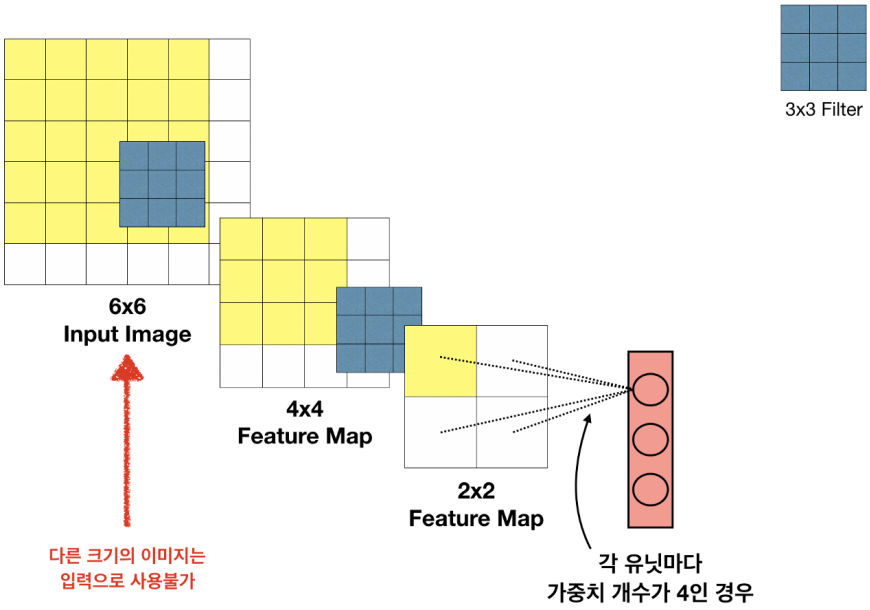
1. 이미지의 위치 정보가 사라집니다.

   Convolution 연산을 사용하면 이미지의 위치 정보를 일정부분 유지할 수 있습니다.

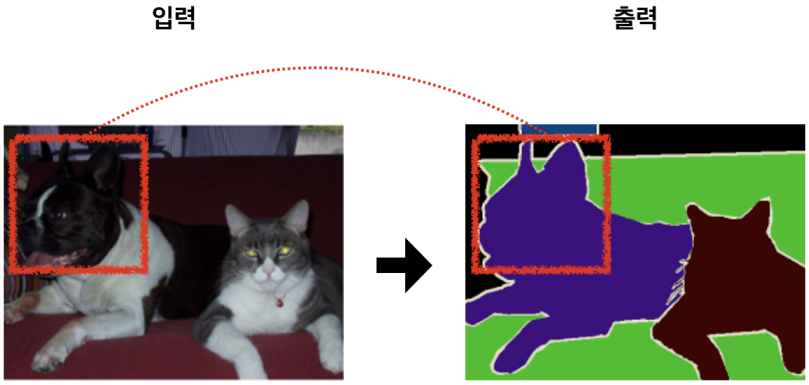


2. 입력 이미지 크기가 고정된다.

   하지만 convolution을 이용하면 입력 이미지 크기를 고정하지지 않아도 된다.



Segmentation의 목적은 원본 이미지의 각 픽셀에 대해 클래스를 구분하고 인스턴스 및 배경을 분할하는 것으로 위치 정보가 매우 중요하다.



위치 정보가 매우 중요함

저자는 이러한 FC-layer의 한계를 보완하기 위해 모든 FC-layer를 Convolution layer로 대체하는 방법을 택하였습니다.

