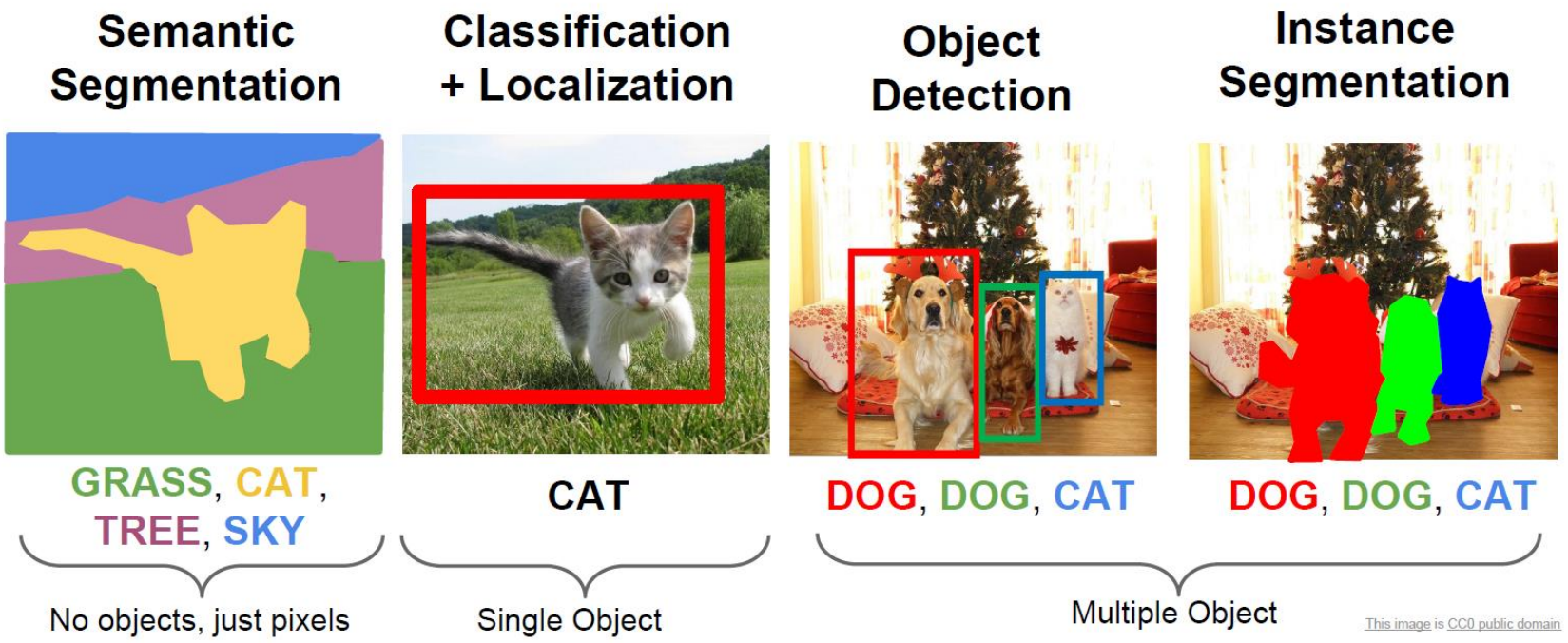
Deep Learning Applications

2017.09.23

최건호

INDEX

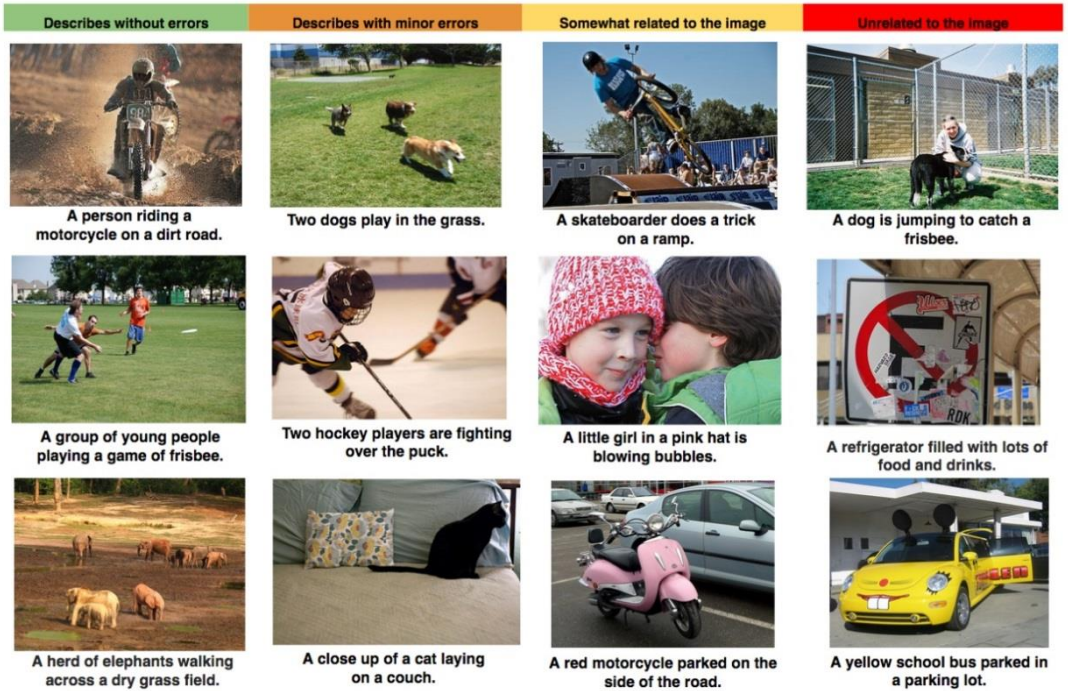
CONTENTS



(출처: CS231n 2017 chapter 11)

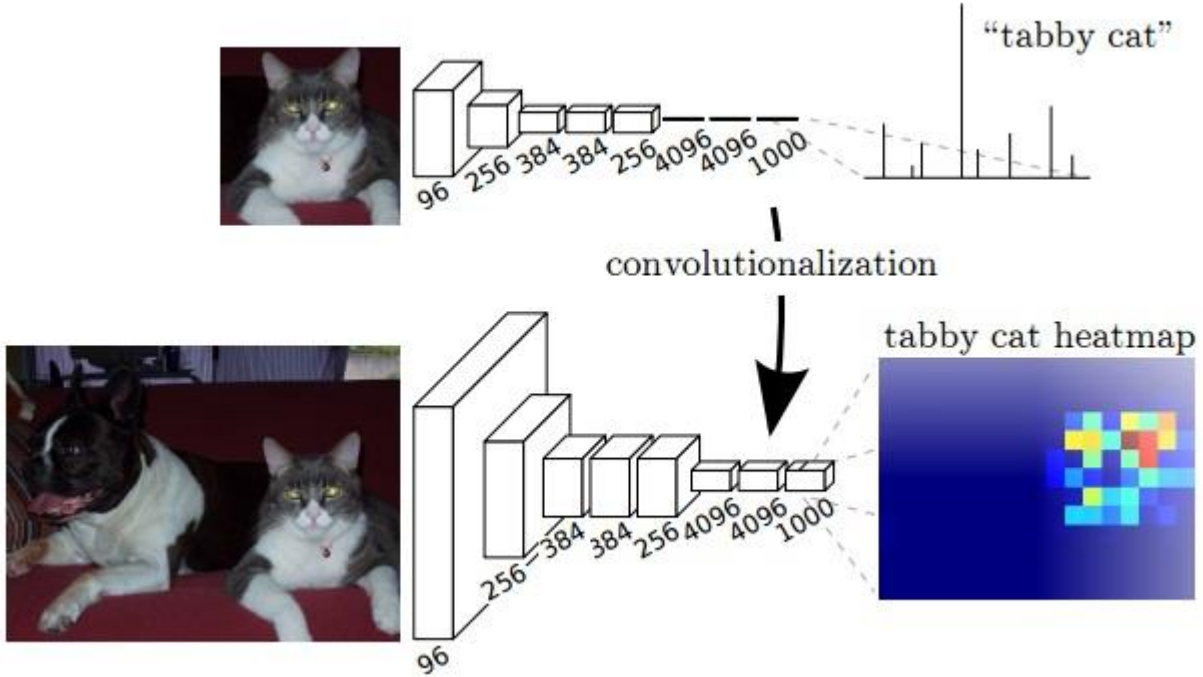
CONTENTS

Image Captioning



(출처: CS231n 2017 chapter 11)

Semantic Segmentation

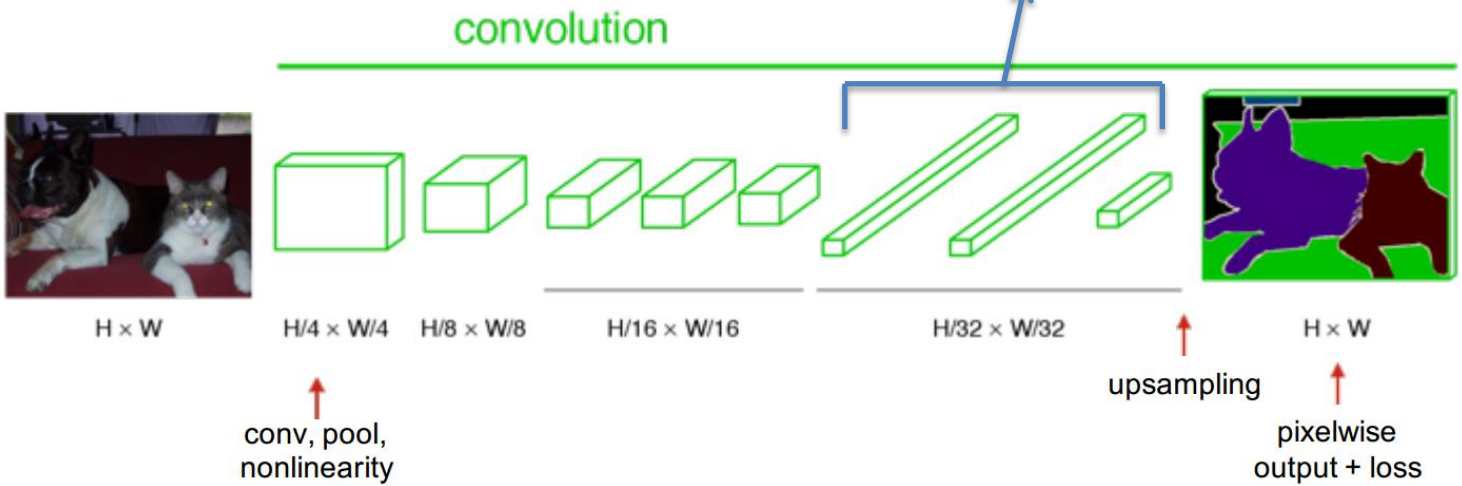


Classification에서 클래스를 결정하는 부분은 어디에 해당할까?

https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf

Semantic Segmentation

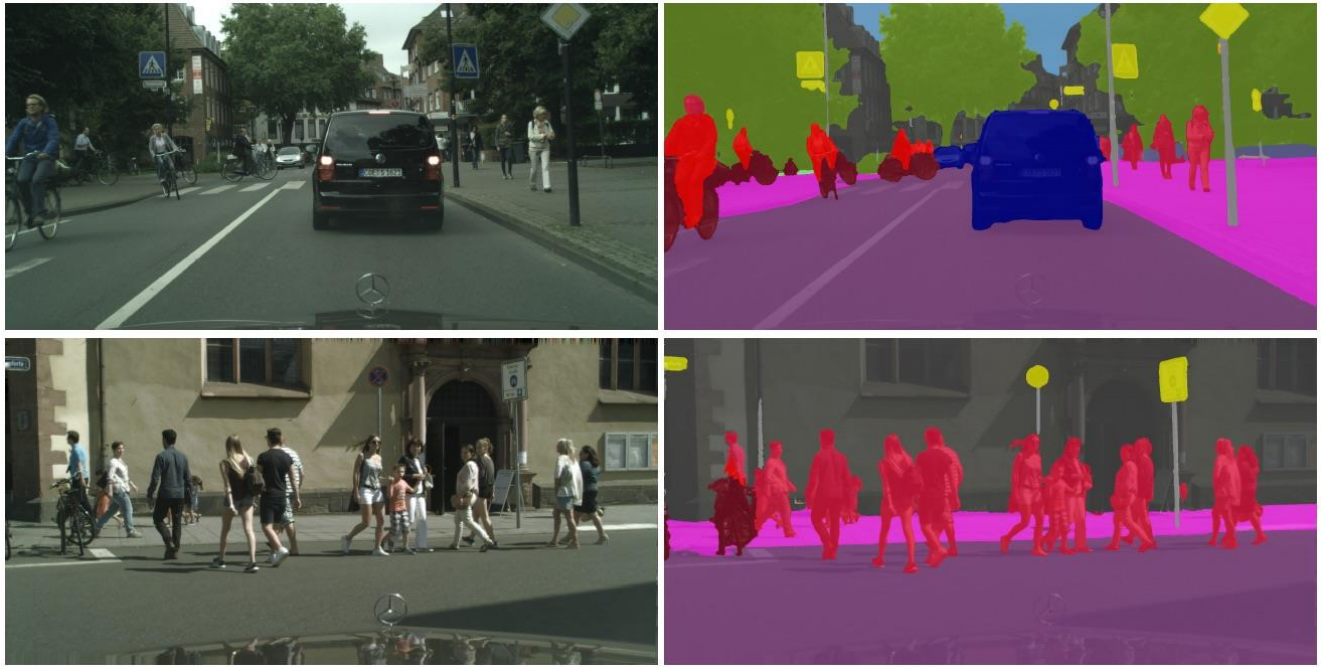
Fully connected network를
convolution 연산으로 대체



Pre-trained classification 모델이 이미지의 특징을 잘 잡아내니까
이를 Segmentation task에 적용해보자

https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf

Semantic Segmentation

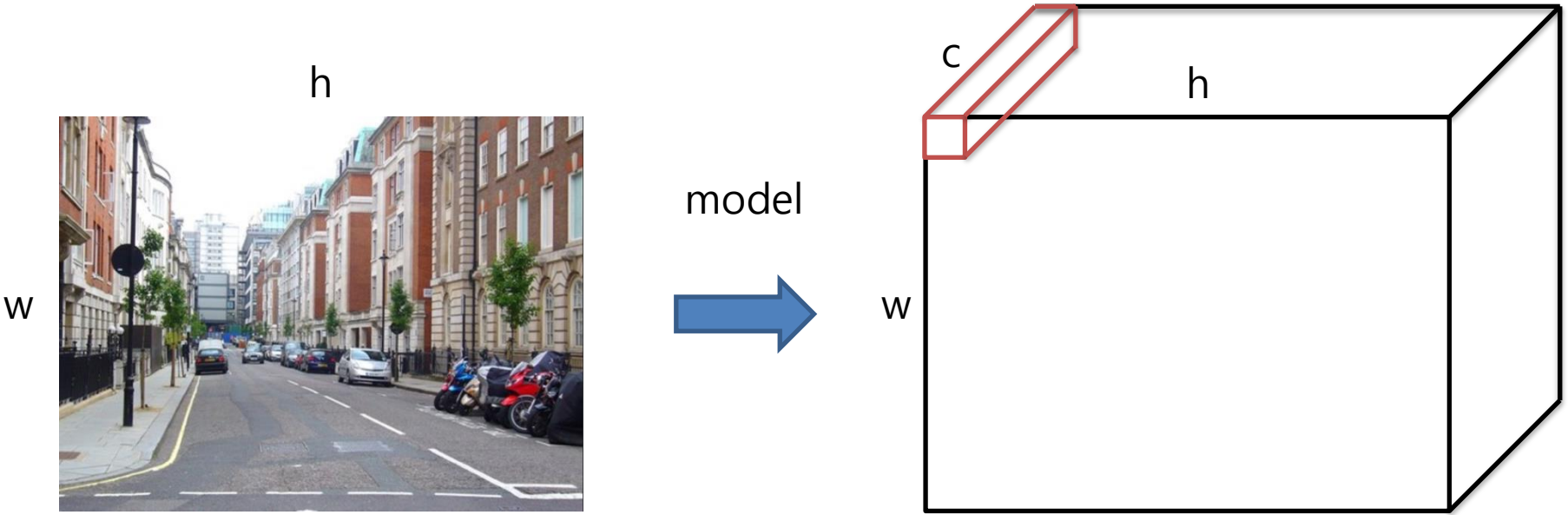


class 가 있을 수도 있고 없을 수도 있음

http://vladlen.info/publications/feature-space-optimization-for-semantic-video-segmentation/

Semantic Segmentation

정해진 클래스가 있는 경우



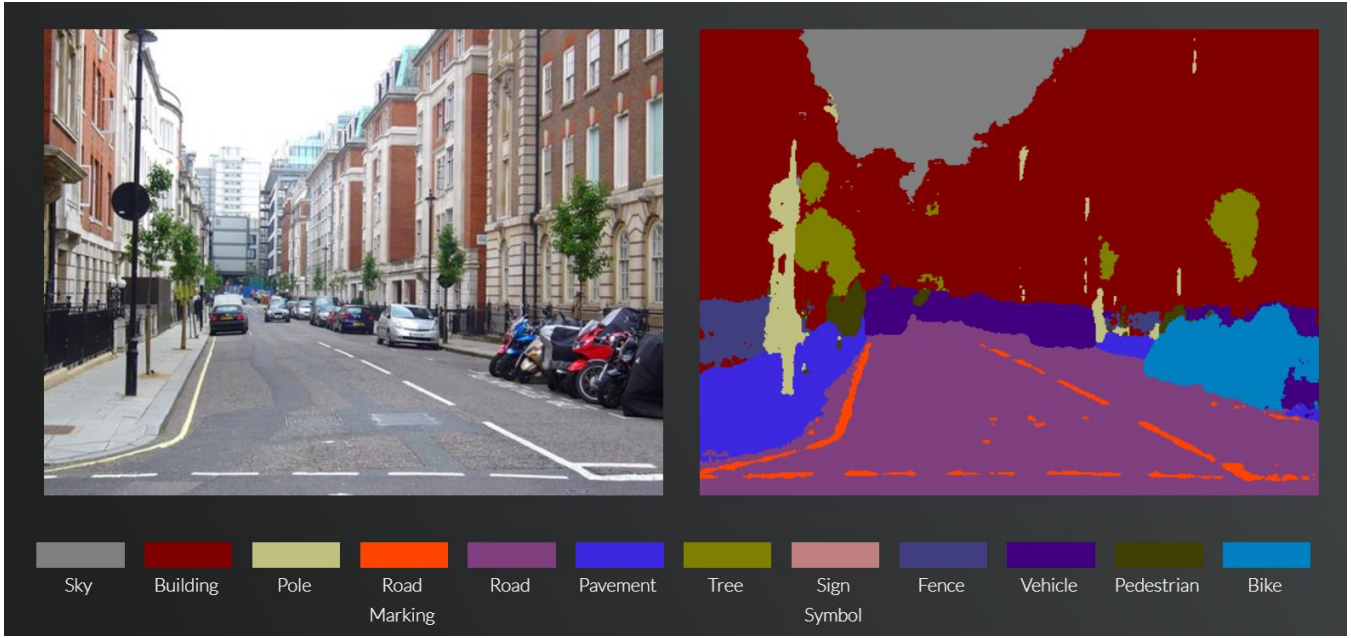
c: 클래스의 개수

픽셀 하나당 각 클래스 중 어디에 해당하는지 softmax로 구할 수 있음

http://mi.eng.cam.ac.uk/projects/segnet/#demo

Semantic Segmentation

정해진 클래스가 있는 경우

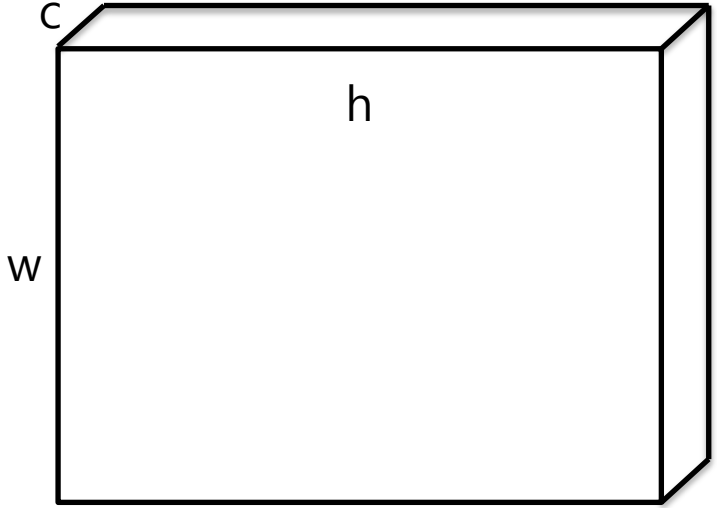


http://mi.eng.cam.ac.uk/projects/segnet/#demo

Semantic Segmentation

정해진 클래스가 없는 경우

model



Output 또한 하나의 이미지

http://mi.eng.cam.ac.uk/projects/segnet/#demo

Semantic Segmentation

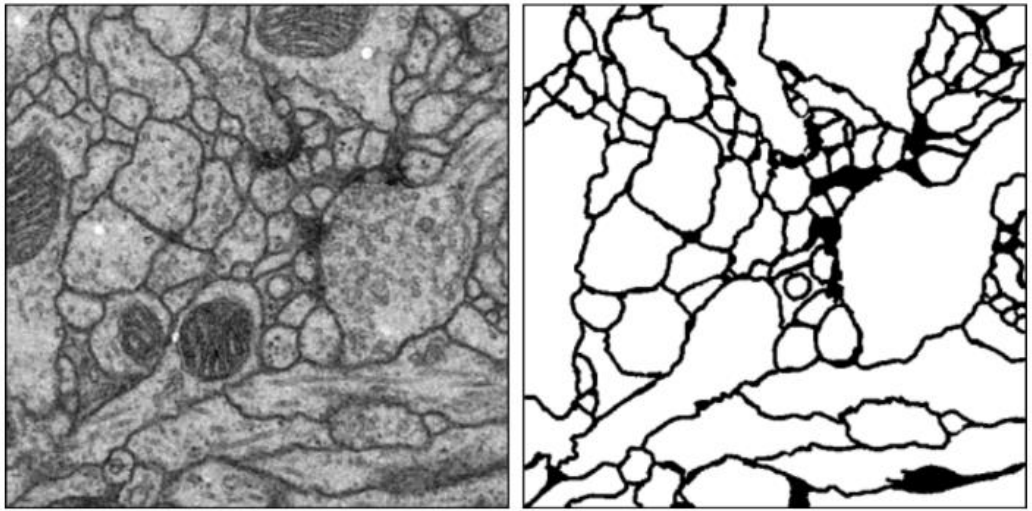
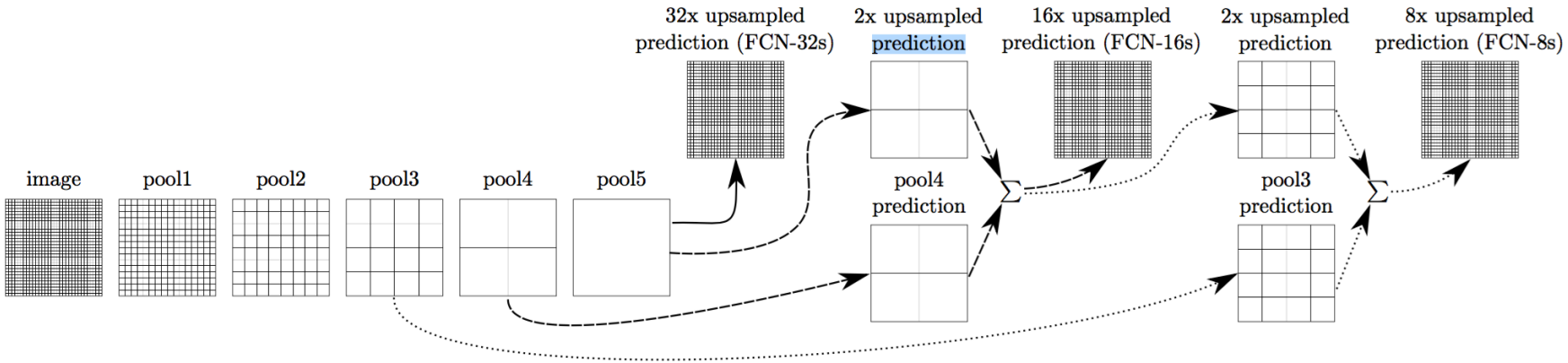


Figure 1. An example of EM image (left) and its cell membrane segmentation result (right).

Semantic Segmentation



적당한 위치에서 upsampling(=transposed convolution)
하고 summation으로 합치는 과정을 거침.

몇 배 늘렸는지에 따라 FCN-Xs로 부름

https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf

Semantic Segmentation

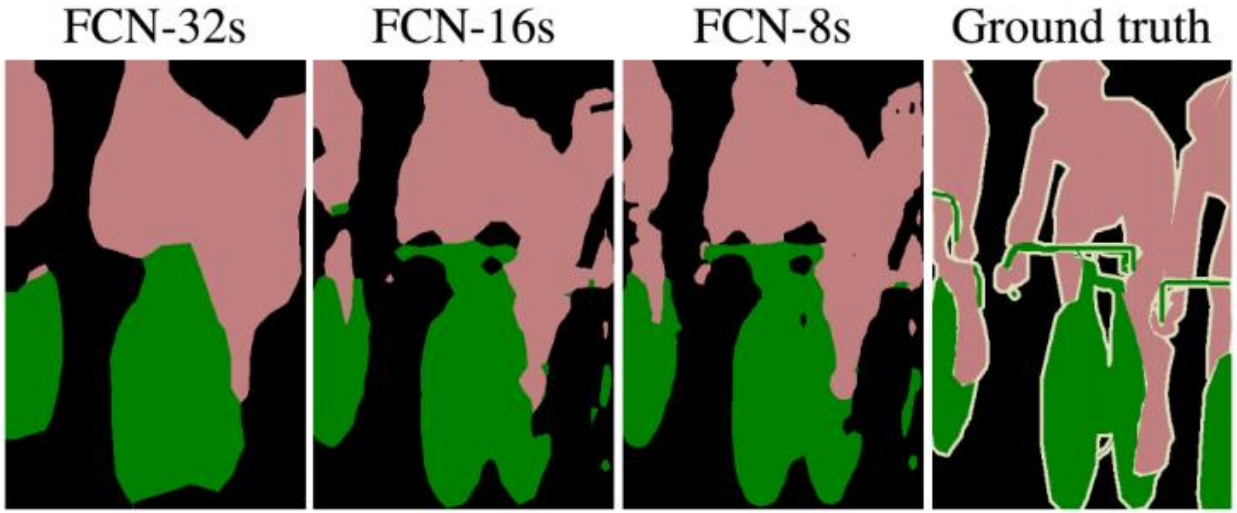
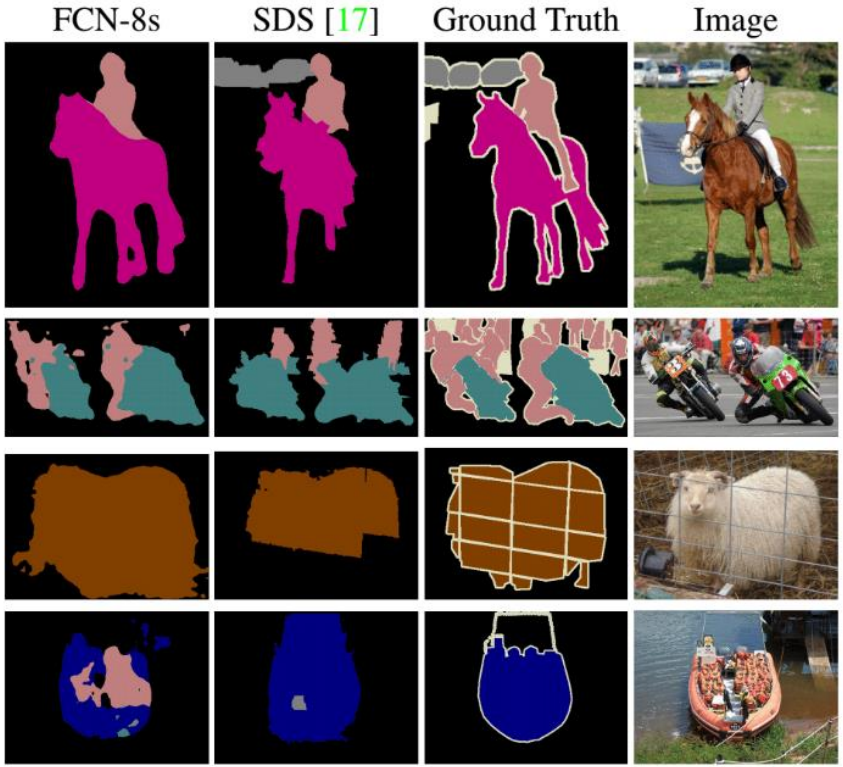


Figure 4. Refining fully convolutional nets by fusing information from layers with different strides improves segmentation detail. The first three images show the output from our 32, 16, and 8 pixel stride nets (see Figure 3).

