4️⃣

Semantic Segmentation

강의 소개

1강과 3강의 Image classification은 사진이 주어졌을 때 사진 전체를 카테고리로 분류합니다.

반면 Semantic Segmentation은 사진이 주어졌을 때 사진 내 각 픽셀을 카테고리로 분류하는 task 입니다.

즉, 하나의 사진이 아닌, **사진에 있는 모든 물체들을 분류**한다는 것입니다. 본 강의에서는 먼저 최초의 end-to-end segmentation 모델 FCN을 시작으로 Hypercolumn 모델을 배웁니다.

다음으로 segmentation의 breakthrough라고 볼 수 있는 UNet 모델에 대해 공부하고 Pytorch 코드 실습을 합니다.

끝으로 최근까지 좋은 성능을 보이고 있는 DeepLab v3에 대해 배웁니다.

Further Reading

- Checkerboard artifacts: https://distill.pub/2016/deconv-checkerboard/
- FCN: https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Long_Fully_Co
- UNet: https://arxiv.org/pdf/1505.04597.pdf

Semantic segmentation

What is semantic segmentation?

Segementation

- Classify **each pixel** of an image into **a category**
- Don't care about instances
  - Only care about **semantic category**
  - ex. 각 사람(insatnce)들을 분류 X, 사람(semantic category)이면 같은 카테고리에 O

Where can semantic segmentation be applied to?

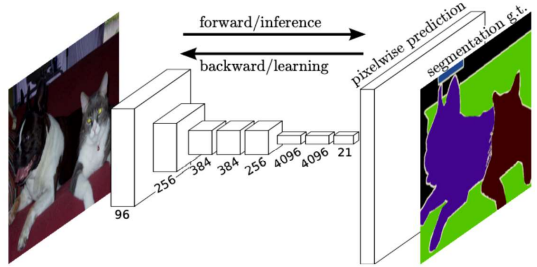
- medical images
- autonomous driving
- computational photography

Semantic segmentation architectures

Fully Convolutional Networks (FCN)

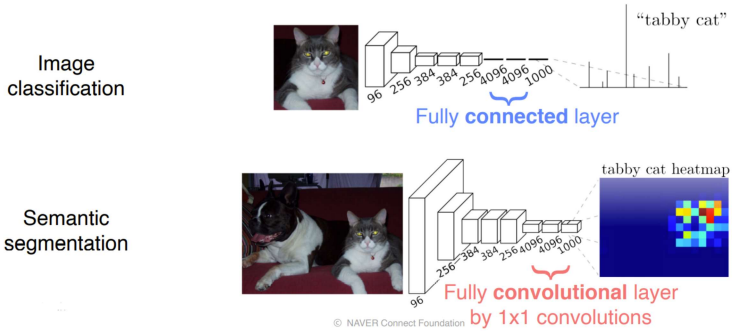
Fully convolutional networks

- The first **end-to-end architecture** for **semantic segmentation**
- Take an image of **an arbitrary size** as **input** + **output a segmentation map** of the corresponding size to the output



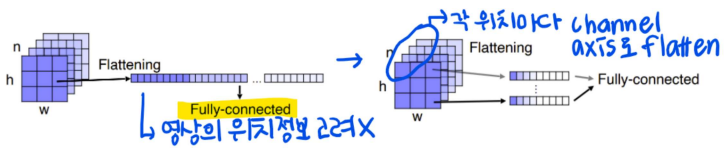
Fully connected VS. Fully convolutional

- `Fully connected layer`
  - output a **fixed dimensional** vector + **discard spatial coordinates**
  - ∵ dimension이 fix되지 않으면, fc layer 사용 불가능
- `Fully convolutional layer`
  - output **a classification map** which has **spatial coordinates ⇒ 호환성** ↑
  - 1x1 convolution 으로 구성 → 각 위치마다 분류 결과를 출력하는 map

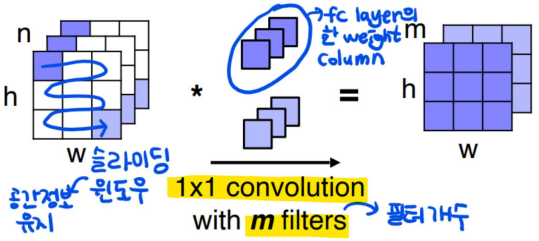


Interpreting fully connected layers as 1x1 convolutions

- `A fully connected layer`
  - classifies **a single feature vector**

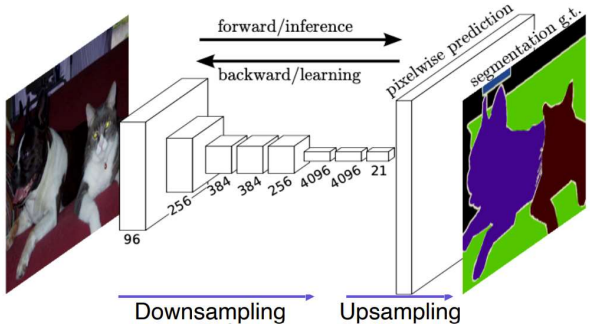


- `A 1x1 convolution layer`
  - classifies **every feature vector** of the convolutional feature map
  - (그림) fc layer의 한 weight column (?)
  - 문제 → predicted **score map** is in **a very low-resolution**
    - ∵ for having **a large receptive field**, several spatial **pooling layers** are deployed
  - 해결 → **enlarge** the score map by **upsampling**