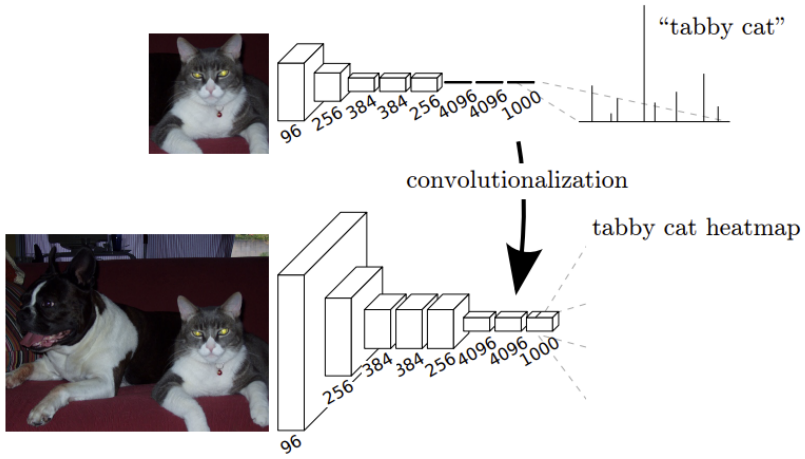
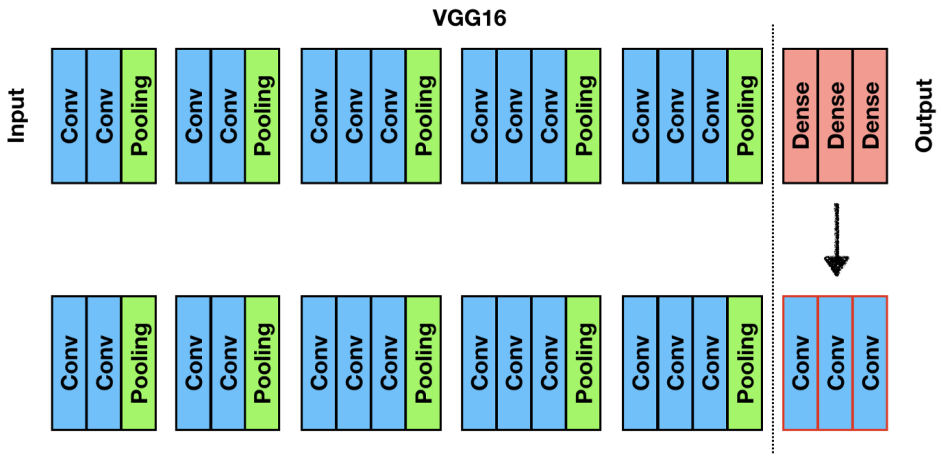


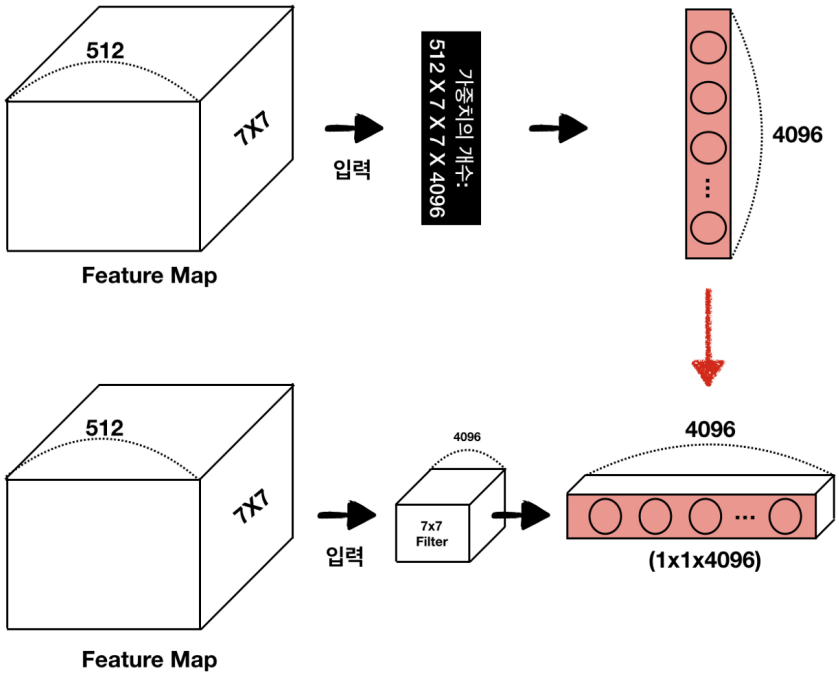
Figure 2. Transforming fully connected layers into convolution layers enables a classification net to output a heatmap. Adding layers and a spatial loss (as in Figure 1) produces an efficient machine for end-to-end dense learning.