Semantic Segmentation



https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf

Semantic Segmentation

Deconvolution Network

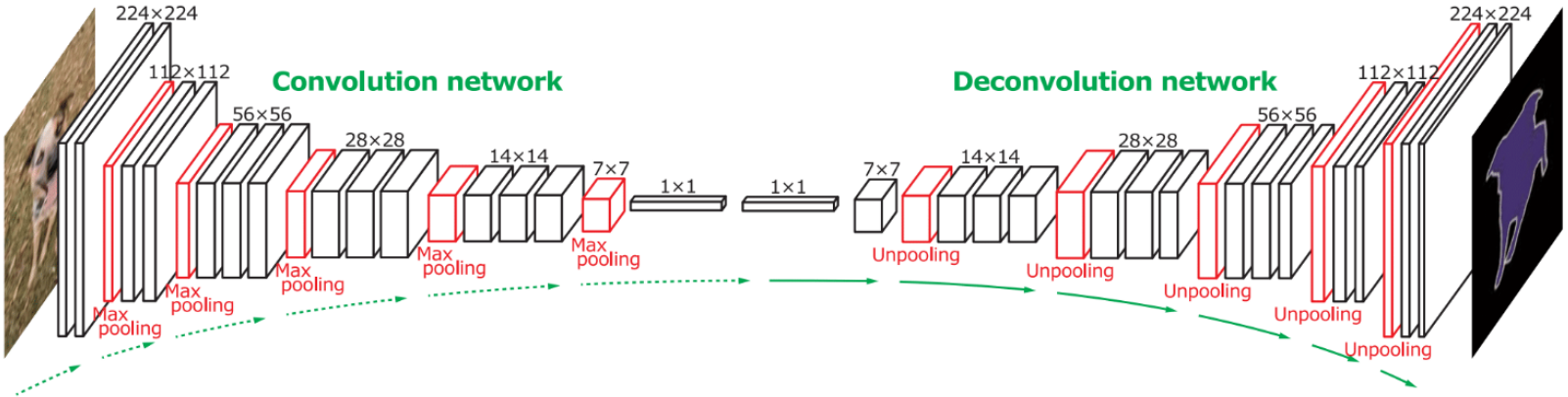


Figure 2. Overall architecture of the proposed network. On top of the convolution network based on VGG 16-layer net, we put a multi-layer deconvolution network to generate the accurate segmentation map of an input proposal. Given a feature representation obtained from the convolution network, dense pixel-wise class prediction map is constructed through multiple series of unpooling, deconvolution and rectification operations.

https://arxiv.org/pdf/1505.04366.pdf

Semantic Segmentation

Deconvolution Network

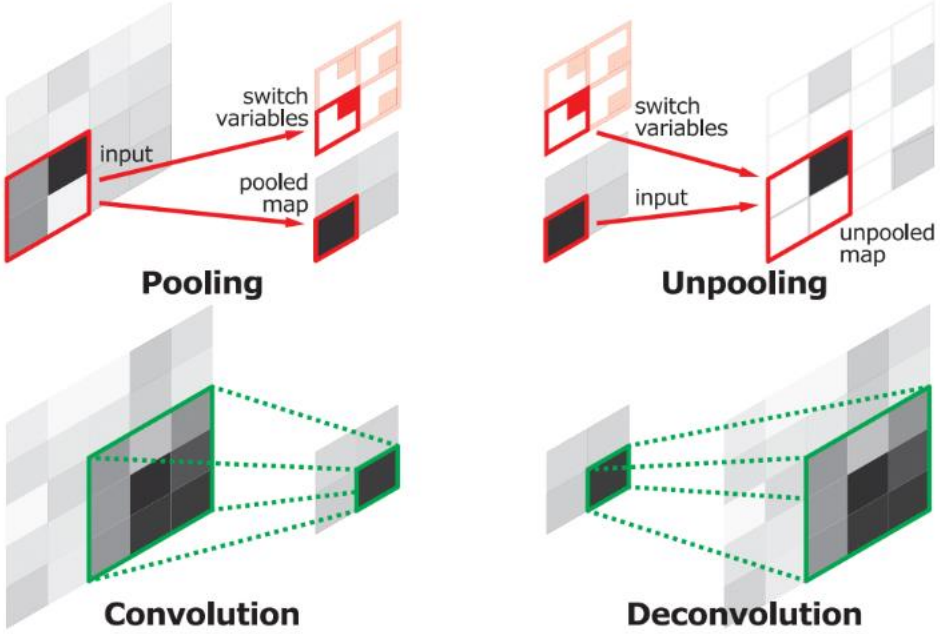
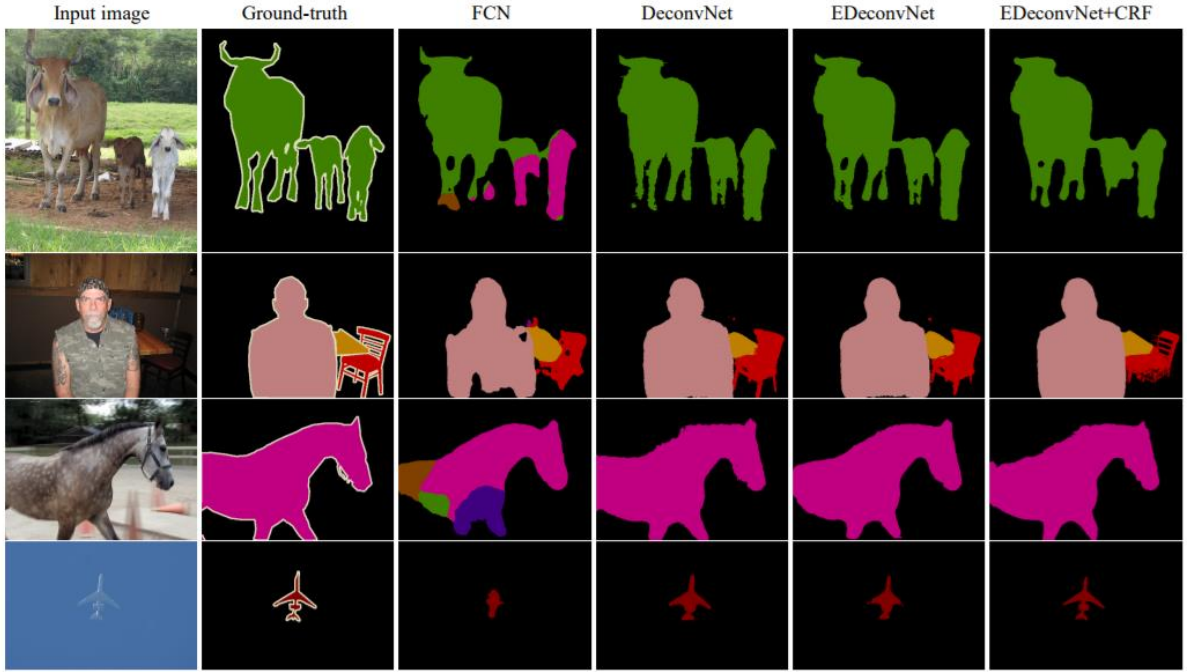


Figure 3. Illustration of deconvolution and unpooling operations.

https://arxiv.org/pdf/1505.04366.pdf

Semantic Segmentation

Deconvolution Network



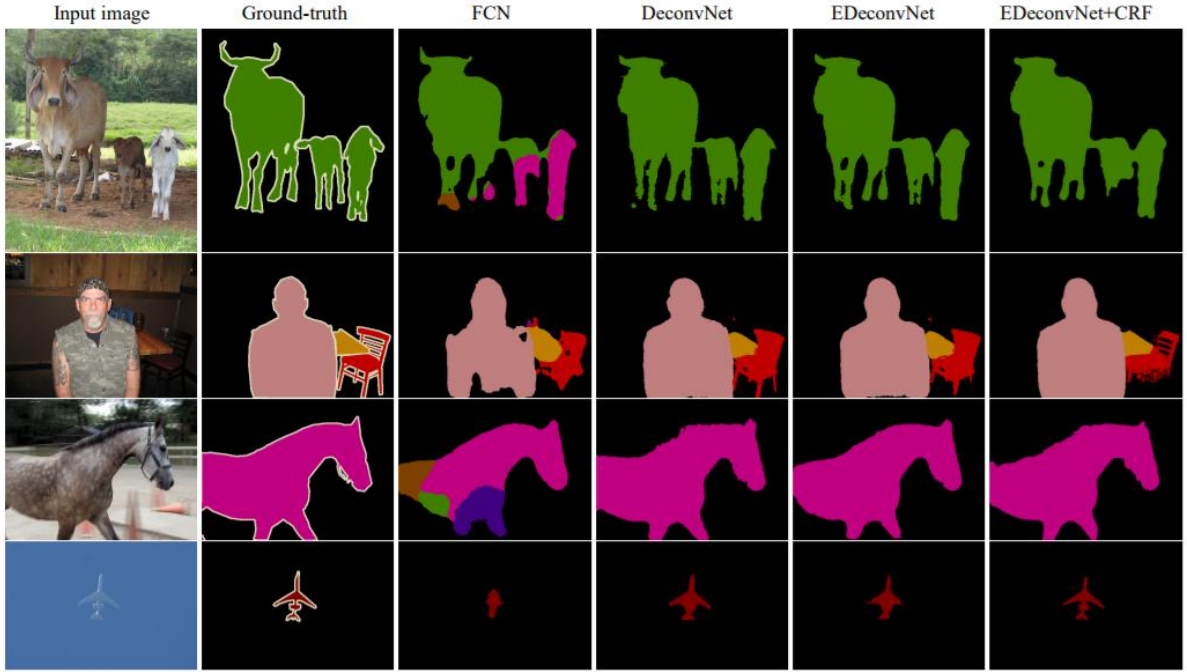
(a) Examples that our method produces better results than FCN [17].

https://arxiv.org/pdf/1505.04366.pdf

Semantic Segmentation

Deconvolution Network

FCN+DeconvNet Ensemble



(a) Examples that our method produces better results than FCN [17].

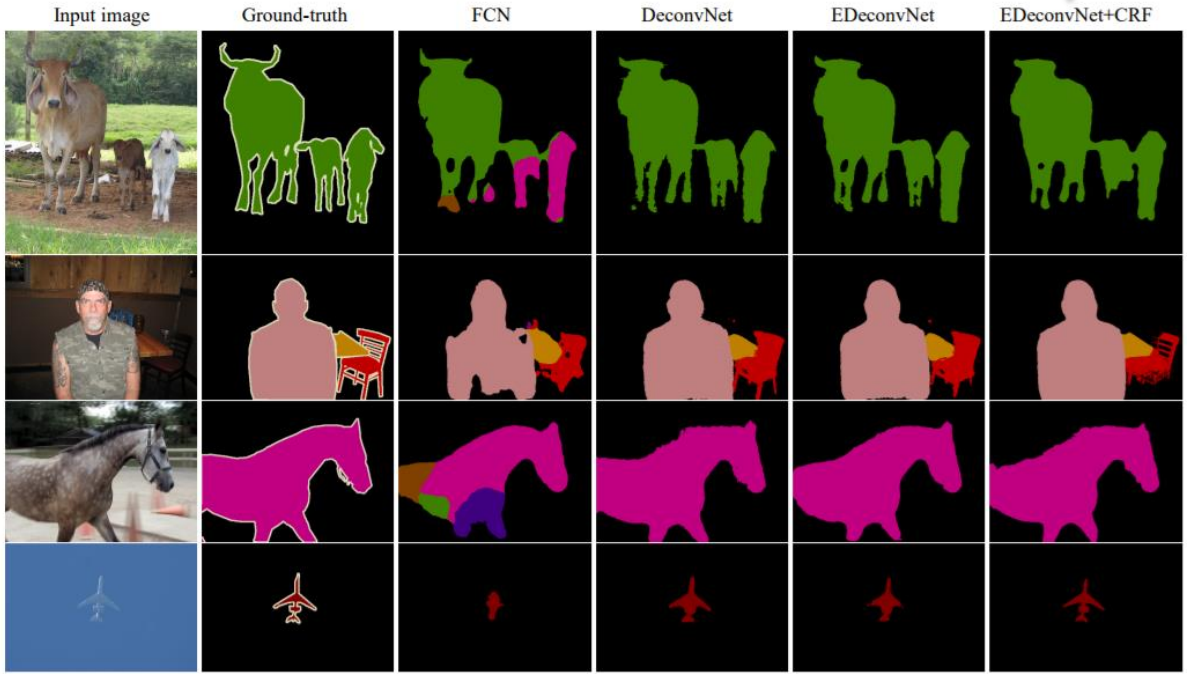
https://arxiv.org/pdf/1505.04366.pdf

Semantic Segmentation

Deconvolution Network

FCN+DeconvNet Ensemble

Post processing으로 CRF 사용



(a) Examples that our method produces better results than FCN [17].

https://arxiv.org/pdf/1505.04366.pdf

Semantic Segmentation

5.2. Ensemble with FCN

Our algorithm based on the deconvolution network has complementary characteristics to the approaches relying on FCN; our deconvolution network is appropriate to capture the fine-details of an object, whereas FCN is typically good at extracting the overall shape of an object. In addition, instance-wise prediction is useful for handling objects with various scales, while fully convolutional network with a coarse scale may be advantageous to capture context within image. Exploiting these heterogeneous properties may lead to better results, and we take advantage of the benefit of both algorithms through ensemble.

FCN은 Overall Shape를 잘 잡아내고 Deconvolution Network은 detail을 잘 잡아내기 때문에 Ensemble 한다.

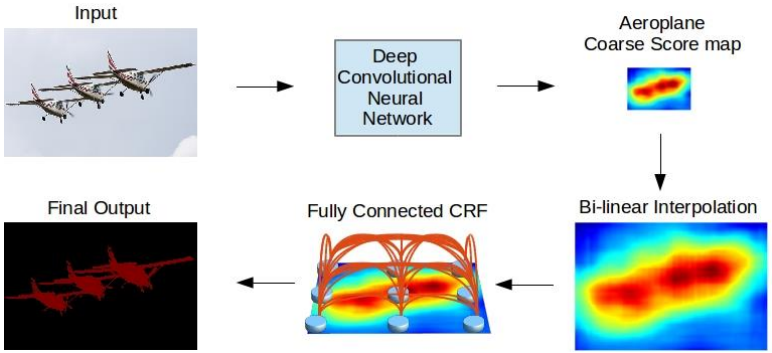
Semantic Segmentation

Deconvolution Network

5.2. Ensemble with FCN

Our algorithm based on the deconvolution network has complementary characteristics to the approaches relying on FCN; our deconvolution network is appropriate to capture the fine-details of an object, whereas FCN is typically good at extracting the overall shape of an object. In addition, instance-wise prediction is useful for handling objects with various scales, while fully convolutional network with a coarse scale may be advantageous to capture context within image. Exploiting these heterogeneous properties may lead to better results, and we take advantage of the benefit of both algorithms through ensemble.

FCN은 Overall Shape를 잘 잡아내고 Deconvolution Network은 detail을 잘 잡아내기 때문에 Ensemble 한다.



CRF는 주변 픽셀정보로부터 해당 픽셀에 대한 값을 구하는 방식

https://arxiv.org/pdf/1505.04366.pdf

Semantic Segmentation

Dilated Convolution

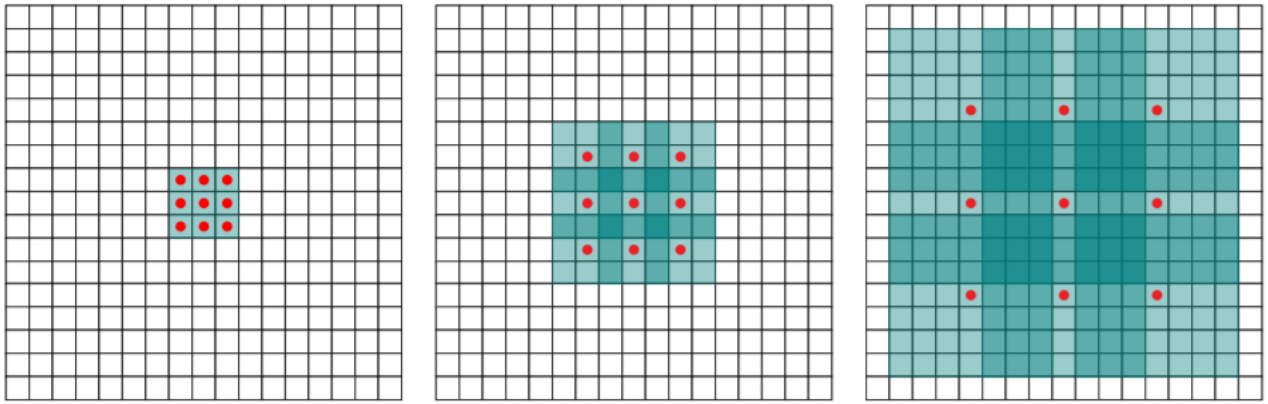
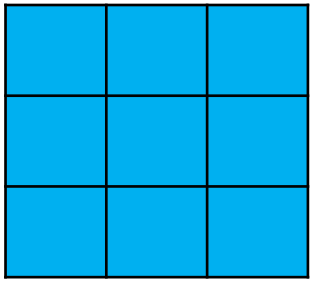


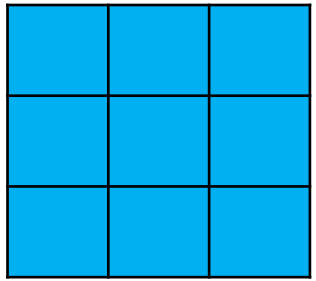
Figure 1: Systematic dilation supports exponential expansion of the receptive field without loss of resolution or coverage. (a) F_1 is produced from F_0 by a 1-dilated convolution; each element in F_1 has a receptive field of 3×3. (b) F_2 is produced from F_1 by a 2-dilated convolution; each element in F_2 has a receptive field of 7×7. (c) F_3 is produced from F_2 by a 4-dilated convolution; each element in F_3 has a receptive field of 15×15. The number of parameters associated with each layer is identical. The receptive field grows exponentially while the number of parameters grows linearly.