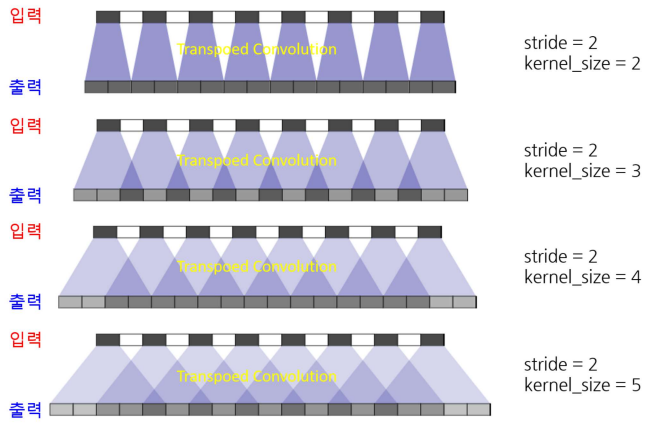
Upsampling

- resize **a small activation map** to the size of **the input image**
  - `Unpooling`
    - pooling, stride 제거 → high-resolution
    - but, receptive field ↓ ⇒ 영상의 전반적인 context 파악 불가능
  - `Transposed convolution`
  - `Upsample and convolution`
    - pooling, stride 적용 → receptive field ↑ 한 후에, upsampling으로 high-resolution

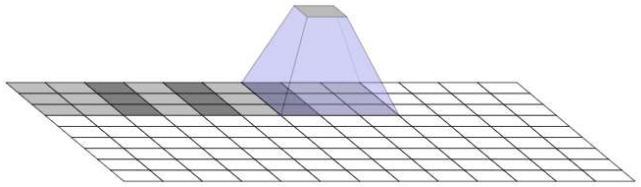


Transposed convolution

- work by **swapping** the forward and backward passes of convolution
  - `Checkerboard artifacts` → due to **uneven overlaps**
    - overlap된 영역을 따로 tuning 해야
  - ▼ 📎 checkerboard artifact
    - Transposed Convolution 또한 Convolution 연산이므로 커널이 슬라이딩 윈도우 방식으로 이동하면서 연산이 진행됩니다. 특히, **kernel의 크기와 stride의 크기에 따라서 transposed convolution 연산의 overlap 영역이 발생할 수 있습니다.**
    - 이 때 Transposed Convolution을 어떻게 사용하느냐에 따라서 overlap이 없을 수도 있고 또는 많이 생길 수도 있습니다.

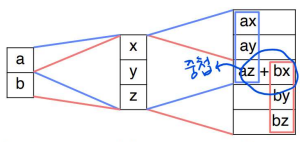


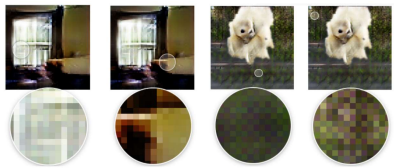
- 이러한 overlap 패턴은 2차원에서도 같은 원리로 발생합니다. 심지어 2개의 차원에서 동시에 겹치는 영역이 발생하여 앞의 1차원에서보다 배로 겹치게 됩니다. 위 그림과 같이 `stride = 2, kernel_size = 3`인 경우 가장 많이 겹친 곳은 4번 겹친 것을 확인할 수 있습니다.



- 지금 까지 내용을 정리하면 **Transposed Convolution을 이용하여 Upsampling을 할 때,** kernel_size % stride≠0 **인 경우에 어떤 영역의 중복 연산이 배로 발생하게 되고 그 영역은 계속 누적되어 큰 값을 가지게 됩니다.** 데이터의 차원이 늘어날수록 겹치는 구간의 겹치는 횟수가 배로 늘어나게 됩니다.
- 이렇게 **연산이 누적되는 횟수의 차이가 발생하게 되어 checkboard artifact가 발생**하게 됩니다.

