Fully Convolutional Neural Network

Semantic Segementation 기초와 Fully Convolutional Neural Network for Semantic Segmentation 논문 이해하기

INDEX

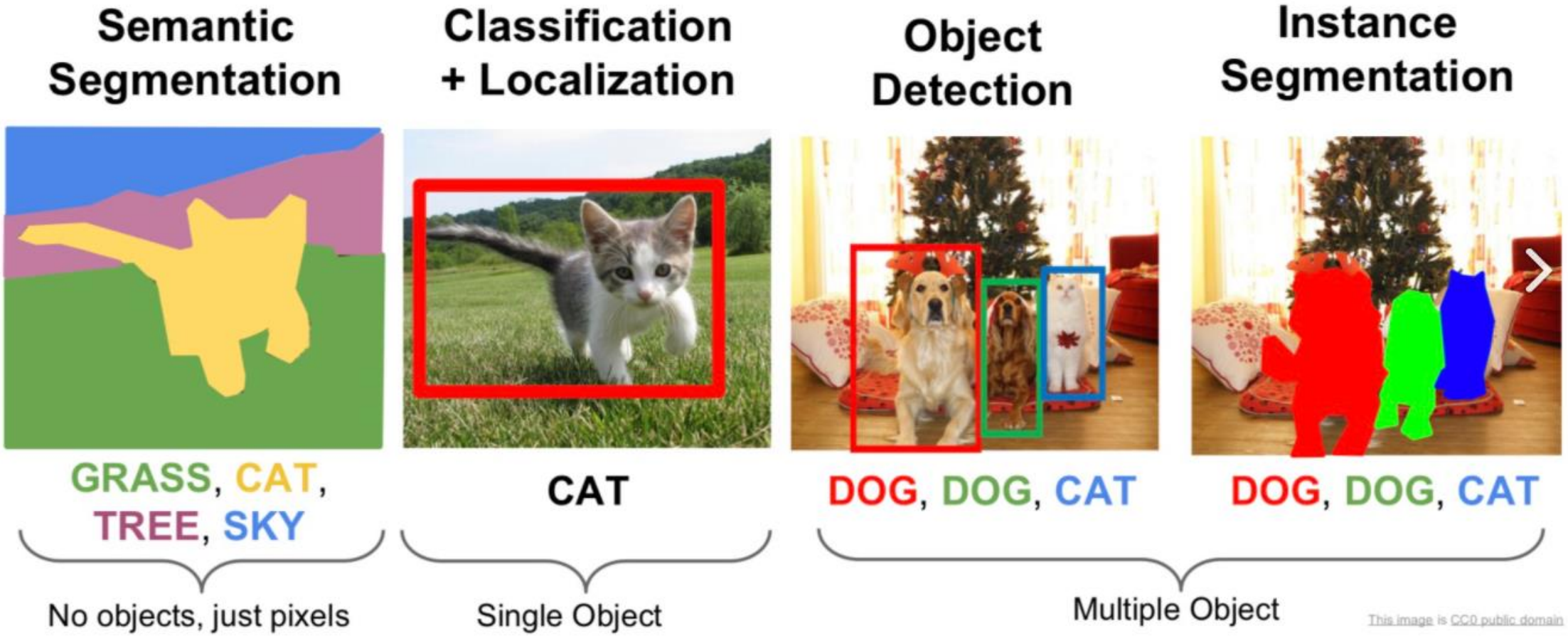
Semantic Segmentation

Code Review

Fully Convolutional Networks

1. Semantic Segmentation이란?

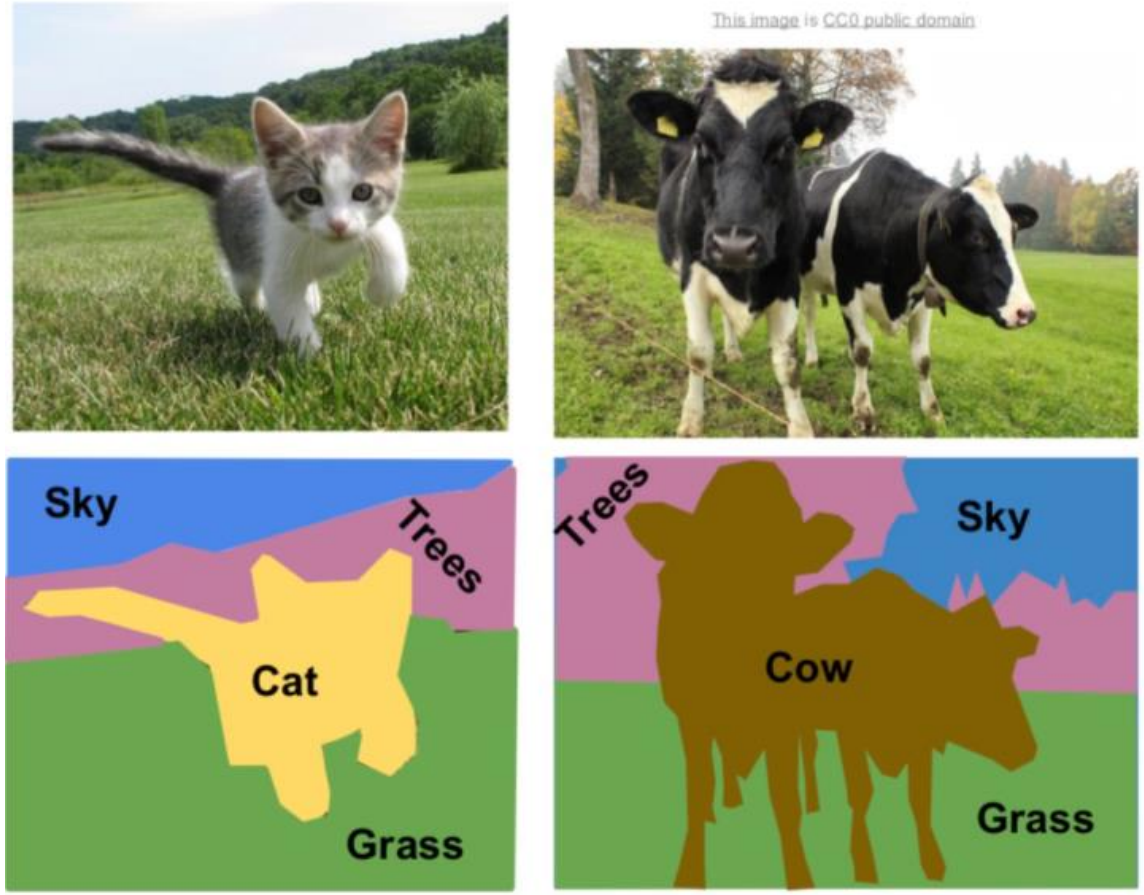
- (1) Computer Vision Tasks



출처 : CS231n, 11. Detection and Segmentation

1. Semantic Segmentation이란?

- (2) Semantic Segmentation



출처 : CS231n, 11. Detection and Segmentation

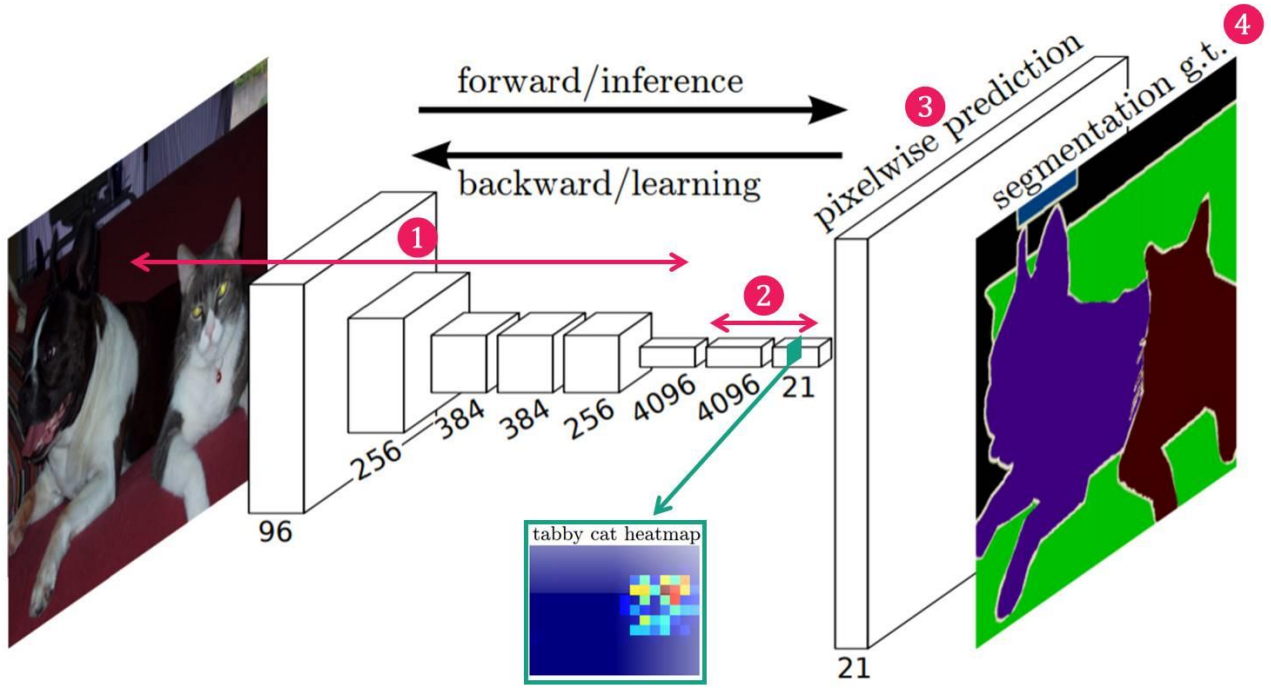
FCN

0. Abstract

(1) Abstract

Fully Convolutional Networks for semantic Segmentation[1]의 Contribution

1. End-to-end 방식의 Semantic Segmentation 방법을 제안
2. AlexNet을 시작으로 하는 CNN 계열 모델을 사용
3. Fully Convolutional + Skip Architecture 두가지 방법을 도입
4. 임의의 입력값에 대해서도 효율적인 Segmentation 결과를 생성 가능하고 SOTA 점수 달성



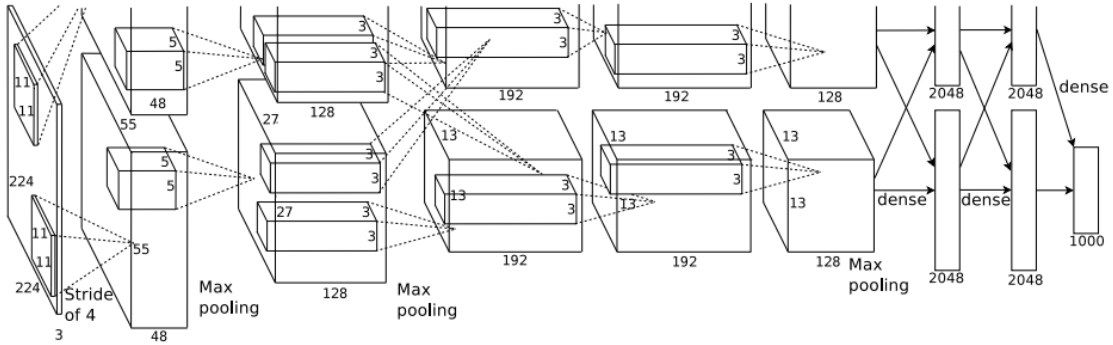
출처 : J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015

1. Introduction

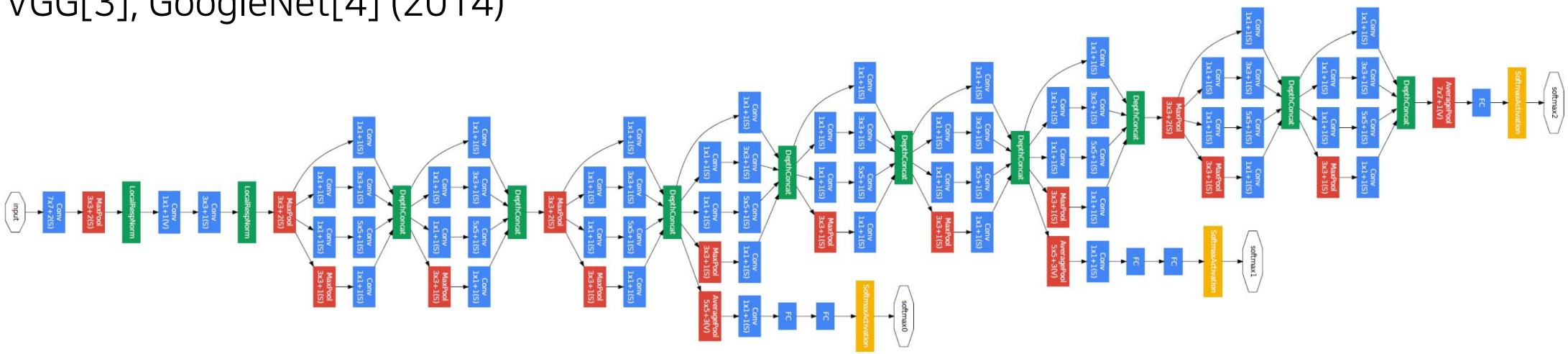
• (1) Convolutional networks

Recognition에서 Deep Learning 네트워크의 개발에 따른 다른 분야로의 확장

AlexNet (2012) [2]



VGG[3], GoogleNet[4] (2014)



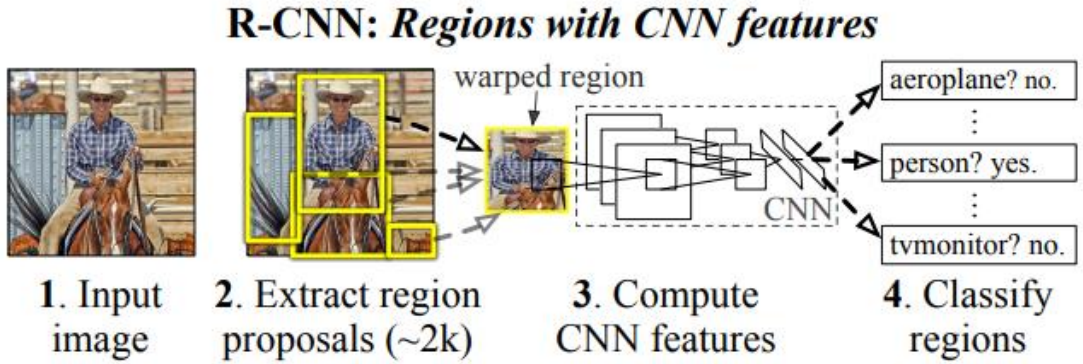
출처 : J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015

1 Introduction

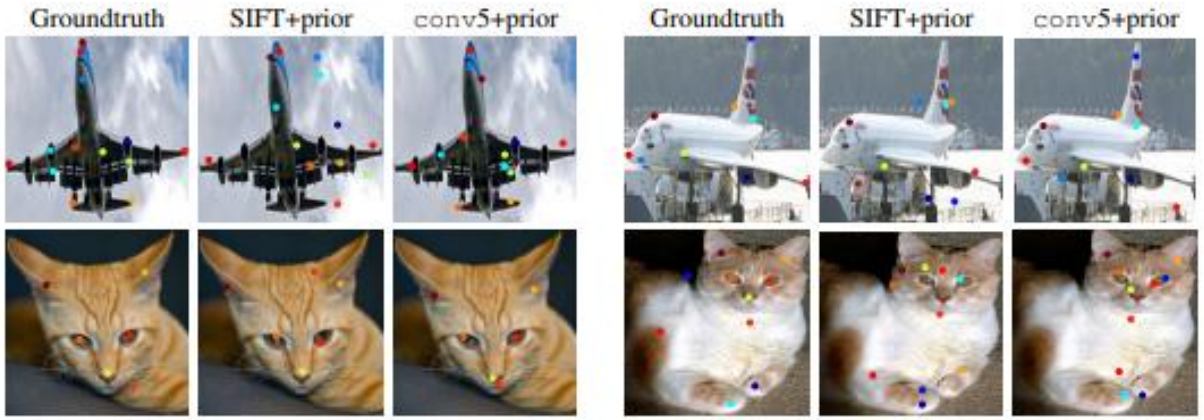
- (1) Convolutional networks

Recognition에서 Deep Learning 네트워크의 개발에 따른 다른 분야로의 확장

Bounding Box object detection [5]



Key point prediction [6]