VGG16을 예로 들어보겠습니다. 마지막의 Dense Layer를 Convolution layer로 변경했습니다.

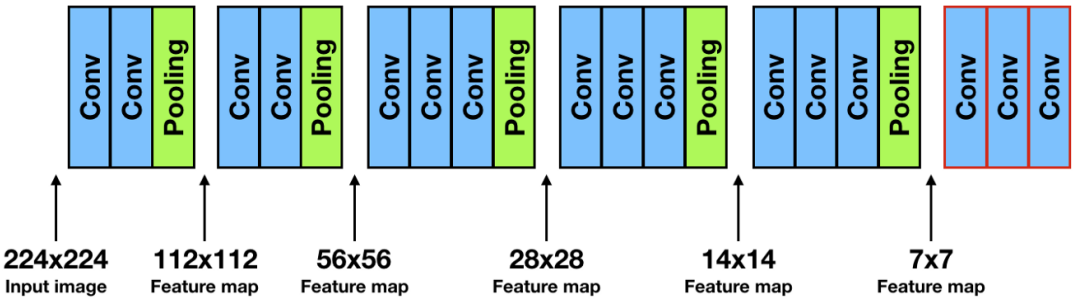


Dense Layer에서 Conv Layer로 변환하는 방식은 다음과 같다.

첫 번째 Fully-connected layer를 (7 * 7 * 512) 를 4096개의 output으로 내 뱉는 FC layer와 동일한 parameter를 가지게 하려면 7*7 * 4096의 Convolution으로 바꾸면 됩니다. 그러면 아래와 같은 형상이 됩니다.



VGG16에서 다섯 번째 max-pooling(size: 2x2, stride : 2)연산 후 Feaeture map의 크기는 7x7이 된다. 입력 이미지의 크기가 224x224인 경우



Convolutionalization을 통해 Feature map은 원본 이미지의 위치 정보를 내포할 수 있게 되었습니다.

그러나 Semantic Segmentation의 최종 목적인 픽셀 단위 예측과 비교했을 때, FCN의 출력 Feature map은 너무 coarse(거친, 알맹이 큰)합니다.

따라서, Coarse map을 원본 이미지 크기에 가까운 Dense map으로 변환해줄 필요가 있다. 적어도 input image size * 1/32 보다는 해상도가 높을 필요가 있다.

Deconvolution

Coarse map에서 Dense map을 얻는 몇 가지 방법이 있습니다.

- Interpolation
- Deconvolution
- Unpooling
- Shift and stitch

물론 Pooling을 사용하지 않거나, Pooling의 stride를 줄임으로써 Feature map의 크기가 작아지는 것을 처음부터 피할 수도 있습니다.

그러나, 이 경우 필터가 더 세밀한 부분을 볼 수 있지만, Receptive Field가 줄어들어 이미지의 컨텍스트를 놓치게 됩니다.

또한, Pooling의 중요한 역할 중 하나는 특징맵의 크기를 줄임으로써 학습 파라미터의 수를 감소 시키는 것인데, 이러한 과정이 사라지면 파라미터의 수가 급격히 증가하고 이로인해 더 많은학습 시간을 요구하게 됩니다.

따라서, Coarse Feature map을 Dense map으로 Upsampling하는 방법을 고려해야 합니다.