- 참고 : Transposed Convolution과 Checkboard artifact - gaussian37

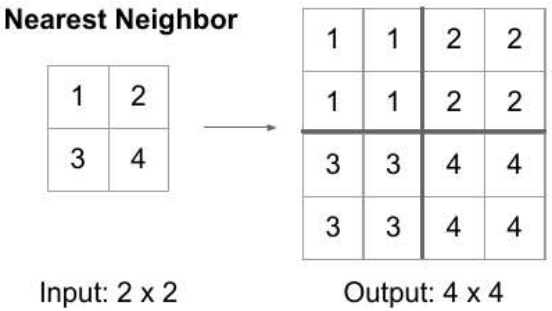




Better approaches for upsampling

- **Avoid overlap issues** in transposed convolution
- **Decompose** into `spatial upsampling` and `feature convolution`
  - {`Nearest-neighbor (NN)`, `Bilinear`} **interpolation** followed by **convolution**

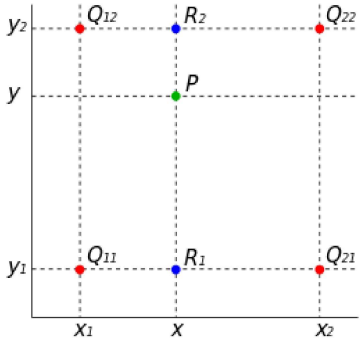
▼ 📎 NN-resize convolution



- `Nearest upsampling`은 Dense layer를 그대로 늘려 빈 구역에 채워 넣는 예입니다. 이름 그대로 scale을 키우는데, 키운 위치에서 **원본에서 가장 가까운 값을 그대로 적용**하는 방법입니다.

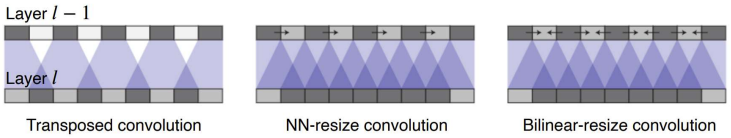
- 원래 2x2 matrix가 있을 때 이를 2배로 키워 4x4 size의 matrix를 만들어 주려고 하는데, 늘린만큼 그 근처에 있는 데이터를 사용하는 방법입니다.
- 참고 : 딥러닝 Segmentation(7) - Upsampling의 다양한 방법 (velog.io)

▼ 📎 Bilinear-resize convolution

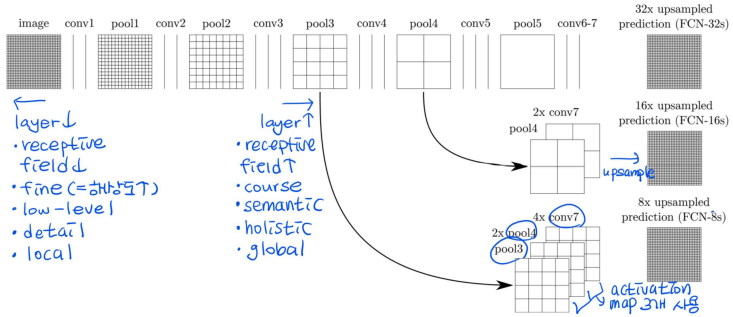


- `Bilinear Interpolation`은 x, y의 2차원에 대해 선형 보간법을 통해 필요한 값을 채우는 방식입니다.
- 위처럼 2x2 matrix를 4x4 size로 upsampling하고 싶을 때, 빈 값을 채워야 하는데 이때 **축을 두 가지 interpolation을 적용**한 것을 Bilinear Interpolation이라고 합니다.
- R1이 Q11, Q21의 x축 방향 interpolation 결과이고, R2는 Q12, Q22의 x축 방향의 interpolation 결과입니다.
- 그리고 R1, R2를 y축 방향으로 interpolation하면 새로운 위치 P의 값을 추정할 수 있습니다. 이러한 선형 보간 방법으로 Interpolation을 하는 방법이 있고 Bicubic interpolation의 경우 삼차보간법을 사용합니다.

- 참고 : 딥러닝 Segmentation(7) - Upsampling의 다양한 방법 (velog.io)



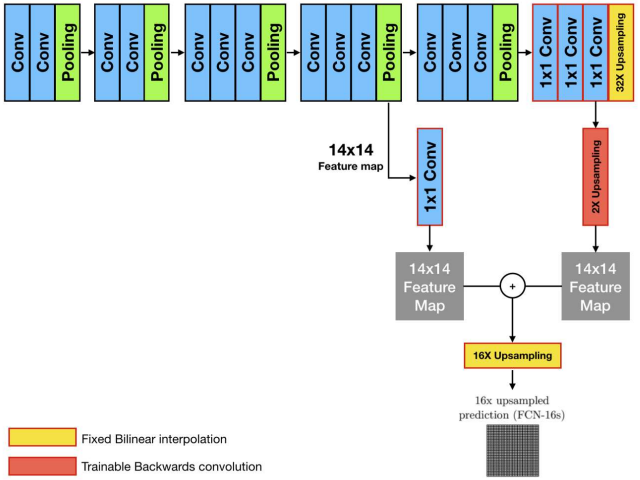
Adding skip connections for enlarging the score map