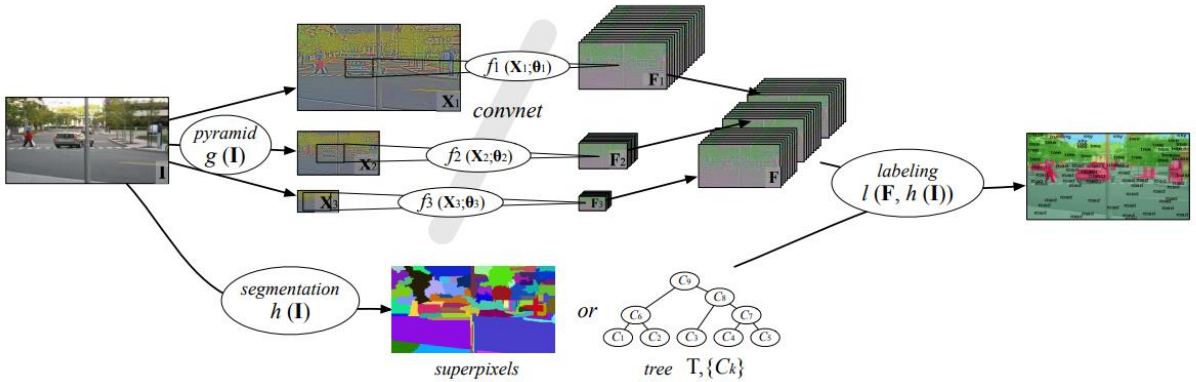
출처 : J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015

1 Introduction

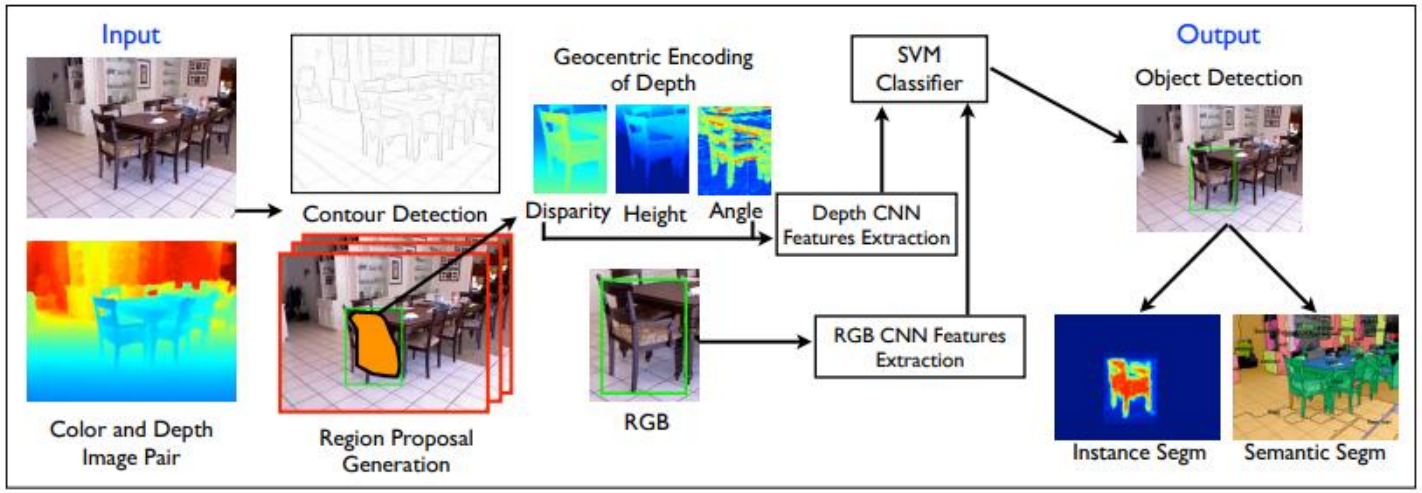
- (1) Convolutional networks

Prior Approaches convnets for semantic segmentation

Learning hierarchical features for scene labeling [7]



Learning Rich Features from RGB-D Images for Object Detection and Segmentation [8]

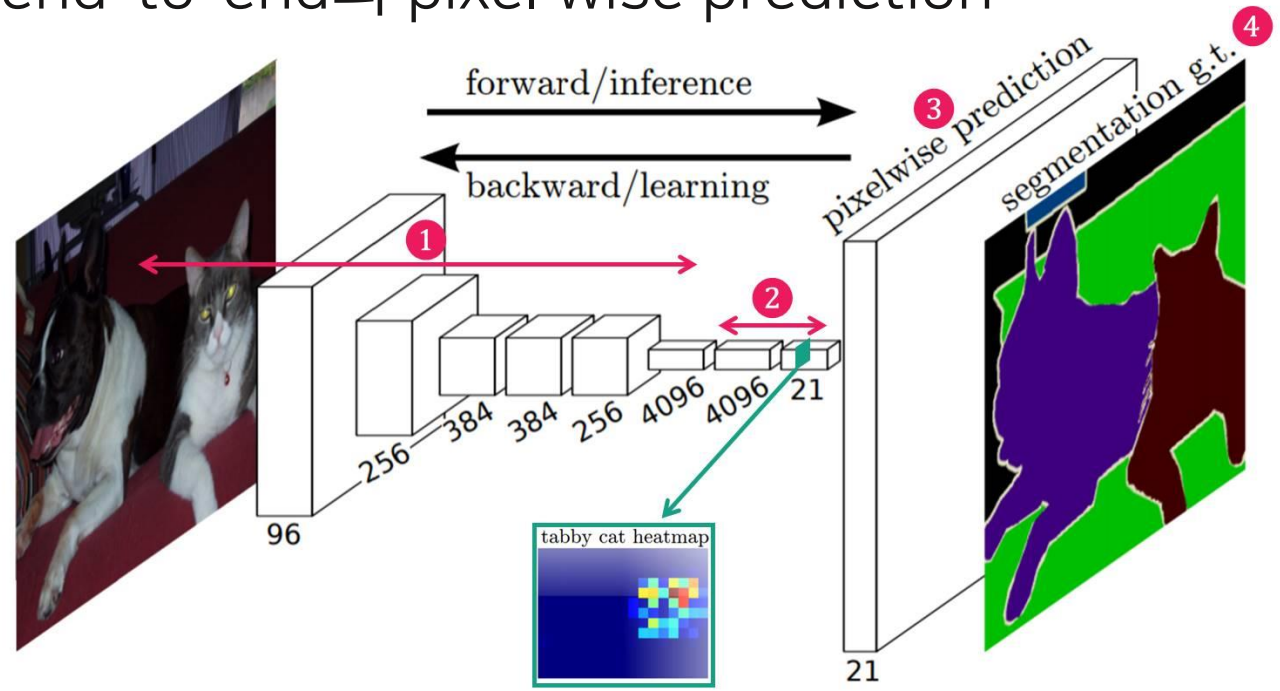


출처 : J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015

1 Introduction

- (2) Fully Convolutional Network (FCN)

end-to-end의 pixel wise prediction



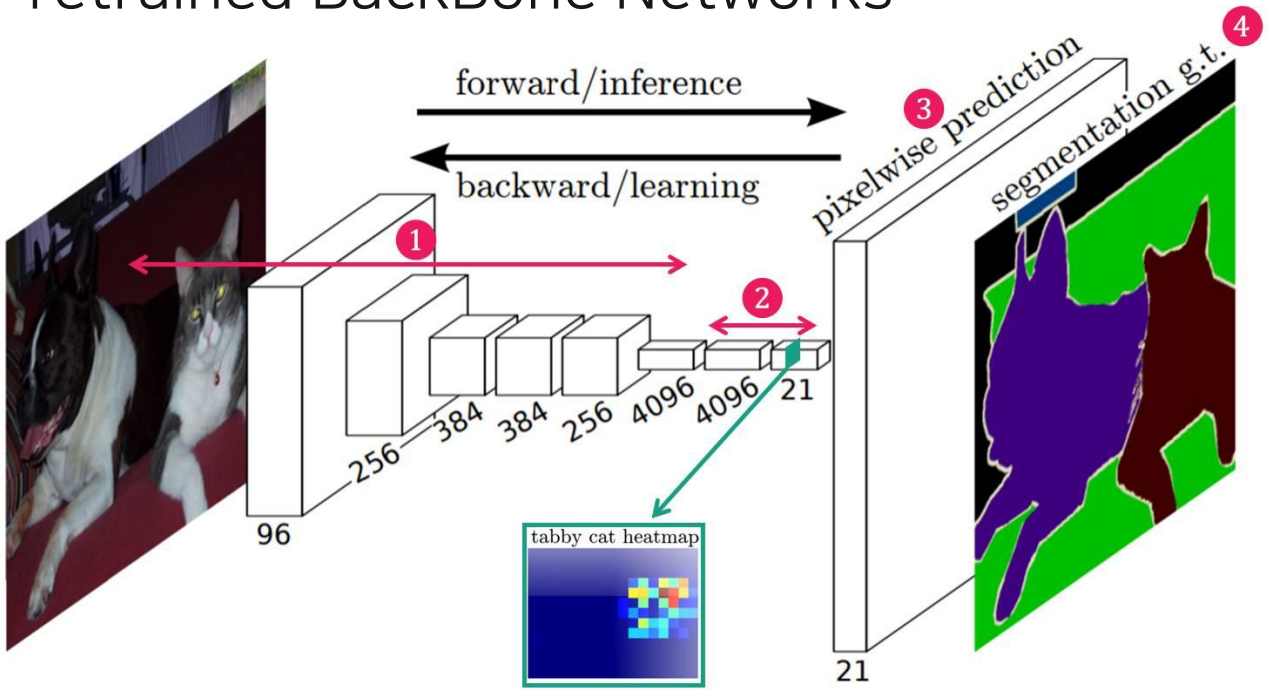
1. End-to-end 방식의 Semantic Segmentation 방법을 제안 (pixel-to-pixel)
   - End-to-end : 입력부터 출력까지의 모든 과정을 한번에 수행하는 방법 (①②③④)
   - forward / inference <-> backward/learning 과정을 통해 한번에 학습과 추론 가능

출처 : J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015

1 Introduction

- (2) Fully Convolutional Network (FCN)

Pretrained BackBone Networks



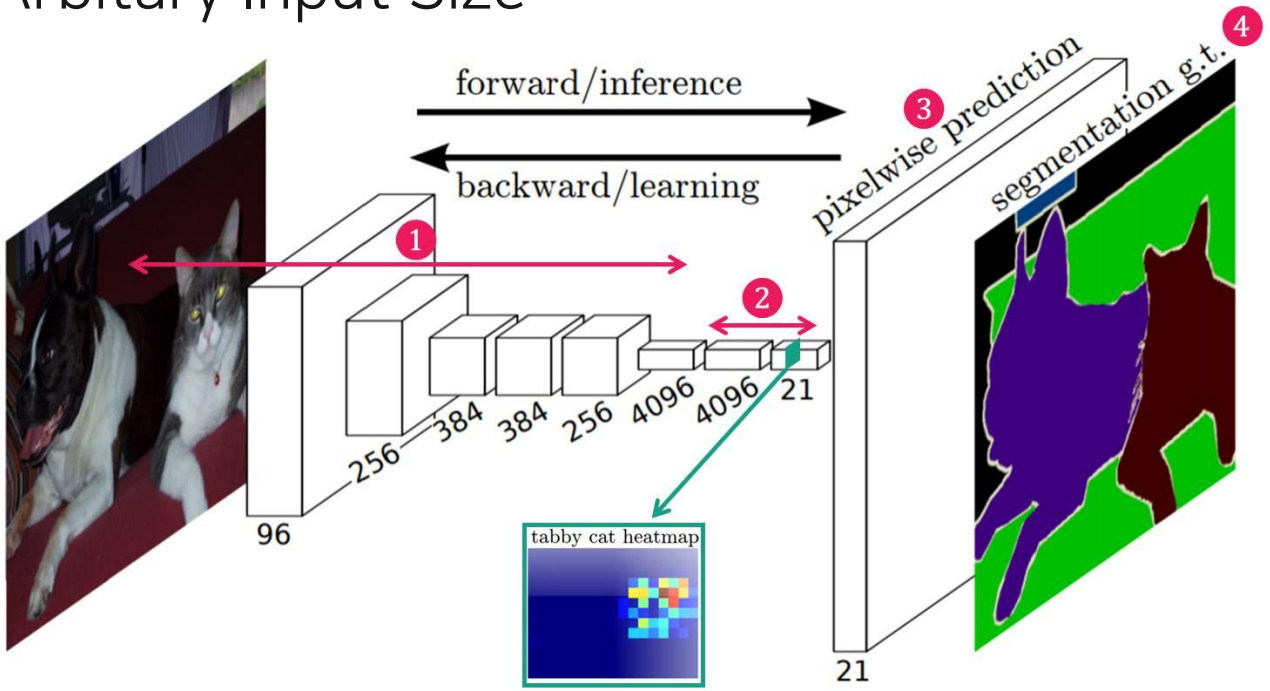
2. Pretraining Model for Supervised
   - VGG, GoogleNet 등과 같이 백본으로 사용하는 모델들을 미리 학습된 웨이트를 사용 가능(①②)
   - Pretrained + FineTuning이 가능함

출처 : J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015

1 Introduction

- (2) Fully Convolutional Network (FCN)

Arbitary Input Size



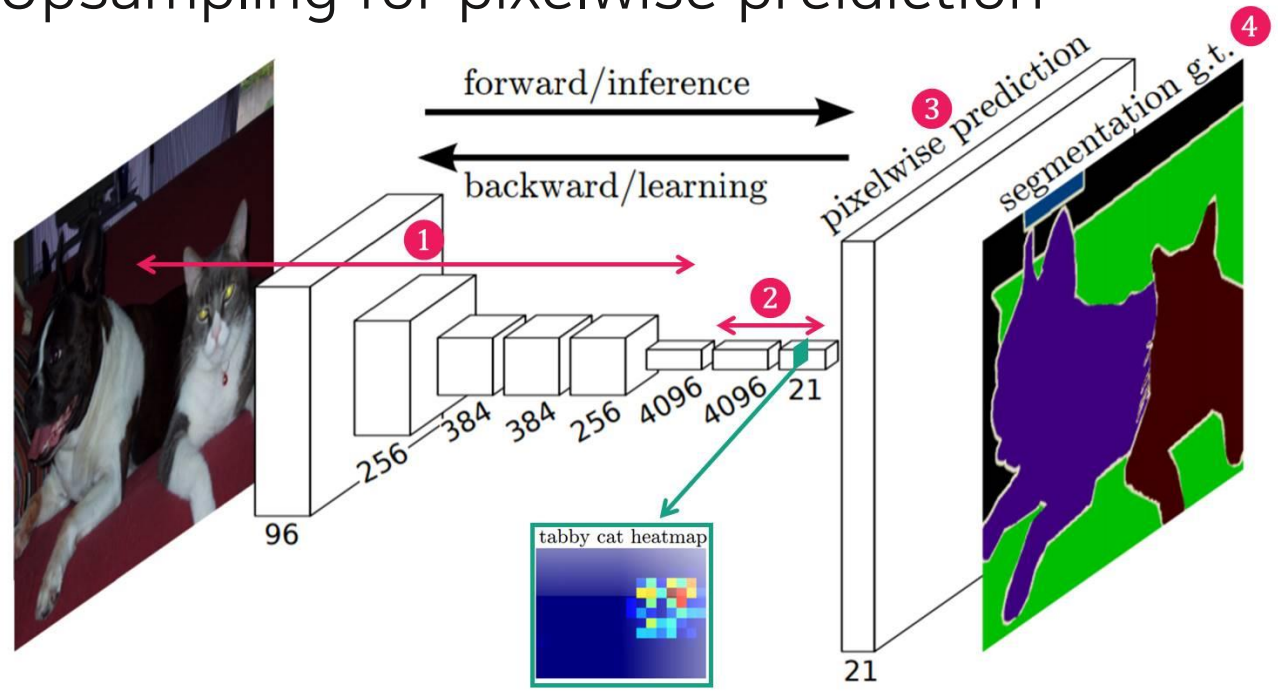
3. 임의의 입력 사이즈에 대해서도 네트워크가 출력을 내보냄 (1x1 Convolution)
   - 어떤 입력 사이즈의 이미지를 넣더라도 동일한 입력 크기를 가지는 출력 결과를 내보냄 (①)
   - 예1) input image size (512, 512) -> output segmentation image size (512, 512)
   - 예2) input image size (256, 256) -> output segmentation image size (256, 256)

출처 : J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015

1 Introduction

- (2) Fully Convolutional Network (FCN)