출처: MULTI-SCALE CONTEXT AGGREGATION BY DILATED CONVOLUTIONS

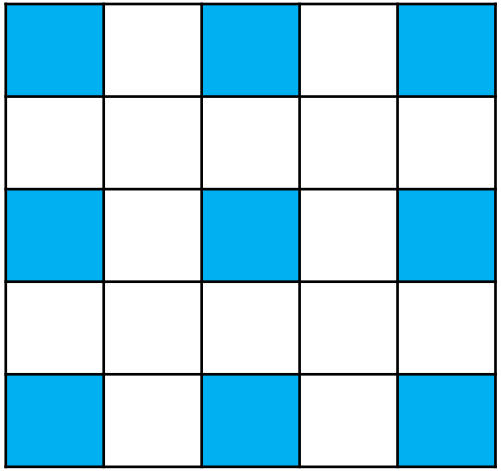
Semantic Segmentation



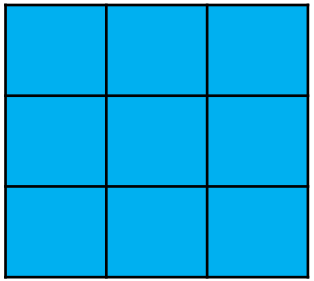
Semantic Segmentation



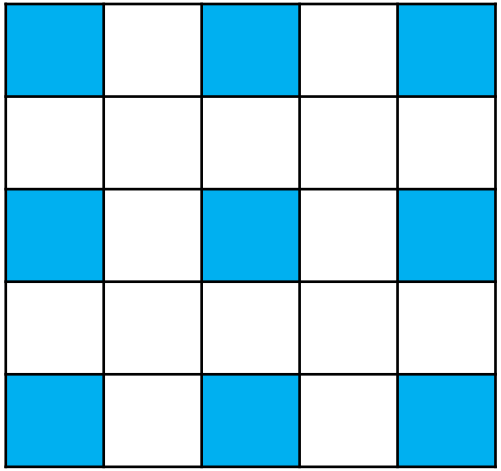
빈 공간은 0 ⇩ dilation=2

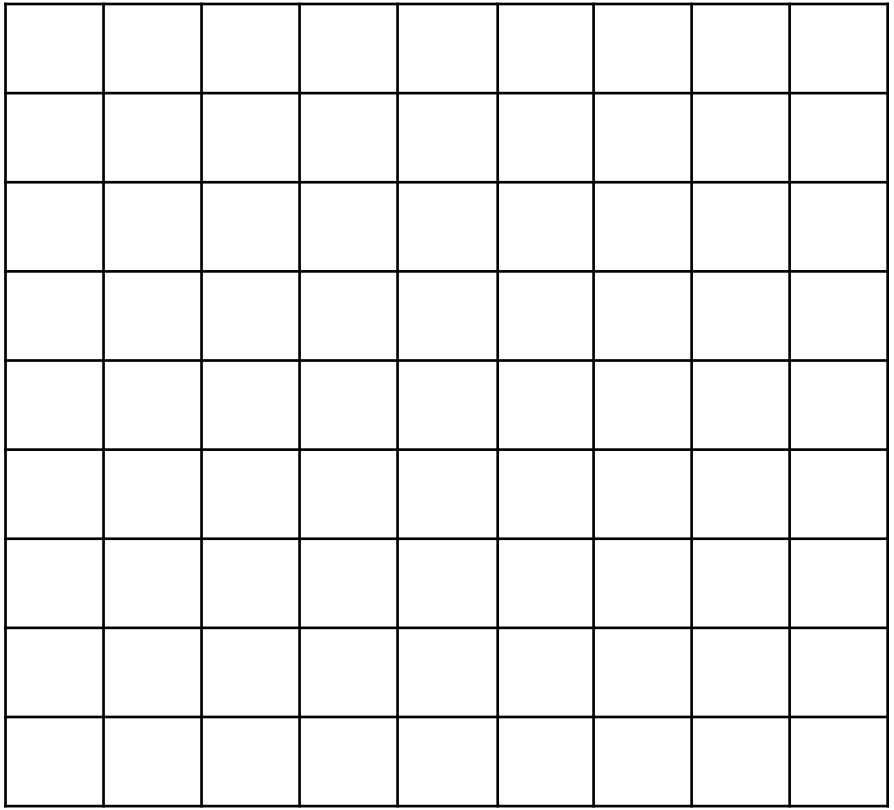


Semantic Segmentation

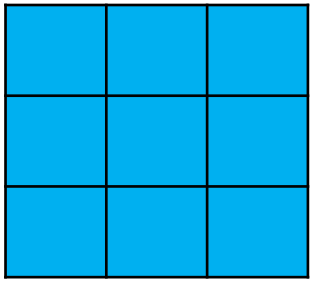


빈 공간은 0 ⇩ dilation=2

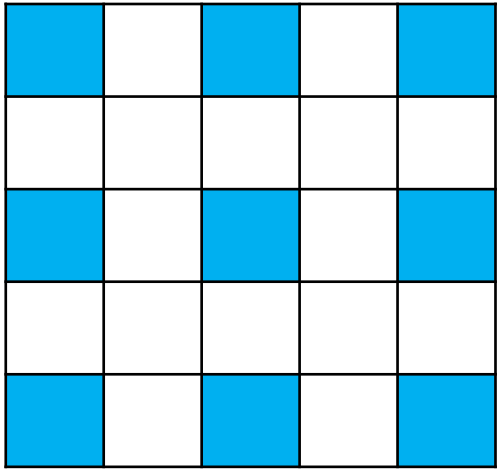


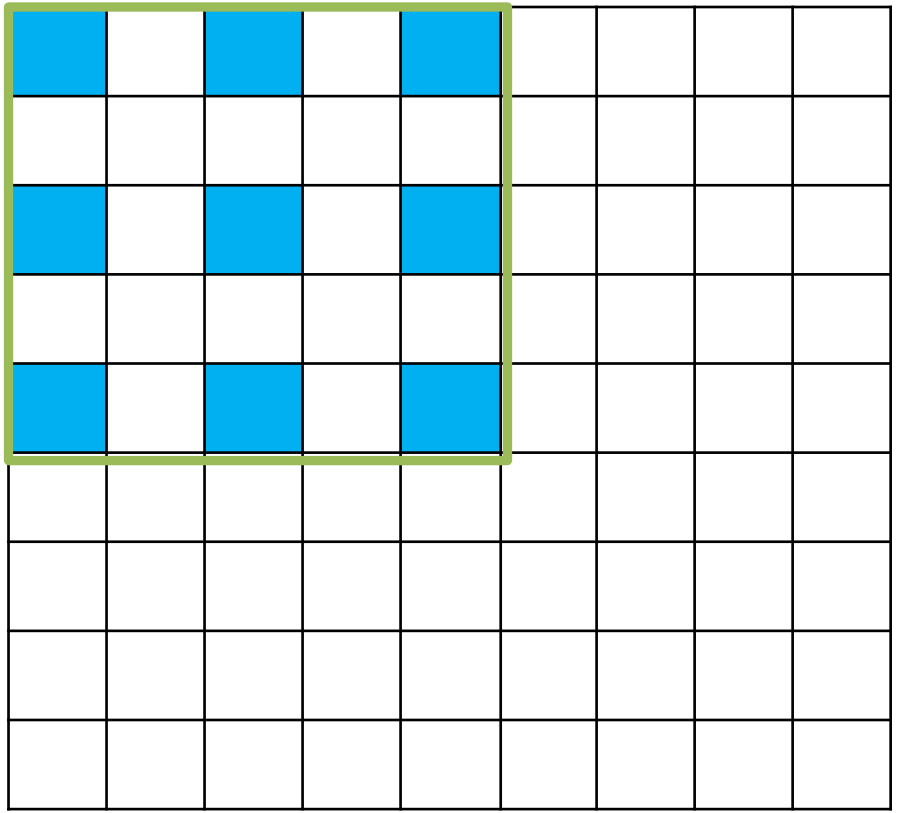


Semantic Segmentation

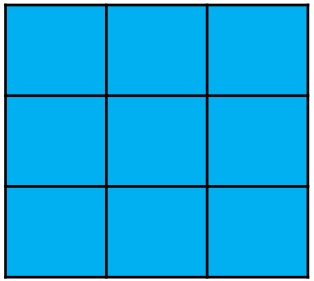


빈 공간은 0 ⇩ dilation=2

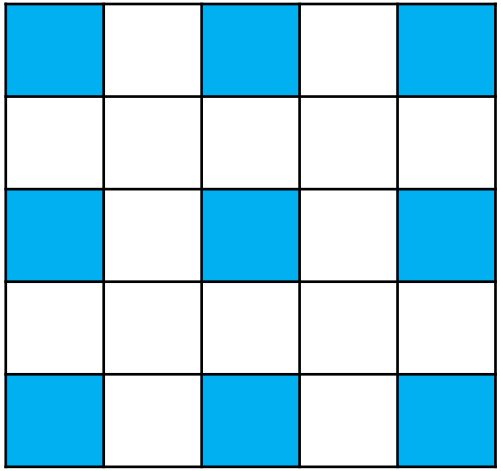


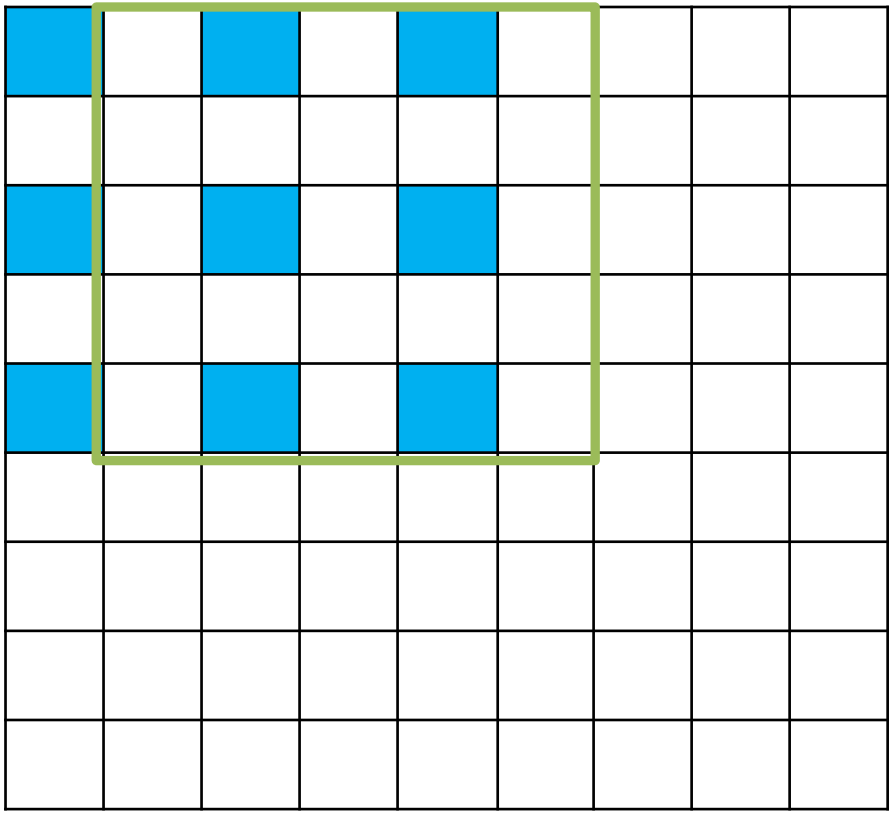


Semantic Segmentation

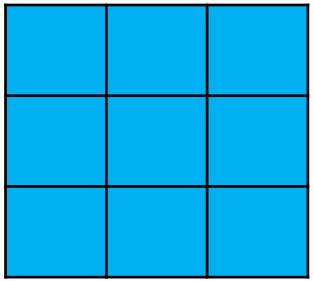


빈 공간은 0 ⇩ dilation=2

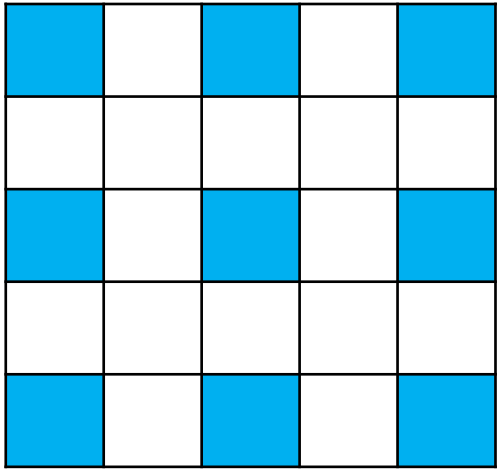


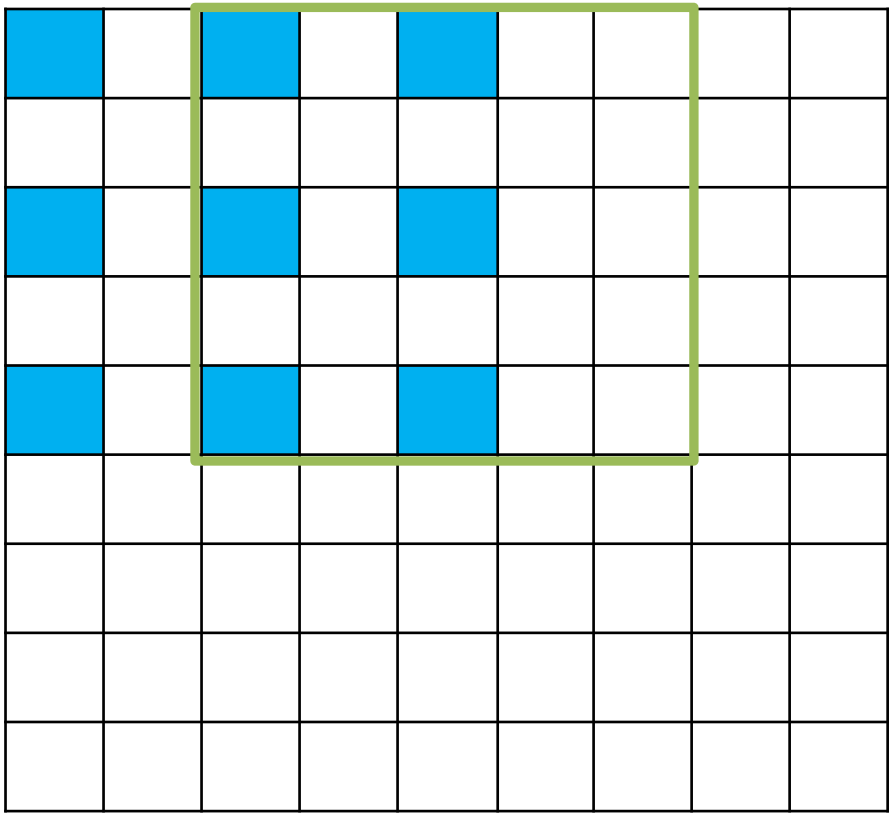


Semantic Segmentation

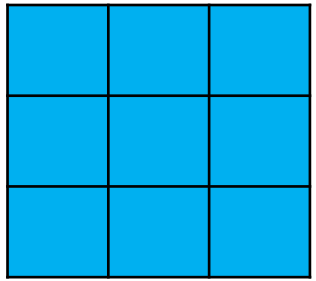


빈 공간은 0 ⇩ dilation=2

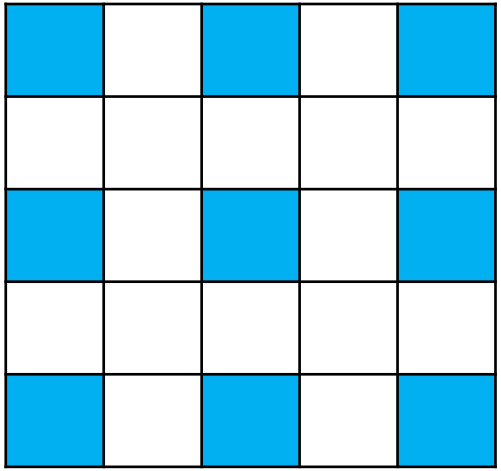


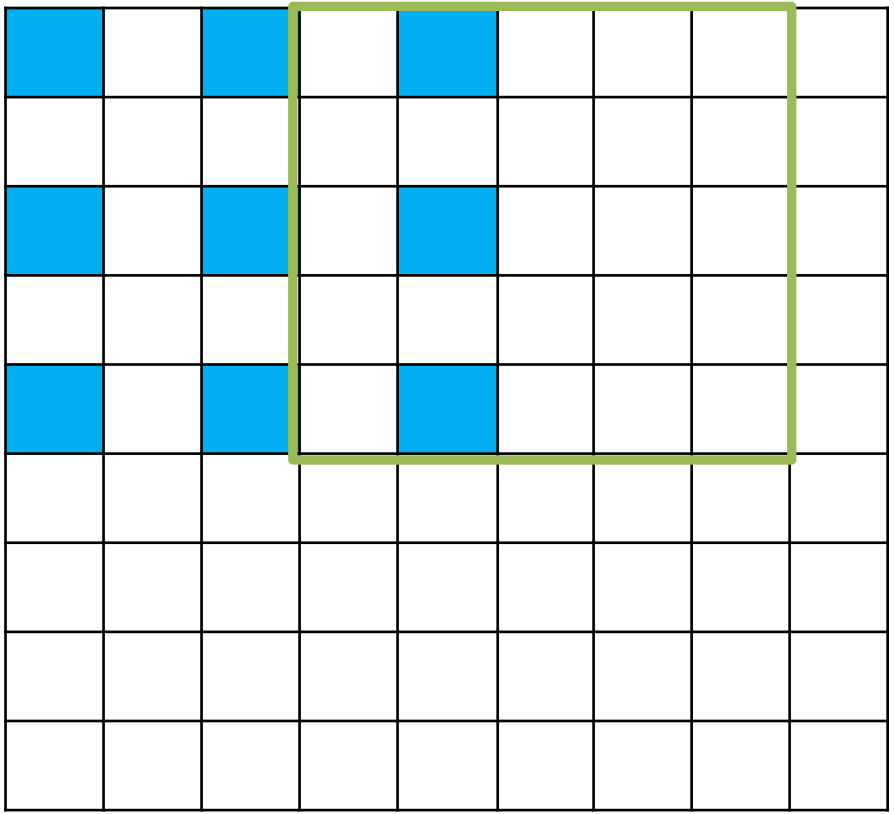


Semantic Segmentation

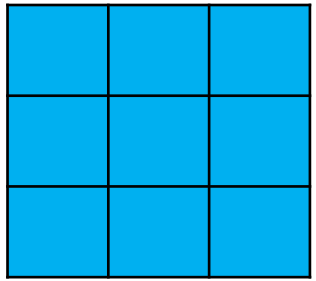


빈 공간은 0 ⇩ dilation=2

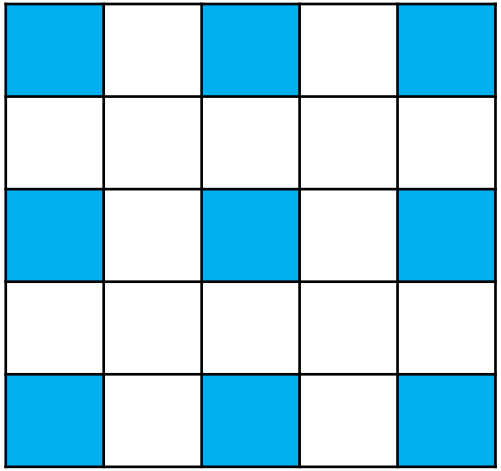


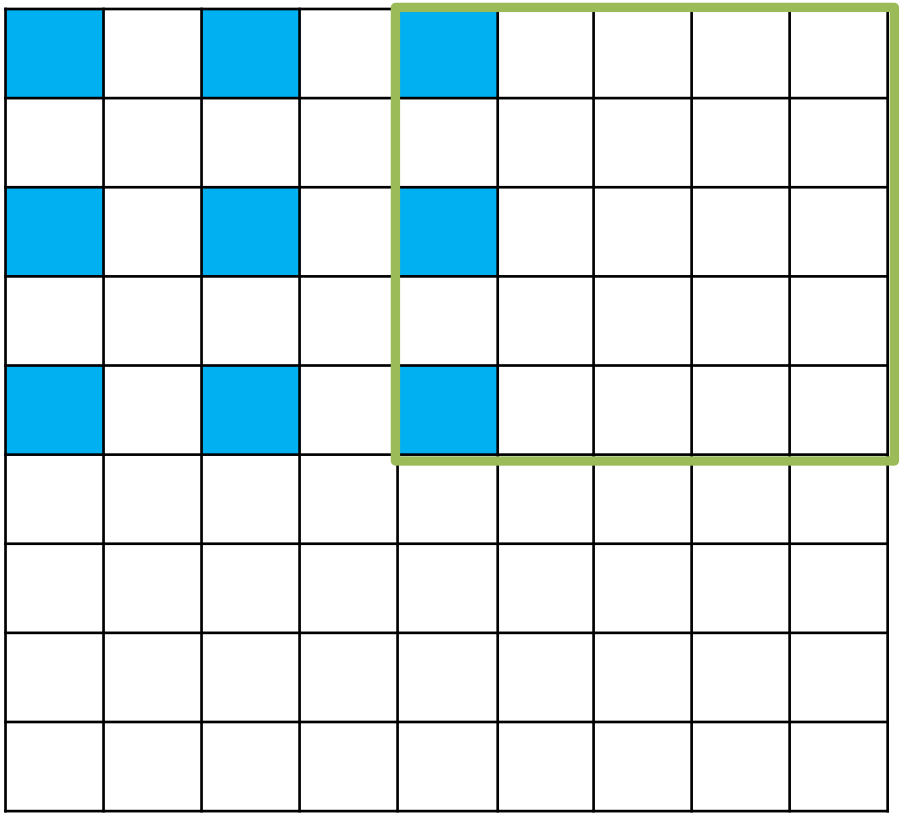


Semantic Segmentation



빈 공간은 0 ⇩ dilation=2





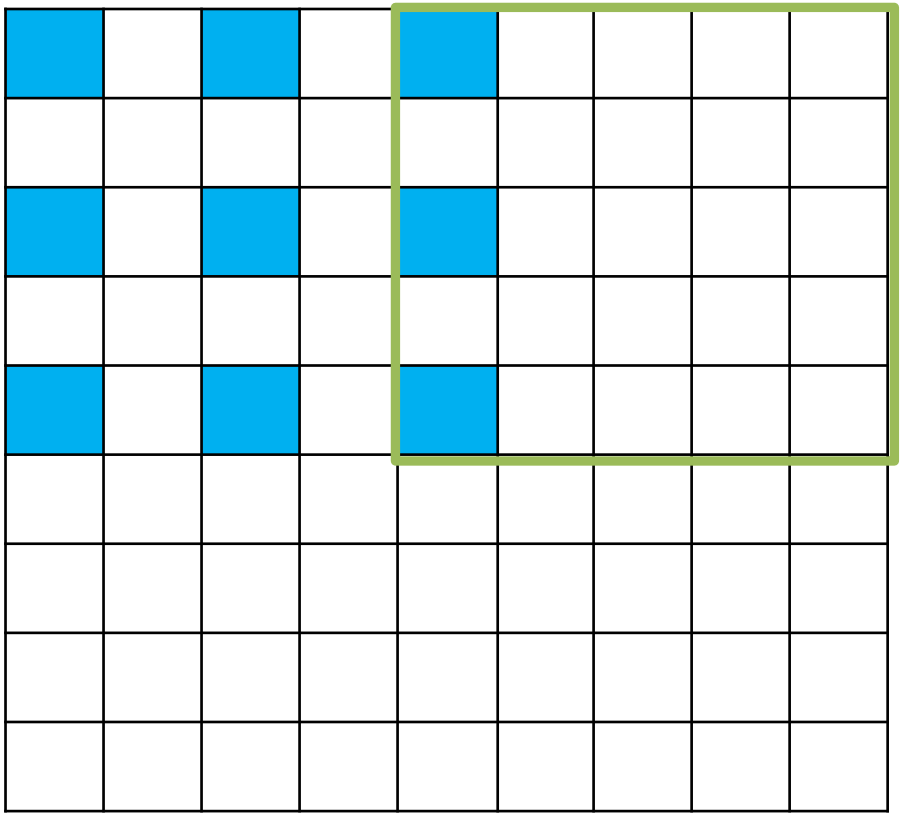
Semantic Segmentation

$$H = floor\left(\frac{H + 2 * P - D * (K - 1) - 1}{S}\right) + 1$$

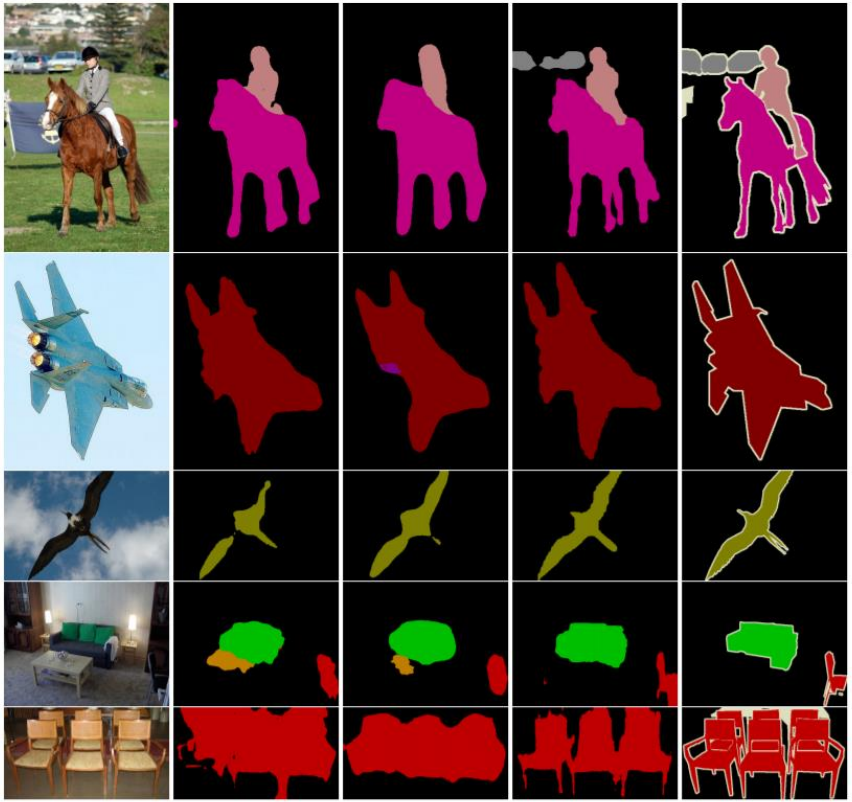
H: height	9
P: padding	0
D: dilation	2
K: kernel	3
S: stride	1

$$H = floor\left(\frac{9 + 2 * 0 - 2 * (3 - 1) - 1}{1}\right) + 1$$

$$= 5$$



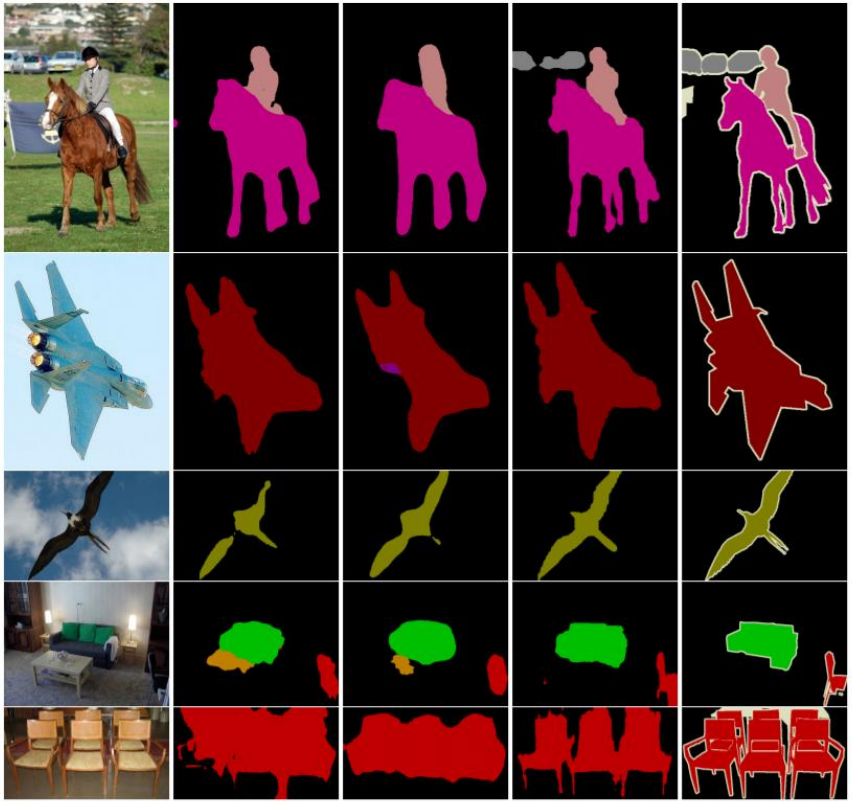
Semantic Segmentation



(a) Image (b) FCN-8s (c) DeepLab (d) Our front end (e) Ground truth

출처: MULTI-SCALE CONTEXT AGGREGATION BY DILATED CONVOLUTIONS

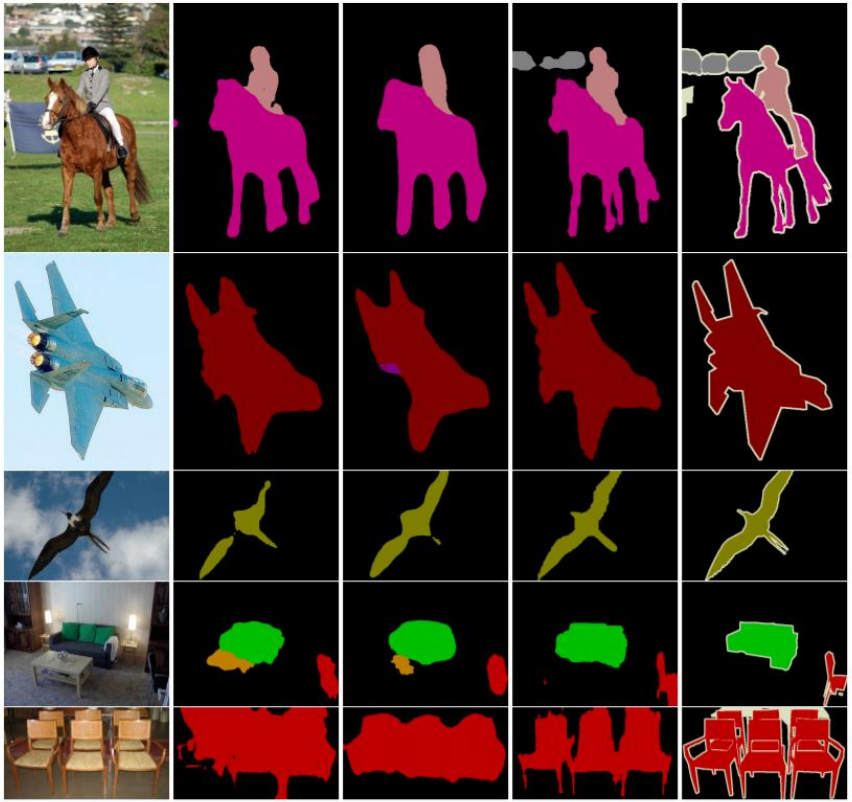
Semantic Segmentation



Dilated(à trous) Convolution은
같은 수의 파라미터로 더 넓은
receptive field를 커버할 수 있게 해줌

출처: MULTI-SCALE CONTEXT AGGREGATION BY DILATED CONVOLUTIONS

Semantic Segmentation

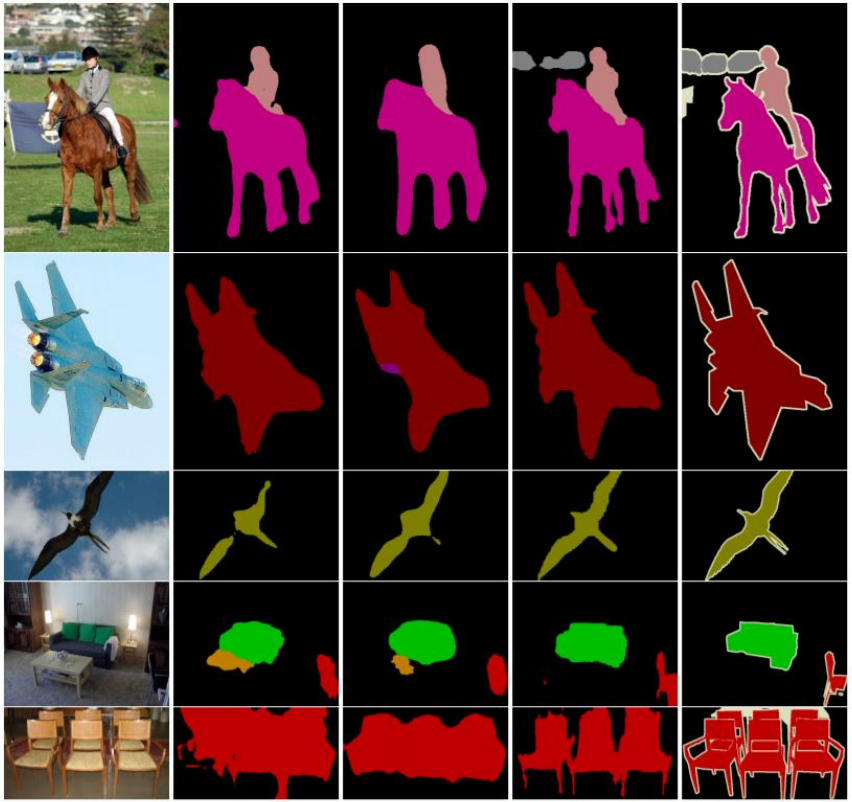


Dilated(à trous) Convolution은
같은 수의 파라미터로 더 넓은
receptive field를 커버할 수 있게 해줌

또한 Dilation=1일 때는 일반 convolution
연산과 같기 때문에 general하게 적용하기
편함

출처: MULTI-SCALE CONTEXT AGGREGATION BY DILATED CONVOLUTIONS

Semantic Segmentation



Dilated(à trous) Convolution은
같은 수의 파라미터로 더 넓은
receptive field를 커버할 수 있게 해줌

또한 Dilation=1일 때는 일반 convolution
연산과 같기 때문에 general하게 적용하기
편함

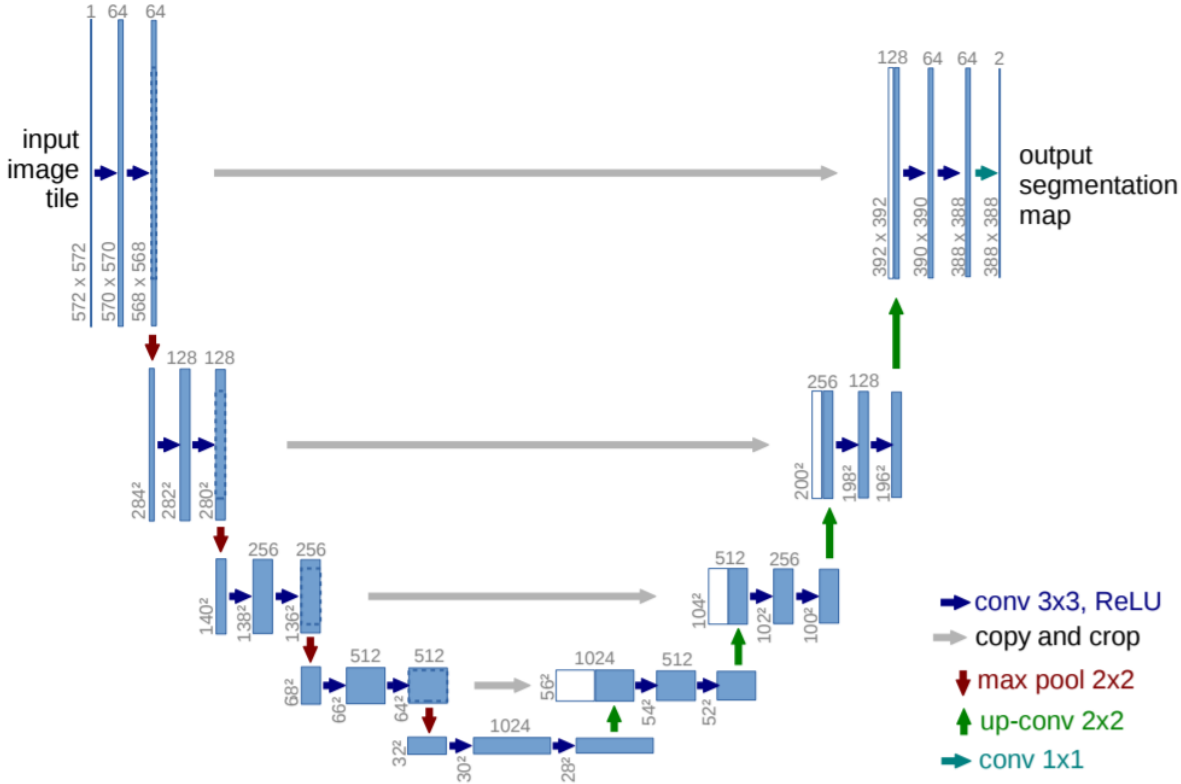
```
class torch.nn.Conv2d(in_channels,
out_channels, kernel_size, stride=1,
padding=0, dilation=1, groups=1, bias=True)
```

[source]

출처: MULTI-SCALE CONTEXT AGGREGATION BY DILATED CONVOLUTIONS

Semantic Segmentation

U-Net



Semantic Segmentation

U-Net

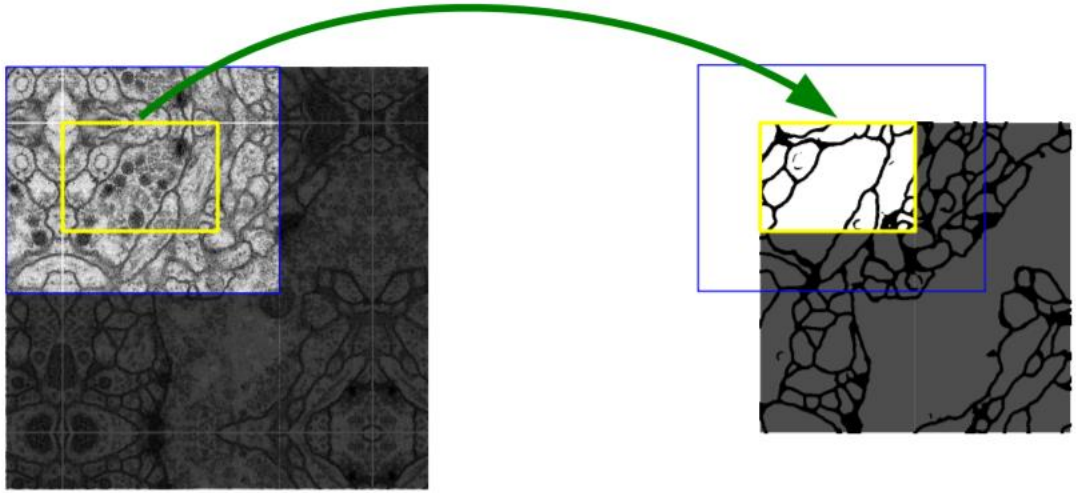
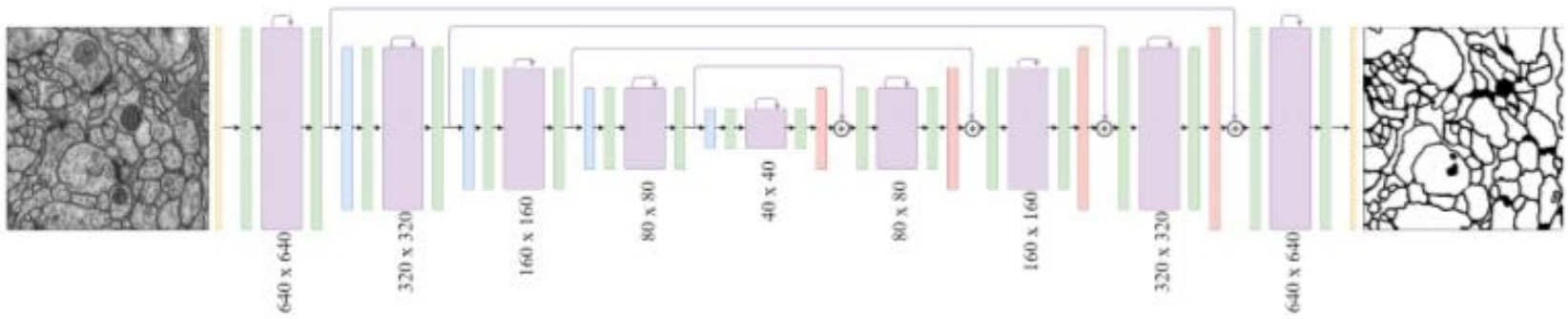


Fig. 2. Overlap-tile strategy for seamless segmentation of arbitrary large images (here segmentation of neuronal structures in EM stacks). Prediction of the segmentation in the yellow area, requires image data within the blue area as input. Missing input data is extrapolated by mirroring

CycleGAN 에서도 Encoder-Decoder로 U-Net을 사용함

Semantic Segmentation

Fusion-Net



Convolutional layer

Downsampling with Maxpooling layer

Residual block with 3 Convolutional layers and 1 skip connection

Upsampling with Deconvolutional layer

FusionNet: A deep fully residual convolutional neural network for image segmentation in connectomics

Semantic Segmentation

Fusion-Net

Semantic Segmentation

Fusion-Net

모듈화!

Semantic Segmentation

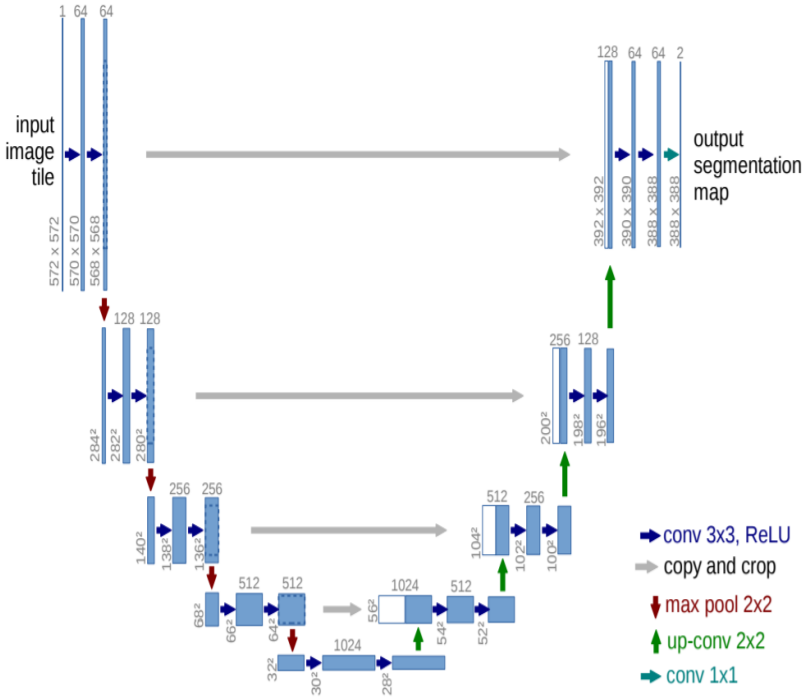
Fusion-Net

```python
def conv_block(in_dim,out_dim,act_fn):
    model = nn.Sequential(
        nn.Conv2d(in_dim,out_dim, kernel_size=3, stride=1, padding=1),
        nn.BatchNorm2d(out_dim),
        act_fn,
    )
    return model


def conv_trans_block(in_dim,out_dim,act_fn):
    model = nn.Sequential(
        nn.ConvTranspose2d(in_dim,out_dim, kernel_size=3, stride=2, padding=1,output_padding=1),
        nn.BatchNorm2d(out_dim),
        act_fn,
    )
    return model


def conv_block_2(in_dim,out_dim,act_fn):
    model = nn.Sequential(
        conv_block(in_dim,out_dim,act_fn),
        nn.Conv2d(out_dim,out_dim, kernel_size=3, stride=1, padding=1),
        nn.BatchNorm2d(out_dim),
    )
    return model


def conv_block_3(in_dim,out_dim,act_fn):
    model = nn.Sequential(
        conv_block(in_dim,out_dim,act_fn),
        conv_block(out_dim,out_dim,act_fn),
        nn.Conv2d(out_dim,out_dim, kernel_size=3, stride=1, padding=1),
        nn.BatchNorm2d(out_dim),
    )
    return model
```

자주 쓰는 모듈, 블록 별로
만들어서 가져다 쓰면 편함

Semantic Segmentation

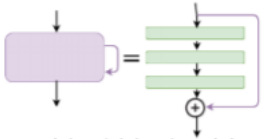
U-Net




```python
from Basic_blocks import *

class UnetGenerator(nn.Module):

    def __init__(self,in_dim,out_dim,num_filter):
        super(UnetGenerator,self).__init__()
        self.in_dim = in_dim
        self.out_dim = out_dim
        self.num_filter = num_filter
        act_fn = nn.LeakyReLU(0.2, inplace=True)

        print("\n------Initiating U-Net------\n")

        self.down_1 = conv_block_2(self.in_dim,self.num_filter,act_fn)
        self.pool_1 = maxpool()
        self.down_2 = conv_block_2(self.num_filter*1,self.num_filter*2,act_fn)
        self.pool_2 = maxpool()
        self.down_3 = conv_block_2(self.num_filter*2,self.num_filter*4,act_fn)
        self.pool_3 = maxpool()
        self.down_4 = conv_block_2(self.num_filter*4,self.num_filter*8,act_fn)
        self.pool_4 = maxpool()

        self.bridge = conv_block_2(self.num_filter*8,self.num_filter*16,act_fn)

        self.trans_1 = conv_trans_block(self.num_filter*16,self.num_filter*8,act_fn)
        self.up_1 = conv_block_2(self.num_filter*16,self.num_filter*8,act_fn)
        self.trans_2 = conv_trans_block(self.num_filter*8,self.num_filter*4,act_fn)
        self.up_2 = conv_block_2(self.num_filter*8,self.num_filter*4,act_fn)
        self.trans_3 = conv_trans_block(self.num_filter*4,self.num_filter*2,act_fn)
        self.up_3 = conv_block_2(self.num_filter*4,self.num_filter*2,act_fn)
        self.trans_4 = conv_trans_block(self.num_filter*2,self.num_filter*1,act_fn)
        self.up_4 = conv_block_2(self.num_filter*2,self.num_filter*1,act_fn)

        self.out = nn.Sequential(
            nn.Conv2d(self.num_filter,self.out_dim,3,1,1),
            nn.Tanh(),
        )
```

Semantic Segmentation

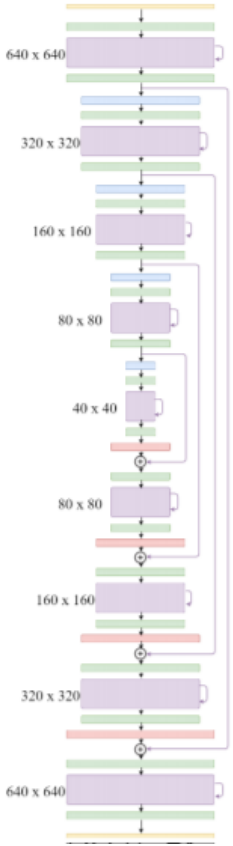
Fusion-Net



Residual block with
3 Convolutional layers
and 1 skip connection

```python
from Basic_blocks import *

class Conv_residual_conv(nn.Module):

    def __init__(self,in_dim,out_dim,act_fn):
        super(Conv_residual_conv,self).__init__()
        self.in_dim = in_dim
        self.out_dim = out_dim
        act_fn = act_fn

        self.conv_1 = conv_block(self.in_dim,self.out_dim,act_fn)
        self.conv_2 = conv_block_3(self.out_dim,self.out_dim,act_fn)
        self.conv_3 = conv_block(self.out_dim,self.out_dim,act_fn)

    def forward(self,input):
        conv_1 = self.conv_1(input)
        conv_2 = self.conv_2(conv_1)
        res = conv_1 + conv_2
        conv_3 = self.conv_3(res)
        return conv_3
```

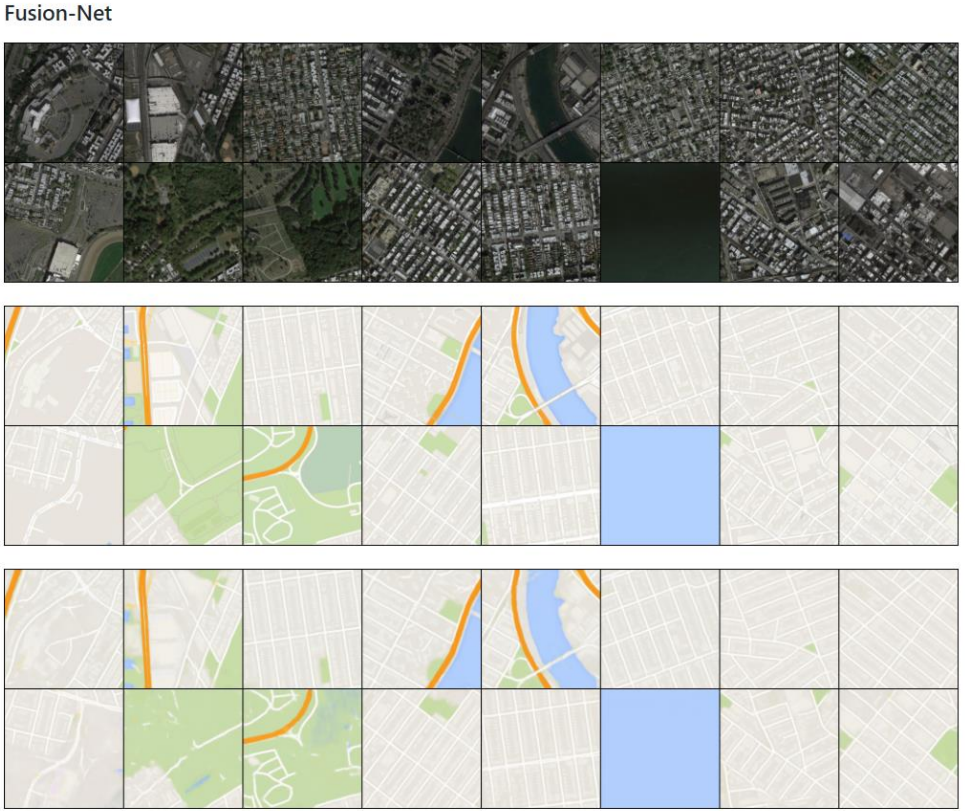
Semantic Segmentation



Fusion-Net

```python
class FusionGenerator(nn.Module):

    def __init__(self,input_nc, output_nc, ngf):
        super(FusionGenerator,self).__init__()
        self.in_dim = input_nc
        self.out_dim = ngf
        self.final_out_dim = output_nc
        act_fn = nn.LeakyReLU(0.2, inplace=True)
        act_fn_2 = nn.ReLU()

        print("\n------Initiating FusionNet------\n")

        # encoder

        self.down_1 = Conv_residual_conv(self.in_dim, self.out_dim, act_fn)
        self.pool_1 = maxpool()
        self.down_2 = Conv_residual_conv(self.out_dim, self.out_dim * 2, act_fn)
        self.pool_2 = maxpool()
        self.down_3 = Conv_residual_conv(self.out_dim * 2, self.out_dim * 4, act_fn)
        self.pool_3 = maxpool()
        self.down_4 = Conv_residual_conv(self.out_dim * 4, self.out_dim * 8, act_fn)
        self.pool_4 = maxpool()

        # bridge

        self.bridge = Conv_residual_conv(self.out_dim * 8, self.out_dim * 16, act_fn)

        # decoder

        self.deconv_1 = conv_trans_block(self.out_dim * 16, self.out_dim * 8, act_fn_2)
        self.up_1 = Conv_residual_conv(self.out_dim * 8, self.out_dim * 8, act_fn_2)
        self.deconv_2 = conv_trans_block(self.out_dim * 8, self.out_dim * 4, act_fn_2)
        self.up_2 = Conv_residual_conv(self.out_dim * 4, self.out_dim * 4, act_fn_2)
        self.deconv_3 = conv_trans_block(self.out_dim * 4, self.out_dim * 2, act_fn_2)
        self.up_3 = Conv_residual_conv(self.out_dim * 2, self.out_dim * 2, act_fn_2)
        self.deconv_4 = conv_trans_block(self.out_dim * 2, self.out_dim, act_fn_2)
        self.up_4 = Conv_residual_conv(self.out_dim, self.out_dim, act_fn_2)

        # output

        self.out = nn.Conv2d(self.out_dim,self.final_out_dim, kernel_size=3, stride=1, padding=1)
        self.out_2 = nn.Tanh()
```