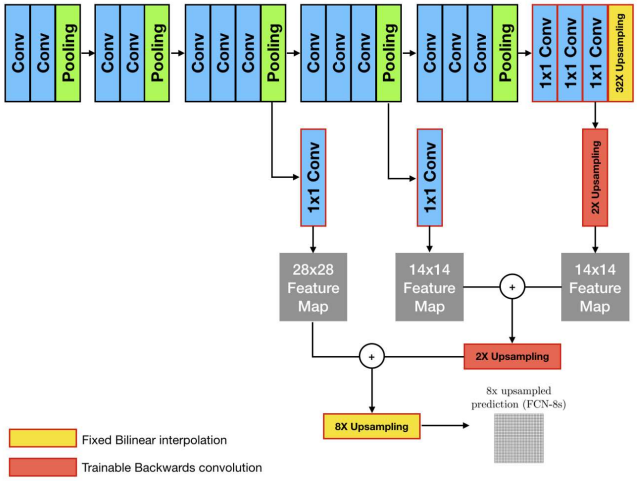
▼ skip architecture ?

- 정확하고 상세한 구분(Segmentation)을 얻기 위해
- **Deep & Coarse**(추상적인) 레이어의 의미적(Semantic) 정보와 **Shallow & fine** 층의 외관적(appearance) 정보를 결합한 **Skip architecture**를 정의
- 시각화 모델을 통해 입력 이미지에 대해 **얕은 층에서는 주로 직선 및 곡선, 색상 등의 낮은 수준의 특징에 활성화**되고, **깊은 층에서는 보다 복잡하고 포괄적인 개체 정보에 활성화**된다는 것을 확인할 수 있다.
- 또한 얕은 층에선 local feature를 깊은 층에선 global feature를 감지한다고 볼 수 있다. FCNs 연구팀은 이러한 직관을 기반으로 앞에서 구한

Dense map에 얕은 층의 정보를 결합하는 방식으로 Segmentation의 품질을 개선하였다.



FCN-16s

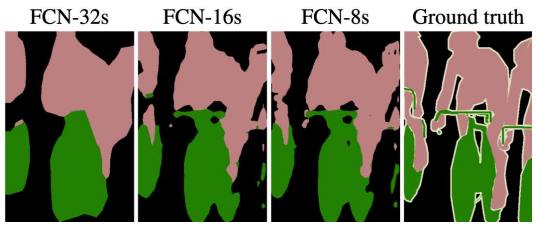


FCN-8s

- 참고 : FCN 논문 리뷰 — Fully Convolutional Networks for Semantic Segmentation | by 강준영 | Medium
- 읽어보기 : 2. FCN :: Time Traveler (tistory.com)
- 읽어보기 : semantic segmentation의 목적과 대표 알고리즘 FCN의 원리 by bskyvision

Features of FCN

- `Faster`
  - The end-to-end architecture that **does not depend on other hand-crafted components**
- `Accurate`
  - Feature representations and classifiers are jointly **optimized (?)**



- 중간 layer를 upsampling하면 resolution ↑

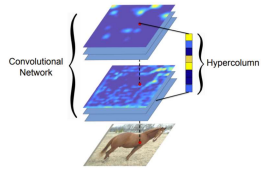
▼ 📎 FCN-32s / FCN-16s / FCN-8s

- FCN-32s = input image의 1/32의 크기에서 32배만큼 upsampling 한 결과이고
- FCN-16s = (pool5의 결과를 2배 upsampling + pool4의 결과)를 16배 upsampling 한 것
- FCN-8s = (FCN-16s의 중간 결과를 2배 upsampling + pool3에서의 결과)를 8배 upsampling 한 것
- 참고 : [머신러닝 공부] 딥러닝/FCN (Fully Convolutional Networks) (tistory.com)

Hypercolumns for object segmentation

Fully convolutional networks

- CNN layers typically use the **output** of the last layer as **feature representation**
  - Too **coarse spatially**
- `Hypercolumn` at a pixel is **a stacked vector** of all CNN units on that pixel
  - **Fine localized information** is extracted from earlier layers



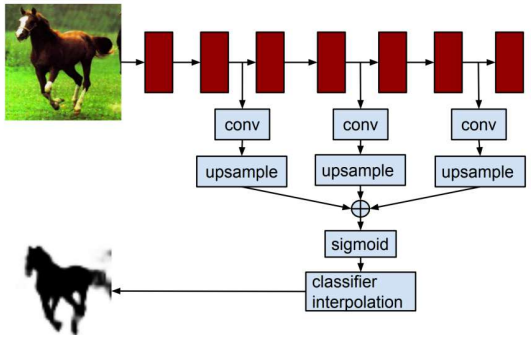
- **Coarse semantic information** is extracted from latter layers

▼ 📎 hypercolumn

- 참고 : 렌더링을 응용한 이미지 세분화 'PointRend' (tistory.com)

Overall architecture

- Concurrent work
  - very similar to FCN (?)
- Difference → Apply to **each bounding box (?)**
  - bounding box를 찾기 위해 third party 사용 → end to end X