Upsampling for pixelwise preidiction



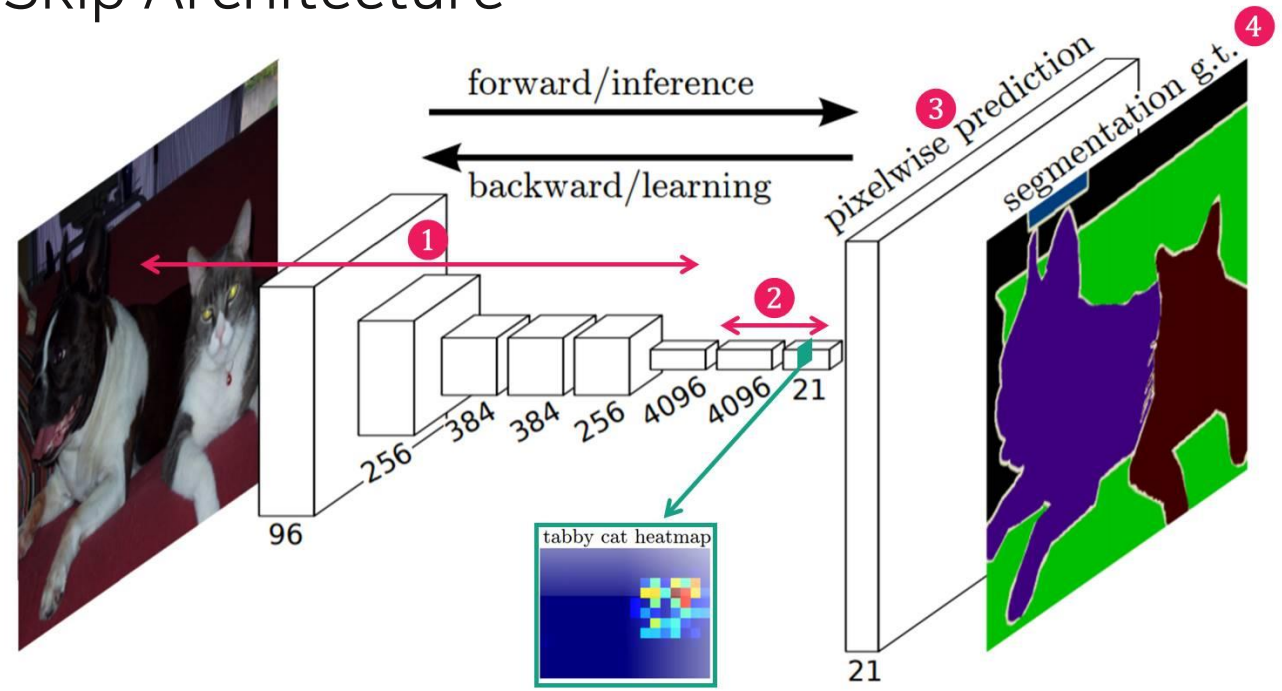
4. Upsampling을 이용해서 Pixel Wise Classfication을 가능하게 함 (②->③)
   - Upsampling (Transposed Convolution)

출처 : J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015

1 Introduction

- (2) Fully Convolutional Network (FCN)

Skip Architecture



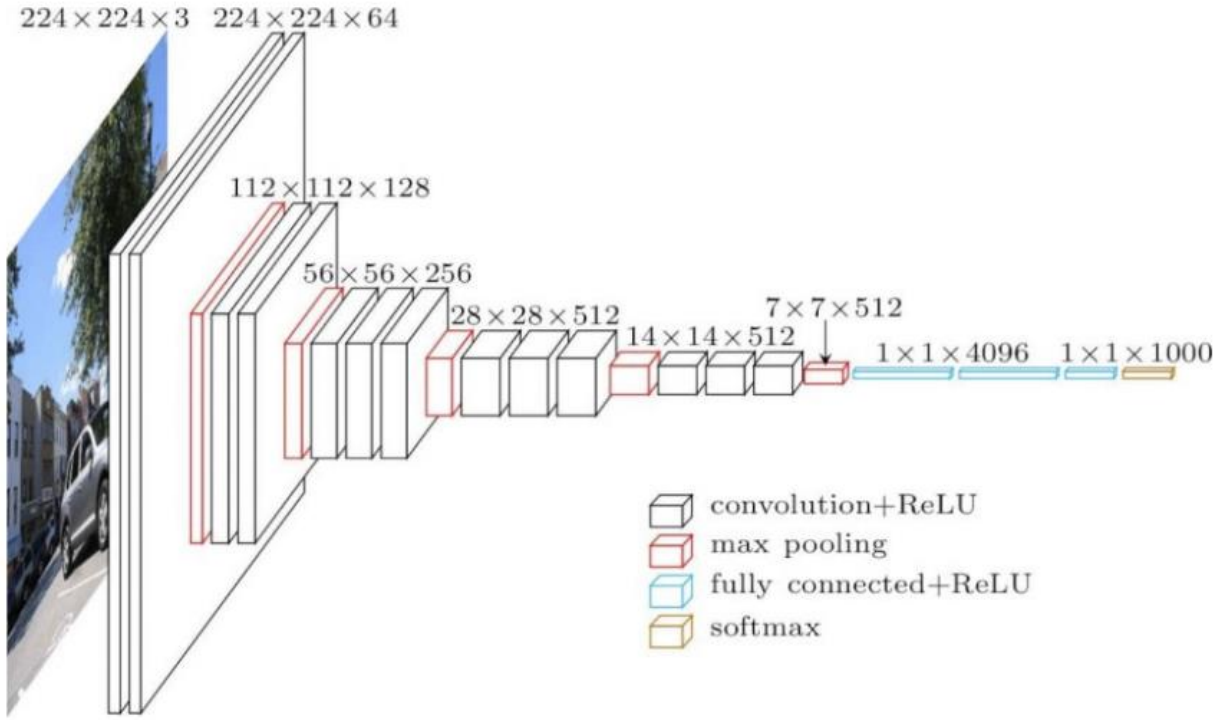
5. 낮은 층의 정보와 높은 층의 정보를 skip architecture를 통해서 결합해줌
   - 직선 및 곡선, 색상 등의 낮은 수준의 특징 (local feature) - ①
   - 복잡하고 포괄적인 개체 정보가 활성화 (global feature) - ②

출처 : J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015

2. Related Work

(1) VGG

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

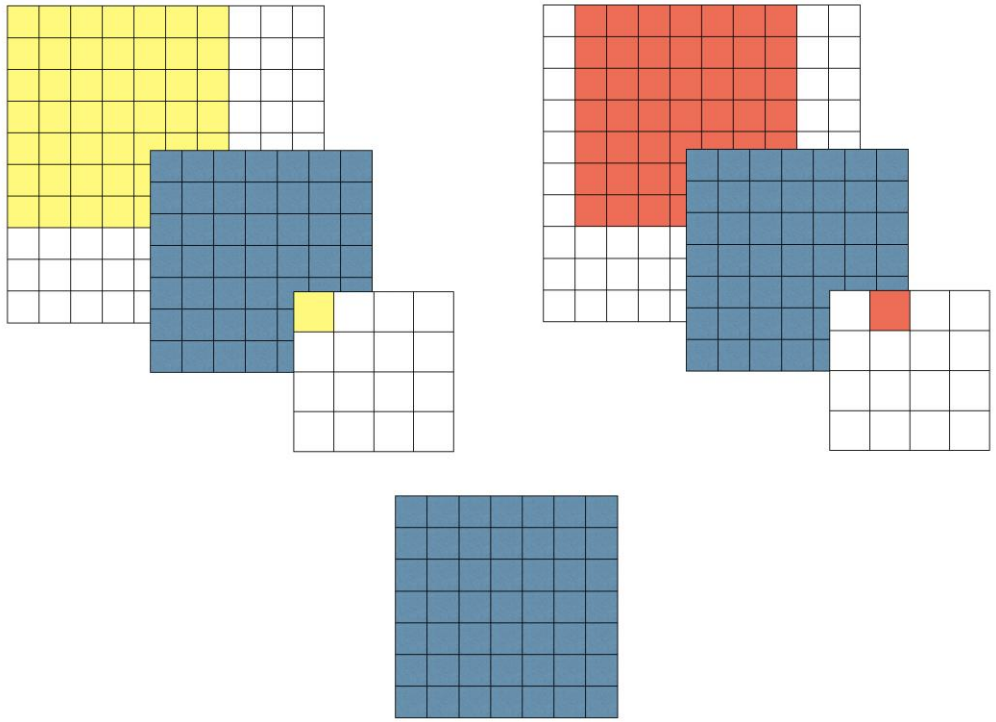


VGG16 Architecture

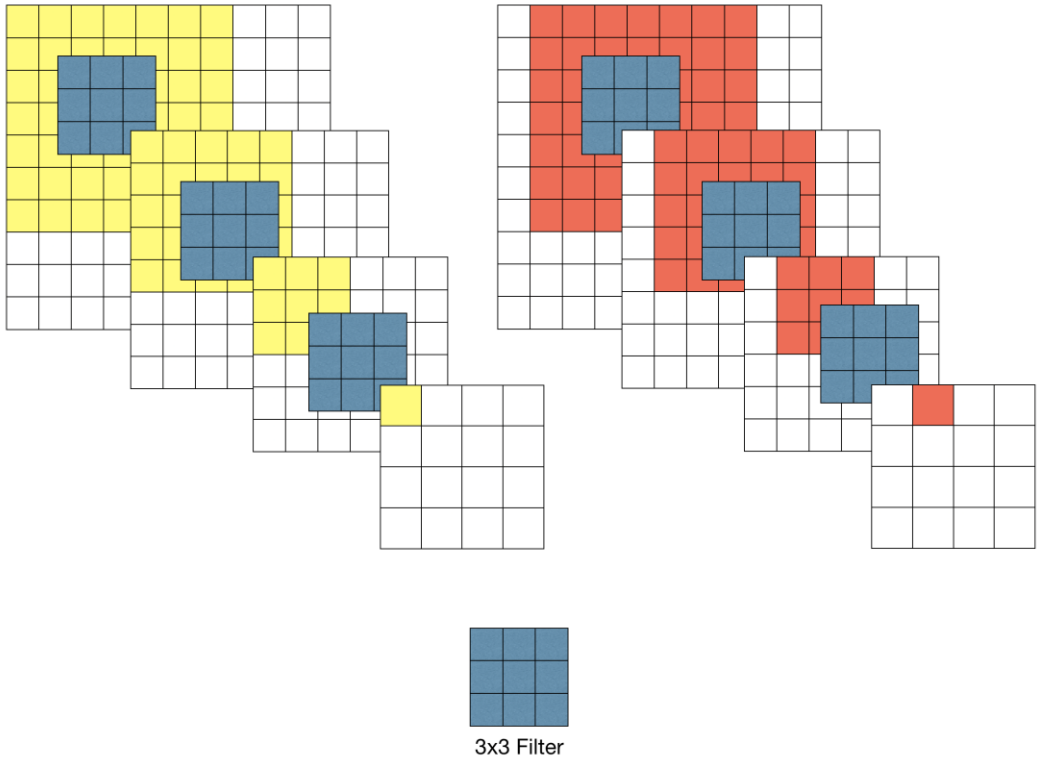
이미지 출처 : https://neurohive.io/en/popular-networks/vgg16/

2. Related Work

(1) VGG



7x7 Filter



3x3 Filter

이미지 출처 : https://medium.com/@msmapark2/vgg16-%EB%85%BC%EB%AC%B8-%EB%A6%AC%EB%B7%B0-very-deep-convolutional-networks-for-large-scale-image-recognition-6f748235242a

2. Related Work

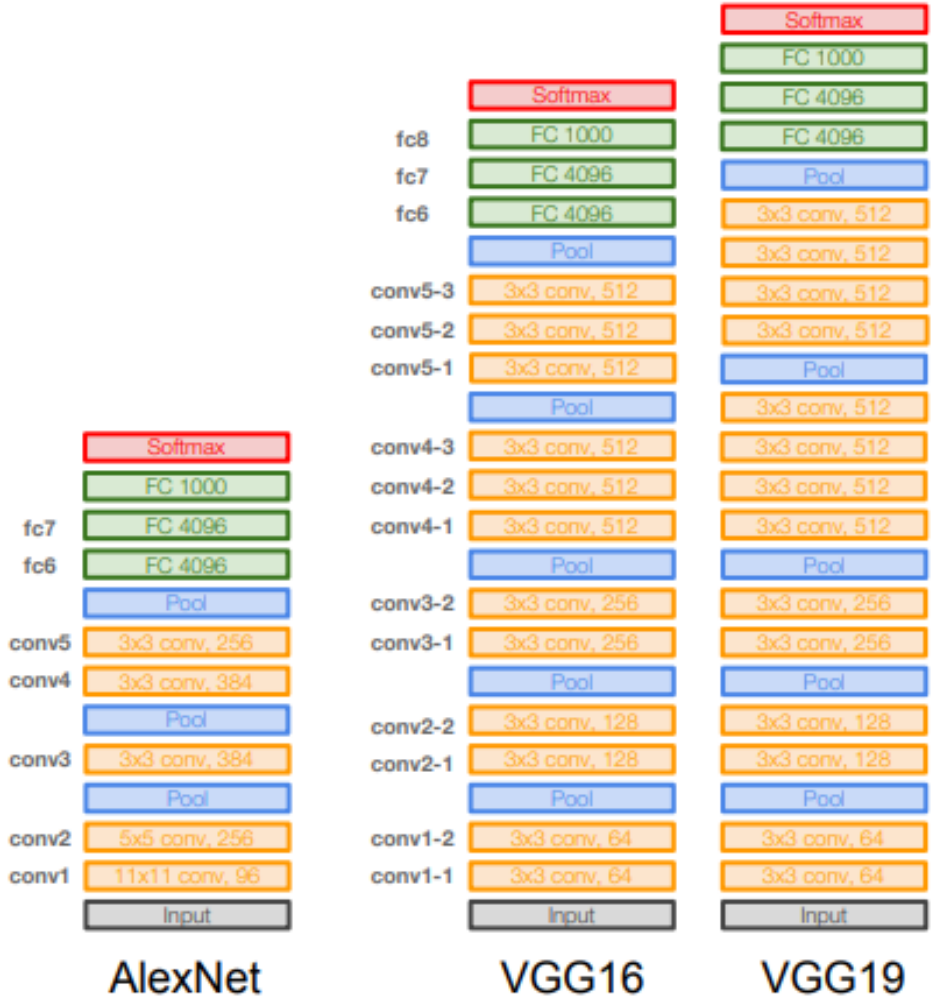
(1) VGG

Case Study: VGGNet

[Simonyan and Zisserman, 2014]

Details:

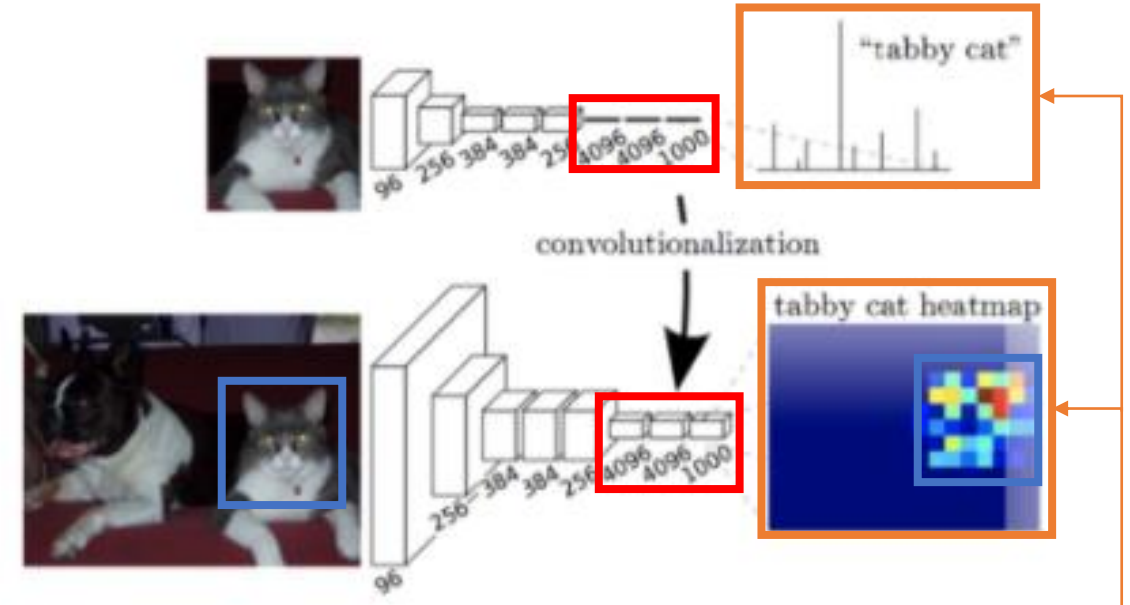
- ILSVRC'14 2nd in classification, 1st in localization
- Similar training procedure as Krizhevsky 2012
- No Local Response Normalisation (LRN)
- Use VGG16 or VGG19 (VGG19 only slightly better, more memory)
- Use ensembles for best results
- FC7 features generalize well to other tasks



출처 : CS231n

2. Related Work
(2) Fully Convolutional Networks

정의 : Fully Connected Layer를 1x1 Convolution으로 변경



Fully Connected vs Fully Convolution

- Fully Connected Layer는 단순하게 개 vs 고양이의 정보만 제공
- Fully Convolutional Layer는 위치 정보를 같이 제공