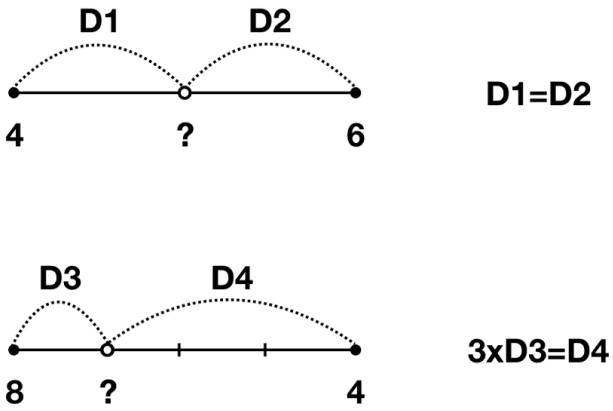
또한, Pooling의 중요한 역할 중 하나는 특징맵의 크기를 줄임으로써 학습 파라미터의 수를 감소 시키는 것인데, 이러한 과정이 사라지면 파라미터의 수가 급격히 증가하고 이로인해 더

많은 학습 시간을 요구하게 됩니다.

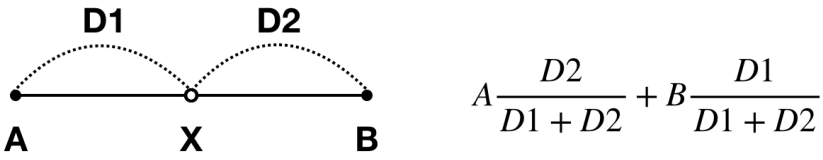
Interpolation

만약 10x10이미지를 320x320이미지로 확대하려면 어떻게 해야할까?

대표적인 방법이 바로 Bilinear interpolation이다. Bilinear interpolation을 이해하기 위해서는 linear interpolation을 우선적으로 이해할 필요가 있다.

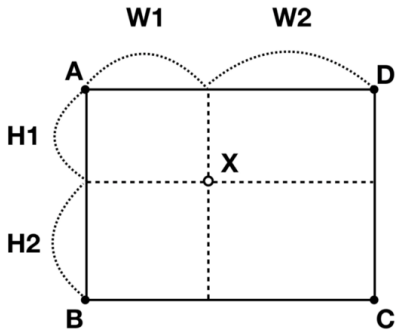


위의 값은 직관적ㅇ로 5, 아래 값은 7임을 직관적으로 예측할 수 있습니다. 이렇게 두 지점 사이의 값을 추정할 때 직관적으로 사용하는 방법이 바로 Linear interploation입니다.



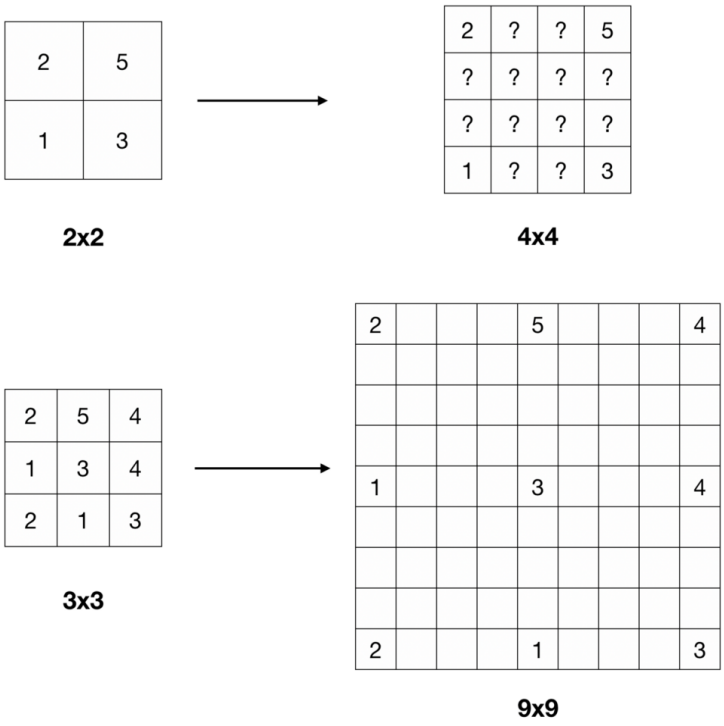
요런 느낌?!