Semantic Segmentation

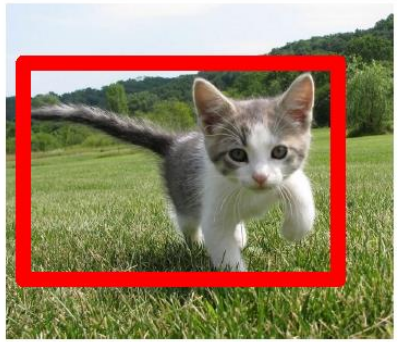


혹시라도 코드가 궁금하신 분은 아래의 링크에 가시면 u-net, fusion-net 둘 다 있습니다.

https://github.com/GunhoChoi/Kind-PyTorch-Tutorial/tree/master/12_Semantic_Segmentation

Classification & Localization

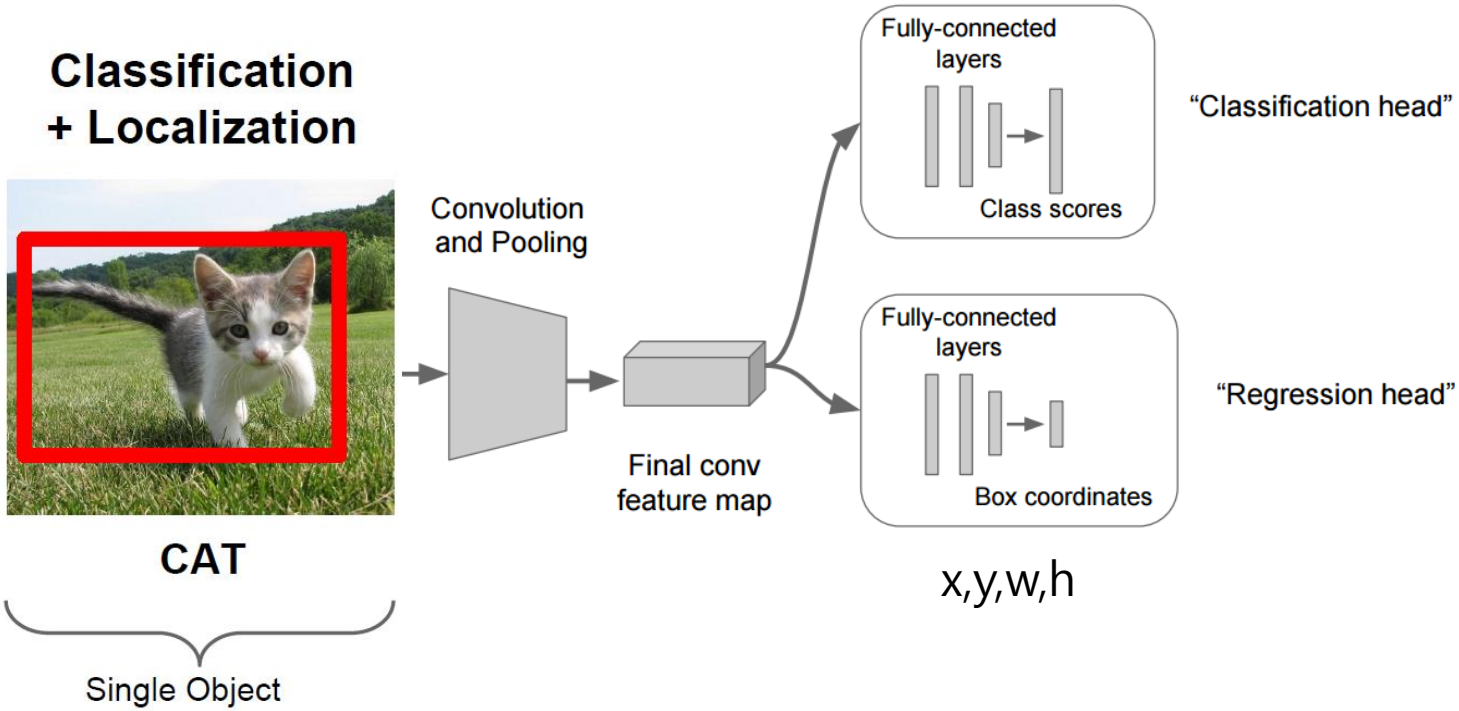
Classification
+ Localization



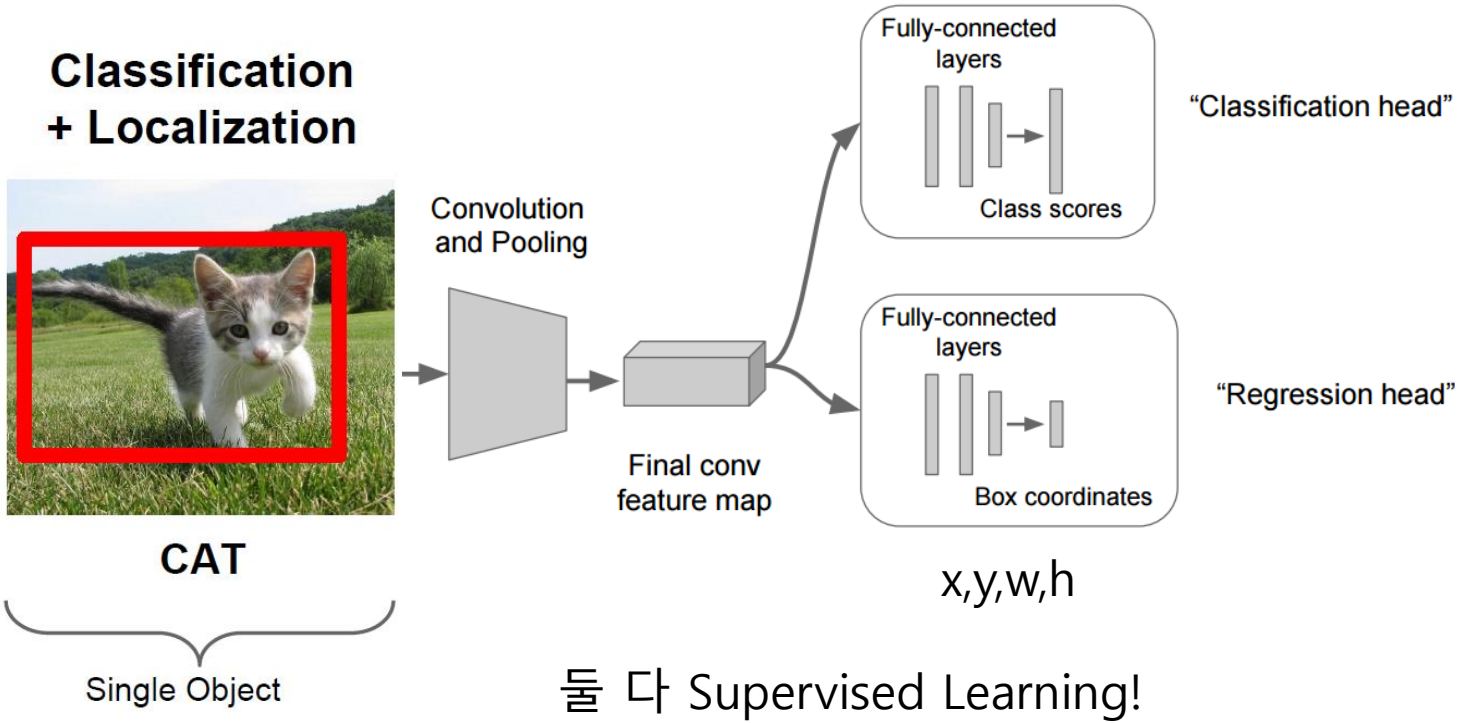
CAT

Single Object

Classification & Localization

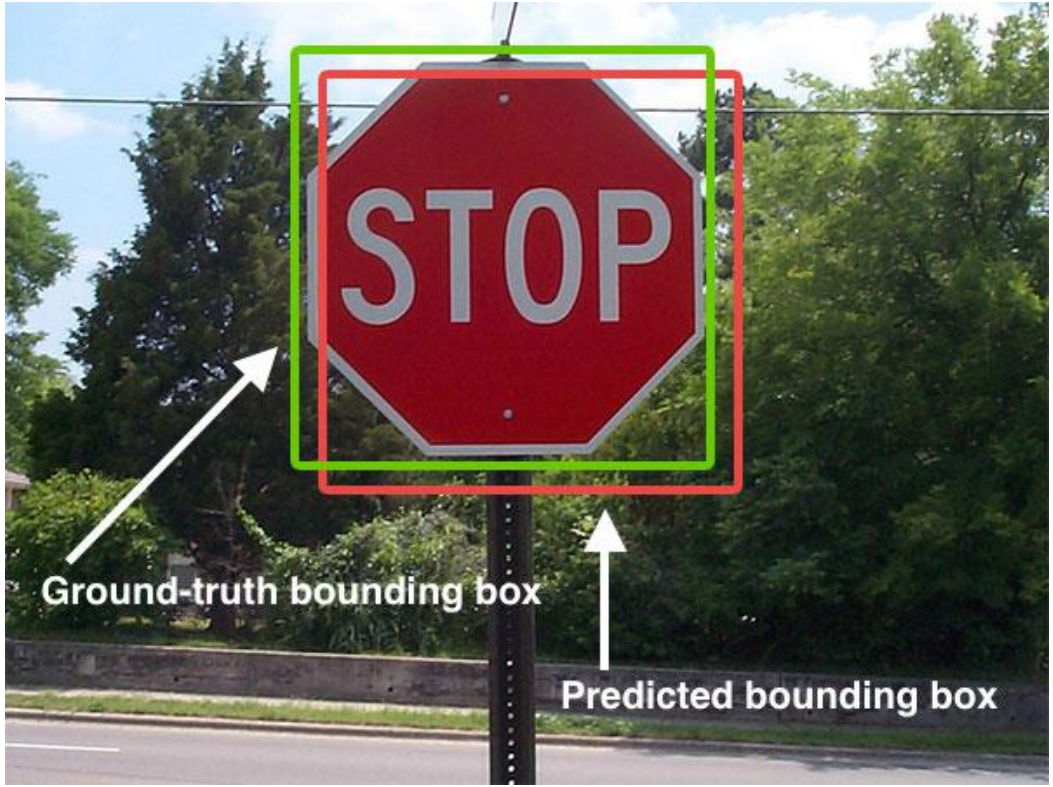


Classification & Localization



둘 다 Supervised Learning!
데이터만 있다면 행복하게 모델을 만들 수 있다.

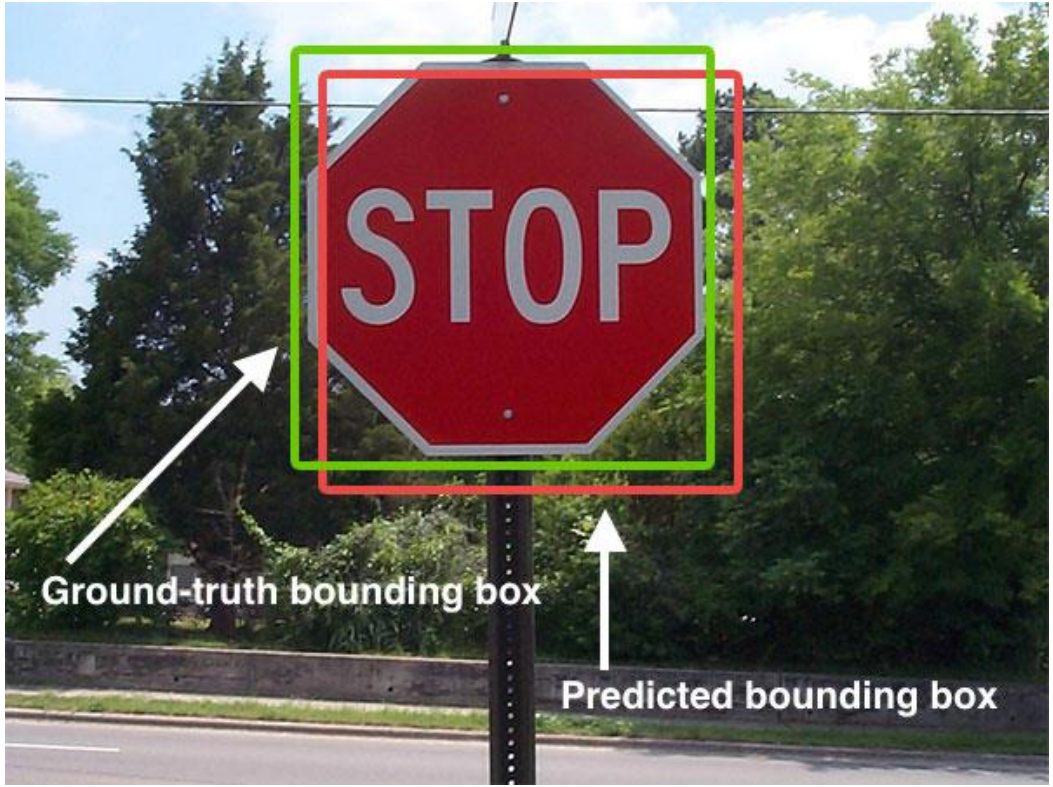
Classification & Localization



Localization이나 Detection을 보다 보면 IOU라는게 자주 등장하는데 이게 뭘까?

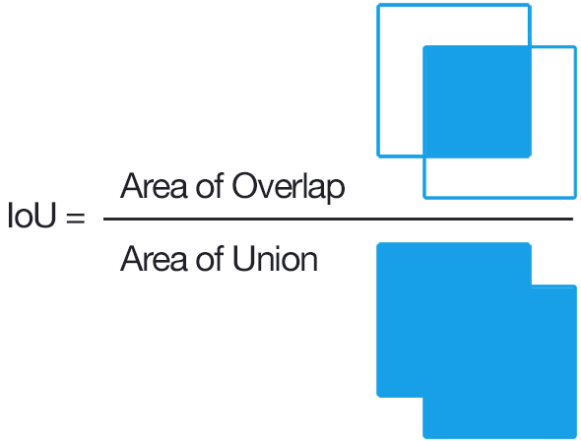
http://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

Classification & Localization



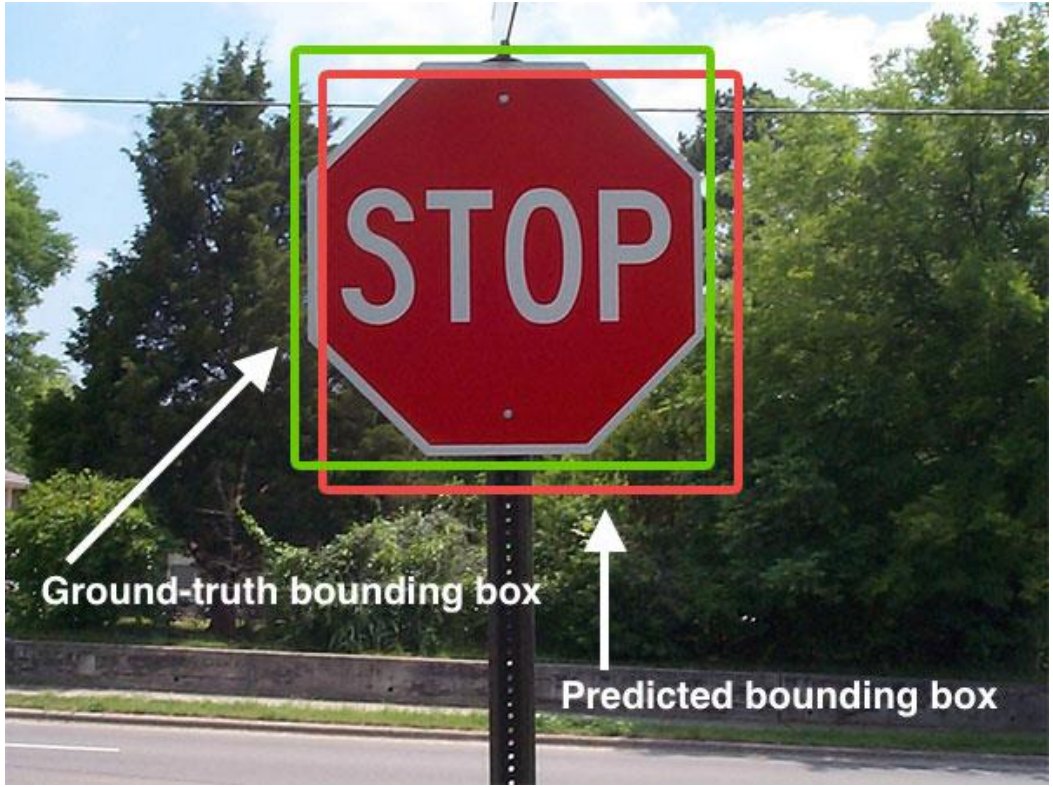
Localization이나 Detection을 보다 보면 IOU라는게 자주 등장하는데 이게 뭘까?

IOU = Intersection Over Union



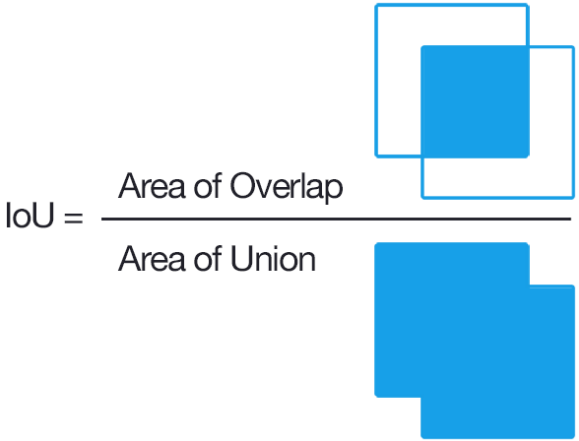
http://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

Classification & Localization



Localization이나 Detection을 보다 보면 IOU라는게 자주 등장하는데 이게 뭘까?

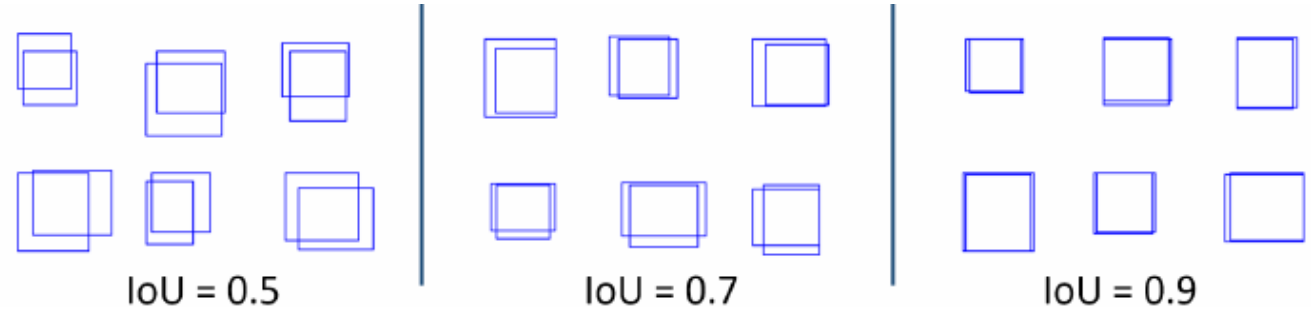
IOU = Intersection Over Union



성능측정 메트릭

http://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

Classification & Localization



맞고 틀리고의 Threshold 는 정하기 나름인데
보통 0.5를 기준으로 많이 사용함

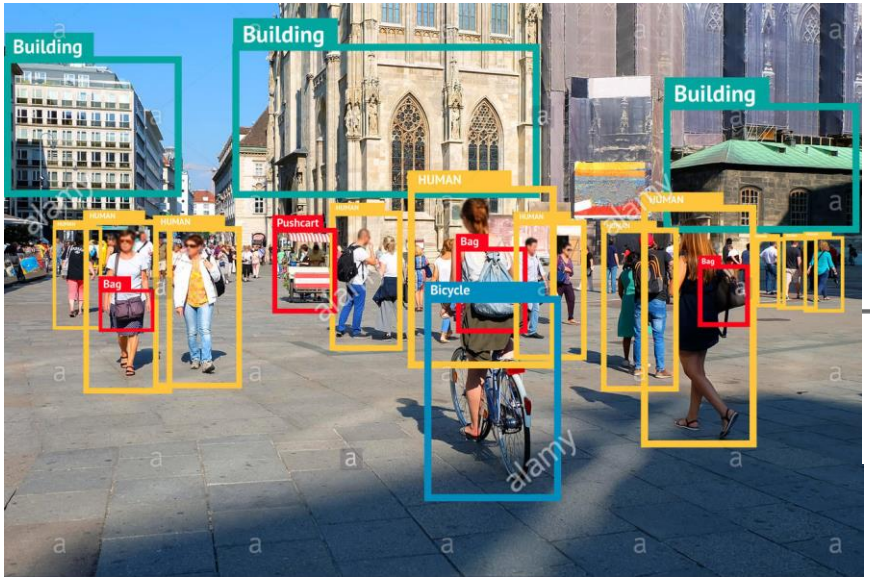
http://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

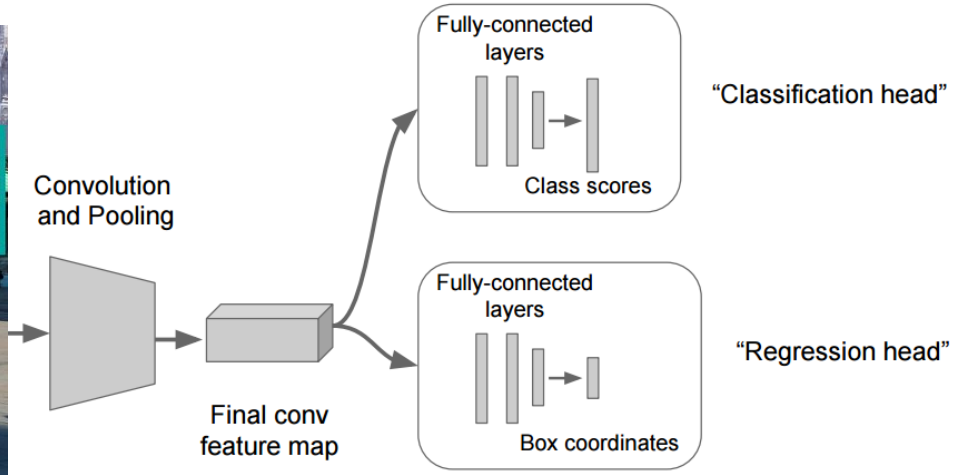
Classification & Localization

[x,y,w,h] 외에도 Pose Estimation도 같은 방식으로 가능하다

http://www.cs.cmu.edu/~ILIM/projects/IM/humanpose/sample_detection_leeds.png

Classification & Localization

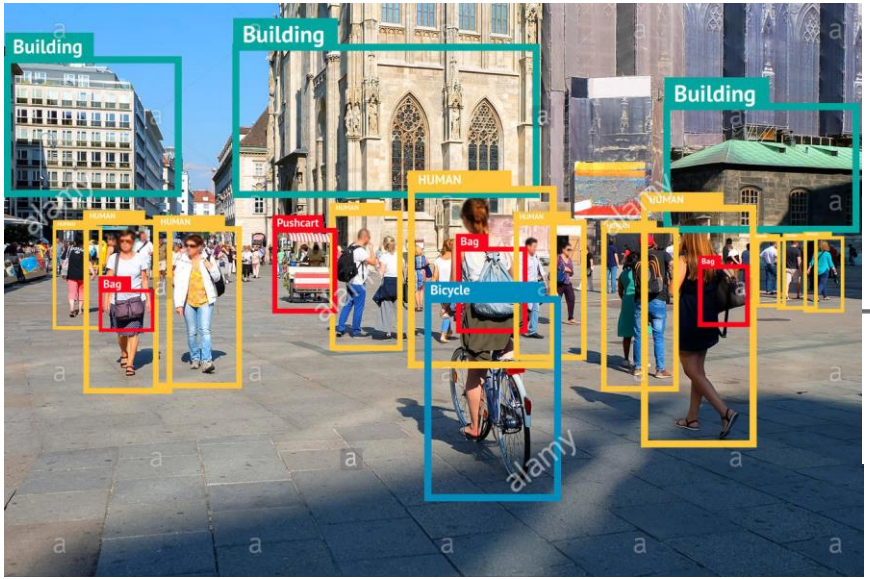


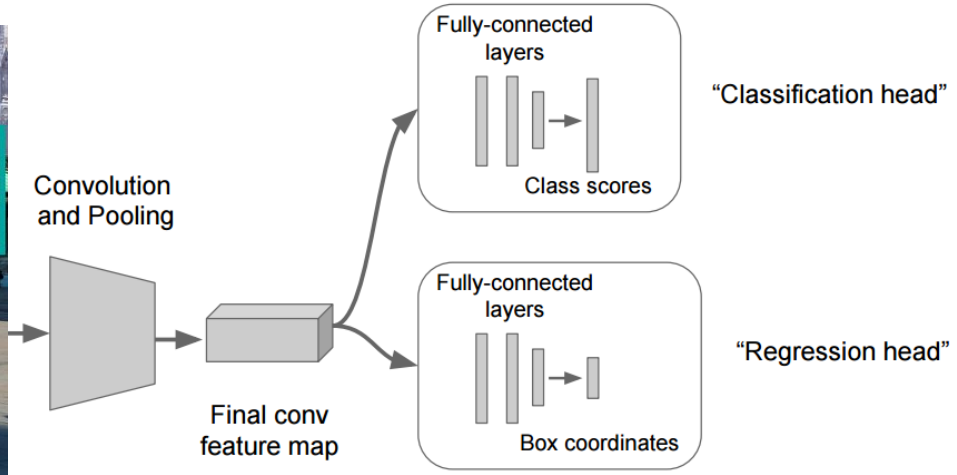


만약 몇 개의 물체가 있는지 모른다면
몇 개의 head를 사용해야 할까??

http://www.alamy.com/stock-photo-machine-learning-object-detection-and-artificial-intelligence-concept-140769075.html

Classification & Localization





만약 몇 개의 물체가 있는지 모른다면
몇 개의 head를 사용해야 할까??

아예 다른 방식이 필요함
Object Detection!

http://www.alamy.com/stock-photo-machine-learning-object-detection-and-artificial-intelligence-concept-140769075.html

Object Detection

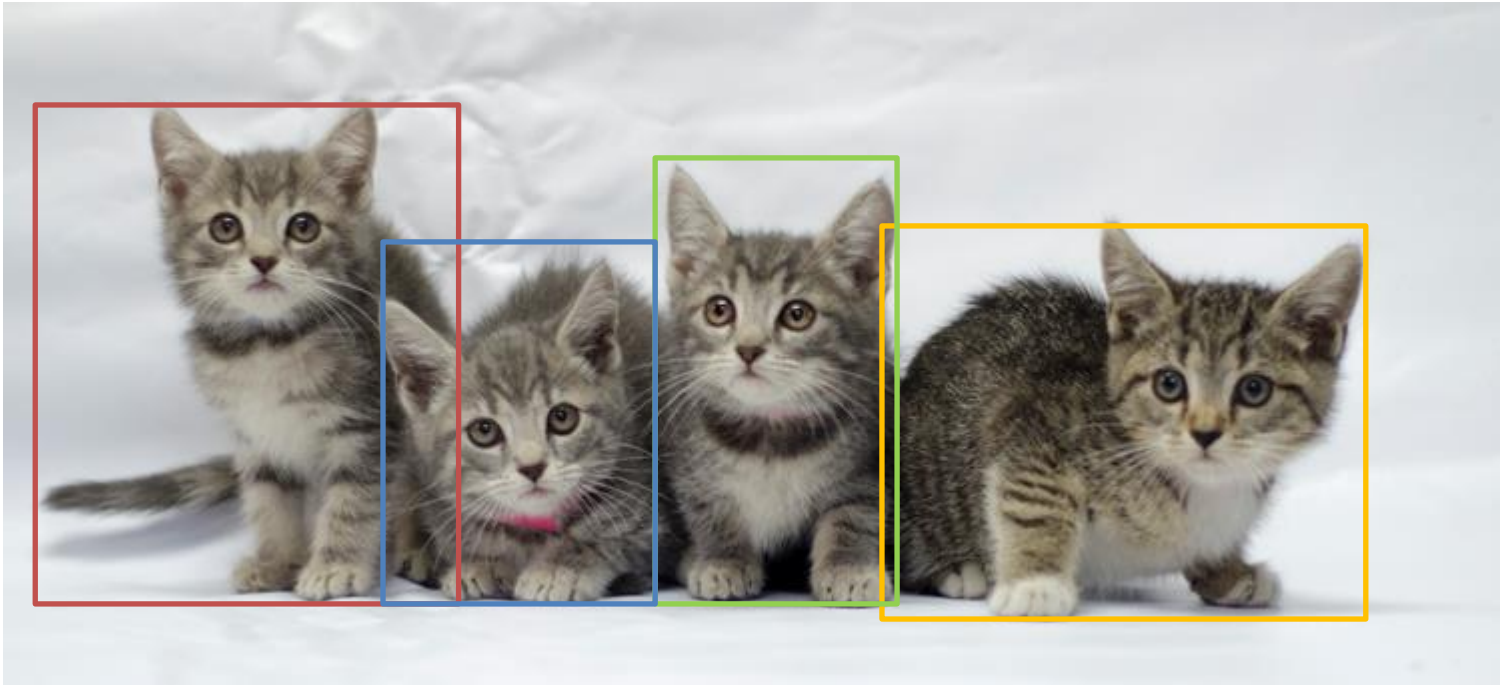
Sliding Window

쭉 지나가면서 어떤 클래스에 해당하는지 Classify

https://www.washingtonpost.com/blogs/local/files/2014/10/4-kittens1.jpg

Object Detection

Sliding Window



하지만 각 객체들의 크기도 다르고 들어오는 가능한 모든 입력 크기마다 모델을 학습시키는 것은 Naïve 한 접근방식이어서 아무도 안 씀

https://www.washingtonpost.com/blogs/local/files/2014/10/4-kittens1.jpg

Object Detection

물체가 있을법한 곳을 먼저 찾자

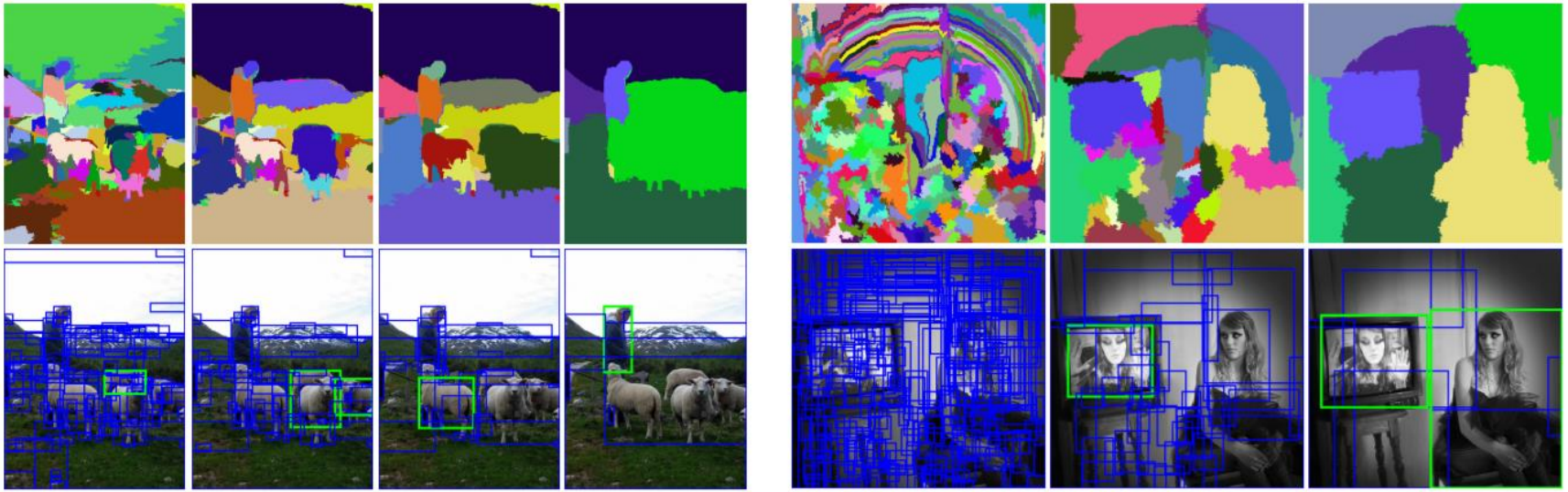


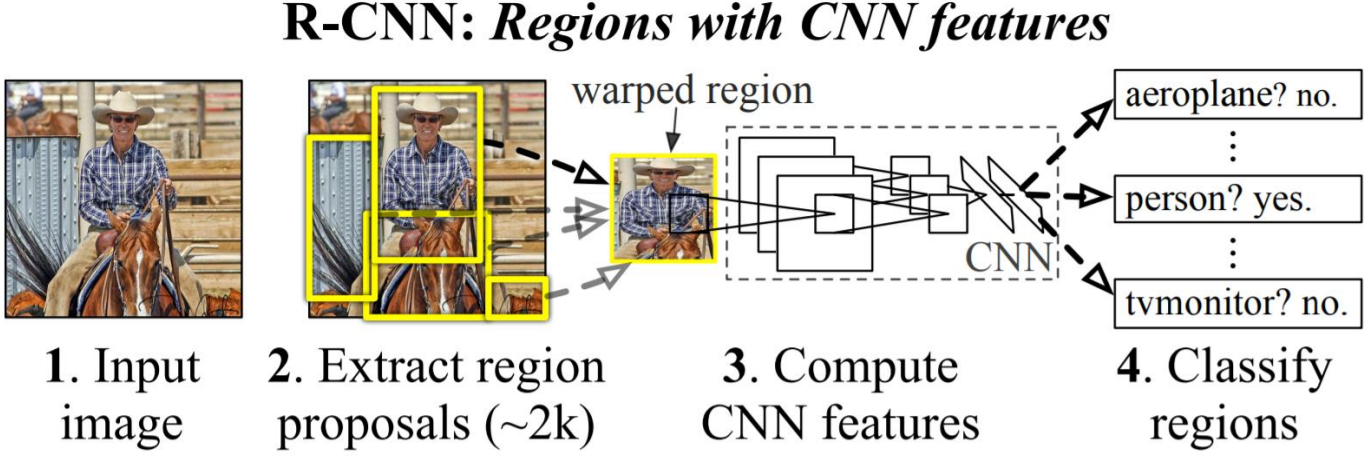
Figure 2: Two examples of our selective search showing the necessity of different scales. On the left we find many objects at different scales. On the right we necessarily find the objects at different scales as the girl is contained by the tv.