1. 이미지의 위치정보를 기억하기 위함
2. 임의의 입력 크기에 대해서도 일관성있는 결과를 생성하려는 의도

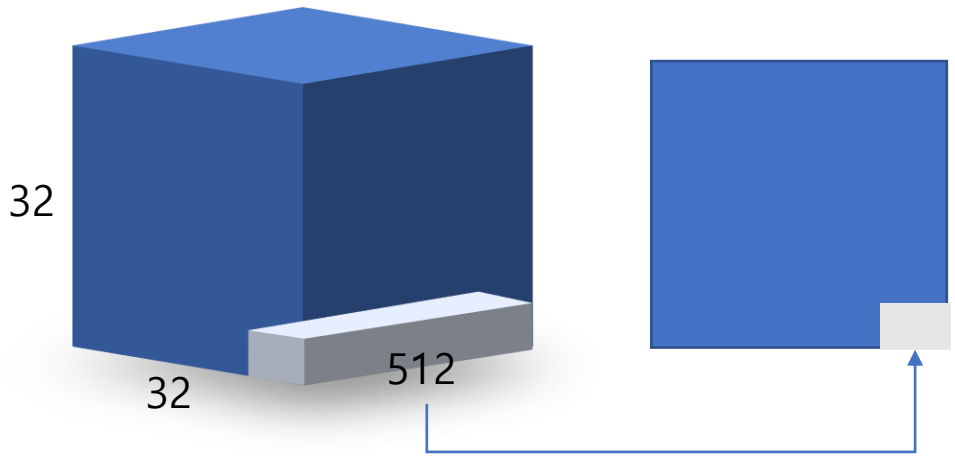
(Fully connected layer는 입력의 크기가 동일해야 하는데, Convolution 는 상관없음)

출처 : J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015

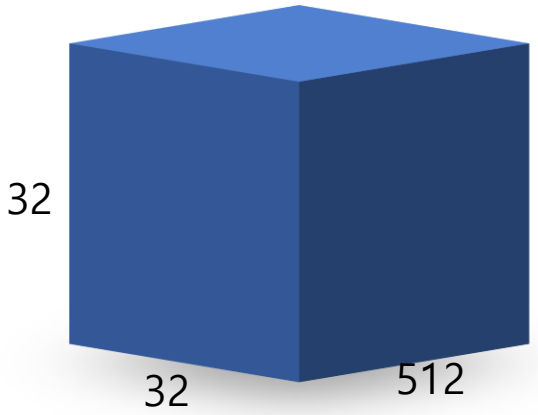
2. Related Work

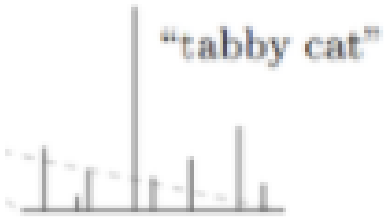
(2) Fully Convolutional Networks

각 픽셀의 위치정보를 해치지 않은채로 특징 추출



각 픽셀의 위치정보를 해침





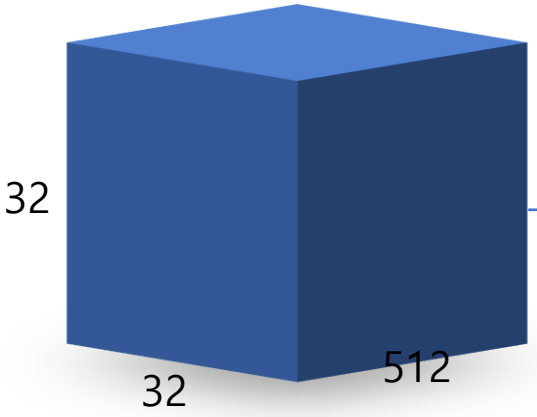
32x32x512x(output 개수)

Output 중 Cat에 대한 필터

출처 : J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015

2. Related Work
(2) Fully Convolutional Networks

1x1 Convolution을 사용할 경우, 임의의 입력값에 대해서도 상관 없는 이유
-> Convolution은 kernel의 파라미터에 의해 영향을 받고, 이미지 혹은 레이어의 크기에 대해서는 상관 없음



입력

nn.Conv2d(input_channel, output_channel, 1, 1)

-> Height, width와 상관이 없음

nn.Linear(input_channel * height * width, output_size)

-> Height, width와 상관이 있음

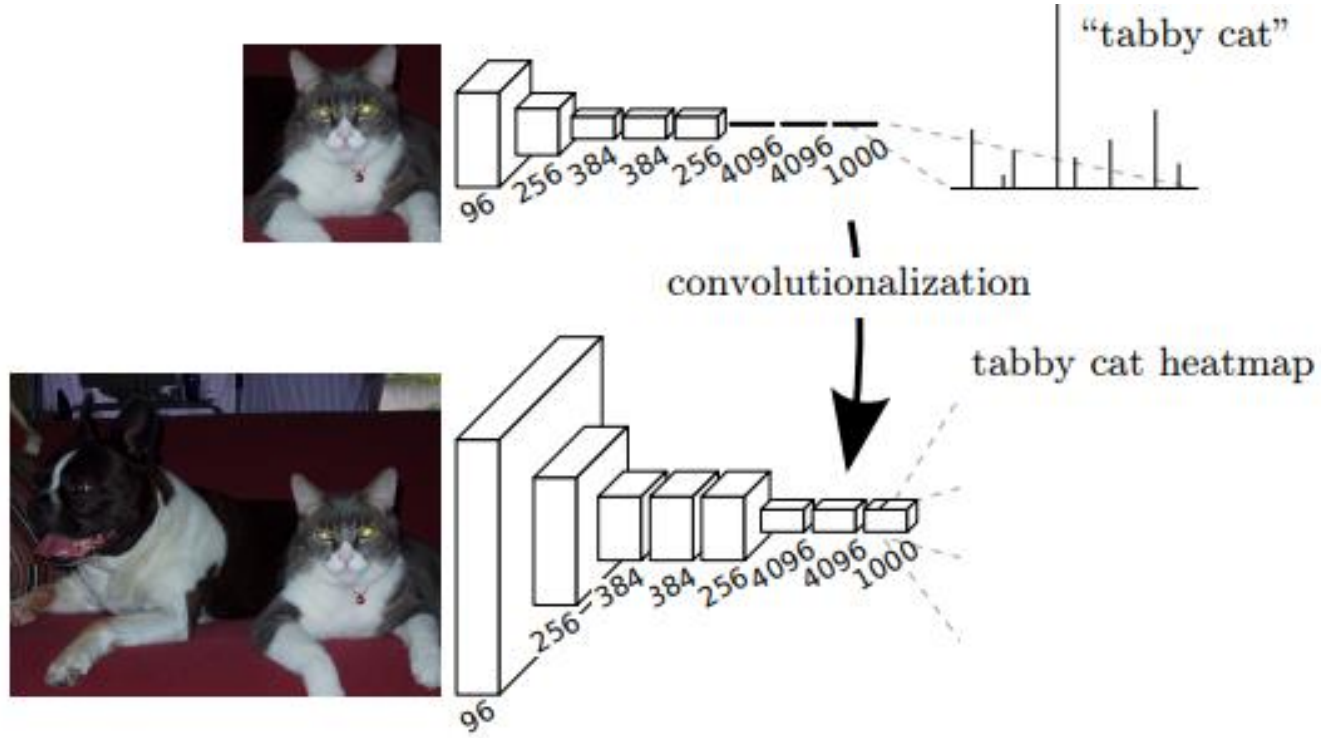
출처 : J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015

3. Fully convolution networks

(1) Adapting classifiers for dense prediction

Fully Connected Layer to Convolution Layer

1. Fixed sized inputs -> Arbitary Image Size
2. spatial coordinates를 기억
3. Computational Efficiency



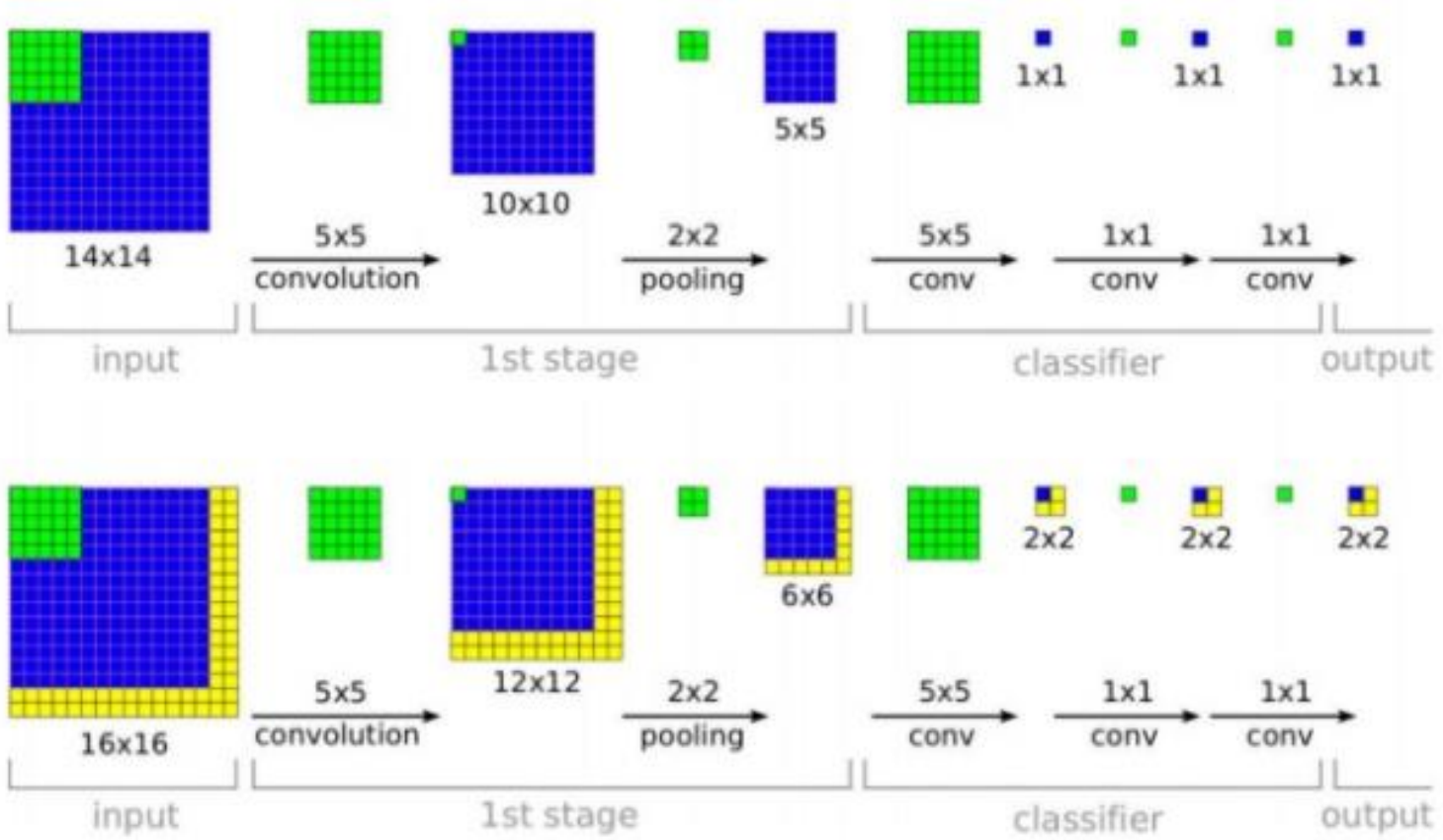
출처 : J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015

3. Fully convolution networks

(2) Shift-and-stitch is filter rarefaction

- Convolution layer의 filters 와 stride를 수정

1. Max-Pooling에 의해서 사라진 정보의 손실을 방지하기 위한 방법



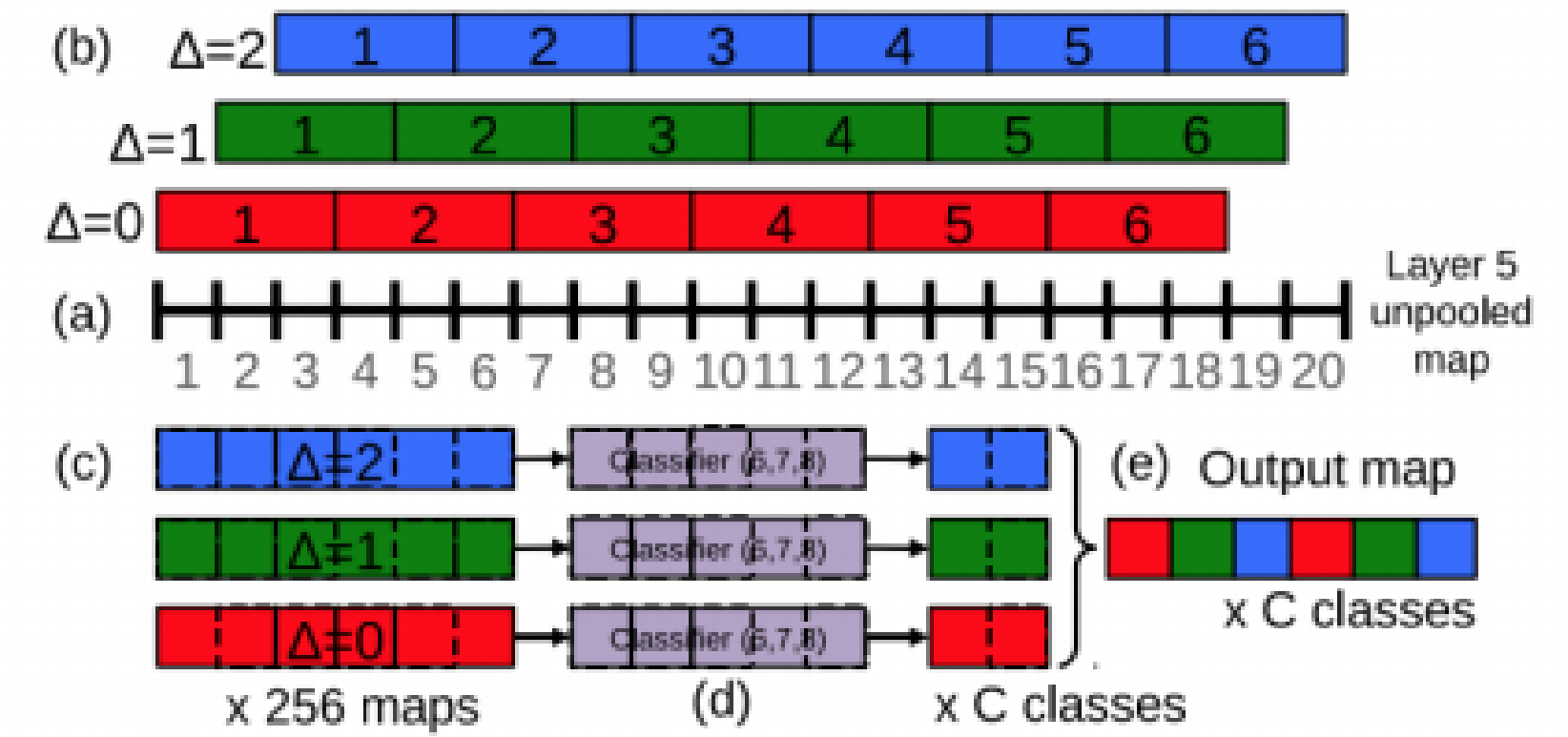
출처 : J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015

3. Fully convolution networks

(2) Shift-and-stitch is filter rarefaction

Convolution layer의 filters 와 stride를 수정

1. Max-Pooling에 의해서 사라진 정보의 손실을 방지하기 위한 방법
2. OverFeat[9] 에서는 Input shifting and output interlacing 을 사용



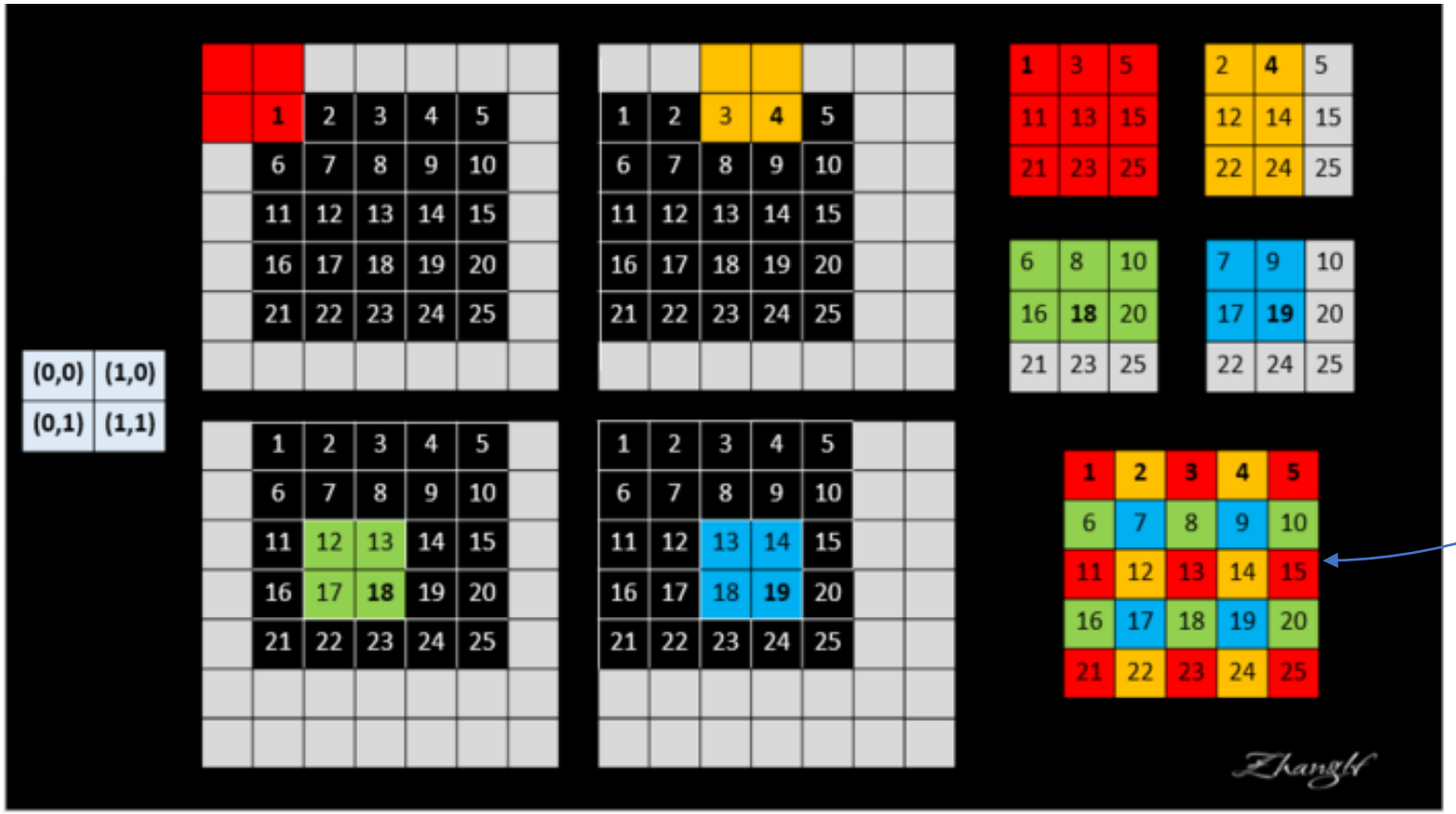
출처 : J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015