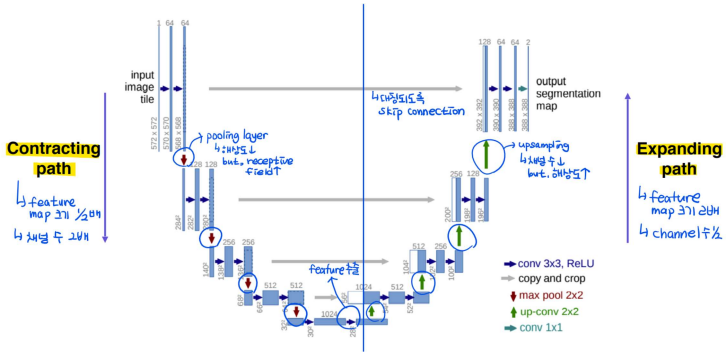


U-Net

U-Net

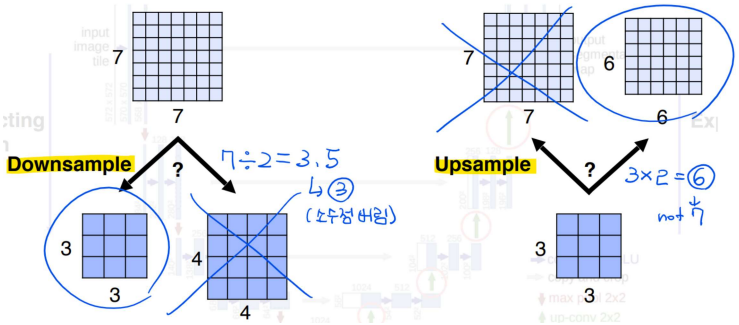
- Built upon `fully convolutional networks`
  - Share the same FCN property
- Predict a dense map by concatenating feature maps from contracting path

  - Similar to `skip connections` in FCN
- Yield more **precise segmentations**



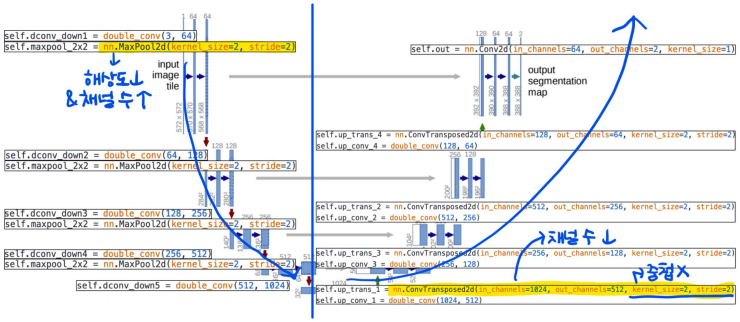
- `Contracting Path`
  - Repeatedly applying **3x3 convolutions**
  - **Doubling** the number of feature **channels**
  - Being used to capture **holistic context**
- `Expanding Path`
  - Repeatedly applying **2x2 convolutions (?)**
  - **Halving** the number of feature **channels**
  - **Concatenating** the corresponding feature maps from the contracting path → **skip connection**
    - Concatenation of feature maps provides **localized information**

An even number is required for input and feature sizes



- feature size를 odd number로 하면 안 됨
  - ∵ downsampling / upsampling 시 예상했던 값이 나오지 않을 수 있음

PyTorch code for U-Net




```
def double_conv(in_channels, out_channels):
  return nn.Sequential(
    nn.Conv2d(in_channels, out_channels, 3),
    nn.ReLU(inplace=True), # inplace 연산은 결과값을 새로운 변수에 값을 저장하는 대신 기
    nn.Conv2d(out_channels, out_channels, 3),
    nn.ReLU(inplace=True)
  )
```

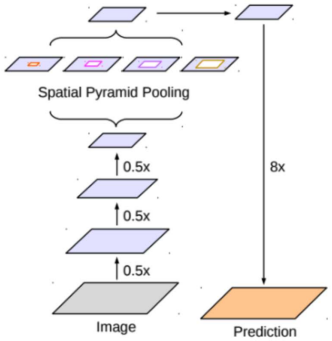

▼ 📎 kernel_size = 2 & stride = 2 이면, 중첩이 생기지 않는 이유

- uneven overlap은 커널 사이즈(output window size)가 stride로 나눠지지 않을 때 생긴다.
- 참고 : Checkerboard Artifacts 체커보드 아티팩트 (tistory.com)

DeepLab

- DeepLab v1 (2015)
  - Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected **CRFs**. ICLR 2015.
- DeepLab v2 (2017)
  - DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, **Atrous Convolution**, and Fully Connected CRFs. TPAMI 2017.
- DeepLab v3 (2017)
  - Rethinking **Atrous Convolution** for Semantic Image Segmentation. arXiv 2017.
- DeepLab v3+ (2018)
  - Encoder-Decoder with **Atrous Separable Convolution** for Semantic Image Segmentation. ECCV 2018.