Bottom-up 방식으로 일정한 개수의 그럴듯한 region들을 Proposal 하는 방식

출처:Segmentation as Selective Search for Object Recognition

Object Detection

Selective Search 로 뽑힌 Region들을 input으로 받는 모델을 만들어보자



출처: Rich feature hierarchies for accurate object detection and semantic segmentation

Object Detection

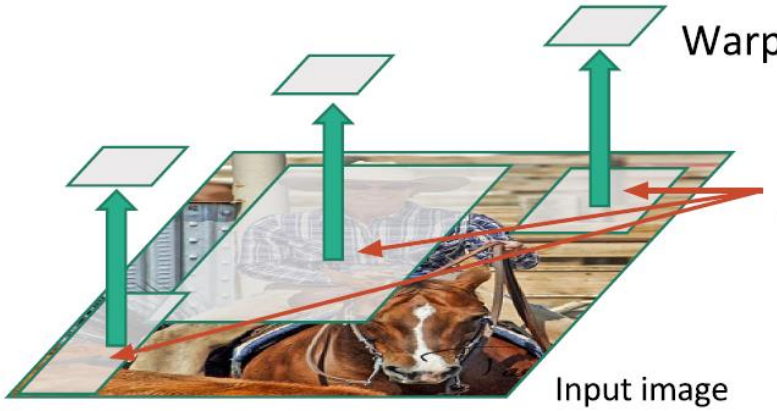
Input image

Object Detection

Selective Search로 Region 뽑기

출처: CS231n_2017_Lecture_11_slides

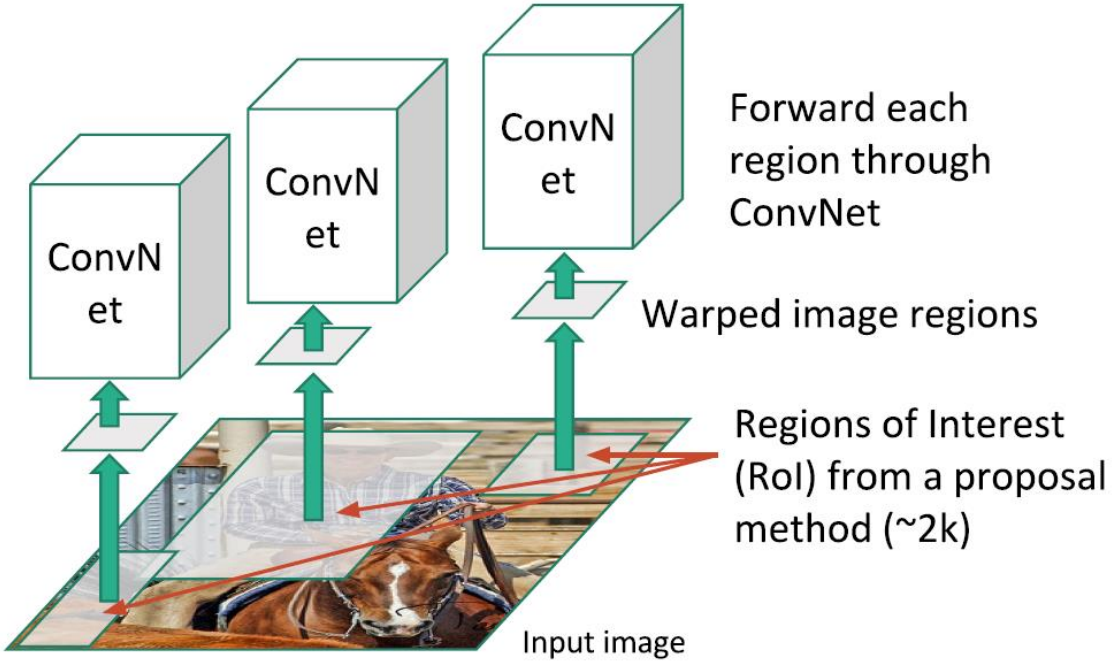
Object Detection



하나의 네트워크에 들어가야 하므로
크기를 동일하게 warp

출처: CS231n_2017_Lecture_11_slides

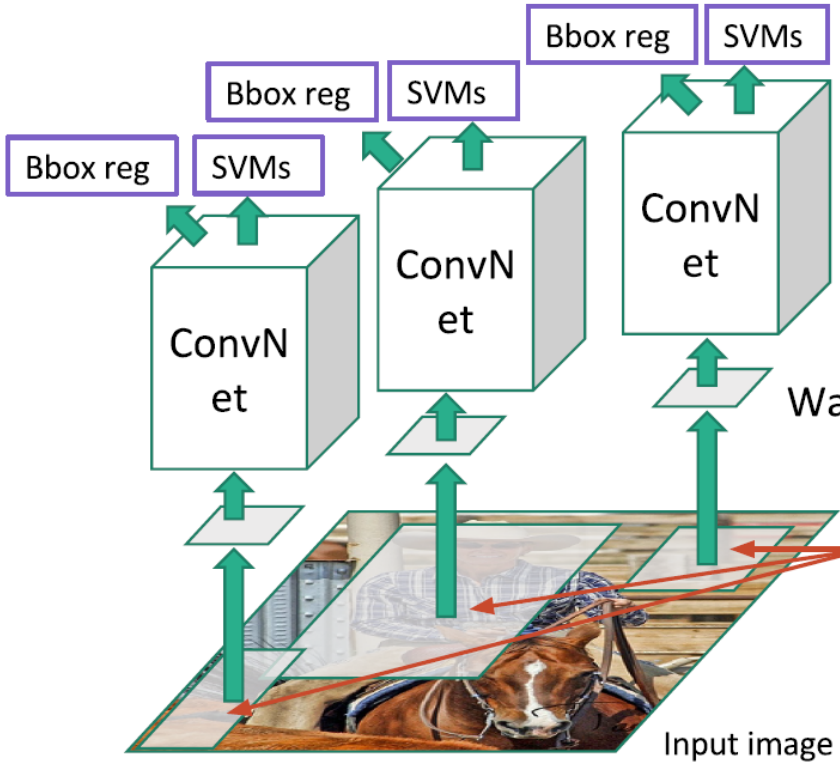
Object Detection



모델에 전달

출처: CS231n_2017_Lecture_11_slides

Object Detection



Linear Regression for bounding box offsets

Classify regions with SVMs

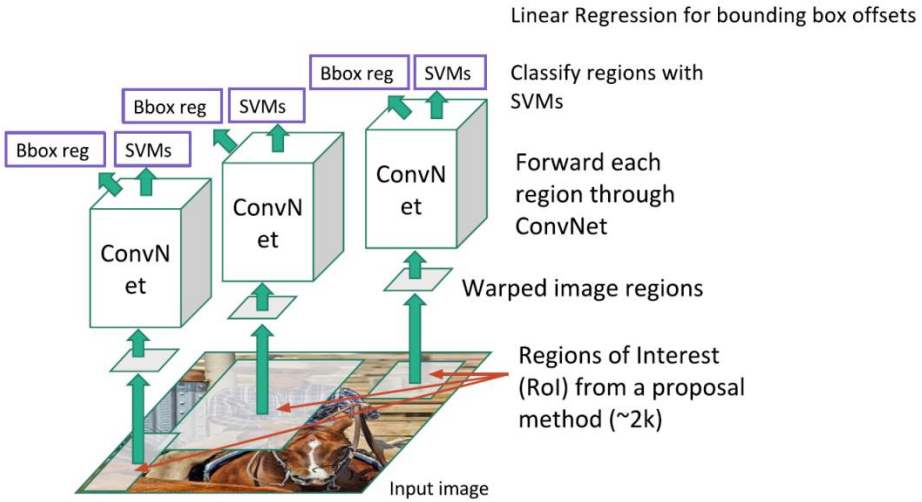
Forward each region through ConvNet

Bounding Box가 정확하지 않을 수 있기 때문에 고쳐주는 부분이 Bbox reg에 해당

SVM으로 클래스에 대해서 학습

출처: CS231n_2017_Lecture_11_slides

Object Detection



<R-CNN의 문제점>

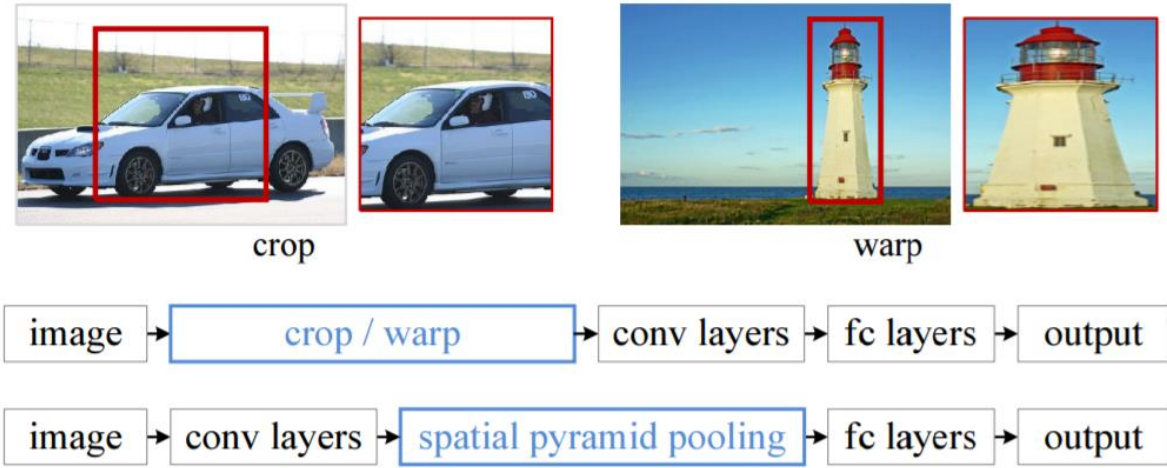
1. 학습도 느리고 test-time에도 느림
2. 2000개의 이미지를 저장해야 하므로 용량도 많이 차지함
3. Warp된 이미지를 전달하면서 형태가 많이 바뀌기도 함

Object Detection

Figure 1: Top: cropping or warping to fit a fixed size. Middle: a conventional CNN. Bottom: our spatial pyramid pooling network structure.

출처: Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition

Object Detection



Warp한 일정 크기 이미지를 전달

Figure 1: Top: cropping or warping to fit a fixed size. Middle: a conventional CNN. Bottom: our spatial pyramid pooling network structure.

출처: Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition

Object Detection

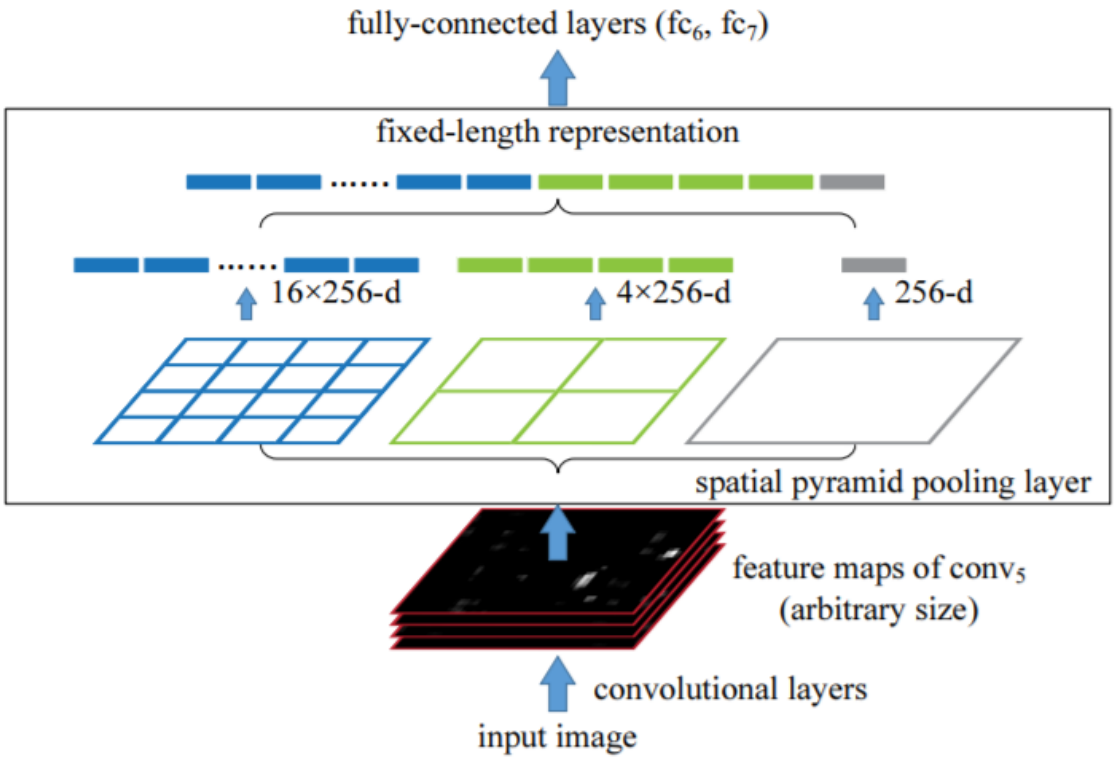
Figure 1: Top: cropping or warping to fit a fixed size. Middle: a conventional CNN. Bottom: our spatial pyramid pooling network structure.

Warp한 일정 크기 이미지를 전달

먼저 CNN 모델을 통과한 이후 Spatial Pyramid Pooling을 통해 일정 크기로 맞춰줌

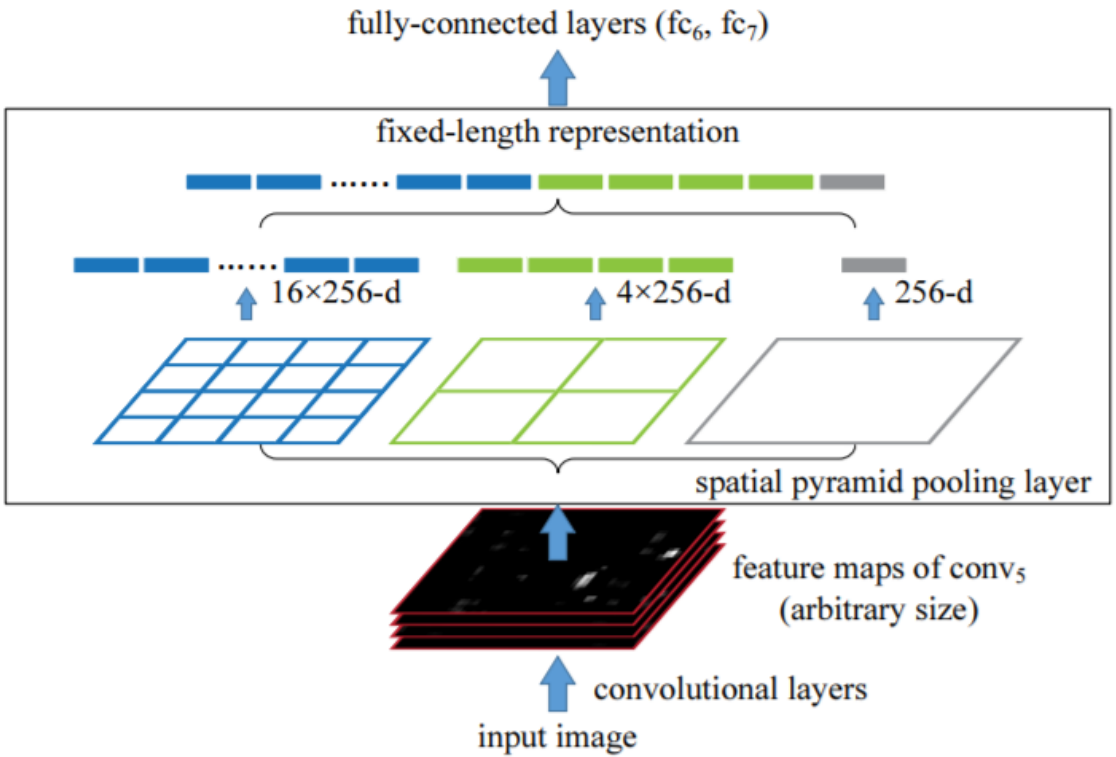
출처: Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition

Object Detection



출처: Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition

Object Detection

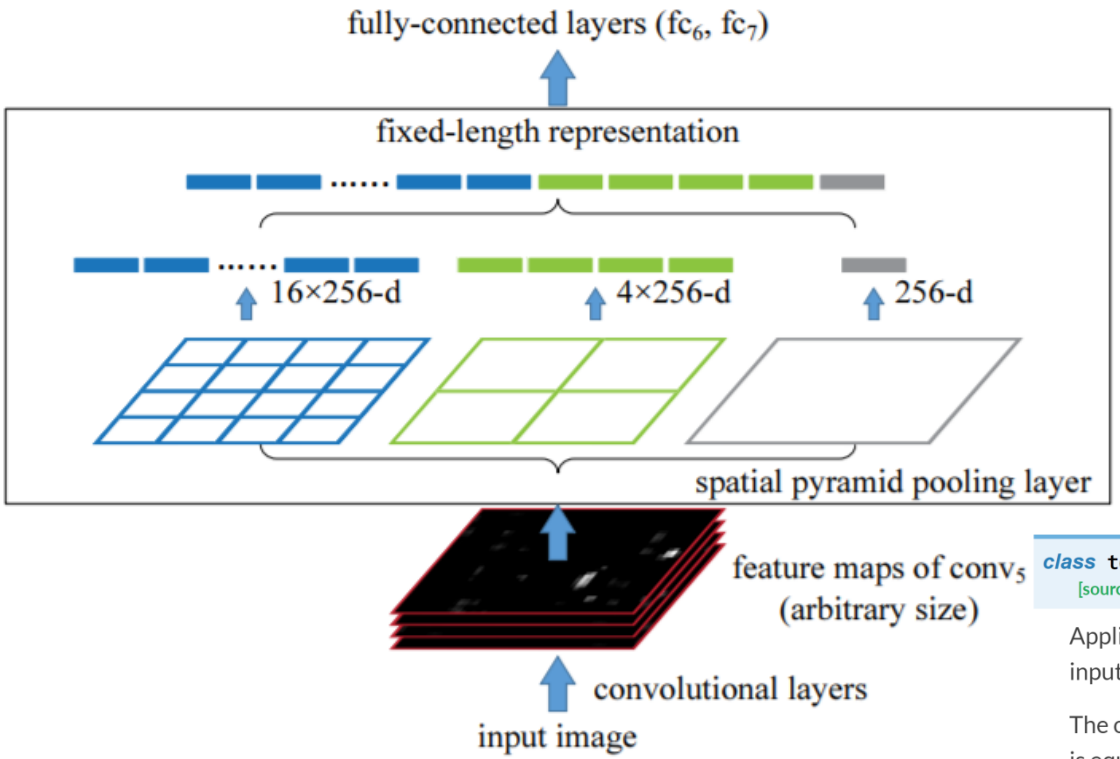


어떠한 크기의 input이 들어와도 output의 크기가 일정하도록 pool -ing을 해줌으로써 fully connected layer를 사용할 수 있게 하는 방법

출처: Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition

Object Detection

SPPNet



어떠한 크기의 input이 들어와도 output의 크기가 일정하도록 pool -ing을 해줌으로써 fully connected layer를 사용할 수 있게 하는 방법

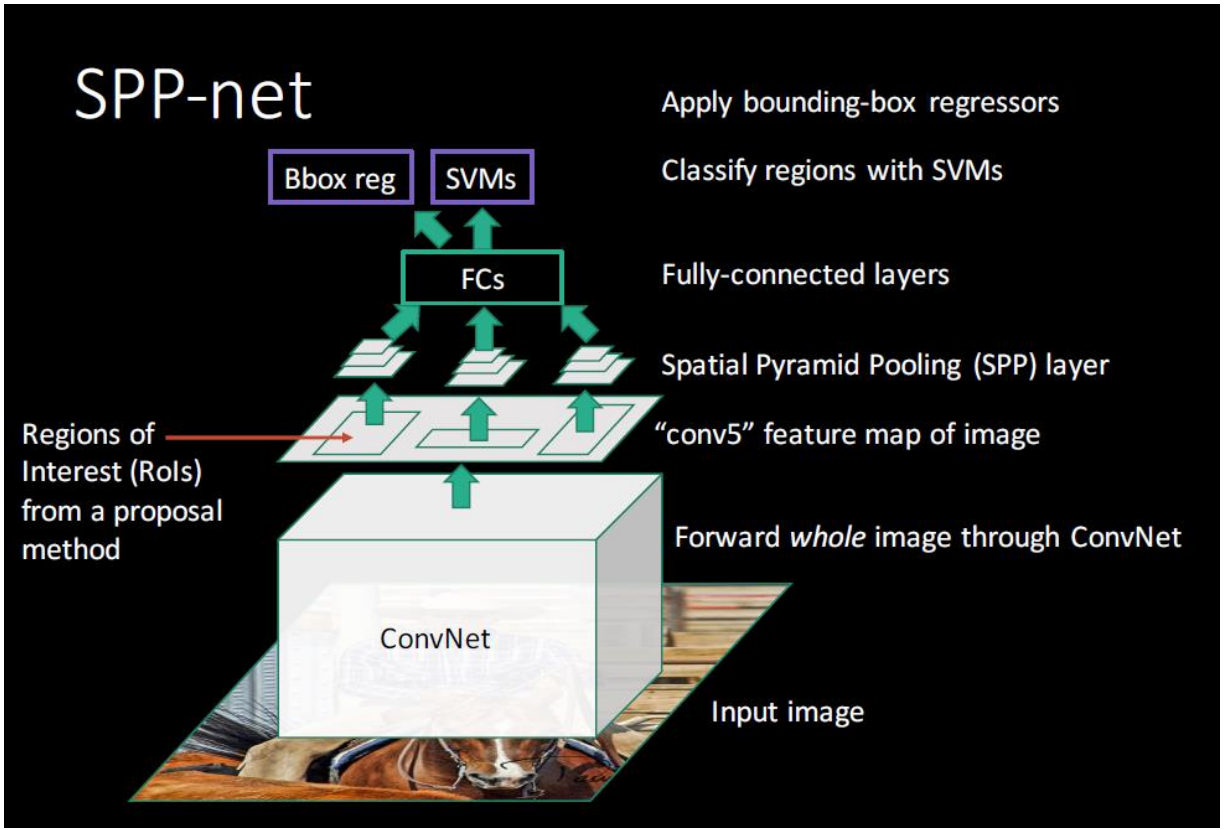
class `torch.nn.AdaptiveMaxPool2d`(*output_size, return_indices=False*)
[source]

Applies a 2D adaptive max pooling over an input signal composed of several input planes.

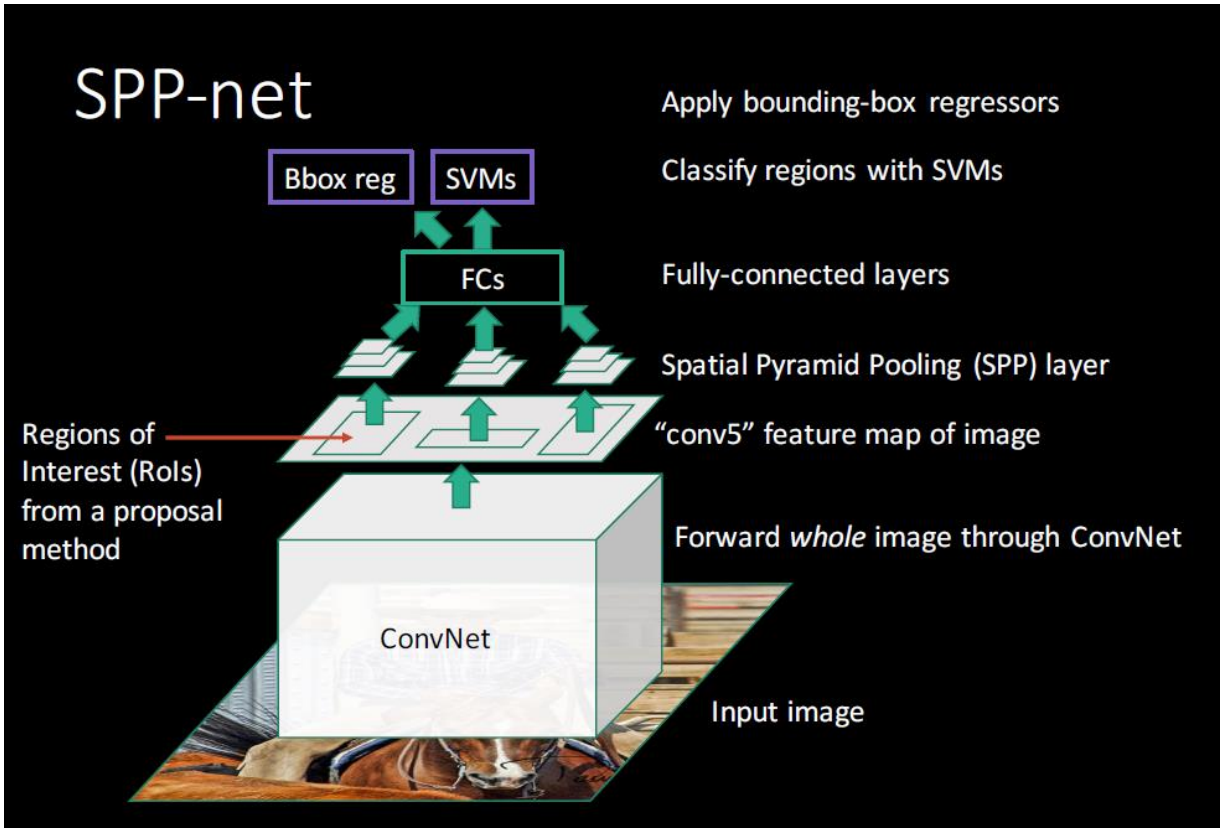
The output is of size H x W, for any input size. The number of output features is equal to the number of input planes.