3. Fully convolution networks

(2) Shift-and-stitch is filter rarefaction

Convolution layer의 filters 와 stride를 수정

1. Max-Pooling에 의해서 사라진 정보의 손실을 방지하기 위한 방법
2. OverFeat[9] 에서는 Input shifting and output interlacing 을 사용



4개의 좌표를 시작으로 하는 2x2 Maxpooling (Stride 2)를 적용하고 결합

출처 : Jonathan Long, Evan Shelhamer and Trevor Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015

3. Fully convolution networks

(2) Shift-and-stitch is filter rarefaction

Convolution layer의 filters 와 stride를 수정

1. Max-Pooling에 의해서 사라진 정보의 손실을 방지하기 위한 방법
2. OverFeat[9] 에서는 Input shifting and output interlacing 을 사용
3. 하지만, 사실 이게 Filters와 strides를 바꾸는 것과 같은 결과를 가짐
   - 이는 이후에 적용은 하지 않고, Skip Architecture를 적용하는게 더 좋았음

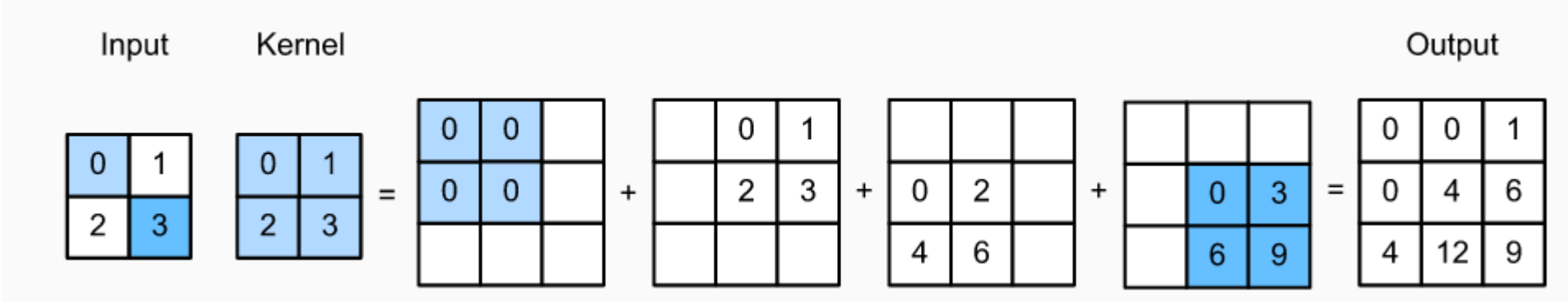
출처 : J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015

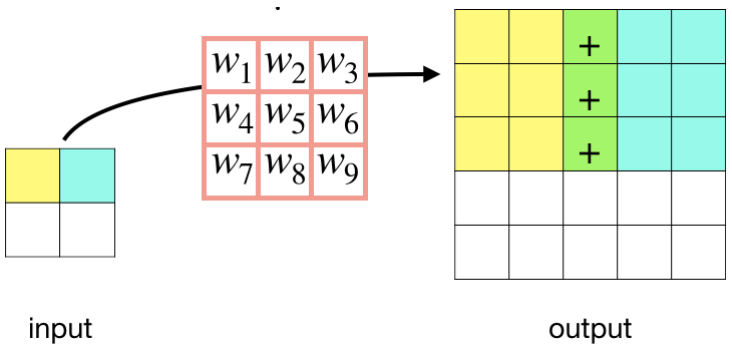
3. Fully convolution networks

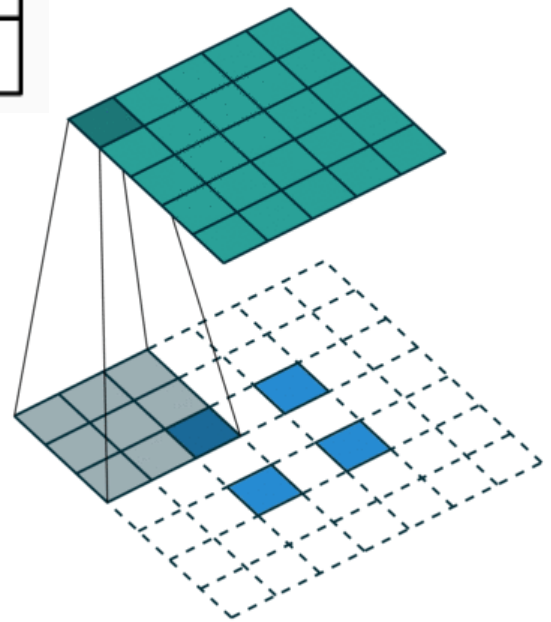
(3) Upsampling is backwards strided convolution

- Transposed Convolution을 이용해서 coarse -> dense output으로 변경

  의미 : Max Pooling에 의해서 감소한 이미지의 크기를 원본 이미지의 크기로 복원







출처 : https://github.com/vdumoulin/conv_arithmetic

3. Fully convolution networks

(3) Upsampling is backwards strided convolution

Transposed Convolution을 이용해서 coarse -> dense output으로 변경

Convolution 된 결과에 Deconvolution을 해도 그대로 나오지 않음. Convolution을 입력값에 대해 미분한 값에 Y를 곱한 값이 출력됨 (의미상으로 Convolution의 Transpose를 계산한 것임)

$$TransposedConvolution(Y, W) = Y \cdot \frac{\partial Conv(X,W)}{\partial X}$$

크기는 같지만, 연산의 결과는 다를 수 있기에 Deconvolution이라는 표현은 수학적으로는 정확하지 않음

출처 : https://github.com/vdumoulin/conv_arithmetic

3. Fully convolution networks

(3) Upsampling is backwards strided convolution

Convolution vs Transposed Convolution

kernel (3x3)

$W_{0,0}$	$W_{0,1}$	$W_{0,2}$
$W_{1,0}$	$W_{1,1}$	$W_{1,2}$
$W_{2,0}$	$W_{2,1}$	$W_{2,2}$

input (4x4)

$X_{0,0}$	$X_{0,1}$	$X_{0,2}$	$X_{0,3}$
$X_{1,0}$	$X_{1,1}$	$X_{1,2}$	$X_{1,3}$
$X_{2,0}$	$X_{2,1}$	$X_{2,2}$	$X_{2,3}$
$X_{3,0}$	$X_{3,1}$	$X_{3,2}$	$X_{3,3}$

output (2x2)

Y_0	Y_1
Y_2	Y_3

$W_{0,0}$	$W_{0,1}$	$W_{0,2}$	0	$W_{1,0}$	$W_{1,1}$	$W_{1,2}$	0	$W_{2,0}$	$W_{2,1}$	$W_{2,2}$	0	0	0	0	0
0	$W_{0,0}$	$W_{0,1}$	$W_{0,2}$	0	$W_{1,0}$	$W_{1,1}$	$W_{1,2}$	0	$W_{2,0}$	$W_{2,1}$	$W_{2,2}$	0	0	0	0
0	0	0	0	$W_{0,0}$	$W_{0,1}$	$W_{0,2}$	0	$W_{1,0}$	$W_{1,1}$	$W_{1,2}$	0	$W_{2,0}$	$W_{2,1}$	$W_{2,2}$	0
0	0	0	0	0	$W_{0,0}$	$W_{0,1}$	$W_{0,2}$	0	$W_{1,0}$	$W_{1,1}$	$W_{1,2}$	0	$W_{2,0}$	$W_{2,1}$	$W_{2,2}$

$X_{0,0}$
$X_{0,1}$
$X_{0,2}$
$X_{0,3}$
$X_{1,0}$
$X_{1,1}$
$X_{1,2}$
$X_{1,3}$
$X_{2,0}$
$X_{2,1}$
$X_{2,2}$
$X_{2,3}$
$X_{3,0}$
$X_{3,1}$
$X_{3,2}$
$X_{3,3}$

Y_0
Y_1
Y_2
Y_3

출처 : https://github.com/vdumoulin/conv_arithmetic

3. Fully convolution networks

(3) Upsampling is backwards strided convolution

Convolution vs Transposed Convolution

input (4x4)

X0, 0	X0, 1	X0, 2	X0, 3
X1, 0	X1, 1	X1, 2	X1, 3
X2, 0	X2, 1	X2, 2	X2, 3
X3, 0	X3, 1	X3, 2	X3, 3

W0, 0	0	0	0
W0, 1	W0, 0	0	0
W0, 2	W0, 1	0	0
0	W0, 2	0	0
W1, 0	0	W0, 0	0
W1, 1	W1, 0	W0, 1	W0, 0
W1, 2	W1, 1	W0, 2	W0, 1
0	W1, 2	0	W0, 2
W2, 0	0	W1, 0	0
W2, 1	W2, 0	W1, 1	W1, 0
W2, 2	W2, 1	W1, 2	W1, 1
0	W2, 2	0	W1, 2
0	0	W2, 0	0
0	0	W2, 1	W2, 0
0	0	W2, 2	W2, 1
0	0	0	W2, 2

×

Y0
Y1
Y2
Y3

=

X_new0, 0
X_new0, 1
X_new0, 2
X_new0, 3
X_new1, 0
X_new1, 1
X_new1, 2
X_new1, 3
X_new2, 0
X_new2, 1
X_new2, 2
X_new2, 3
X_new3, 0
X_new3, 1
X_new3, 2
X_new3, 3

$$C = \frac{\partial X^{(l+1)}}{\partial X^{(l)}} = \frac{\partial Vec(X^{(l+1)})}{\partial Vec(X^{(l)})}$$

$$= \begin{bmatrix} \frac{\partial x_{11}^{(l+1)}}{\partial x_{11}^{(l)}} & \frac{\partial x_{12}^{(l+1)}}{\partial x_{11}^{(l)}} & \frac{\partial x_{13}^{(l+1)}}{\partial x_{11}^{(l)}} & \frac{\partial x_{14}^{(l+1)}}{\partial x_{11}^{(l)}} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial x_{11}^{(l+1)}}{\partial x_{44}^{(l)}} & \frac{\partial x_{12}^{(l+1)}}{\partial x_{44}^{(l)}} & \frac{\partial x_{13}^{(l+1)}}{\partial x_{44}^{(l)}} & \frac{\partial x_{14}^{(l+1)}}{\partial x_{44}^{(l)}} \end{bmatrix}$$

$$= \begin{bmatrix} w_{11} & 0 & 0 & 0 \\ w_{12} & w_{11} & 0 & 0 \\ w_{13} & w_{12} & 0 & 0 \\ 0 & w_{13} & 0 & 0 \\ w_{21} & 0 & w_{11} & 0 \\ w_{22} & w_{21} & w_{12} & w_{11} \\ w_{23} & w_{22} & w_{13} & w_{12} \\ 0 & w_{23} & 0 & w_{13} \\ w_{31} & 0 & w_{21} & 0 \\ w_{32} & w_{31} & w_{22} & w_{21} \\ w_{33} & w_{32} & w_{23} & w_{22} \\ 0 & w_{33} & 0 & w_{23} \\ 0 & 0 & w_{31} & 0 \\ 0 & 0 & w_{32} & w_{31} \\ 0 & 0 & w_{33} & w_{32} \\ 0 & 0 & 0 & w_{33} \end{bmatrix}$$

출력값을 입력값으로 미분한 값 (출력값 : Convolution한 결과)