Bilnear interpolation은 이러한 Liner interpolation을 2차원으로 확장한 것이다.



$$X = \left(A\frac{H2}{H1+H2} + B\frac{H1}{H1+H2}\right)\frac{W2}{W1+W2} + \left(D\frac{H2}{H1+H2} + C\frac{H1}{H1+H2}\right)\frac{W1}{W1+W2}$$

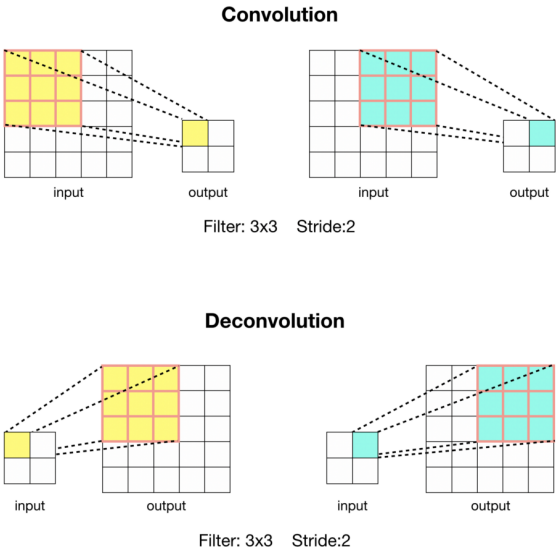
이제 우리는 다음과 같은 Feature map의 빈 영역을 추정할 수 있습니다.



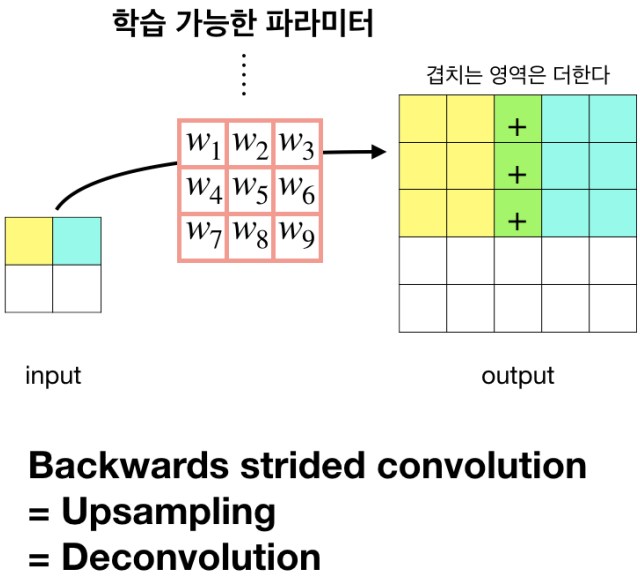
Backwards convolution

Dense prediction을 위한 Upsampling 방법에는 Bilinear interpolation처럼 정해진 방법만 있는 것은 아니다. 즉, Up-sampling도 학습이 가능하다.

Stride가 2 이상인 Convolution 연산의 경우 입력 이미지에 대해 크기가 줄어든 특징 맵을 출력한다. 이것은 Down-sampling에 해당한다.