출처: Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition

Object Detection

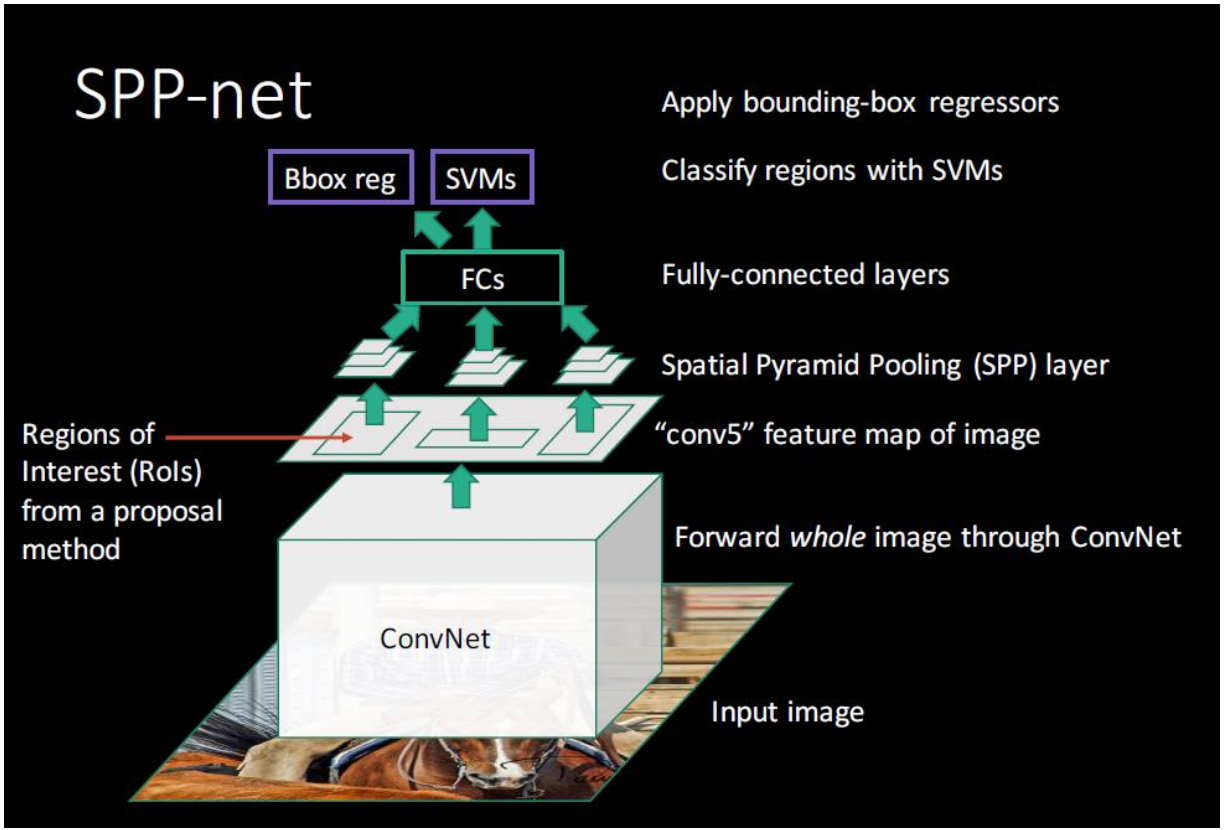


Object Detection



CNN 네트워크가 공유되기 때문에 빨라짐

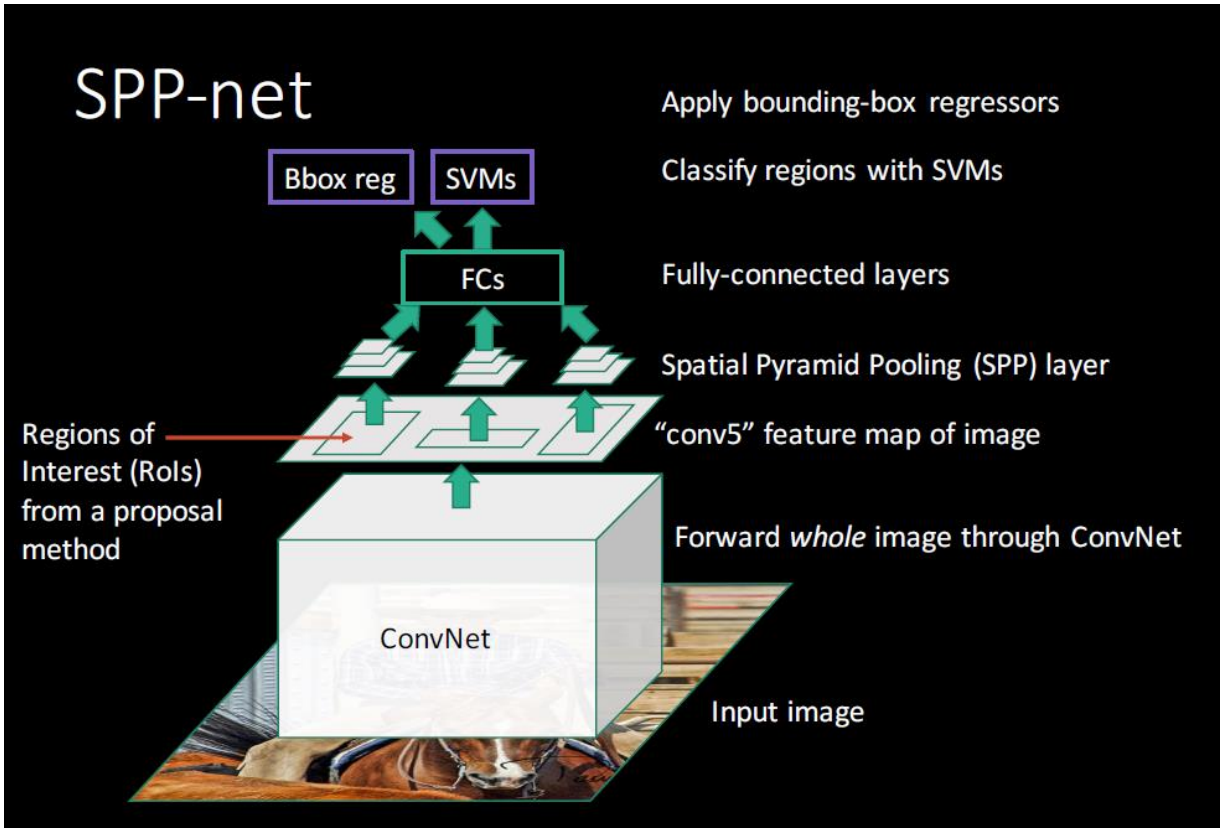
Object Detection



CNN 네트워크가 공유되기 때문에 빨라짐

하지만 Region Proposal에 있어서는 아직 Selective Search를 사용함

Object Detection

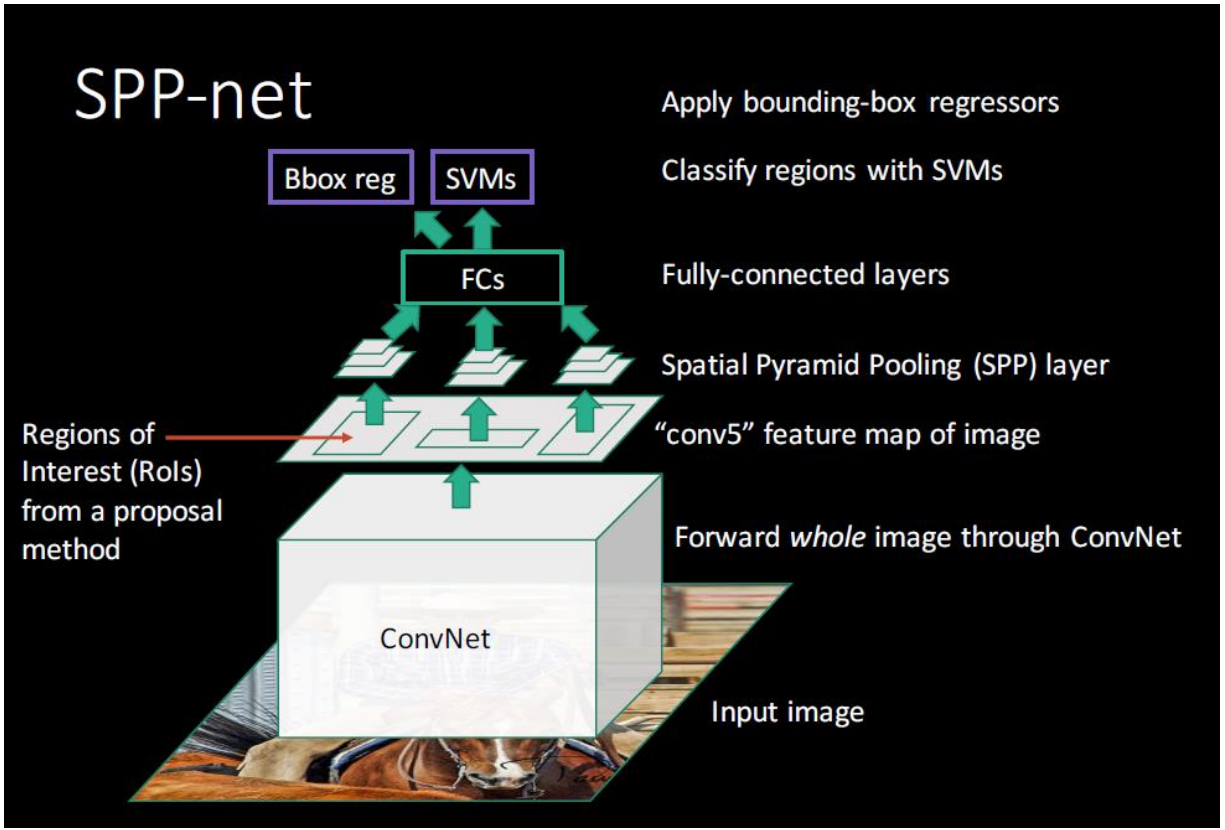


CNN 네트워크가 공유되기 때문에 빨라짐

하지만 Region Proposal에 있어서는 아직 Selective Search를 사용함

중간에 Region Proposal이 끼기 때문에 end-to-end가 아님

Object Detection



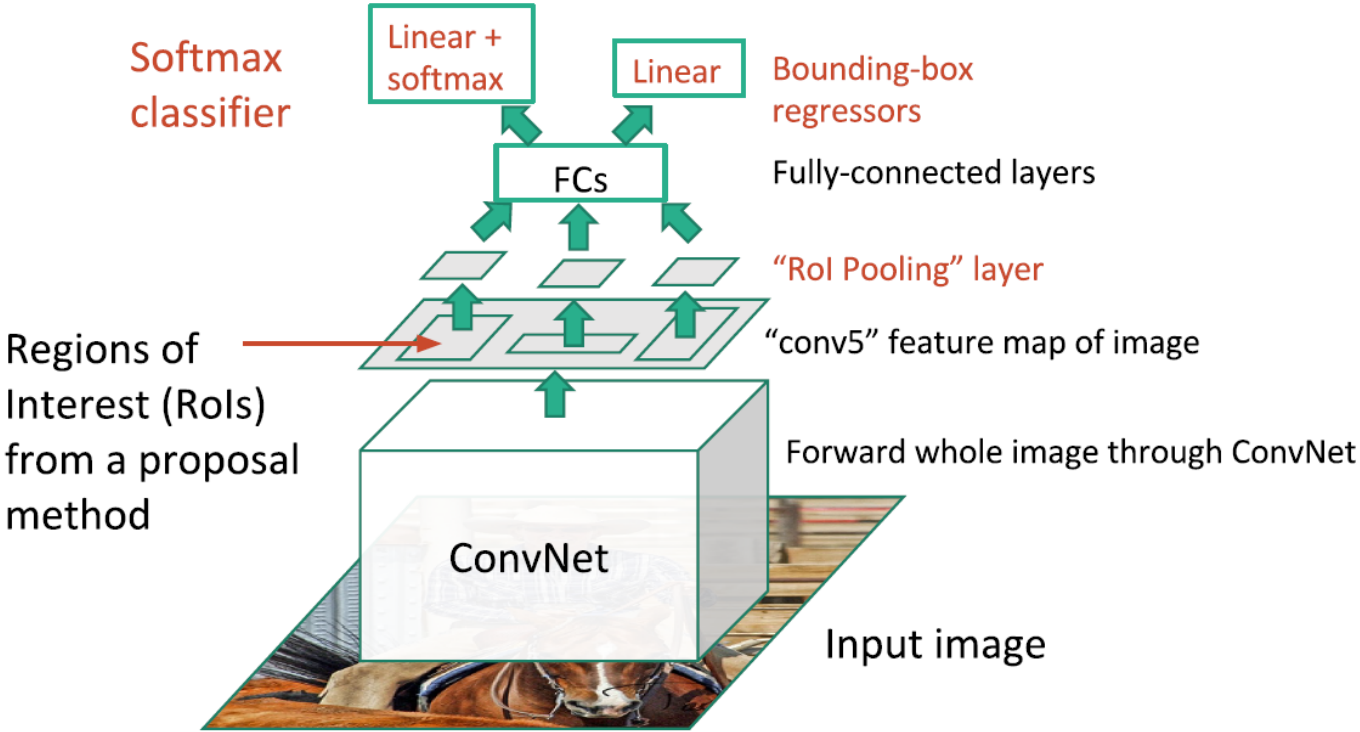
CNN 네트워크가 공유되기 때문에 빨라짐

하지만 Region Proposal에 있어서는 아직 Selective Search를 사용함

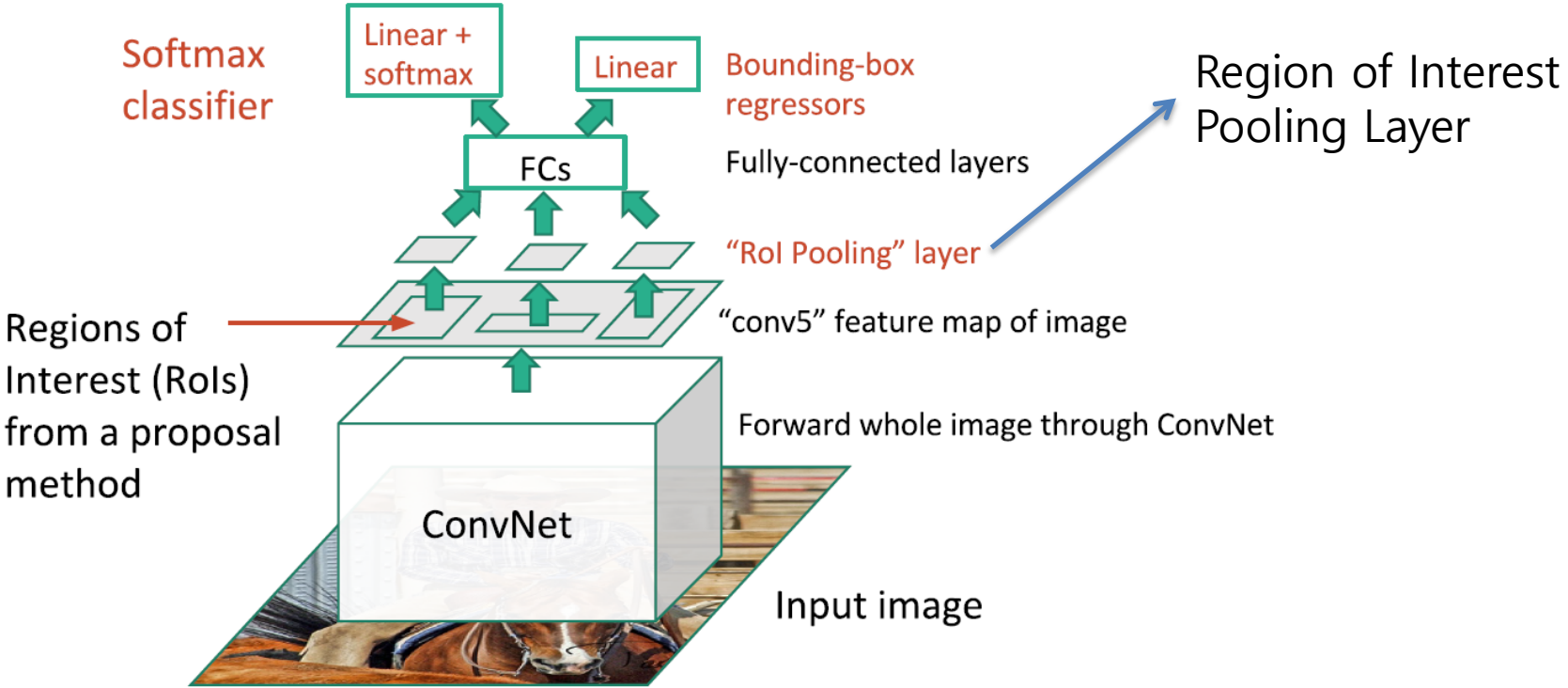
중간에 Region Proposal이 끼기 때문에 end-to-end가 아님

Fast R-CNN도 SPPNet과 비슷한 방식

Object Detection

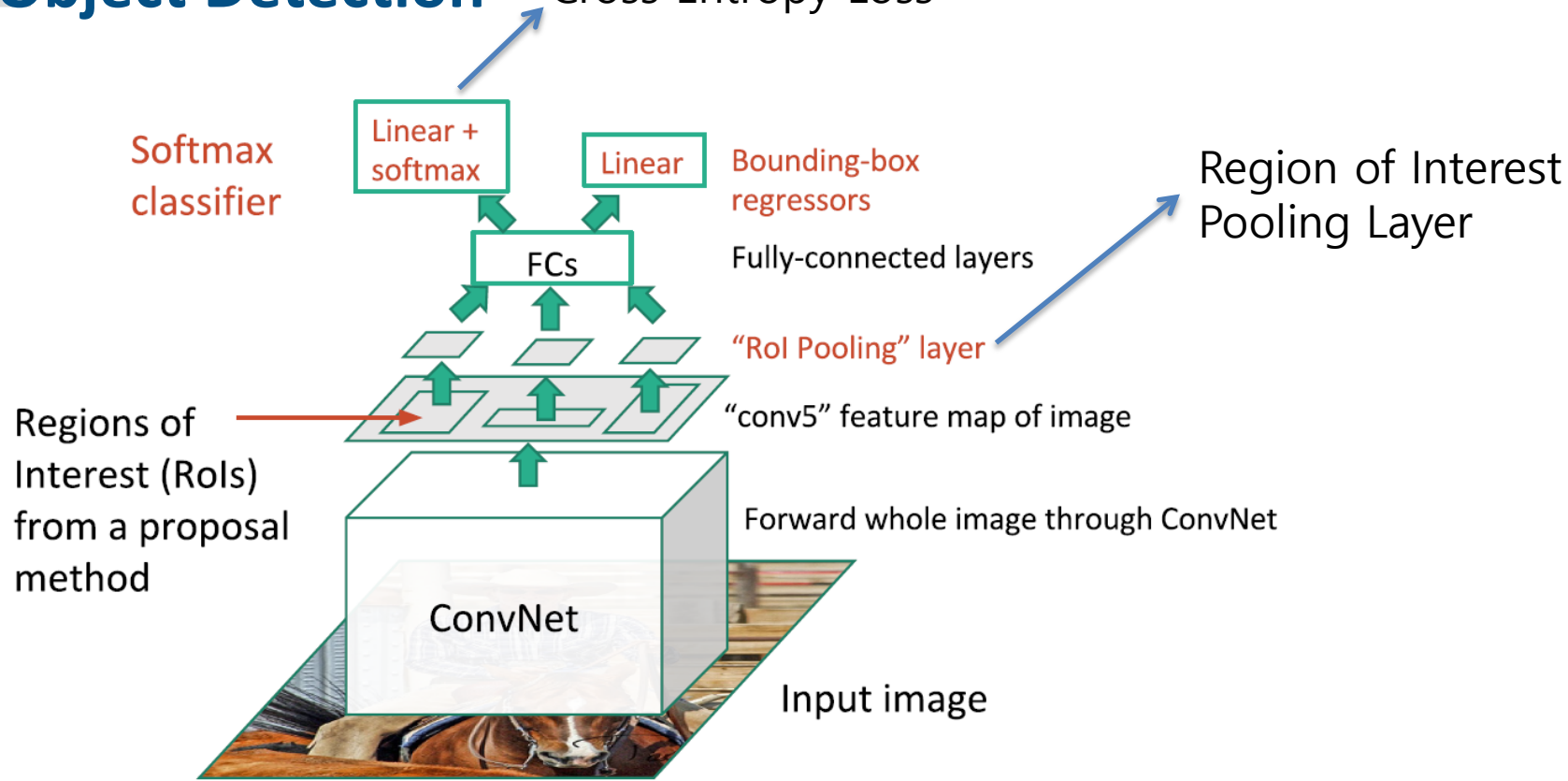


Object Detection



Object Detection

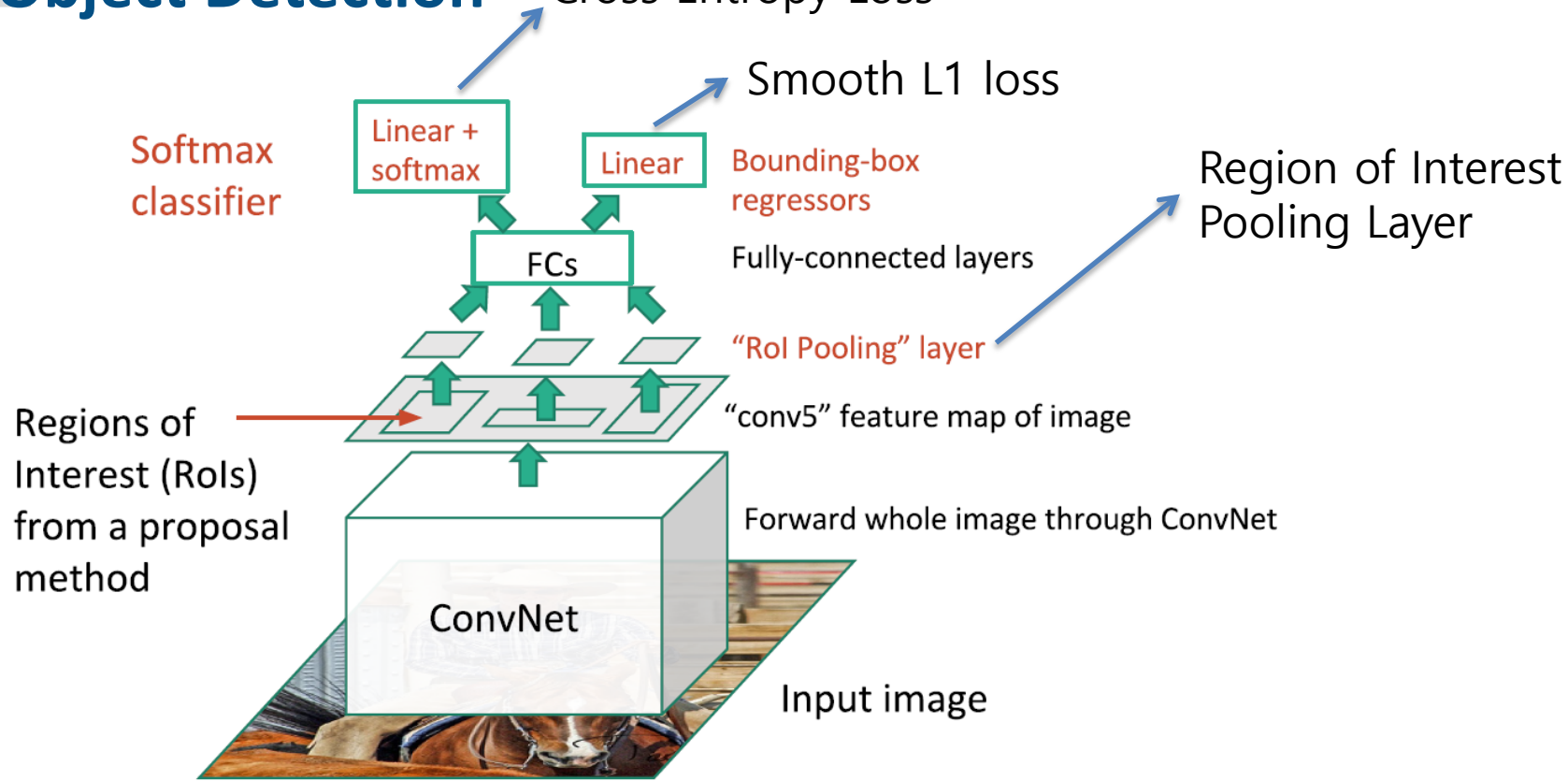
Fast R-CNN



출처: CS231n_2017_Lecture_11_slides

Object Detection

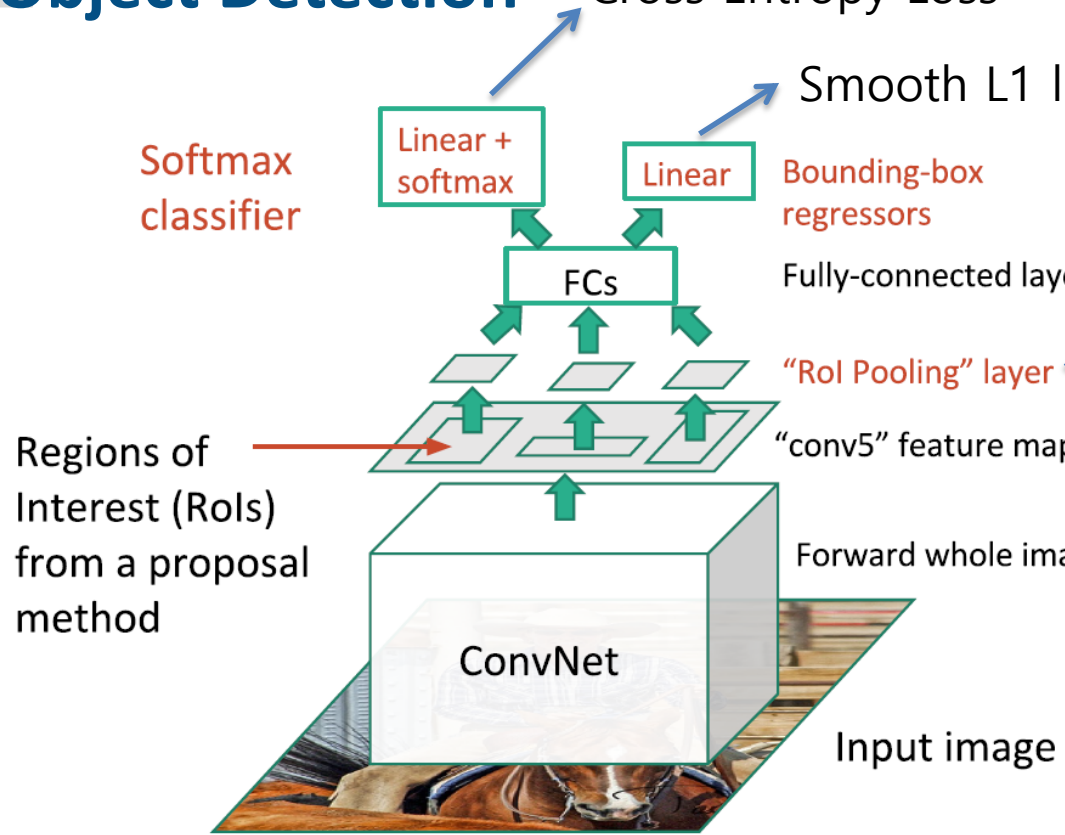
Fast R-CNN



출처: CS231n_2017_Lecture_11_slides

Object Detection

Fast R-CNN



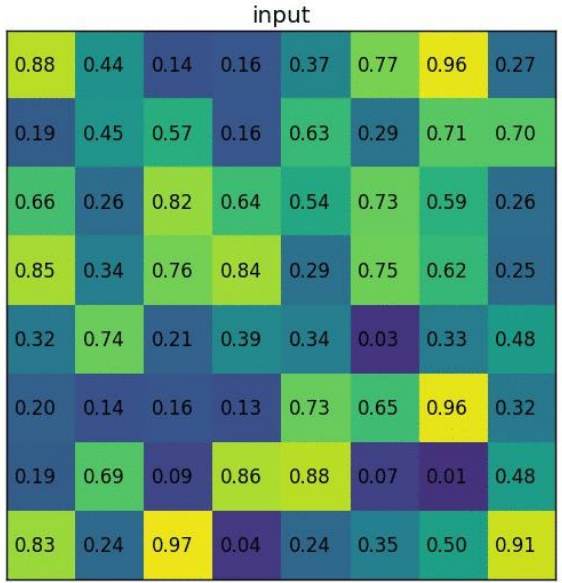
$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases}$$

Smooth L1는 outlier에 덜 민감함

출처: CS231n_2017_Lecture_11_slides

Object Detection

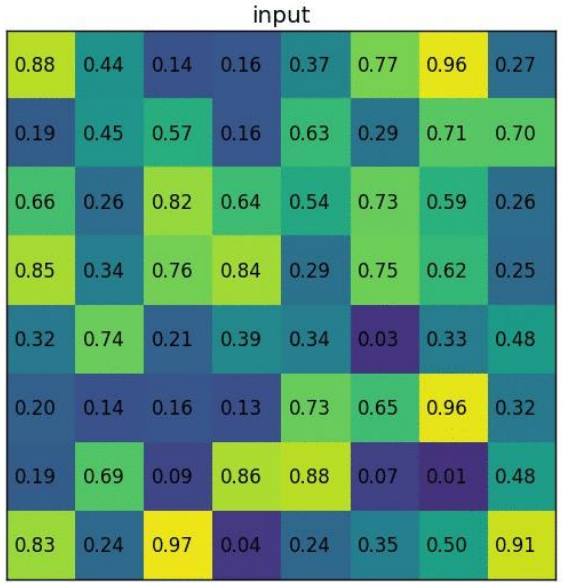
ROI pooling Layer의 동작 원리



사실 Spatial Pyramid Pooling과 비슷함

Object Detection

ROI pooling Layer의 동작 원리

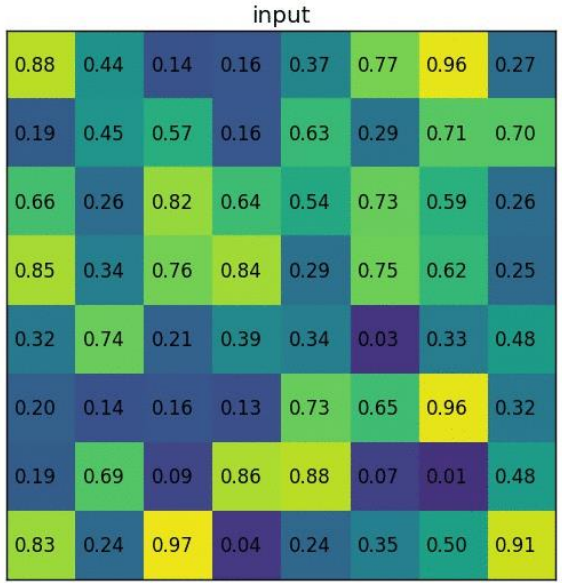


사실 Spatial Pyramid Pooling과 비슷함

CNN을 통과한 Feature Map과 Selective search 를 통해 proposed된 지역이 있으면 원하는 크기대로 Pooling해서 일정한 크기로 전달

Object Detection

ROI pooling Layer의 동작 원리



사실 Spatial Pyramid Pooling과 비슷함

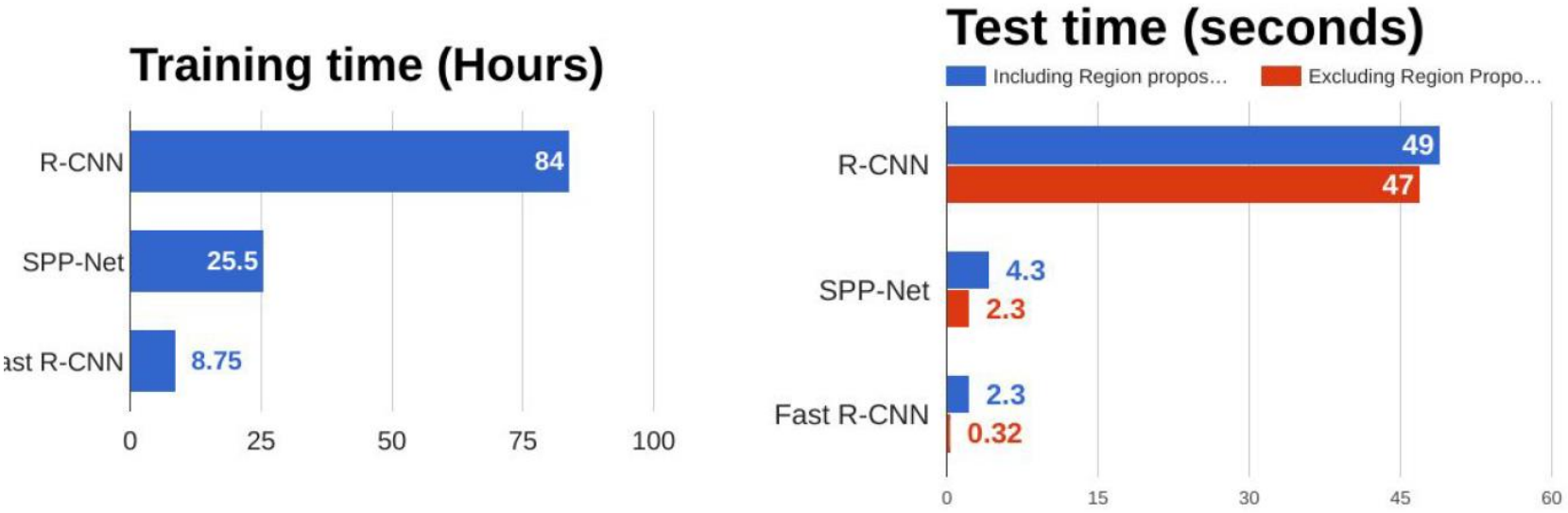
CNN을 통과한 Feature Map과 Selective search 를 통해 proposed된 지역이 있으면 원하는 크기대로 Pooling해서 일정한 크기로 전달