출처 : https://www.slideshare.net/ssuserb208cc1/transposed-convolution

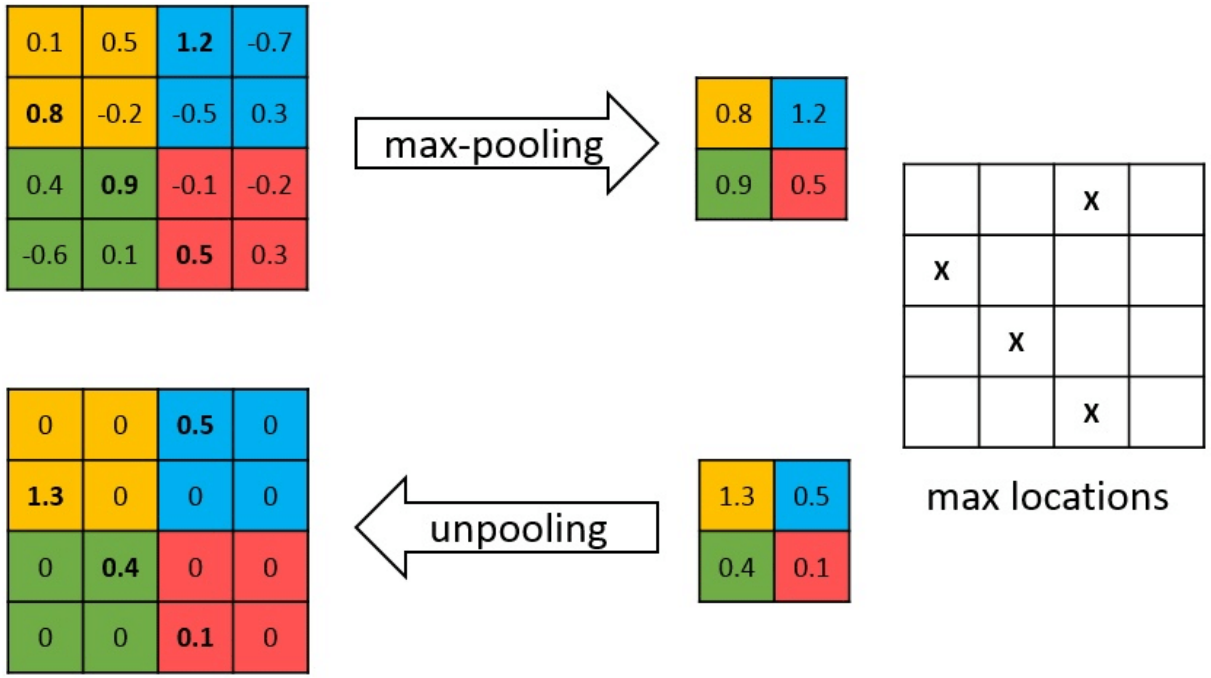
3. Fully convolution networks

(3) Upsampling is backwards strided convolution

그 외의 coarse -> dense output으로 변경하는 방법

1. Pooling을 사용하지 않거나, Pooling의 Stride를 줄임으로서 애초에 Feature map을 늘리는 경우
   - 파라미터가 너무 많아지는 문제가 있음
   - Receptive Field가 작아짐

2. Unpooling을 이용한 방법 (Max Unpooling, Bilinear interpolation)

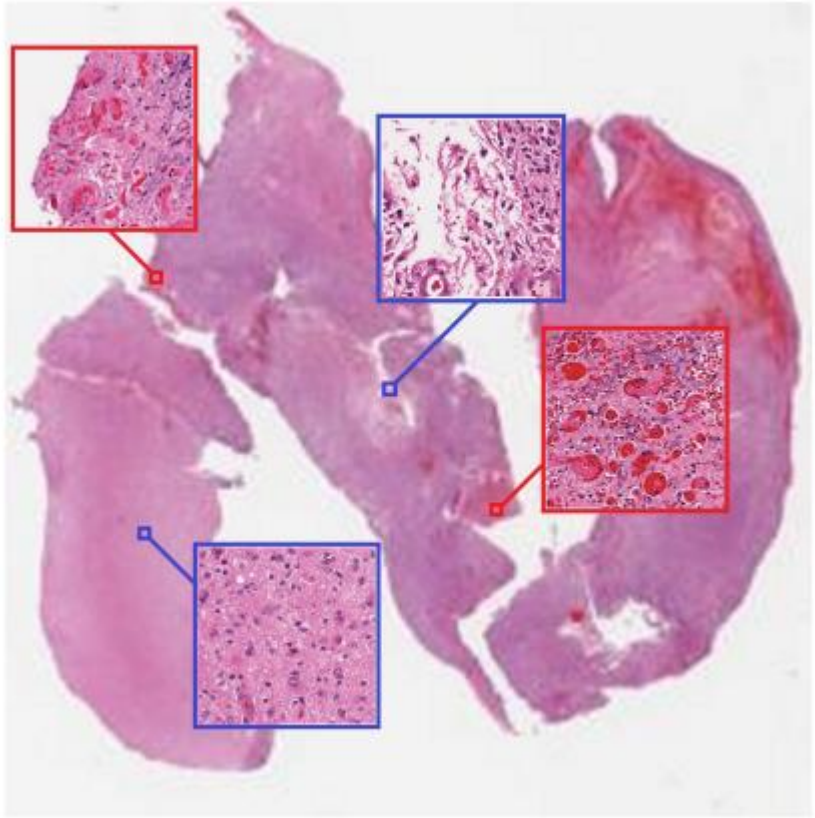


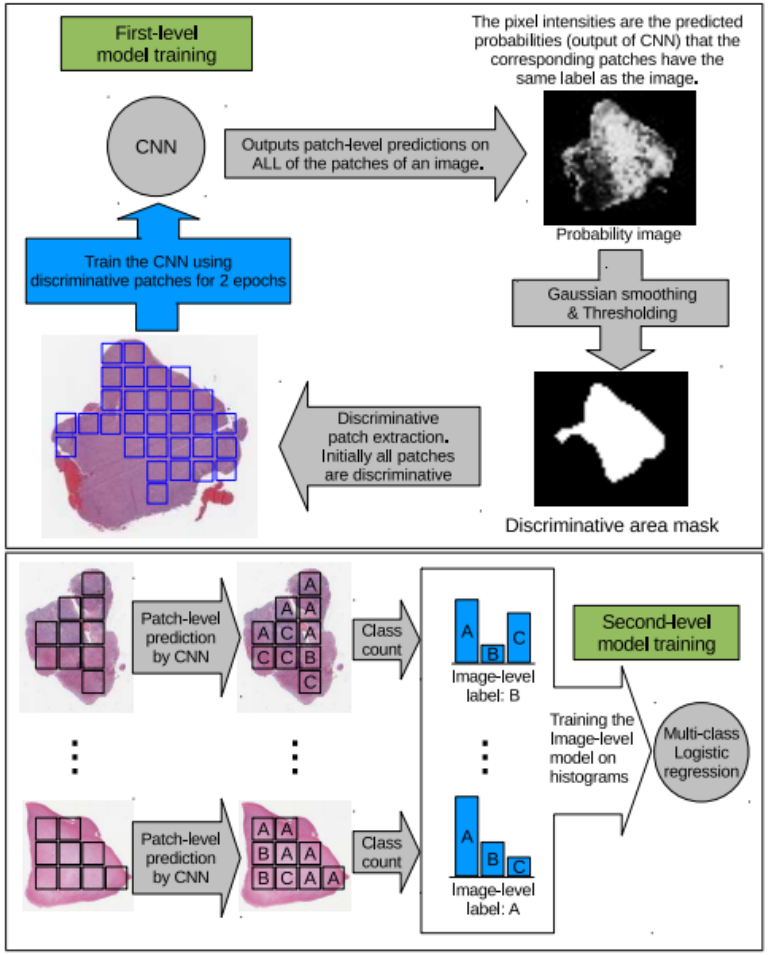
출처 : DeepPainter: Painter Classification Using Deep Convolutional Autoencoders

3. Fully convolution networks

(4) Patchwise training is loss sampling

What is the Patch-wise Training?





출처 : Patch-based Convolutional Neural Network for Whole Slide Tissue Image Classification

3. Fully convolution networks

(4) Patchwise training is loss sampling

Patchwise training vs fully convolutional training

[patchwise training]

[fully convolutional training]

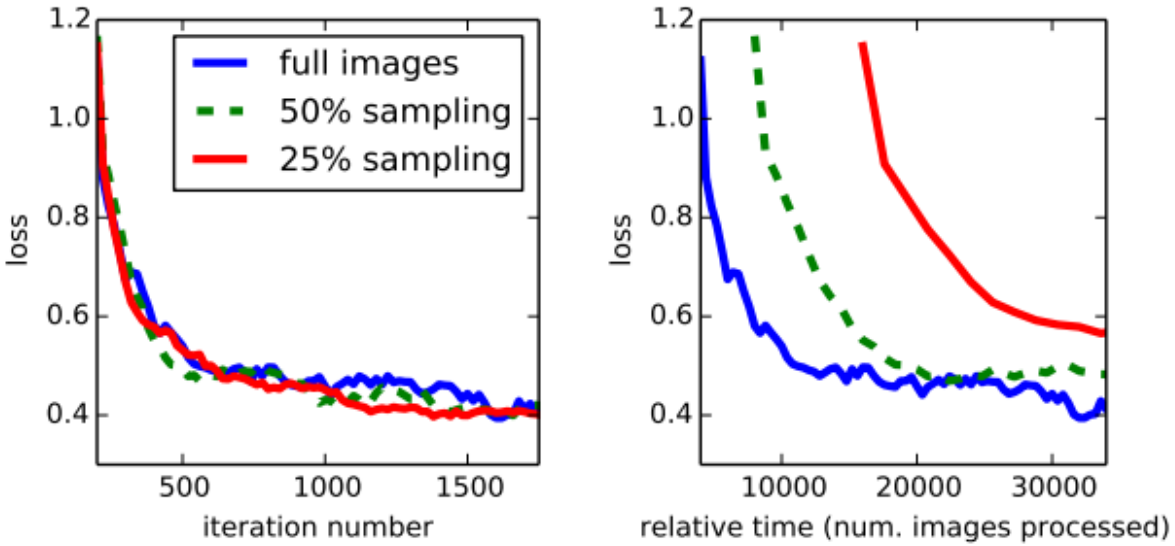
1. patchwise training을 사용시에 효과적인 학습이 가능
   - 배경과 같이 사진의 대부분을 차지하는 영역(redundancy)을 학습에서 제외할 수 있음
   - 샘플링을 통해 클래스간의 균형을 맞출 수 있음
   - Fully convolutional의 경우 pixel이 sptial correlation을 가지는데 이를 해결할 수 있음
   - 학습데이터와 검증데이터간의 분포를 비슷하게 맞출 수 있음
     - 이를 통해 수렴하는 속도도 빨라지는 장점이 있음

출처 : https://stackoverflow.com/questions/42636685/patch-wise-training-and-fully-convolutional-training-in-fcn

3. Fully convolution networks

(4) Patchwise training is loss sampling

Patchwise training vs fully convolutional training



2. 실제 실험결과 Patchwise training의 효과는 거의 없었음
   - patch-wise 와 fully convolutional 방법은 iteration에 대한 loss는 비슷했음
   - 하지만 relative time에서 fully convolutional이 효과가 좋았음
   - class imbalance 같은 경우는 weight를 주거나 loss를 sampling 해서 해결할 수도 있음

출처 : https://stackoverflow.com/questions/42636685/patch-wise-training-and-fully-convolutional-training-in-fcn

4. Segmentation Architecture

(0) Overview

Fully Connected Layer

1. ILSVRC 분류기 (Alexnet, VGG16, GoogleNet)에 FCN을 결합해서 Semantic Segmentation을 수행
   - VGG19의 경우 16과 큰 차이가 없었음
   - GoogleNet의 경우 average pooling layer를 버리고 final loss layer만 사용
2. Pretrained된 모델을 가져와서 FineTuning을 수행
3. Skip architecture를 도입해서 깊은 정보와 얕은 정보를 결합함
4. 이를, PASCAL VOC 2011 segmentation challenge에 실험
5. Loss : pixel multinomial logistic loss / Metric : mean pixel intersection over union (Mean IU)
   - 참고로 모호한 mask들은 무시하고 진행했음

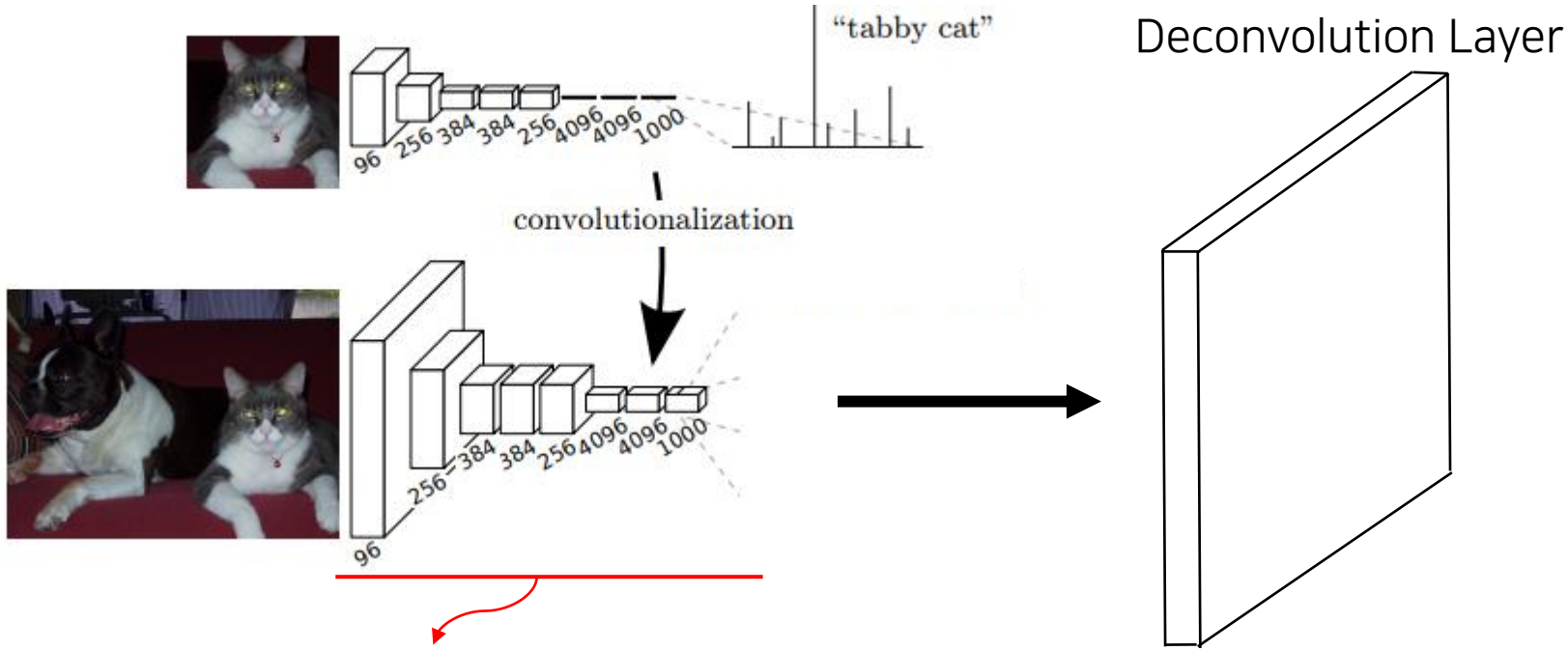
출처 : J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015

4. Segmentation Architecture

(1) From classifier to dense FCN

How to apply ILSVRC Classifier to FCN?

1. ILSVRC 분류기 (Alexnet, VGG16, GoogleNet)에 FCN을 결합해서 Semantic Segmentation을 수행
   - Final Classifier Layer를 버리고 Fully Connected Layer를 1x1 Convolution으로 대체
   - Upsampling을 위한 Deconvolution layer를 추가 (Bilinearly upsampling)



출처 : J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015

4. Segmentation Architecture

(1) From classifier to dense FCN

How to apply ILSVRC Classifier to FCN?

2. Fine Tuning
   - Classification -> Segmentation

	FCN-AlexNet	FCN-VGG16	FCN-GoogLeNet[4]
mean IU	39.8	**56.0**	42.5
forward time	50 ms	210 ms	59 ms
conv. layers	8	16	22
parameters	57M	134M	6M
rf size	355	404	907
max stride	32	32	32

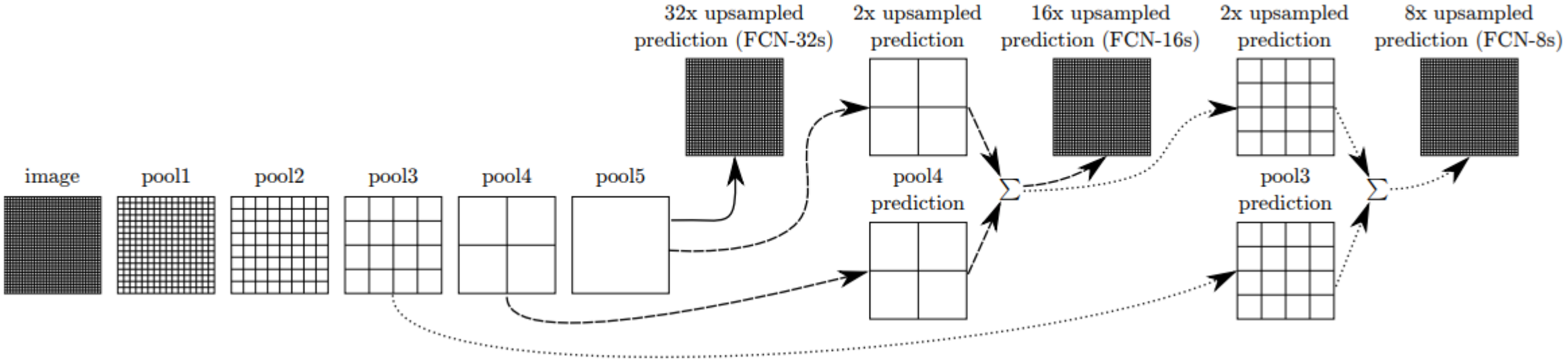
출처 : J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015

4. Segmentation Architecture

(2) Combining what and where

How to apply ILSVRC Classifier to FCN?

3. Segmentation의 성능을 향상시키기 위한 Skip Architecture 사용
   - pooling의 결과와 upsampled의 결과를 결합시킴
   - 얼마나 얕은 정보까지 사용했냐에 따라서 FCN-8s부터 FCN-32s까지 존재
     (8은 2개, 16은 1개, 32는 0개 사용)



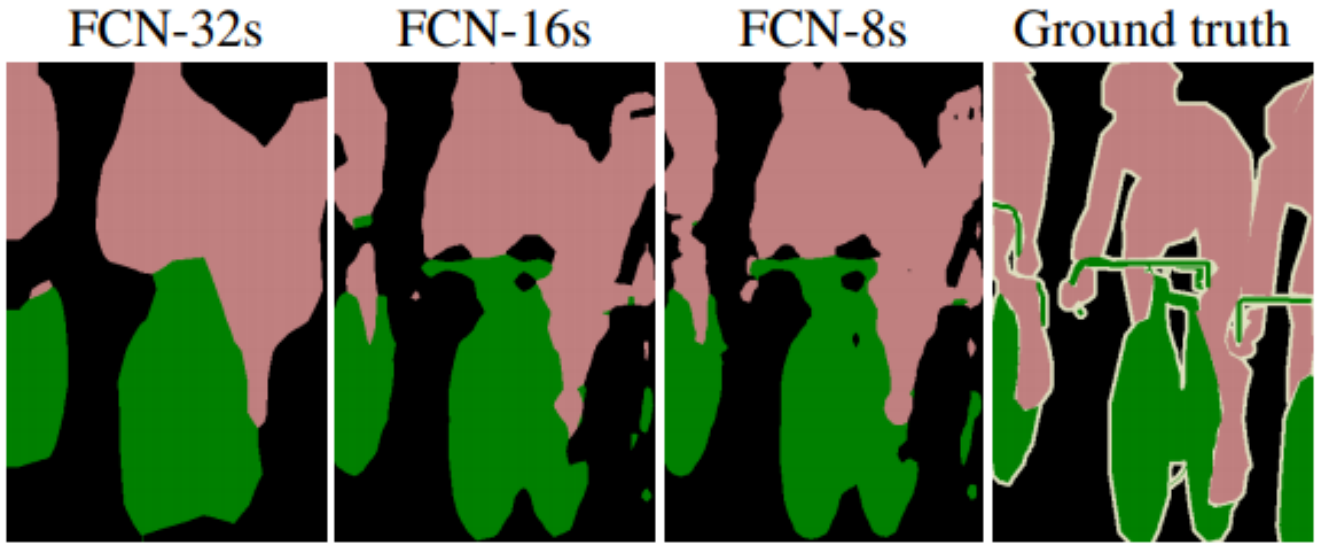
출처 : J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015

4. Segmentation Architecture

(2) Combining what and where

How to apply ILSVRC Classifier to FCN?