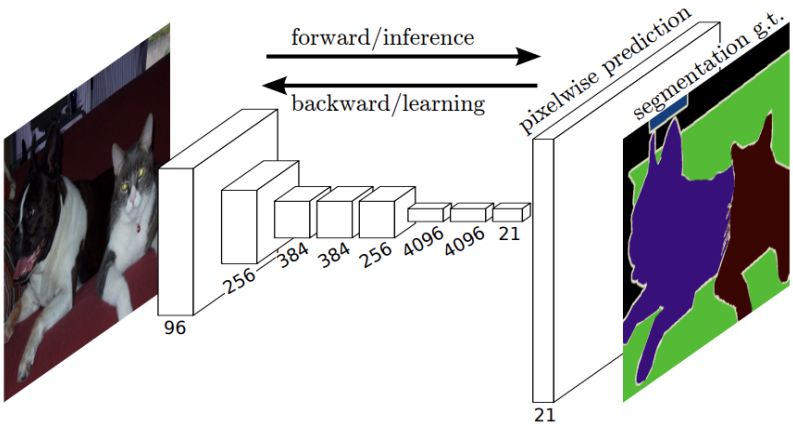
Convolution연산을 반대로 할 경우 자연스럽게 Up-sampling 효과를 볼 수 있다. 또한, 이 때 사용하는 Filter의 가중치 값은 학습 파라미터에 해당한다.



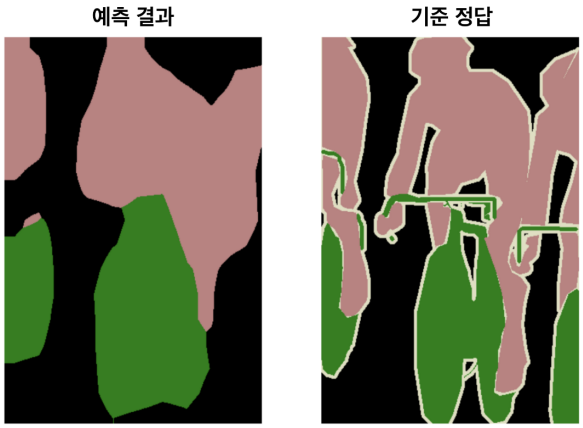
Backwards strided convolution
= Upsampling
= Deconvolution

FCNs에서는 Bilinear interpolation과 Backwards convolution 두 가지 방법을 사용하여 Coarse Feature map으로부터 Dense prediction을 구했다.

초기 Segmentation을 위한 모델은 다음과 같이 VGG모델을 컨볼루션화한 구조에 Bilinear interpolation작업을 더함으로써 얻을 수 있다.



이 처럼, Interpolation을 통해 coarse map에서 dense map을 도출할 수 있었지만, 근본적으로 feature map의 크기가 너무 작기 때문에 예측된 dense map의 정보는 여전히 거칠 수 밖에 없다.

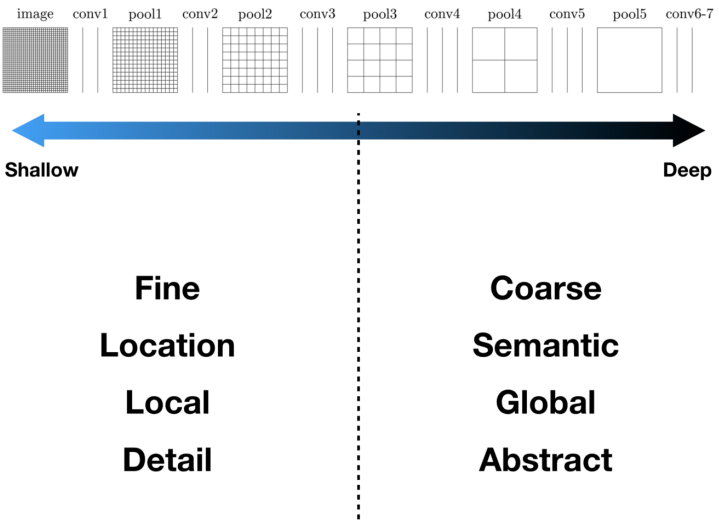


무려 Stride가 32인 Up-sampling 과정에서 얼마나 많은 정보가 손실될 지를 생각해보면 당연한 결과이다.