예시는 8x8 짜리 feature map과 5x7 짜리 proposed region이 전달되었을 때, ROI pool -ing 으로 2x2짜리 map을 뽑는 과정

https://blog.deepsense.ai/region-of-interest-pooling-explained/

Object Detection

R-CNN vs SPP vs Fast R-CNN

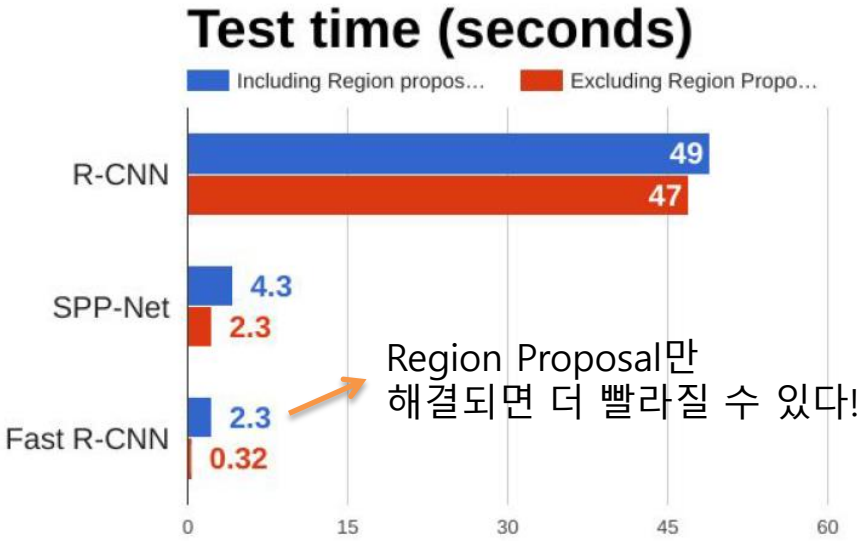


Fast R-CNN은 SVD(Single Value Decomposition) 같은 것도 추가적으로 사용함

출처: CS231n_2017_Lecture_11_slides

Object Detection

R-CNN vs SPP vs Fast R-CNN

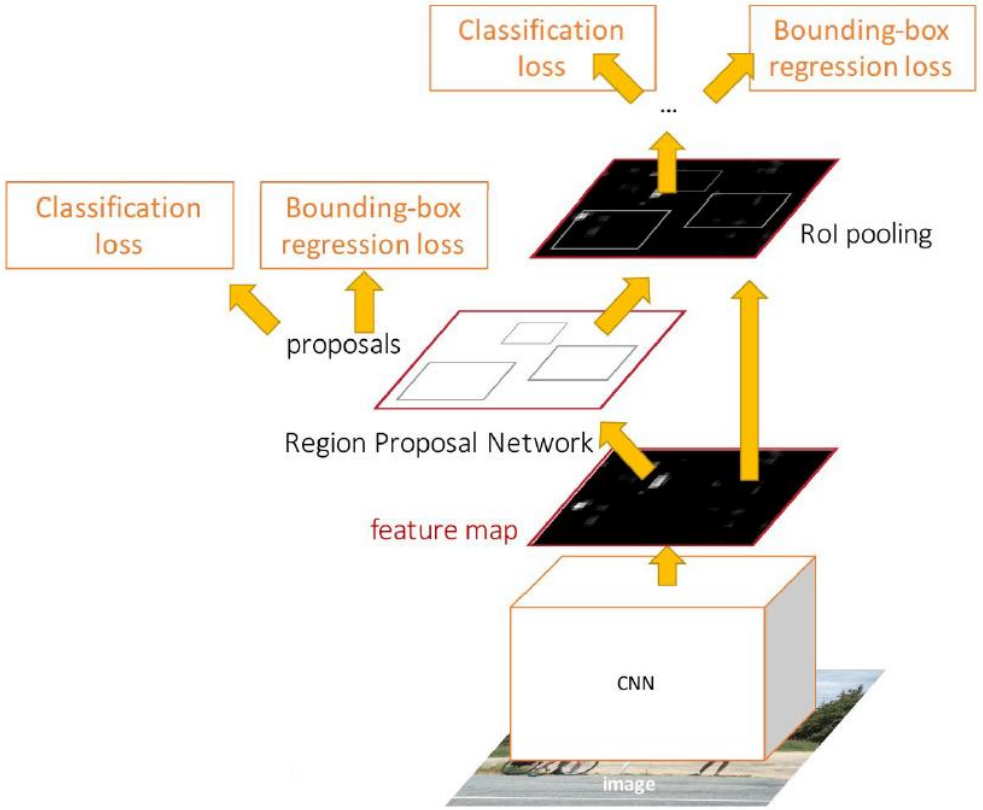


Fast R-CNN은 SVD(Single Value Decomposition) 같은 것도 추가적으로 사용함

출처: CS231n_2017_Lecture_11_slides

Object Detection

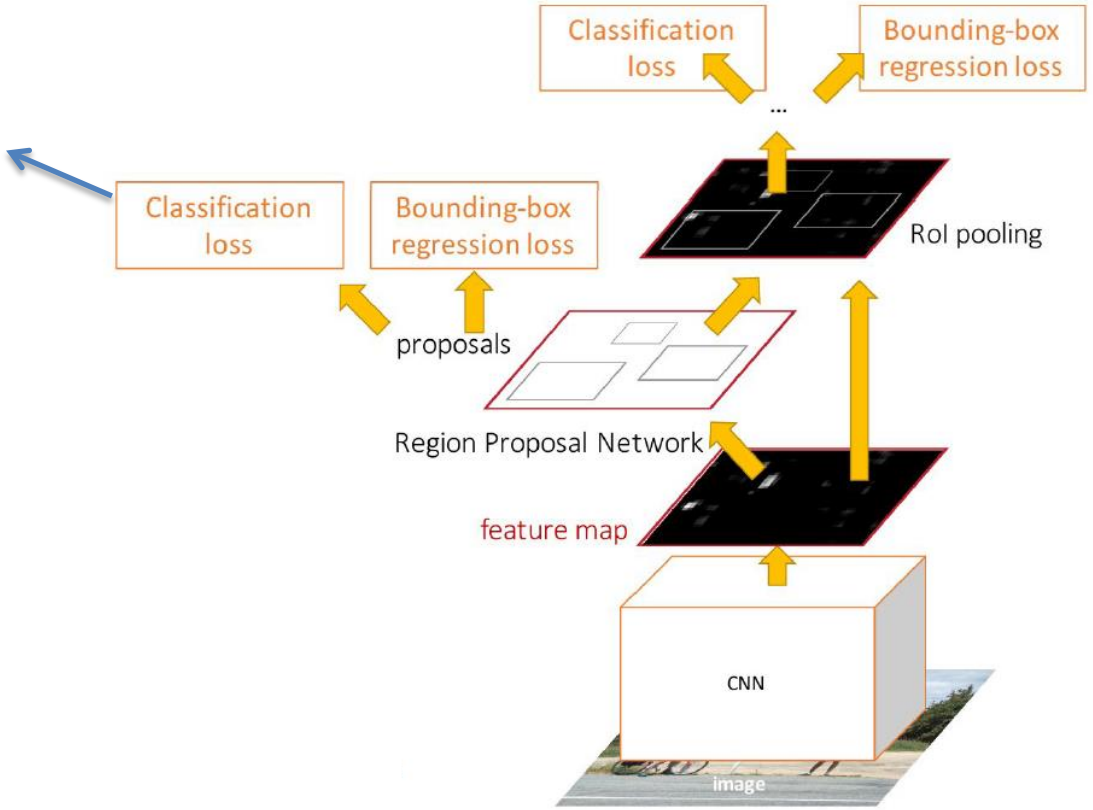
Faster R-CNN



출처: CS231n_2017_Lecture_11_slides

Object Detection

Region Proposal도 Selective Search 대신 Network로 해결해보자!

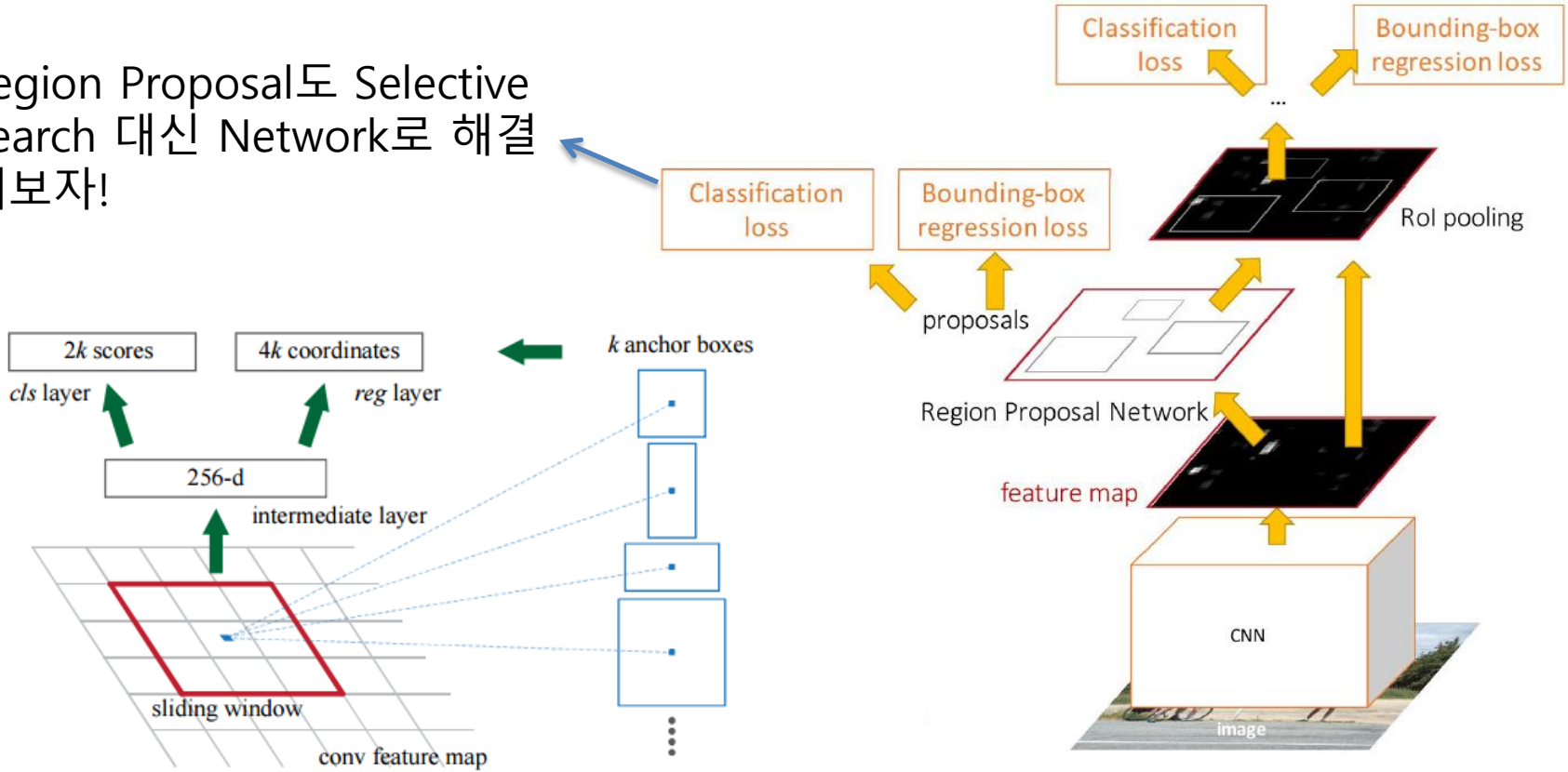


출처: CS231n_2017_Lecture_11_slides

Object Detection

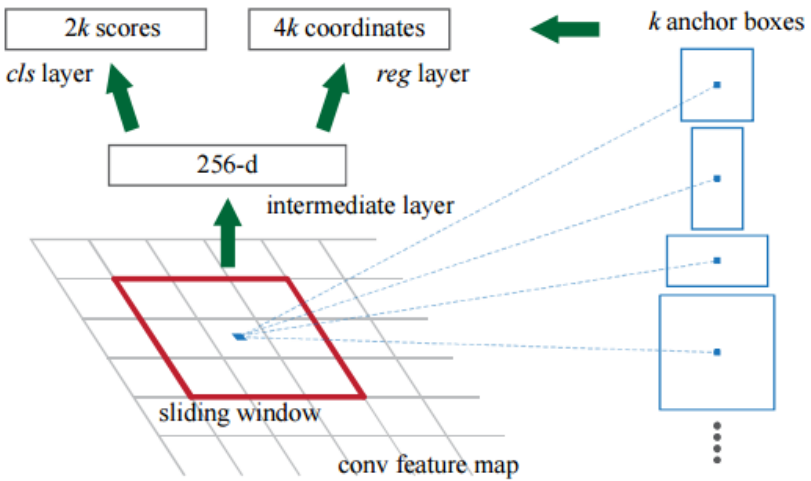
Faster R-CNN

Region Proposal도 Selective Search 대신 Network로 해결해보자!



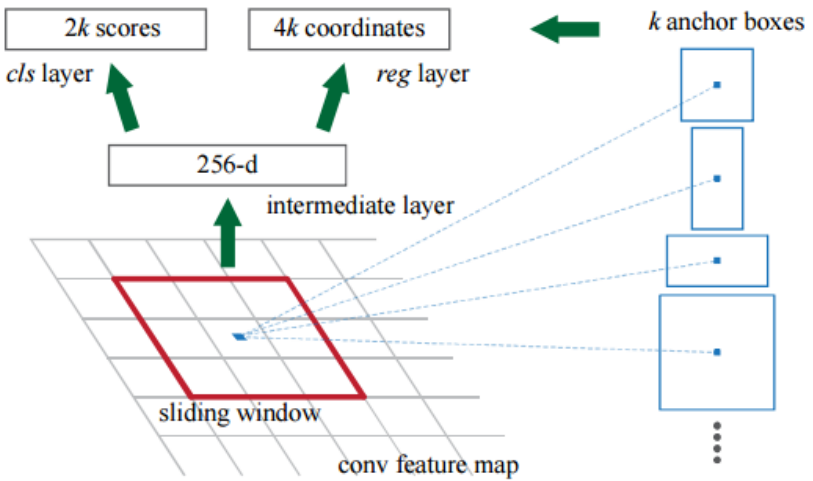
출처: CS231n_2017_Lecture_11_slides

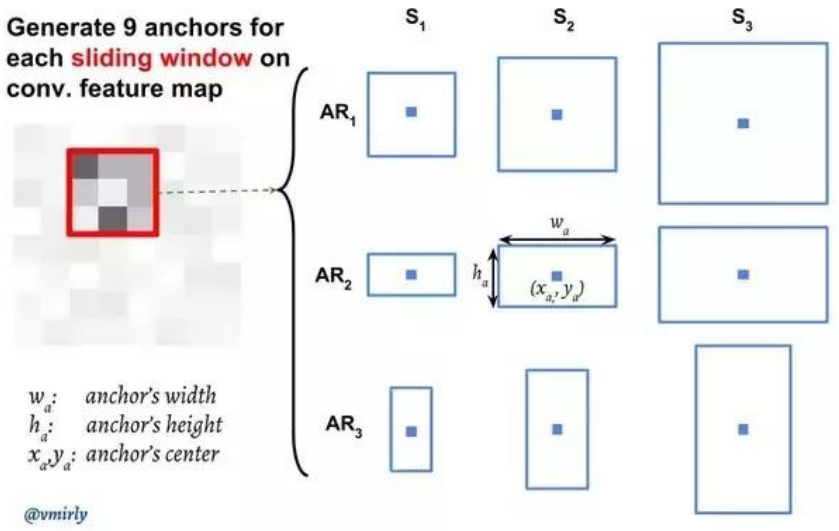
Object Detection



논문에서의 그림

Object Detection

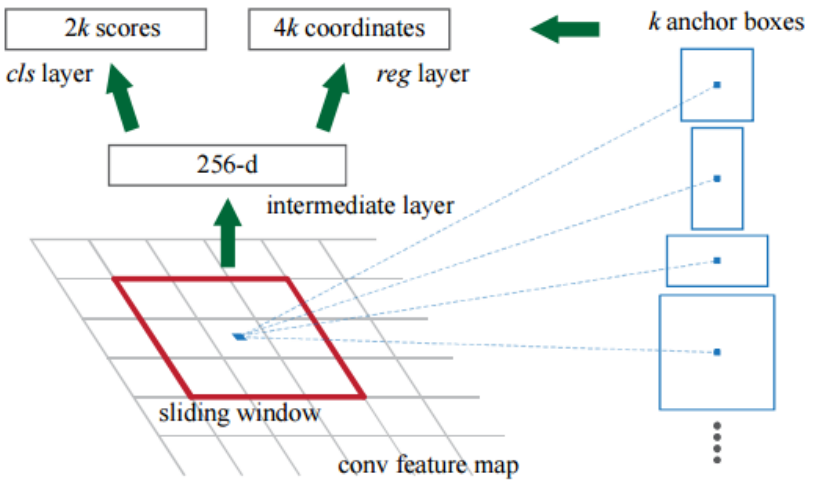


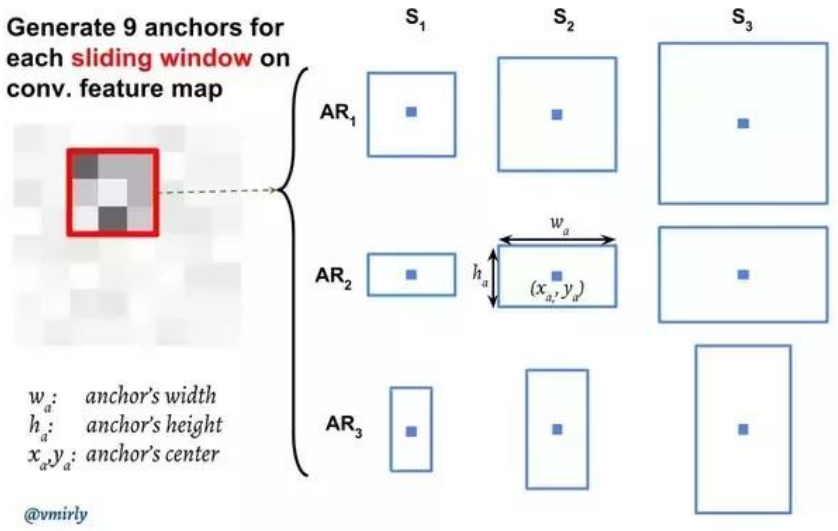


논문에서의 그림

https://www.quora.com/How-does-the-region-proposal-network-RPN-in-Faster-R-CNN-work

Object Detection





논문에서의 그림

https://www.quora.com/How-does-the-region-proposal-network-RPN-in-Faster-R-CNN-work

Feature Map을 몇 개의 anchor를 사용하여 sliding window 방식으로 쭉 지나가면서 물체가 있나 없나(0 or 1), box coordinate(x,y,w,h)을 학습함

Object Detection

학습 과정

Jointly train with 4 losses:

1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classification score (object classes)
4. Final box coordinates

1. 먼저 RPN에서 object가 있는지
2. RPN regression box 학습
3. Classification 학습
4. 최종 Box coordinate 학습

Object Detection

학습 과정

Jointly train with 4 losses:

1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classification score (object classes)
4. Final box coordinates

1. 먼저 RPN에서 object가 있는지
2. RPN regression box 학습
3. Classification 학습
4. 최종 Box coordinate 학습

settings	anchor scales	aspect ratios	mAP (%)
1 scale, 1 ratio	128^2	1:1	65.8
	256^2	1:1	66.7
1 scale, 3 ratios	128^2	{2:1, 1:1, 1:2}	68.8
	256^2	{2:1, 1:1, 1:2}	67.9
3 scales, 1 ratio	$\{128^2, 256^2, 512^2\}$	1:1	**69.8**
3 scales, 3 ratios	$\{128^2, 256^2, 512^2\}$	{2:1, 1:1, 1:2}	**69.9**

anchor box의 scale, ratio에 대한 실험

Object Detection

학습 과정

Jointly train with 4 losses:

1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classification score (object classes)
4. Final box coordinates

1. 먼저 RPN에서 object가 있는지
2. RPN regression box 학습
3. Classification 학습
4. 최종 Box coordinate 학습

settings	anchor scales	aspect ratios	mAP (%)
1 scale, 1 ratio	128^2	1:1	65.8
	256^2	1:1	66.7
1 scale, 3 ratios	128^2	{2:1, 1:1, 1:2}	68.8
	256^2	{2:1, 1:1, 1:2}	67.9
3 scales, 1 ratio	$\{128^2, 256^2, 512^2\}$	1:1	**69.8**
3 scales, 3 ratios	$\{128^2, 256^2, 512^2\}$	{2:1, 1:1, 1:2}	**69.9**

anchor box의 scale, ratio에 대한 실험

method	# proposals	data	mAP (%)
SS	2000	07	66.9†
SS	2000	07+12	70.0
RPN+VGG, unshared	300	07	68.5
RPN+VGG, shared	300	07	69.9
RPN+VGG, shared	300	07+12	**73.2**
RPN+VGG, shared	300	COCO+07+12	**78.8**

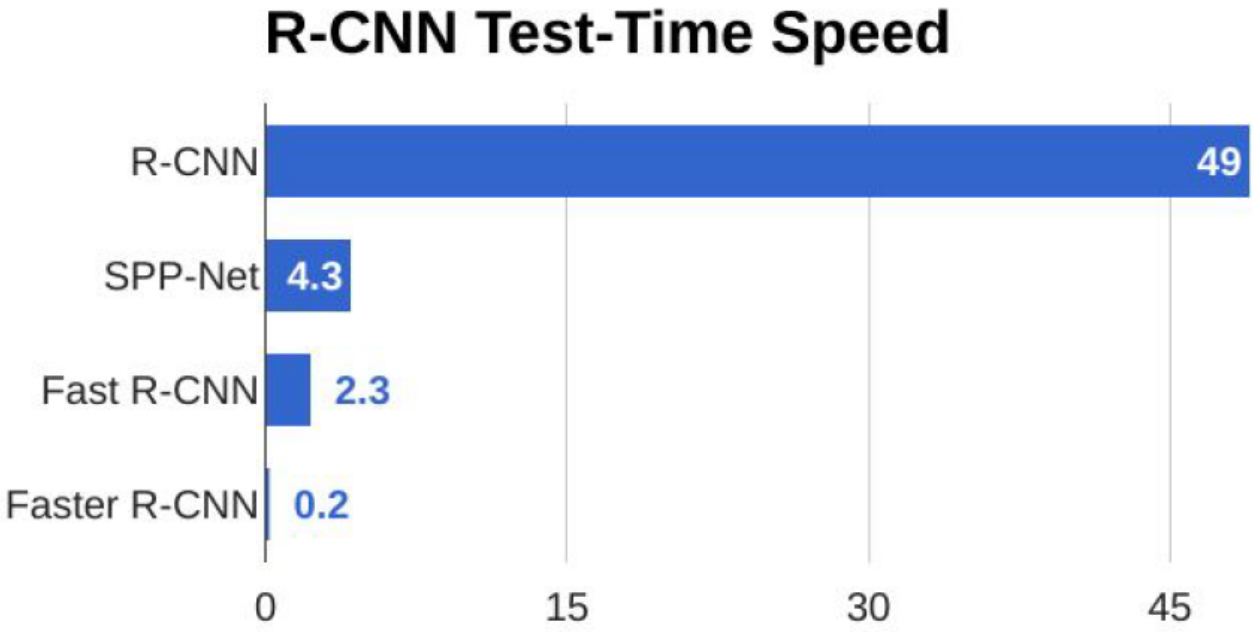
Pascal VOC 2007 test set 결과

Object Detection

	R-CNN	Fast R-CNN	Faster R-CNN
Test time per image (with proposals)	50 seconds	2 seconds	**0.2 seconds**
(Speedup)	1x	25x	**250x**
mAP (VOC 2007)	66.0	**66.9**	**66.9**

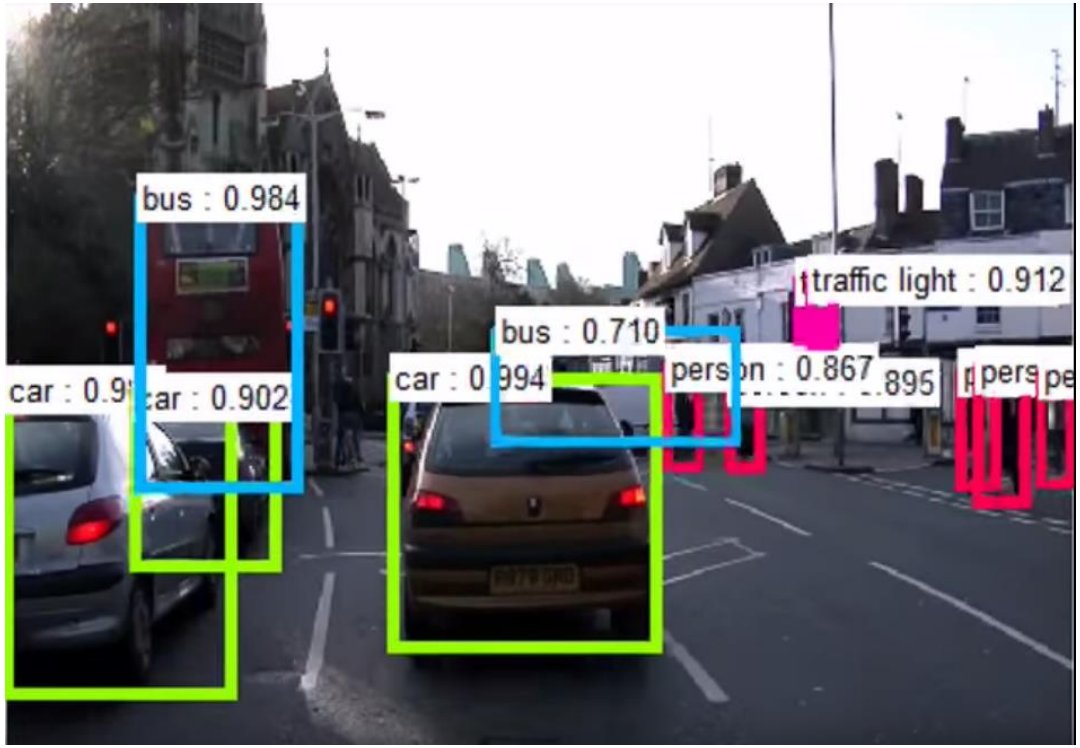
정확도도 높아지고 속도도 빨라지고 거의 Real-time에 가까워짐

Object Detection



좀 더 실감나는 비교 도표

Object Detection



Faster R-CNN + ResNet 101

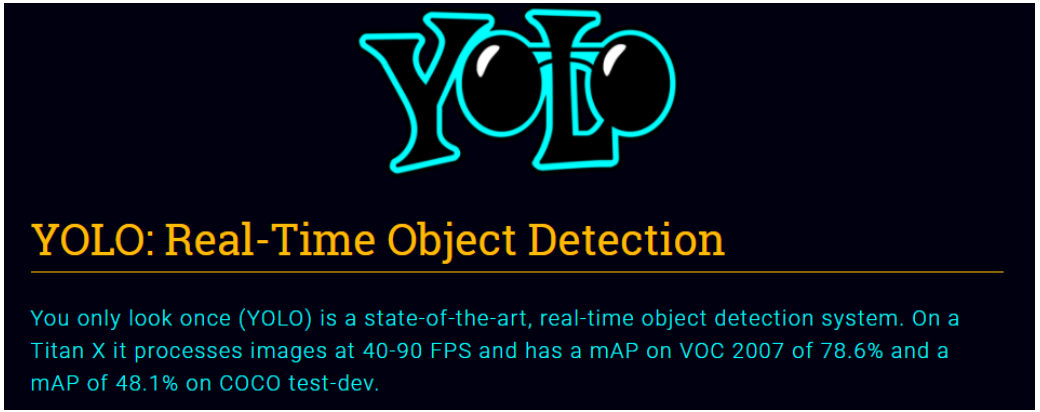
Object Detection

근데 그냥 Region Proposal이고 뭐고 다 한번에 하면 안되나?

Object Detection

근데 그냥 Region Proposal이고 뭐고 다 한번에 하면 안되나?

YOLO(you only look once), SSD(single shot detection)



출처: You Only Look Once: Unified, Real-Time Object Detection

Object Detection

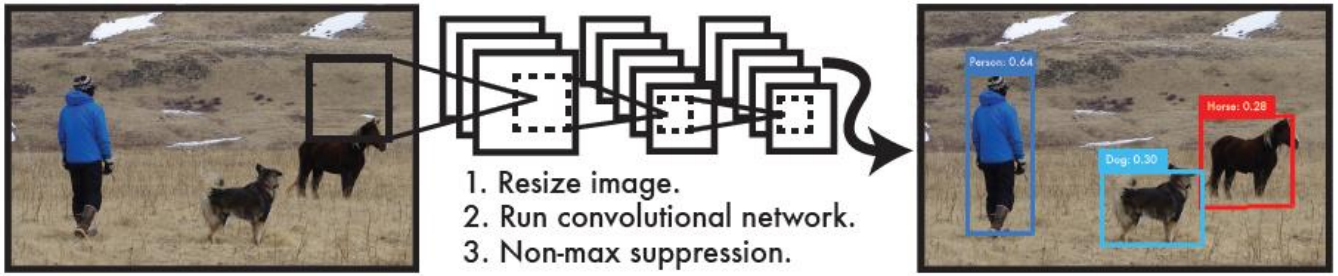
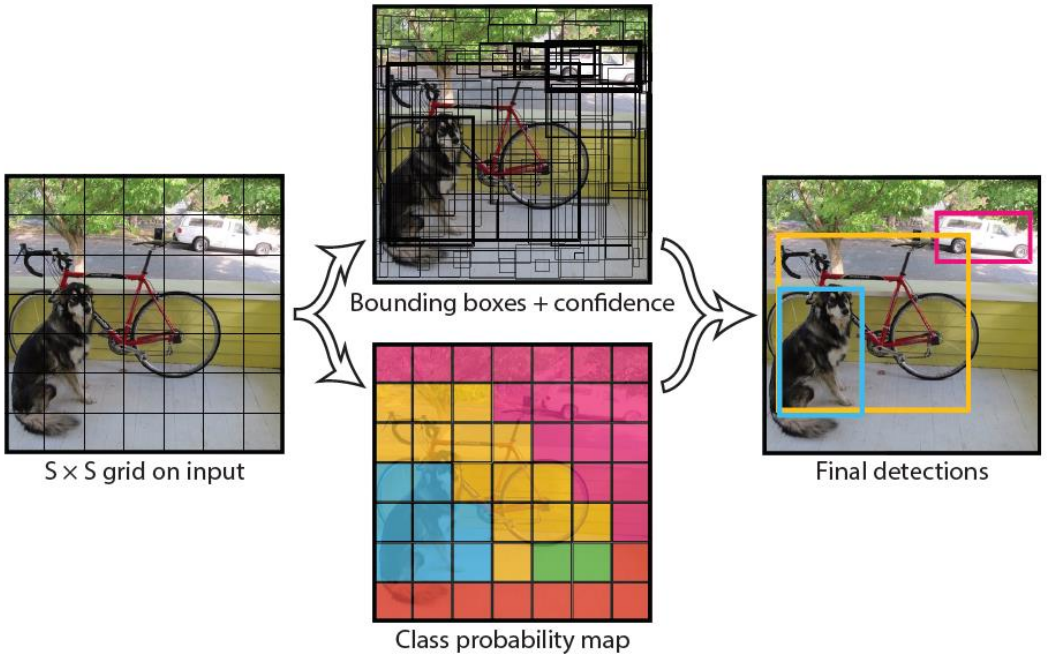


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

사람이 한번 보고 파악하는 것처럼 한번의 feed forward로 끝내자

출처: You Only Look Once: Unified, Real-Time Object Detection

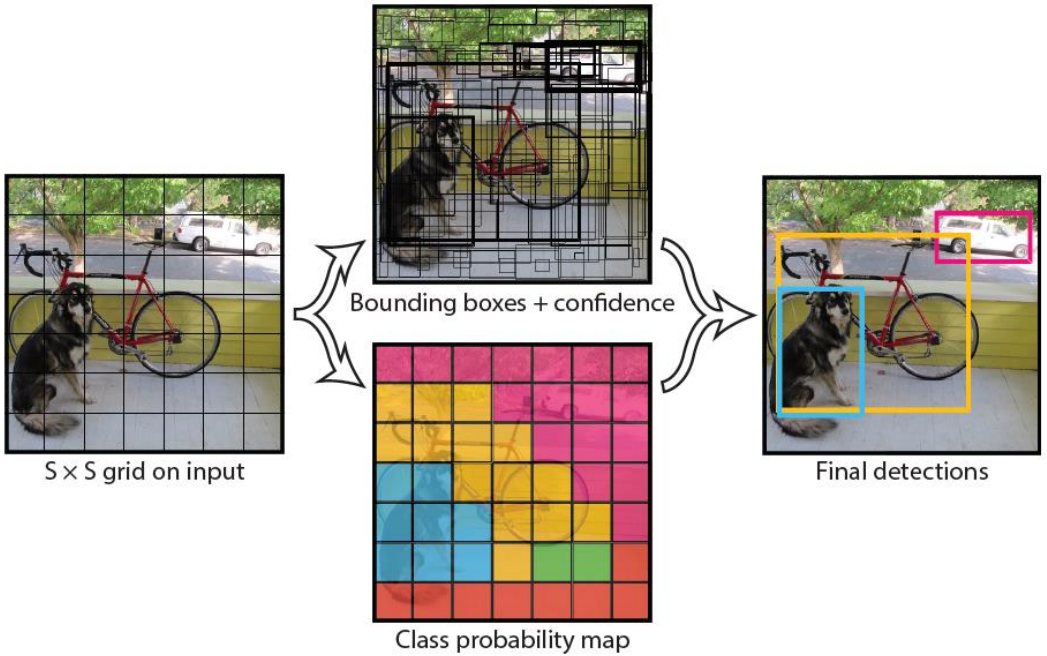
Object Detection



출처: You Only Look Once: Unified, Real-Time Object Detection

Object Detection

1. 우선 이미지를
SxS의 grid로 분할



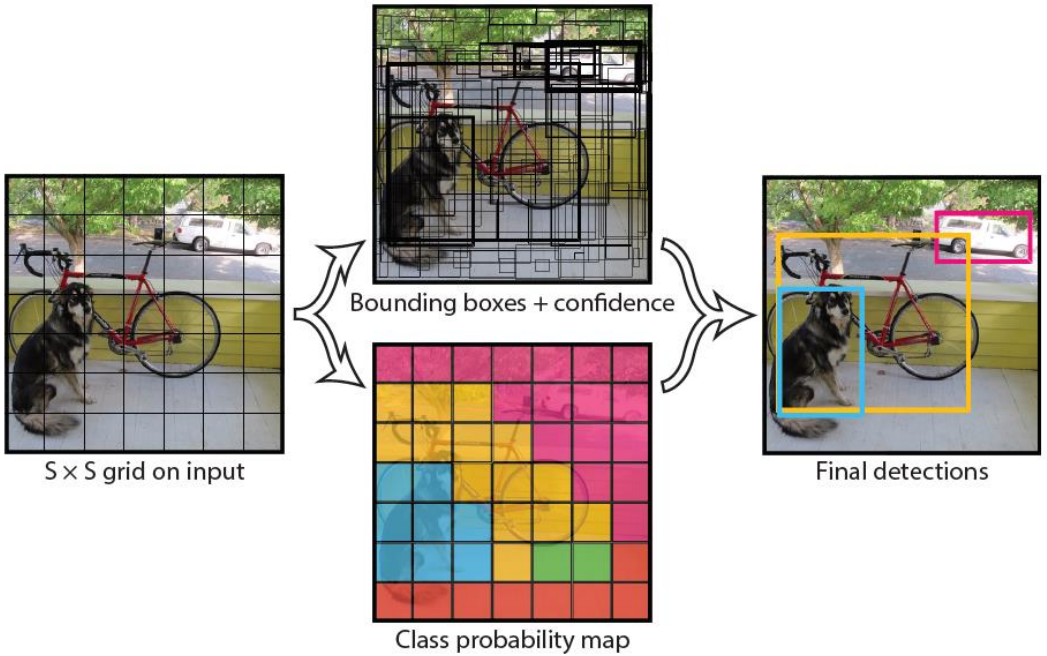
출처: You Only Look Once: Unified, Real-Time Object Detection