3. Segmentation의 성능을 향상시키기 위한 Skip Architecture 사용
   - pooling의 결과와 upsampled의 결과를 결합시킴
   - 얼마나 얕은 정보까지 사용했냐에 따라서 FCN-8s부터 FCN-32s까지 존재
   - 최대한 얕은 정보까지 결합할 수록 성능이 계속 좋아짐



	pixel acc.	mean acc.	mean IU	f.w. IU
FCN-32s-fixed	83.0	59.7	45.4	72.0
FCN-32s	89.1	73.3	59.4	81.4
FCN-16s	90.0	75.7	62.4	83.0
FCN-8s	**90.3**	**75.9**	**62.7**	**83.2**

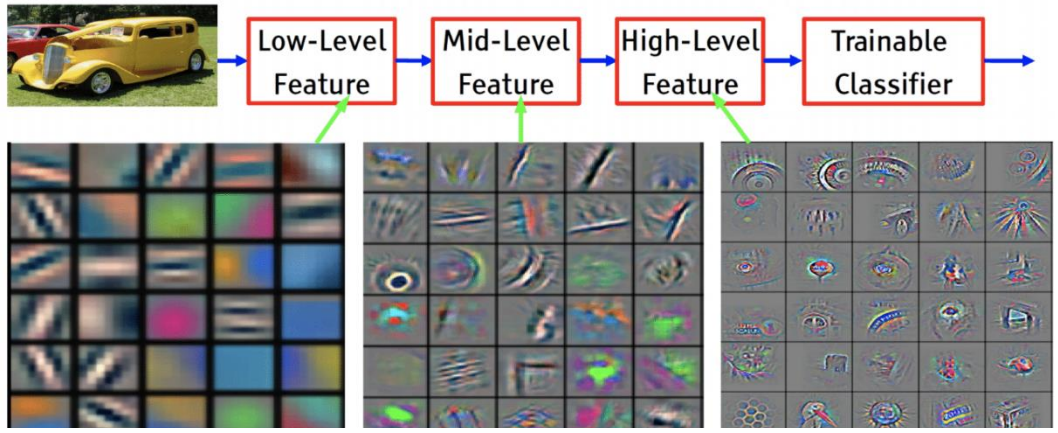
출처 : J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015

4. Segmentation Architecture

(2) Combining what and where

How to apply ILSVRC Classifier to FCN?

얕은 층의 정보는 Local Feature를 깊은 층은 Global Feature를 가지고 있어서 결합하는게 의미가 있음

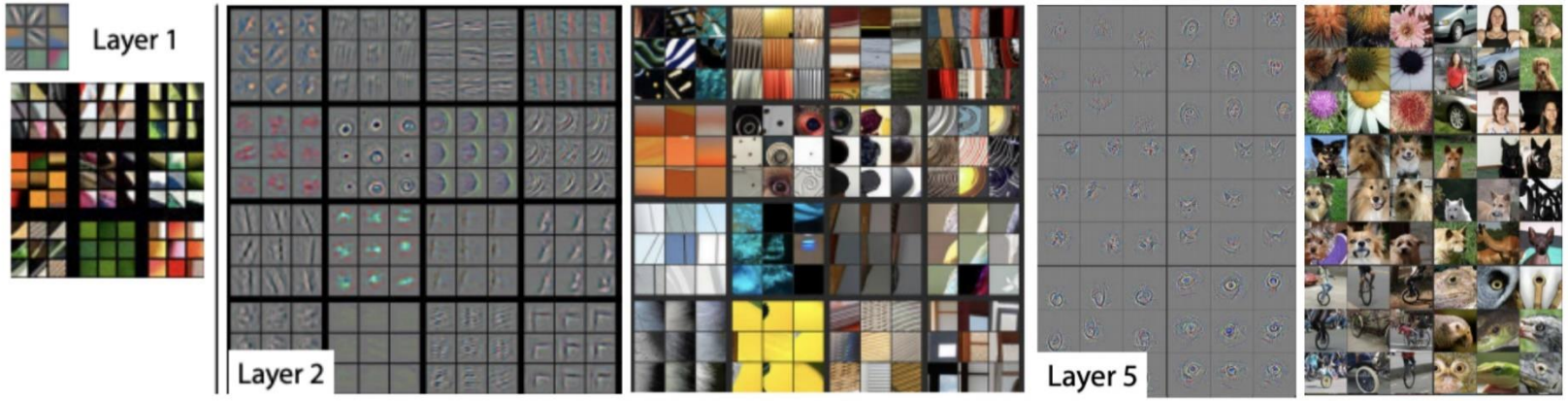


Low Layer

- 직선 및 곡선, 색상 등의 낮은 수준의 특징 (local feature)

High Layer

- 복잡하고 포괄적인 개체 정보가 활성화 (global feature)

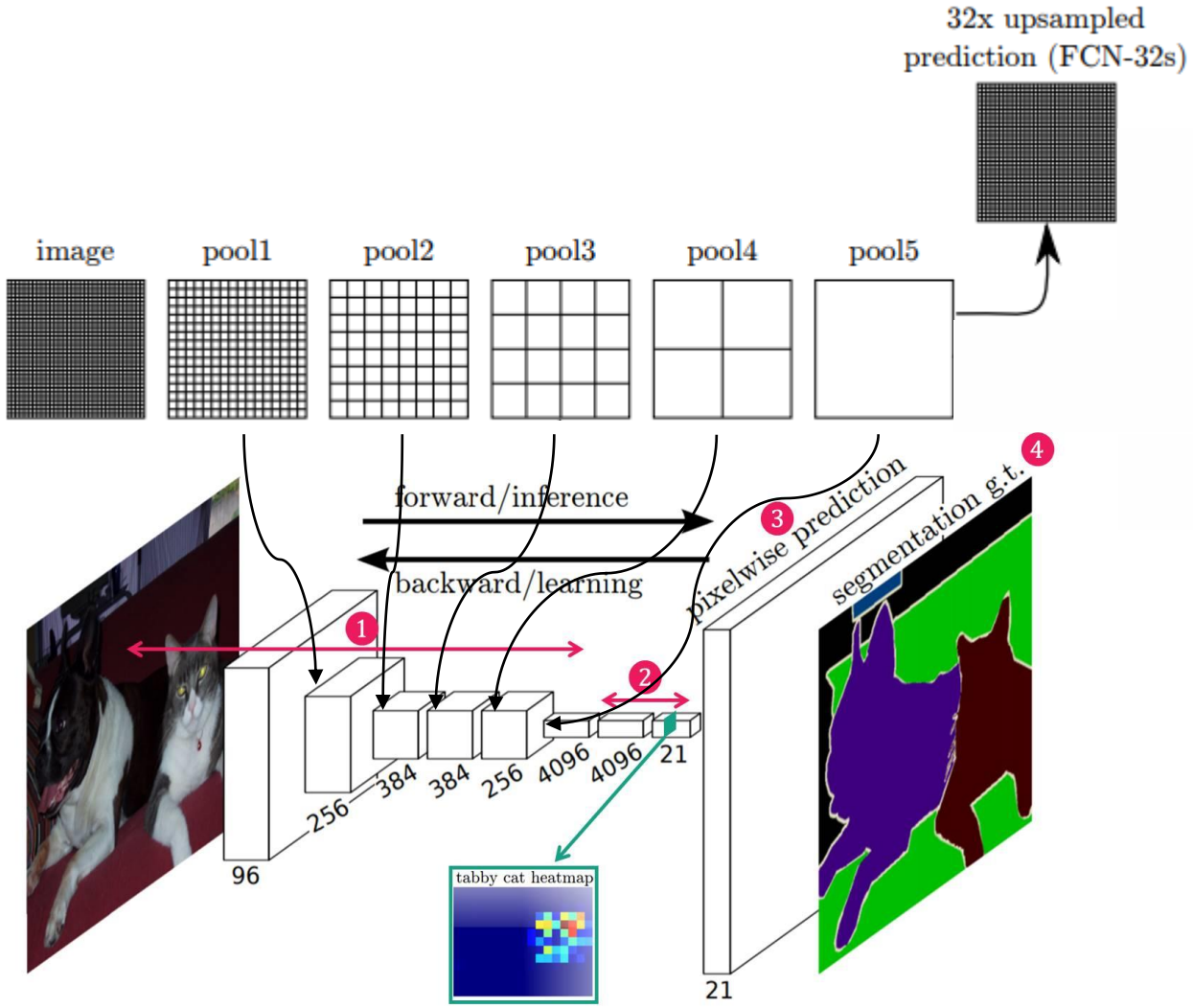


출처 : J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015

4. Segmentation Architecture

(2) Combining what and where

- How to apply ILSVRC Classifier to FCN?



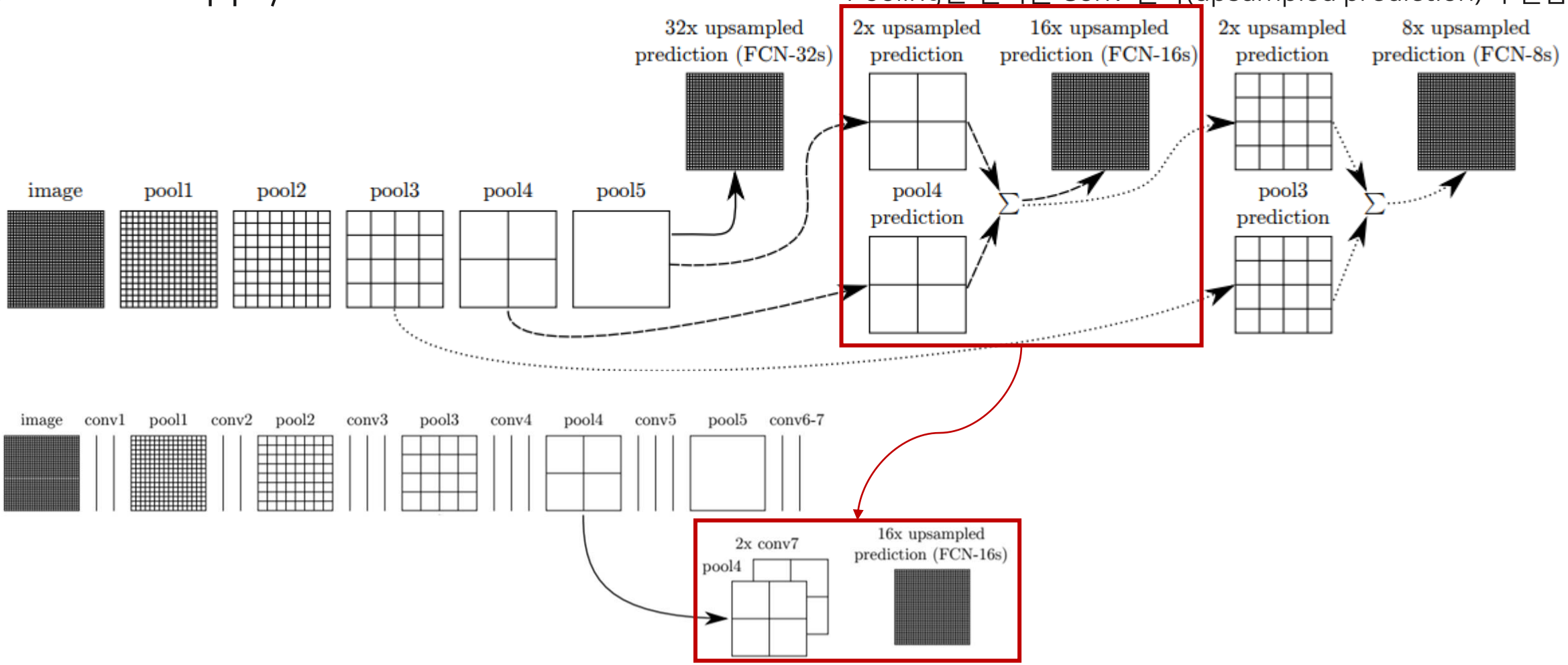
출처 : J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015

4. Segmentation Architecture

(2) Combining what and where

- How to apply ILSVRC Classifier to FCN?

Pooling한 결과를 Conv 결과(upsampled prediction)와 결합



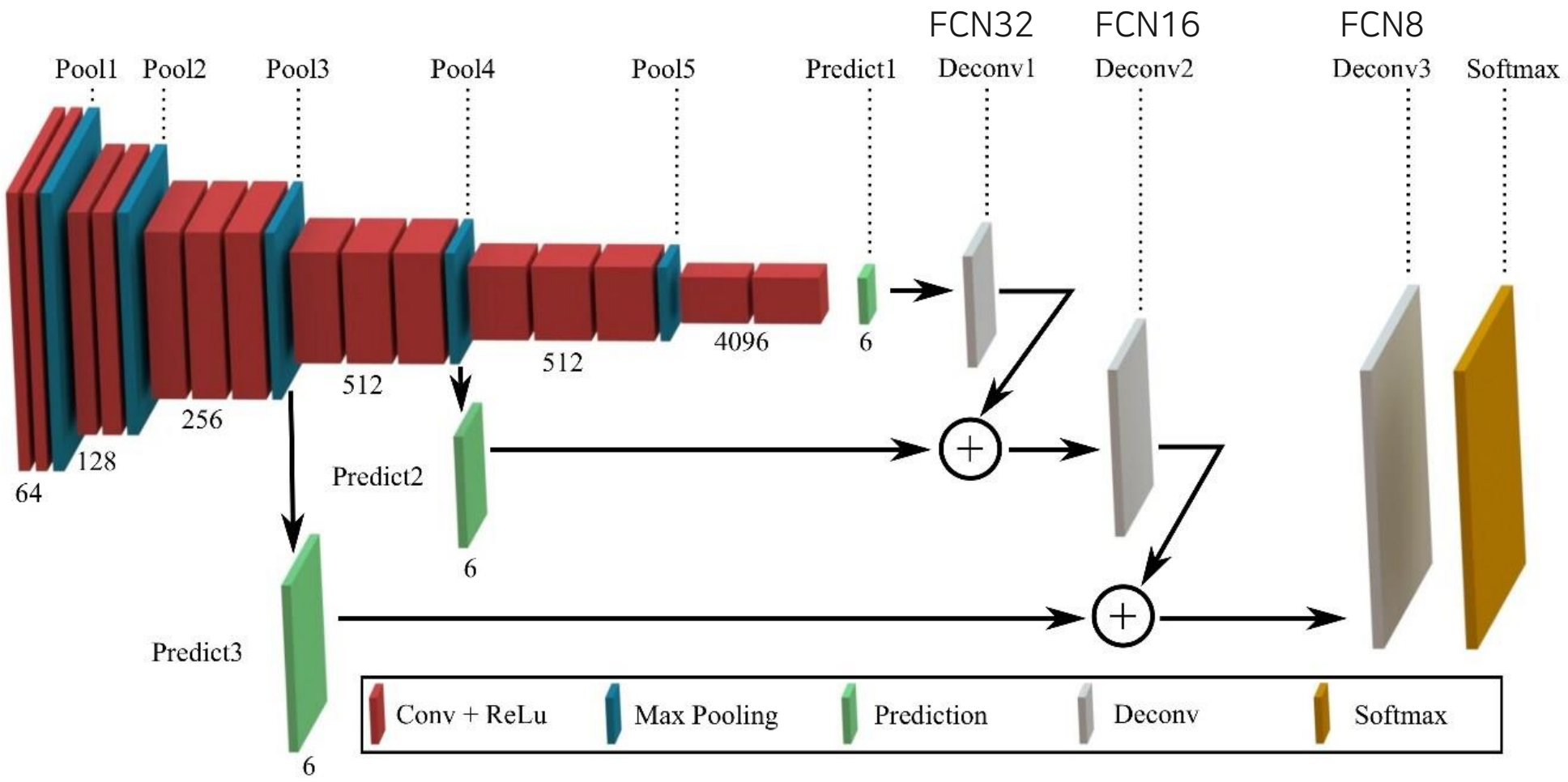
출처 : J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015

4. Segmentation Architecture

(2) Combining what and where

- How to apply ILSVRC Classifier to FCN?

Pooling한 결과를 Conv 결과(upsampled prediction)와 결합



출처 : Investigations on the inference optimization techniques and their impact on multiple hardware platforms for Semantic Segmentation

4. Segmentation Architecture

(2) Combining what and where

How to apply ILSVRC Classifier to FCN?