보다 정교한 Semgentation을 위해서 추가적인 작업이 필요한 이유다.

Skip Architecture

본 논문에서는 정확하고 상세한 구분(Segmentation)을 얻기 위해 Deep & Coarse(추상적인) 레이어의 의미적(Semantic) 정보와 Shallow & fine층의 외관적(appearance)정보를 결합한 Skip architecture를 정의합니다.

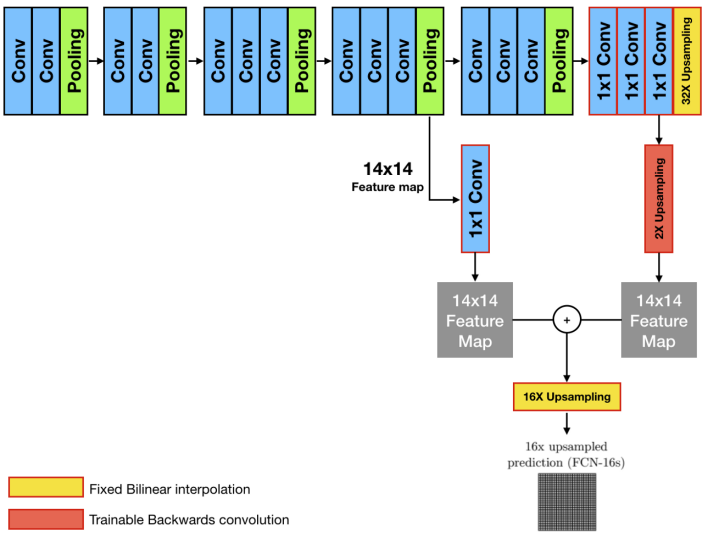


이러한 insight는 *Visualizing and Understanding Convolutional Networks*라는 연구를 통해서도 엿볼 수 있다고 합니다.

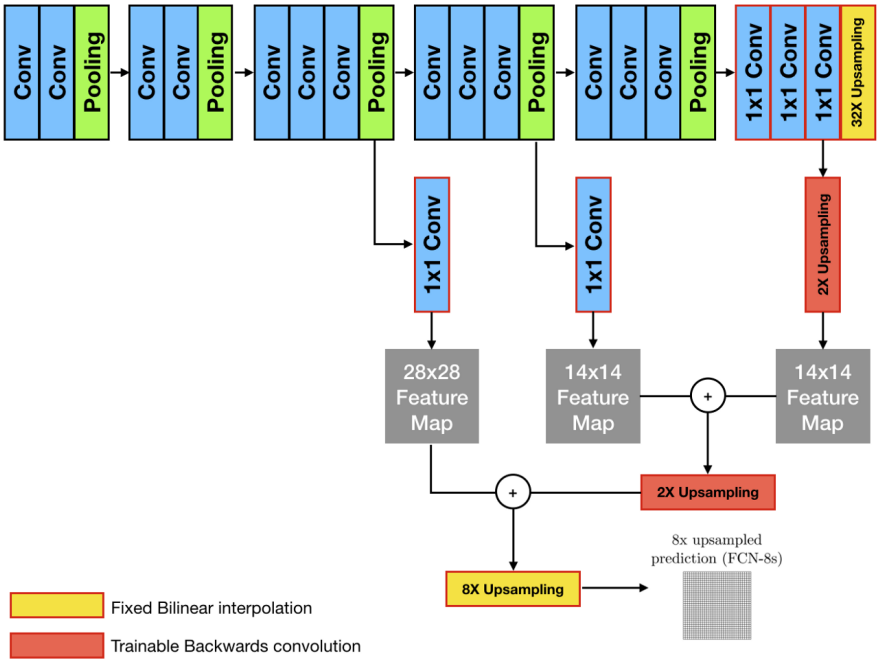
시각화 모델을 통해 입력 이미지에 대해 얕은 층에 대해서는 주로 직선 및 곡선, 색상 등의 낮은 수준의 특징에 활성화되고, 깊은 층에서는 보다 복잡하고 포괄적인 개체 정보에 활성화된다는 것을 확인할 수 있다.