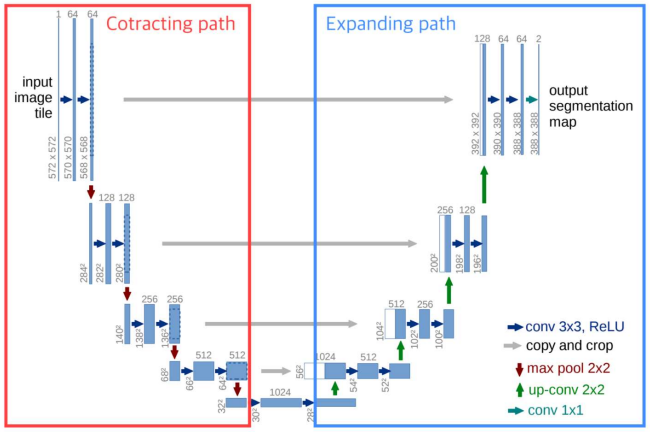
▼ 📎 DeepLab



- 아래 요소들로 이루어져있습니다.
  - Encoder: ResNet with atrous convolution
  - ASPP
  - Decoder: Bilinear upsampling
- 참고 : [DL] Semantic Segmentation (FCN, U-Net, DeepLab V3+) (tistory.com)
- 읽어보기 : [Part VII. Semantic Segmentation] 6. DeepLab [2] - 라온피플 머신러닝 아카데미 - : 네이버 블로그 (naver.com)

▼ 📎 Encoder-Decoder 구조 + 의미

- U-Net 구조 (`Contracing Path` : Encoder ↔ `Expanding Path` : Decoder)

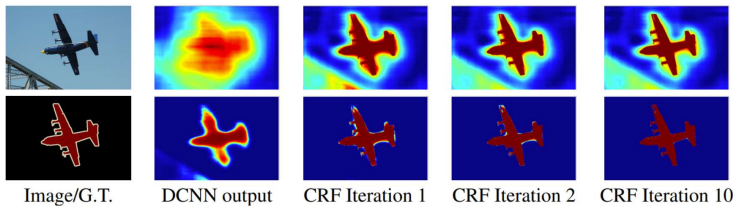


- Encoder-decoder 구조 또한 semantic segmentation을 위한 **CNN 구조**로 자주 활용되고 있습니다.
- 왼쪽의 `encoder` 부분에서는 점진적으로 **spatial dimension을 줄여가면서 고차원의 semantic 정보를 convolution filter가 추출**해낼 수 있게 됩니다.
- 이후 오른쪽의 `decoder` 부분에서는 encoder에서 **spatial dimension 축소로 인해 손실된 spatial 정보를 점진적으로 복원**하여 보다 정교한 boundary segmentation을 완성하게 됩니다.
- U-Net이 여타 encoder-decoder 구조와 다른 점은, 위 그림에서 가운데 놓인 회색 선입니다.
- **Spatial 정보를 복원하는 과정**에서, 이전 encoder feature map 중 **동일한 크기를 지닌 feature map을 가져 와 prior로 활용**함으로써 더 정확한 boundary segmentation이 가능하게 만듭니다.
- 이번에 발표된 DeepLab V3+에서는 U-Net과 유사하게 intermediate connection을 가지는 encoder-decoder 구조를 적용하여, 보다 정교한 object boundary를 예측할 수 있게 됩니다

- 참고 : [DL] Semantic Segmentation (FCN, U-Net, DeepLab V3+) (tistory.com)

Conditional Random Fields (CRFs)

- **CRF post-processes** a segmentation map to be refined to follow **image boundaries**
  - 1st row → score map (**before** softmax)
  - 2nd row → belief map (**after** softmax)

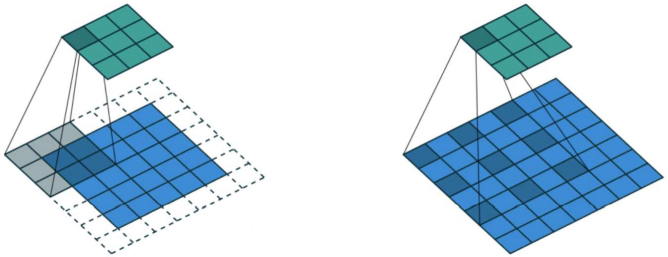


▼ 📎 CRF ?