FCN32

```
# conv1
self.conv1_1 = nn.Conv2d(3, 64, 3, padding=100)
self.relu1_1 = nn.ReLU(inplace=True)
self.conv1_2 = nn.Conv2d(64, 64, 3, padding=1)
self.relu1_2 = nn.ReLU(inplace=True)
self.pool1 = nn.MaxPool2d(2, stride=2, ceil_mode=True)  # 1/2

# conv2
self.conv2_1 = nn.Conv2d(64, 128, 3, padding=1)
self.relu2_1 = nn.ReLU(inplace=True)
self.conv2_2 = nn.Conv2d(128, 128, 3, padding=1)
self.relu2_2 = nn.ReLU(inplace=True)
self.pool2 = nn.MaxPool2d(2, stride=2, ceil_mode=True)  # 1/4

# conv3
self.conv3_1 = nn.Conv2d(128, 256, 3, padding=1)
self.relu3_1 = nn.ReLU(inplace=True)
self.conv3_2 = nn.Conv2d(256, 256, 3, padding=1)
self.relu3_2 = nn.ReLU(inplace=True)
self.conv3_3 = nn.Conv2d(256, 256, 3, padding=1)
self.relu3_3 = nn.ReLU(inplace=True)
self.pool3 = nn.MaxPool2d(2, stride=2, ceil_mode=True)  # 1/8

# conv4
self.conv4_1 = nn.Conv2d(256, 512, 3, padding=1)
self.relu4_1 = nn.ReLU(inplace=True)
self.conv4_2 = nn.Conv2d(512, 512, 3, padding=1)
self.relu4_2 = nn.ReLU(inplace=True)
self.conv4_3 = nn.Conv2d(512, 512, 3, padding=1)
self.relu4_3 = nn.ReLU(inplace=True)
self.pool4 = nn.MaxPool2d(2, stride=2, ceil_mode=True)  # 1/16
```

```
# conv5
self.conv5_1 = nn.Conv2d(512, 512, 3, padding=1)
self.relu5_1 = nn.ReLU(inplace=True)
self.conv5_2 = nn.Conv2d(512, 512, 3, padding=1)
self.relu5_2 = nn.ReLU(inplace=True)
self.conv5_3 = nn.Conv2d(512, 512, 3, padding=1)
self.relu5_3 = nn.ReLU(inplace=True)
self.pool5 = nn.MaxPool2d(2, stride=2, ceil_mode=True)  # 1/32

# fc6
self.fc6 = nn.Conv2d(512, 4096, 7)
self.relu6 = nn.ReLU(inplace=True)
self.drop6 = nn.Dropout2d()

# fc7
self.fc7 = nn.Conv2d(4096, 4096, 1)
self.relu7 = nn.ReLU(inplace=True)
self.drop7 = nn.Dropout2d()

self.score_fr = nn.Conv2d(4096, n_class, 1)
self.upscore = nn.ConvTranspose2d(n_class, n_class, 64, stride=32,
                                  bias=False)
```

FCN16

```
self.score_fr = nn.Conv2d(4096, n_class, 1)
self.score_pool4 = nn.Conv2d(512, n_class, 1)

self.upscore2 = nn.ConvTranspose2d(
    n_class, n_class, 4, stride=2, bias=False)
self.upscore16 = nn.ConvTranspose2d(
    n_class, n_class, 32, stride=16, bias=False)


h = self.score_fr(h)
h = self.upscore2(h)
upscore2 = h  # 1/16

h = self.score_pool4(pool4)
h = h[:, :, 5:5 + upscore2.size()[2], 5:5 + upscore2.size()[3]]
score_pool4c = h  # 1/16

h = upscore2 + score_pool4c

h = self.upscore16(h)
```

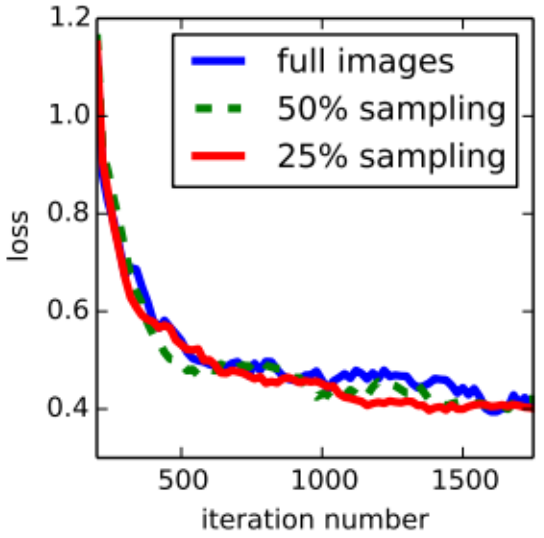
출처 : https://github.com/wkentaro/pytorch-fcn/blob/master/torchfcn/models/fcn16s.py

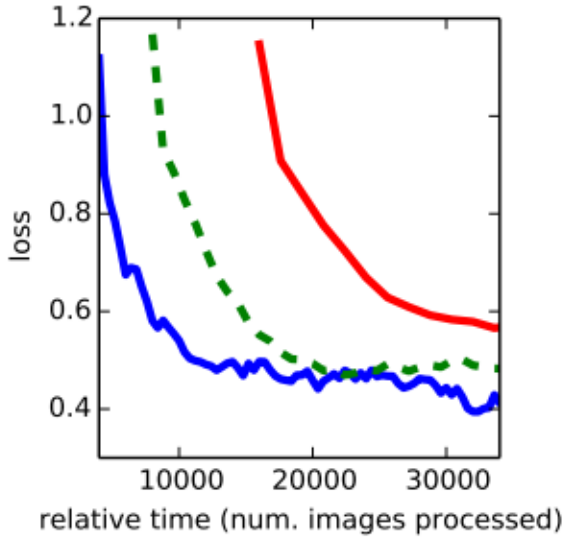
4. Segmentation Architecture

(3) Experiemnt Framework

Optimization, Fine-tuning, Patch Sampling

1. Optimizer
   - SGD with momentum (minibatch size of 20 images)
   - lr : $10^{-3}, 10^{-4}, 5^{-5}$ for FCN-AlexNet, FCN-VGG16, FCN-GoogLeNet
   - momentum : 0.9
   - weight decay : $5^{-4}, 2^{-4}$
   - zero-initialize > random initialize
   - Drop Out
2. Fine-tuning
   - Last Layer vs full fine- tuning
3. Patch Sampling
   - 전체 이미지를 사용하는게 가장 성능이 좋았음





출처 : J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015

4. Segmentation Architecture

(3) Experiemnt Framework

Class Balancing, Dense Prediction, Augmentation

4. Class Balancing
   - weighting / sampling -> 사용 x

5. Dense Prediction
   - Shift and stitch / filter rarefaction equivalent or not -> 사용 x

6. Augmentation
   - “jittering”(잡음/채도) -> 사용효과미비

출처 : J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015

5. Results
(1) Metrics

Pixel Accuracy

클래스별 전체 픽셀에서 예측에 성공한 픽셀의 수

$$\text{pixel accuracy: } \textstyle\sum_i n_{ii} / \sum_i t_i$$

출처 : J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015

5. Results
(1) Metrics

Mean Accuracy

클래스별 전체 픽셀에서 예측에 성공한 픽셀의 수

$$\text{mean accuraccy: } (1/n_{\text{cl}}) \sum_i n_{ii}/t_i$$

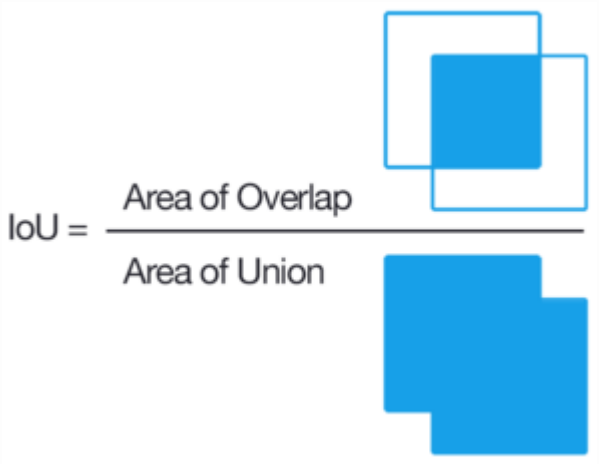
출처 : J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015

5. Results
(1) Metrics

Mean IU

클래스별 전체 픽셀에서 예측에 성공한 픽셀의 수

$$\text{mean IU: } (1/n_{\text{cl}}) \sum_i n_{ii} / \left(t_i + \sum_j n_{ji} - n_{ii} \right)$$



출처 : https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2

5. Results
(1) Metrics

Frequency weighted IU

클래스별 전체 픽셀에서 예측에 성공한 픽셀의 수

frequency weighted IU:

$$\left(\sum_k t_k\right)^{-1} \sum_i t_i n_{ii} / \left(t_i + \sum_j n_{ji} - n_{ii}\right)$$

출처 : J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015

6. Concolusion

(1) Advantages

1. End-to-end 방식의 Semantic Segmentation 방법을 제안
   - 기존의 모델은 전처리 및 후처리가 있거나 복잡한 형태를 가지지만, FCN은 되게 간단한 구조를 가짐
   - 빠른 학습속도와 추론속도를 가지고 있음

2. Pretrained된 모델을 사용할 수가 있음
   - BackBone으로 Pretrained된 모델을 사용하고 마지막의 Fully Connected Layer만 1x1 Convolutional Layer으로 변경해주면 되는 구조
   - Pretrained + Fine Tuning이 가능함

3. Skip Architecture 구조를 이용해서 다양한 resolution의 결과를 합침

4. 다양한 실험과정을 통해서 최적의 결과를 도출했고, 다양한 데이터에 대해 실험을 진행해서 성능 평가

출처 : J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015

6. Concolusion

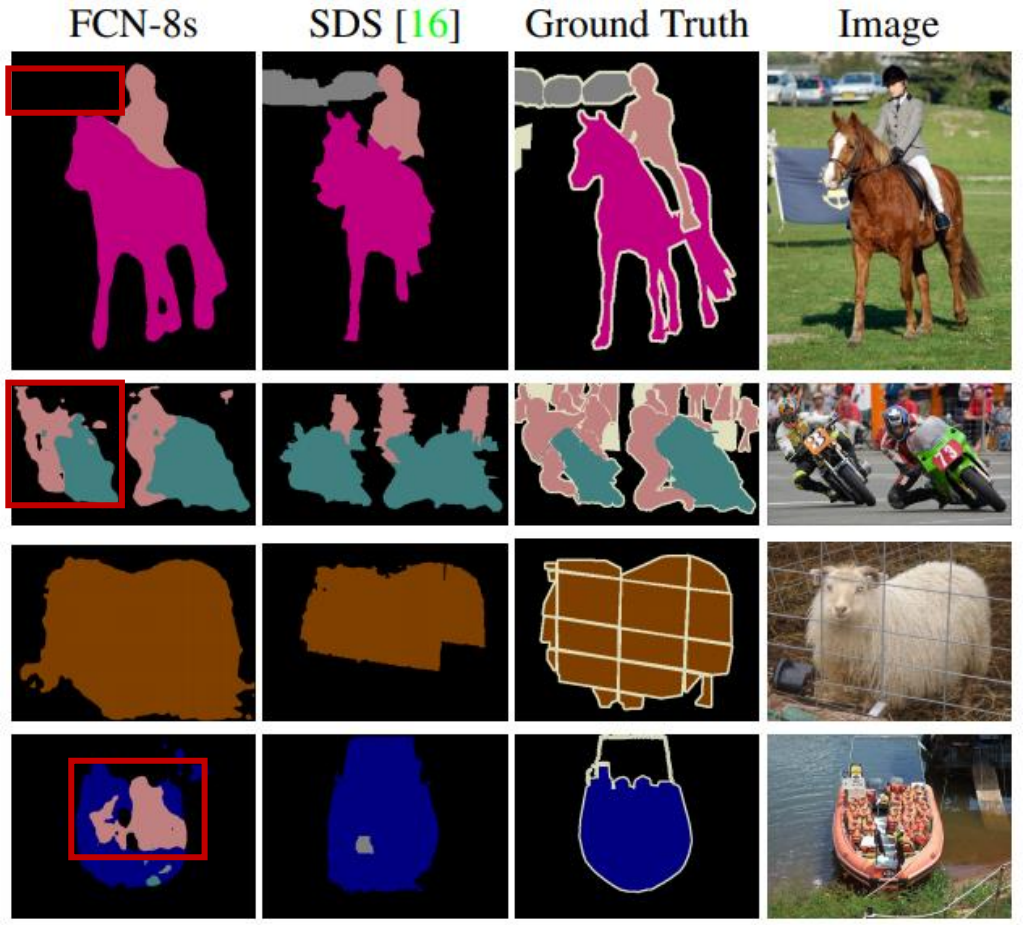
(2) Disadvantages

1. End-to-End 방식이 가지고 있는 문제점들을 보유하고 있음
   - 입력 데이터가 부족한 경우 학습이 잘 되지 않음

2. 세부적인 외곽선들을 잘 못잡는 문제가 있음

3. 작은, 큰 물체들을 잘 잡지 못하는 문제가 있음

4. 큰 물체의 경우 내부에 Object들이 섞이는 문제가 있음



출처 : J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015

7. Appendix

(1) 참고자료

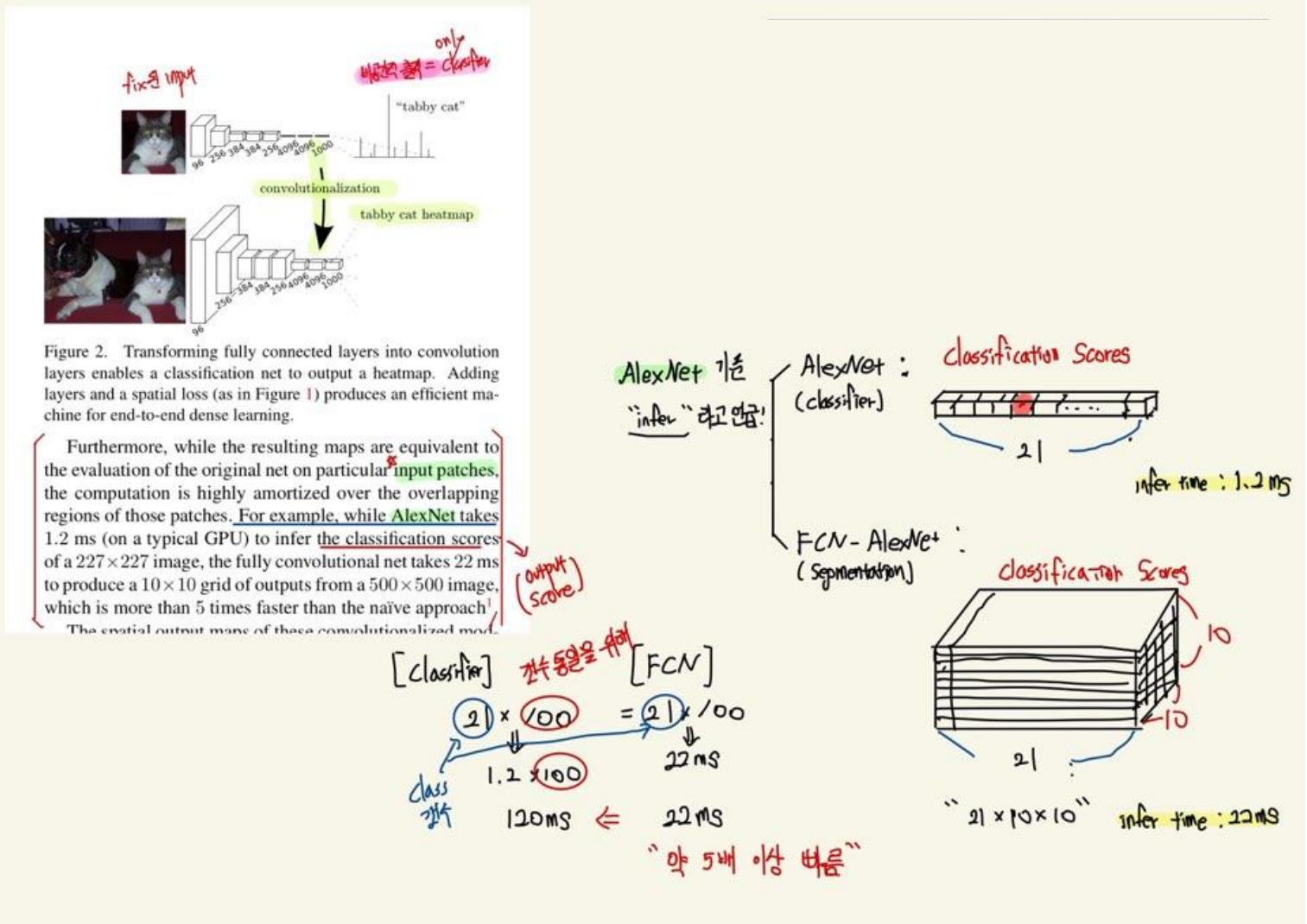
1. J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015
2. A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012
3. K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556, 2014
4. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. CoRR, abs/1409.4842, 2014
5. R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In Computer Vision and Pattern Recognition, 2014
6. J. Long, N. Zhang, and T. Darrell. Do convnets learn correspondence? In NIPS, 2014
7. C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2013
8. S. Gupta, R. Girshick, P. Arbelaez, and J. Malik. Learning rich features from RGB-D images for object detection and segmentation. In ECCV. Springer, 2014
9. P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In ICLR, 2014
10. Investigations on the inference optimization techniques and their impact on multiple hardware platforms for Semantic Segmentation

출처 : J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015

7. Appendix

- (2) 이해하지 못했던 부분 (헷갈린 부분)

1. 500 x 500 -> 10 x 10 output grid 생성



출처 : J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015

감사합니다