또한 얕은 층에선 local feature를 깊은 층에선 global feature를 감지한다고 볼 수 있다.

FCNs연구팀은 이러한 insight를 기반으로 앞에서 구한 Dense np에 얕은 층의 정보를 결합하는 방식으로 Semgentation의 품질을 개선했습니다.



FCN-16s



FCN-8s

각 Pooling에 Prediction을 위해 추가된 Conv layer의 필터는 0으로, Trainable Backwards convolutions은 Bilinear interpolation으로 초기화한 후 학습을 진행하였다.

이러한 Skip Architecture를 통해 다음과 같이 개선된 Segmentation 결과를 얻을 수 있었다.

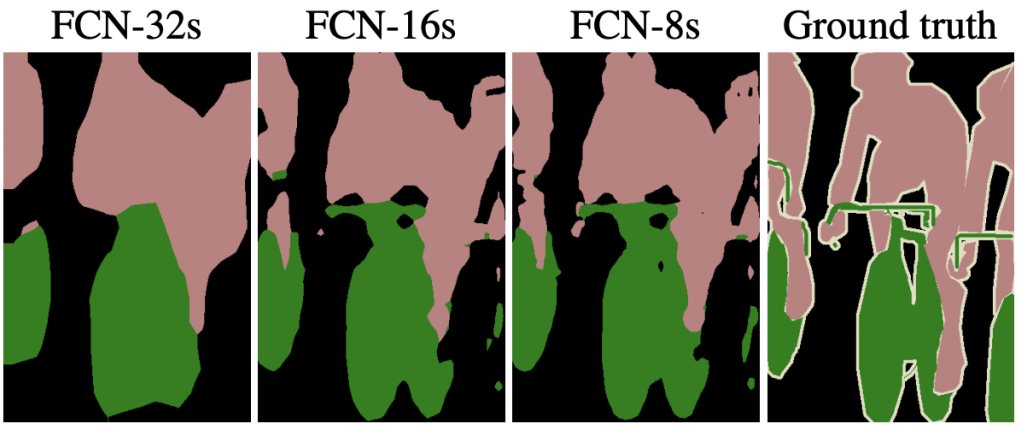


Figure 4. Refining fully convolutional nets by fusing information from layers with different strides improves segmentation detail. The first three images show the output from our 32, 16, and 8 pixel stride nets (see Figure 3).

정리

FCNs는 기존의 딥러닝 기반 이미지 분류를 위해해 학습이 완료된 모델의 구조를 Semantic Segmentation 목적에 맞게 수정하여 Transfer learning하였다.

Convolutionalized 모델을 통해 예측된 Coarse map을 원본 이미지 사이즈와 같이 세밀(Dense)하게 만들기 위해 Up-samplling을 수행하였다.

또한 Deep Neural Network에서 얕은 층의 Local 정보와 깊은층의 Semantic정보를 결합하는 Skip architecture를 통해 보다 정교한 Segmantation 결과를 얻을 수 있었습니다.

FCNs는 End-to-End 방식의 Fully-Convolution Model을 통한 Dense Prediction혹은 Semantic Segmentation의 초석을 닦은 연구로써 이후후 많은 연구들에 영향을 주었습니다.

- U-net Paper Review (***Complete***)
- Deconvolution Paper Review (*Not yet)*
- Visualizing CNN Paper Review (*Not yet)*

Reference

- Fully Convolutional Networks for Semantic Segmentation
- https://medium.com/@msmapark2/fcn-논문-리뷰-fully-convolutional-networks-for-semantic-segmentation-81f016d76204