- Classification과 같이 object-centric한 경우 가능한 높은 수준의 공간적인 불변성(spatial invariance)를 얻기 위해 여러 단계의 conv+pooling을 통해 영상 속에 존재하며 변화에 영향을 크게 받지 않은 강인한 특징을 추출해야하며, 이로인해 detail한 정보보단 global한 정보에 집중하게 됨
- 반면 semantic segmenation은 픽셀 단위의 조밀한 예측이 필요해 classification 네트워크 기반으로 segmentation 망을 구상하게 된다면 계속 feature map의 크기가 줄어들게되는 특성상 detail한 정보들을 잃게 됨
- 이 문제에 대한 해결책으로 FCN에선 skip connection을 사용하였고, dilated conv나 DeepLab에서는 마지막에 오는 pooling layer 2개를 없애고 dilated/atrous conv를 사용함
- 하지만 이러한 방법을 사용하더라도 분명히 한계는 존재하기에 DeepLab에서는 atrous conv에 그치지 않고 **CRF를 후처리 과정으로 사용하여 픽셀 단위 예측의 정확도를 더 높일 수 있게 됨**
- `CRF`는 **segmentation을 수행한 뒤 생기는 segmentation noise를 없애는 용도**로 많이 사용됨
- 참고 + 읽어보기 : DeepLab · Seongkyun Han's blog
- 읽어보기 : Convolutional CRFs for Semantic Segmentation (tistory.com)

Dilated convolution

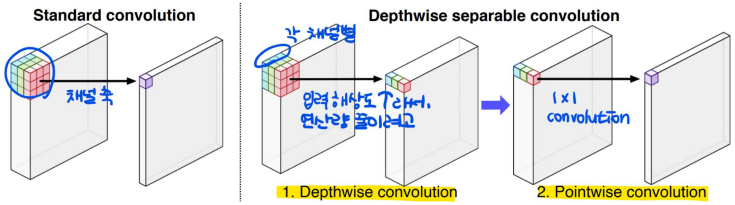
- (= `Atrous convolution`)
- `Dilation factor`
  - **inflate the kernel** by **inserting spaces** between the kernel element
  - Enable exponential **expansion of the receptive field** (단, parameter 개수는 늘어나지 X)



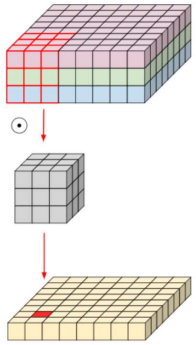
Depthwise separable convolution (proposed by Howard et al.)

- ~~No. parameters~~ → 연산량 ($\because D_F^2$가 곱해짐) ($\because$ parameter 수 * feature map size = 연산량)
  - Standard conv → $D_K^2 M N D_F^2$ (6차)

- Depthwise separable conv → $D_K^2 M D_F^2 + MN D_F^2$ → max(5차, 4차) = (5차)
  - D_k / D_F → Kernel / feature map size
  - M / N → input / output channels

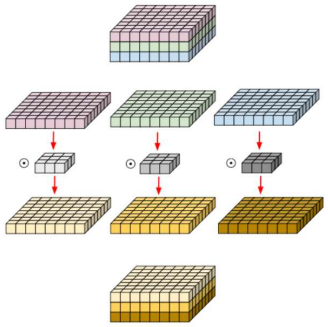


▼ 📎 Depthwise separable convolution



- 위 그림처럼, 입력 이미지가 8×8×3 (H×W×C) 이고, convolution filter 크기가 3×3 (F×F) 이라고 했을 때, filter 한 개가 가지는 파라미터 수는 3×3×3 (F×F×C) = 27 이 됩니다.
- 만약 filter가 4개 존재한다면, 해당 convolution의 총 파라미터 수는 3×3×3×4 (F×F×C×N) 만큼 지니게 됩니다.

- 위 그림처럼, Convolution 연산에서 channel 축을 filter가 한 번에 연산하는 대신에,
- **아래 그림처럼, 같이 입력 영상의 channel 축을 모두 분리시킨 뒤, filter의 channel 축 길이를 항상 1로 가지는 여러 개의 convolution filter로 대체시킨 연산을 depthwise convolution이라고 합니다.**

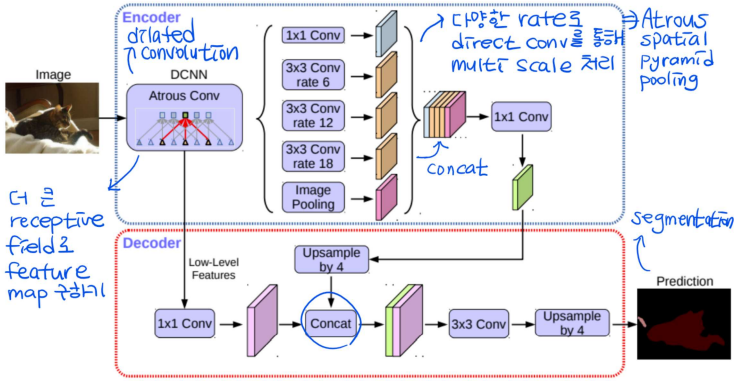


- **이제, 위의 depthwise convolution으로 나온 결과에 대해, 1×1×*C* 크기의 convolution filter(pointwise convolution)를 적용한 것을 depthwise separable convolution 이라 합니다.**
- 이처럼 복잡한 연산을 수행하는 이유는 기존 convolution과 유사한 성능을 보이면서도 사용되는 **파라미터 수와 연산량을 획기적으로 줄일 수 있기 때문**입니다.
  - 예를 들어, 입력값이 8×8×3 라면
  - 16개의 3×3 convolution 필터를 적용할 때 사용되는 파라미터의 개수는
  - Convolution: 3×3×3×16 = 432
  - Depthwise separable convolution: 3×3×3(x1) + 3×16 = 27 + 48 = 75임을 확인할 수 있습니다.
- Depthwise separable convolution은 기존 convolution filter가 spatial dimension과 channel dimension을 동시에 처리하던 것을 따로 분리시

켜 각각 처리한다고 볼 수 있습니다.

- 이 과정에서, 여러 개의 필터가 spatial dimension 처리에 필요한 파라미터를 하나로 공유함으로써 파라미터의 수를 더 줄일 수 있게 됩니다.
- 두 축을 분리시켜 연산을 수행하더라도 최종 결과값은 결국 두 가지 축 모두를 처리한 결과값을 얻을 수 있으므로, 기존 convolution filter가 수행하던 역할을 충분히 대체할 수 있게 됩니다.
- 픽셀 각각에 대해서 label을 예측해야 하는 semantic segmentation은 난이도가 높은 편에 속하기 때문에 CNN 구조가 깊어지고, receptive field를 넓히기 위해 더 많은 파라미터들을 사용하게 되는 상황에서, separable convolution을 잘 활용할 경우, 모델에 필요한 parameter 수를 대폭 줄일 수 있게 되므로, 보다 깊은 구조로 확장하여 성능 향상을 꾀하거나, 기존 대비 메모리 사용량 감소와 속도 향상을 기대할 수 있습니다.
- 참고 : [DL] Semantic Segmentation (FCN, U-Net, DeepLab V3+) (tistory.com)

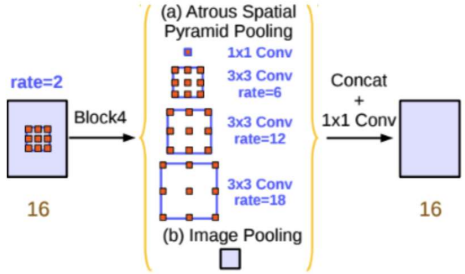
Deeplab v3+



▼ Deeplab v3+

- Encoder: ResNet with atrous convolution → **Xception** (Inception with separable convolution)
  - ResNet을 사용하던 encoder가 앞서 소개드린 separable convolution을 적극 활용한 구조인 **Xception**으로 대체
- ASPP → ASSPP (Atrous Separable Spatial Pyramid Pooling)
  - Multi-scale context를 얻기 위해 활용되던 ASPP에는, **separable convolution과 atrous convolution을 결합**한 atrous separable convolution이 적용된 ASPP로 대체
- Decoder: Bilinear upsampling → Simplified U-Net style decoder
- 참고 : [DL] Semantic Segmentation (FCN, U-Net, DeepLab V3+) (tistory.com)

▼ spatial pyramid pooling



- Semantic segmentaion의 성능을 높이기 위한 방법 중 하나입니다.
- **DeepLab V2에서는 feature map으로부터, 여러 개의 rate가 다른 atrous convolution을 병렬로 적용한 뒤, 이를 다시 합쳐주는 atrous spatial pyramid pooling (ASPP) 기법을 활용할 것을 제안하고 있습니다.**
- 최근 발표된 PSPNet에서도 atrous convolution을 활용하진 않지만, 이와 유사한 pyramid pooling 기법을 적극 활용하고 있습니다.

- 이러한 방법들은 **multi-scale context**를 모델 구조로 구현하여, 보다 정확한 semantic segmentation을 수행할 수 있도록 돕게 됩니다.
- DeepLab 에서는 ASPP를 기본 모듈로 사용하고 있습니다.
- 참고 : [DL] Semantic Segmentation (FCN, U-Net, DeepLab V3+) (tistory.com)

Reference

1. Semantic segmentation
   - Chen et al., Rethinking Atrous Convolution for Semantic Image Segmentation, arXiv 2017
   - Novikov et al., Fully Convolutional Architectures for Multi-Class Segmentation in Chest Radiographs, T-MI 2016
   - Aksoy et al., Semantic Soft Segmentation, SIGGRAPH 2018
2. Semantic segmentation architectures
   - Long et al., Fully Convolutional Networks for Semantic Segmentation, CVPR 2015
   - Hariharan et al., Hypercolumns for Object Segmentation and Fine-Grained localization, CVPR 2015
   - Ronneberger et al., U-Net: Convolutional Networks for Biomedical Image Segmentation, MICCAI 2015
   - Chen et al., Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs, ICLR 2015
   - Howard et al., MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, arXiv 2017
   - Chen et al., Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation, ECCV 2018