Object Detection

Prediction box와 Ground Truth 간의 IOU

2-1. 각 grid 별[x,y,w,h] 및 confidence 예측

1. 우선 이미지를 SxS의 grid로 분할



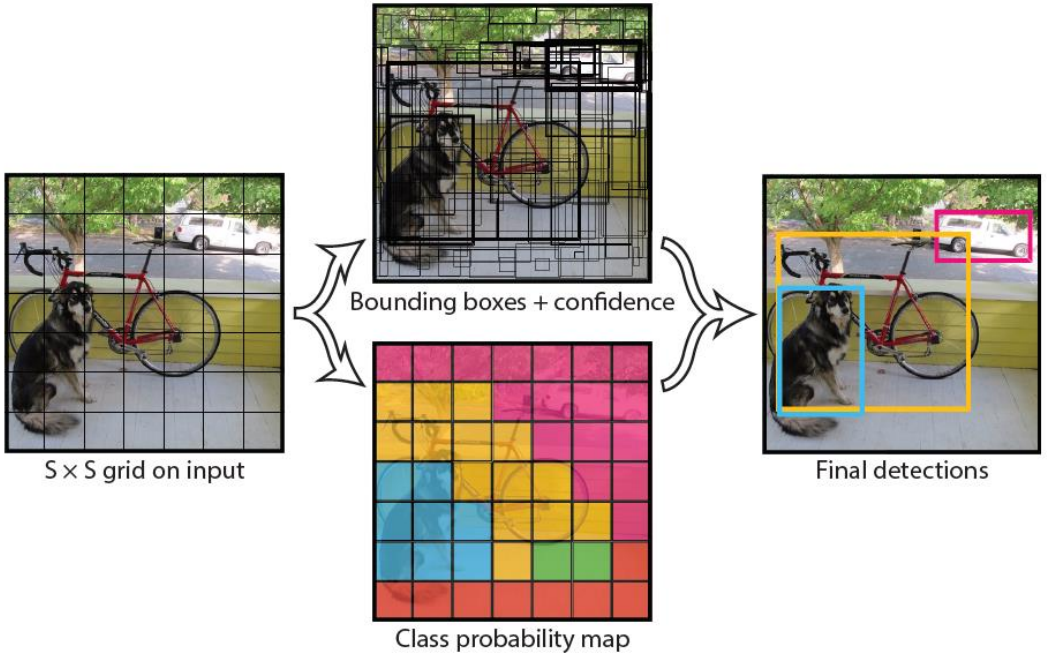
출처: You Only Look Once: Unified, Real-Time Object Detection

Object Detection

Prediction box와 Ground Truth 간의 IOU

2-1. 각 grid 별[x,y,w,h] 및 confidence 예측

1. 우선 이미지를 SxS의 grid로 분할



2-2. 각 grid별 클래스 probability를 구함

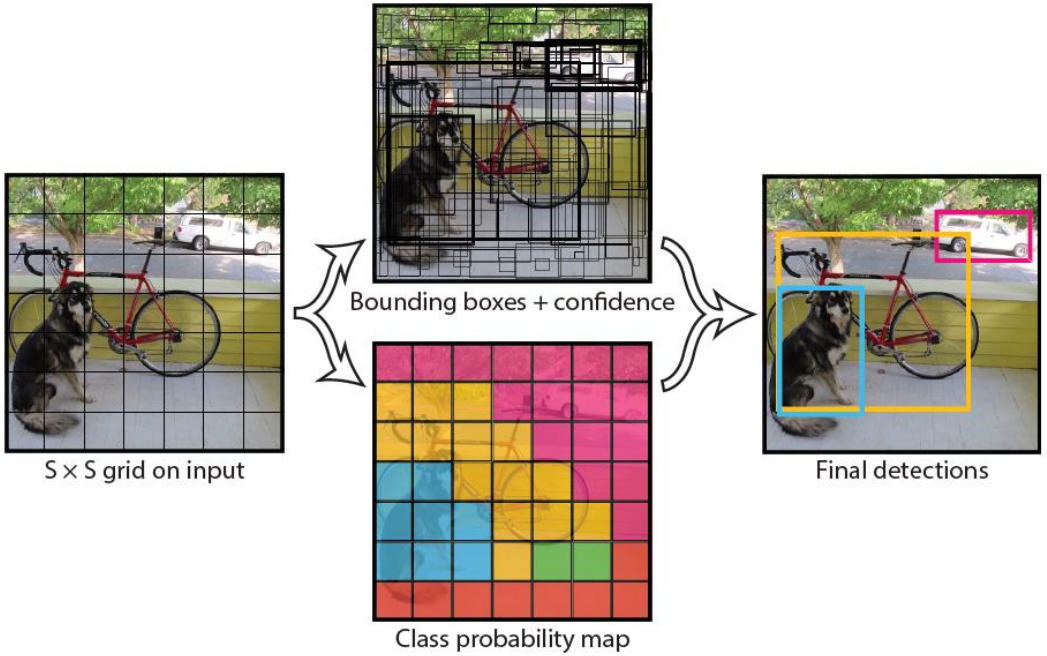
출처: You Only Look Once: Unified, Real-Time Object Detection

Object Detection

Prediction box와 Ground Truth 간의 IOU

2-1. 각 grid 별[x,y,w,h] 및 confidence 예측

1. 우선 이미지를 SxS의 grid로 분할



3. 최종 예측

2-2. 각 grid별 클래스 probability를 구함

출처: You Only Look Once: Unified, Real-Time Object Detection

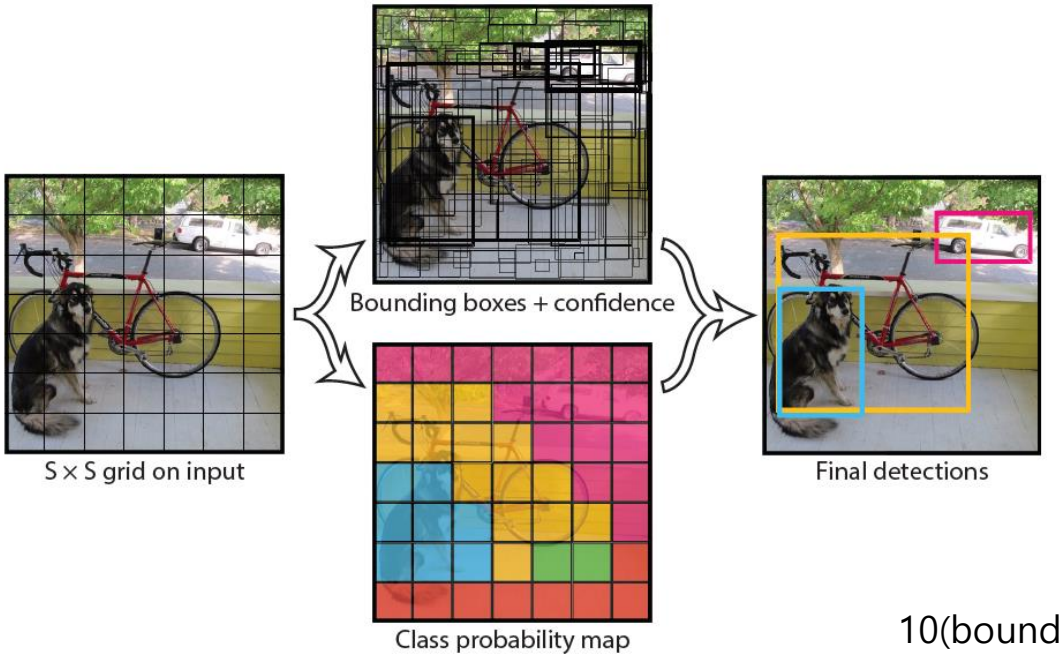
Object Detection

[x, y, w, h, confidence]

B(2) x 5

S = 7
B = 2
C = 20

7(S)x7(S) grid



7x7x30

10(bounding box+confidence)
+20(class probability map)

C(20)

출처: You Only Look Once: Unified, Real-Time Object Detection

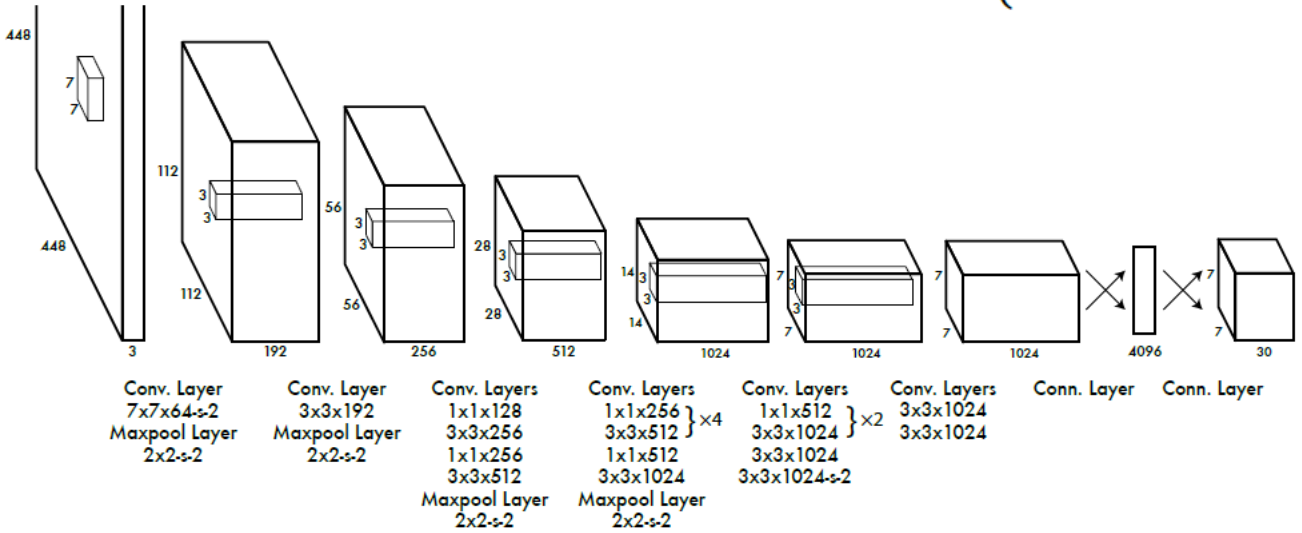
Object Detection

Activation은 Leaky ReLU

$$\phi(x) = \begin{cases} x, & \text{if } x > 0 \\ 0.1x, & \text{otherwise} \end{cases}$$

Pascal VOC Data

resize to 448x448



출처: You Only Look Once: Unified, Real-Time Object Detection

Object Detection

loss function:

$$
\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \qquad (3)
\end{aligned}
$$

출처: You Only Look Once: Unified, Real-Time Object Detection

Object Detection

loss function:

grid별
S x S

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \qquad (3)$$

출처: You Only Look Once: Unified, Real-Time Object Detection

Object Detection

bounding box 별

grid별
S x S

loss function:

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \qquad (3)$$

출처: You Only Look Once: Unified, Real-Time Object Detection

Object Detection

bounding box 별

loss function:

grid별
S x S

해당 박스에 object
가 존재하는지 여부

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \qquad (3)$$

출처: You Only Look Once: Unified, Real-Time Object Detection

Object Detection

bounding box 별

loss function:

grid별
S x S

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

해당 박스에 object 가 존재하는지 여부

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

cell i의 j번째 박스가 그 물체에 대해 responsible 할 때(ex. highest IOU)

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \qquad (3)$$

출처: You Only Look Once: Unified, Real-Time Object Detection

Object Detection

bounding box 별

loss function:

grid별
S x S

학습 비율을 맞춰주기 위해
$\lambda_{coord} = 5$
$\lambda_{noobj} = 0.5$

해당 박스에 object 가 존재하는지 여부

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

cell i의 j번째 박스가 그 물체에 대해 responsible 할 때(ex. highest IOU)

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \qquad (3)$$

출처: You Only Look Once: Unified, Real-Time Object Detection

Object Detection

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

Table 1: Real-Time Systems on PASCAL VOC 2007. Comparing the performance and speed of fast detectors. Fast YOLO is the fastest detector on record for PASCAL VOC detection and is still twice as accurate as any other real-time detector. YOLO is 10 mAP more accurate than the fast version while still well above real-time in speed.

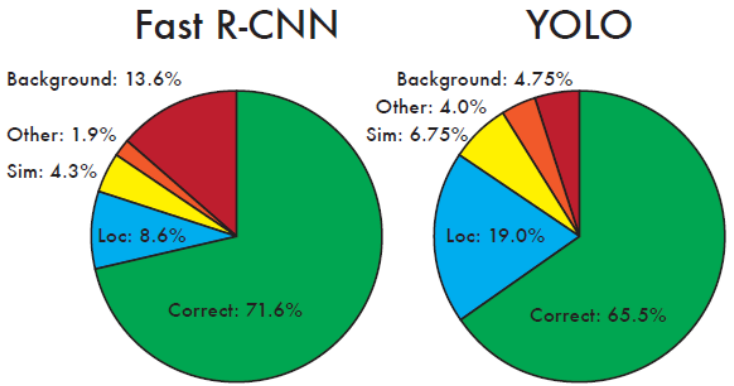


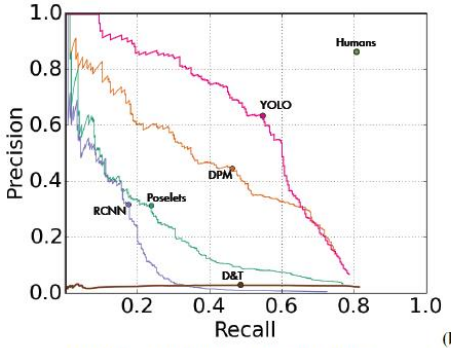
Figure 4: Error Analysis: Fast R-CNN vs. YOLO These charts show the percentage of localization and background errors in the top N detections for various categories (N = # objects in that category).

Yolo는 location을 잘 못 잡지만 아무 물체가 없는 배경을 잡는 확률을 낮음

Fast R-CNN과 반대

출처: You Only Look Once: Unified, Real-Time Object Detection

Object Detection

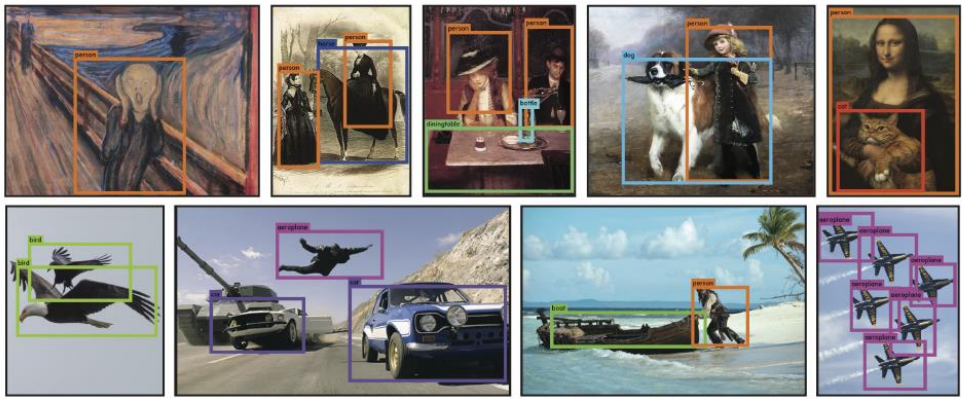


(a) Picasso Dataset precision-recall curves.

	VOC 2007	Picasso		People-Art
	AP	AP	Best F_1	AP
YOLO	**59.2**	**53.3**	**0.590**	**45**
R-CNN	54.2	10.4	0.226	26
DPM	43.2	37.8	0.458	32
Poselets [2]	36.5	17.8	0.271	
D&T [4]	-	1.9	0.051	

(b) Quantitative results on the VOC 2007, Picasso, and People-Art Datasets. The Picasso Dataset evaluates on both AP and best F_1 score.

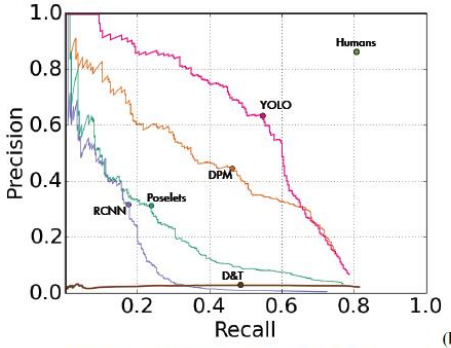
Figure 5: Generalization results on Picasso and People-Art datasets.



Yolo는 artwork에 대해 generalize 수치가 높다

출처: You Only Look Once: Unified, Real-Time Object Detection

Object Detection

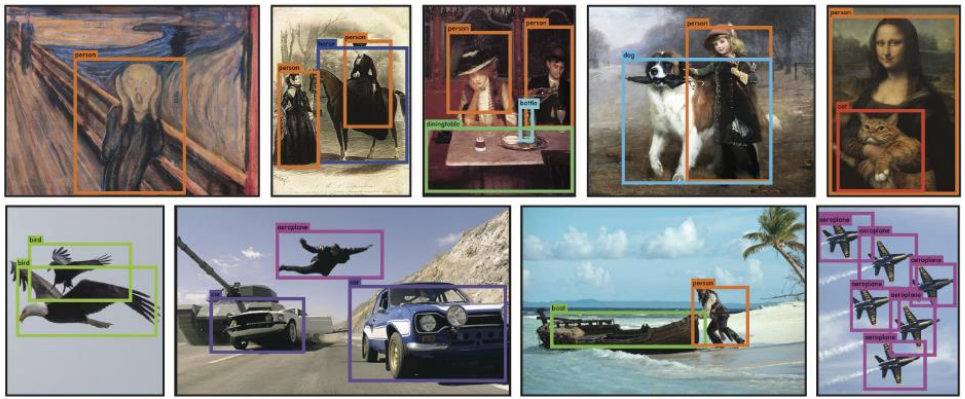


(a) Picasso Dataset precision-recall curves.

	VOC 2007	Picasso		People-Art
	AP	AP	Best F_1	AP
YOLO	**59.2**	**53.3**	**0.590**	**45**
R-CNN	54.2	10.4	0.226	26
DPM	43.2	37.8	0.458	32
Poselets [2]	36.5	17.8	0.271	
D&T [4]	-	1.9	0.051	

(b) Quantitative results on the VOC 2007, Picasso, and People-Art Datasets. The Picasso Dataset evaluates on both AP and best F_1 score.

Figure 5: Generalization results on Picasso and People-Art datasets.

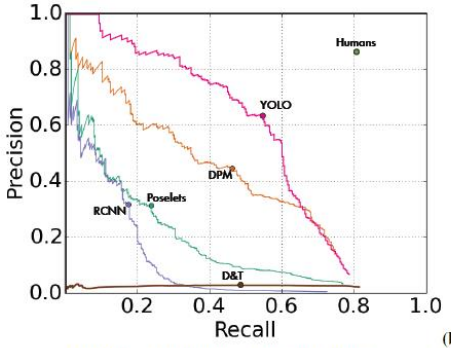


Yolo는 artwork에 대해 generalize 수치가 높다

이유는 이미지 전체를 입력으로 받기 때문에 전체적 위치와 사물의 관계를 통해 파악하기 때문

출처: You Only Look Once: Unified, Real-Time Object Detection

Object Detection

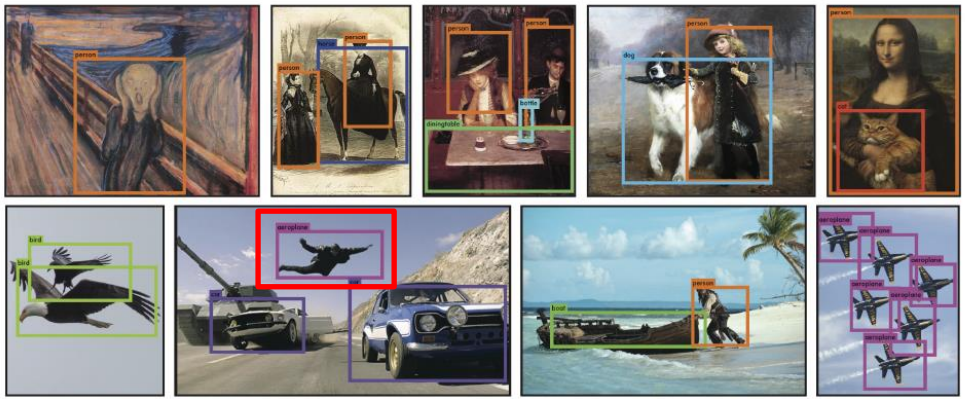


(a) Picasso Dataset precision-recall curves.

	VOC 2007 AP	Picasso AP	Picasso Best F_1	People-Art AP
YOLO	**59.2**	**53.3**	**0.590**	**45**
R-CNN	54.2	10.4	0.226	26
DPM	43.2	37.8	0.458	32
Poselets [2]	36.5	17.8	0.271	
D&T [4]	-	1.9	0.051	

(b) Quantitative results on the VOC 2007, Picasso, and People-Art Datasets. The Picasso Dataset evaluates on both AP and best F_1 score.

Figure 5: Generalization results on Picasso and People-Art datasets.



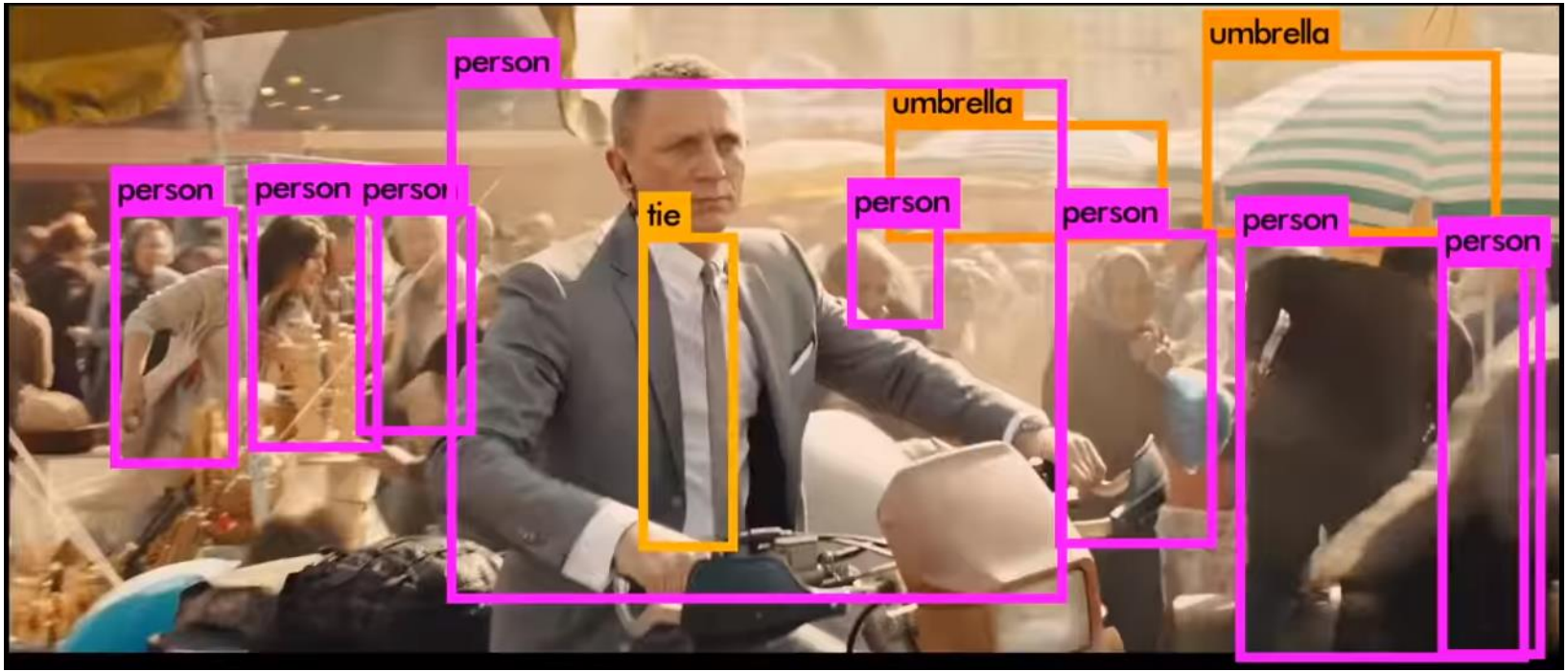
Yolo는 artwork에 대해 generalize 수치가 높다

이유는 이미지 전체를 입력으로 받기 때문에 전체적 위치와 사물의 관계를 통해 파악하기 때문

근데 사람을 비행기로 착각하기도 함

출처: You Only Look Once: Unified, Real-Time Object Detection

Object Detection



https://www.youtube.com/watch?v=VOC3huqHrss

출처: You Only Look Once: Unified, Real-Time Object Detection

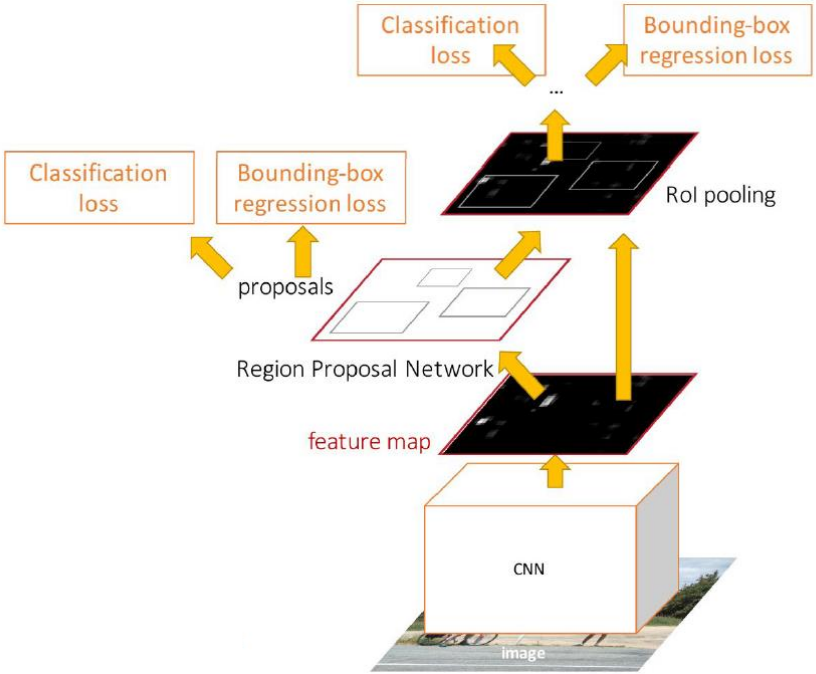
Object Detection

Mask R-CNN

출처: https://arxiv.org/abs/1703.06870

Object Detection

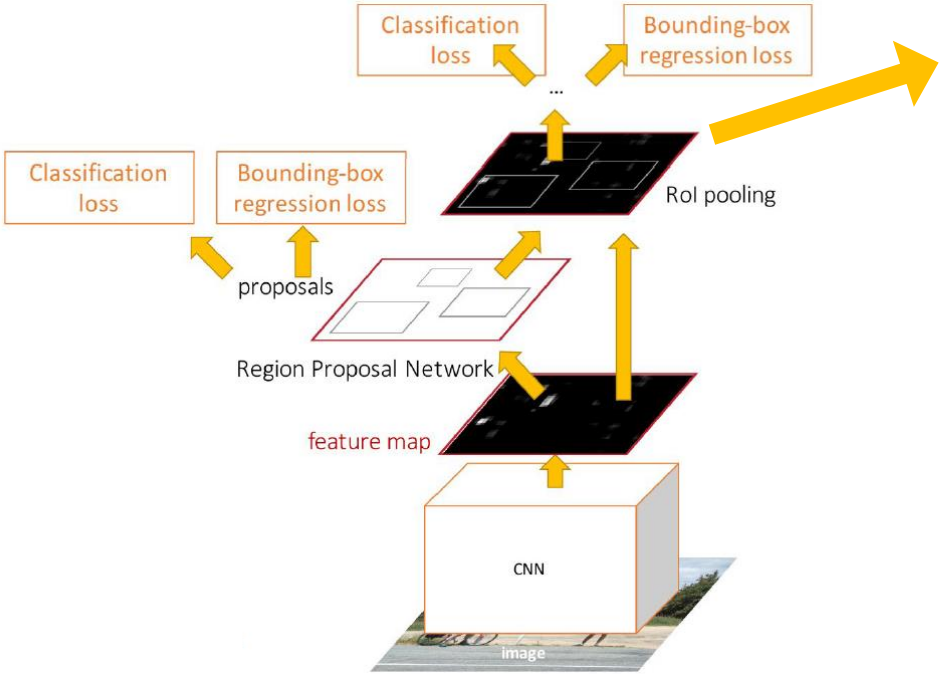
Mask R-CNN = Faster R-CNN + Object Mask



출처: https://arxiv.org/abs/1703.06870

Object Detection

Mask R-CNN = Faster R-CNN + Object Mask

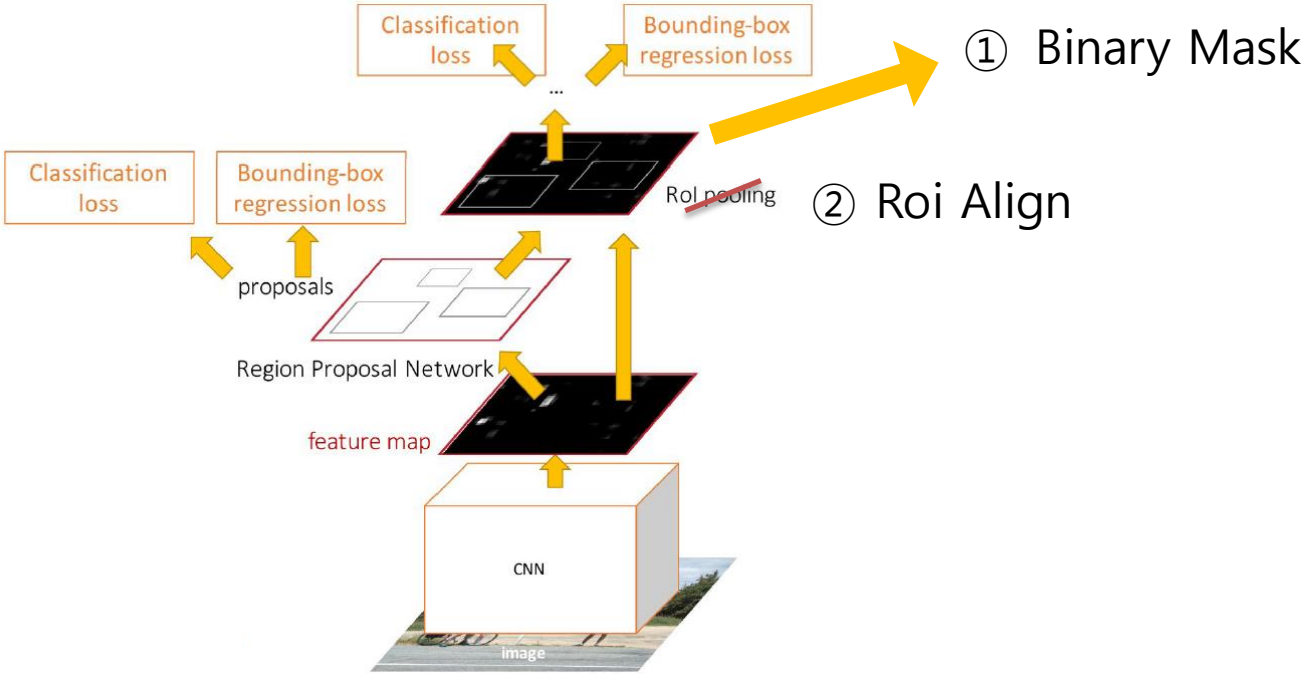


① Binary Mask

출처: https://arxiv.org/abs/1703.06870

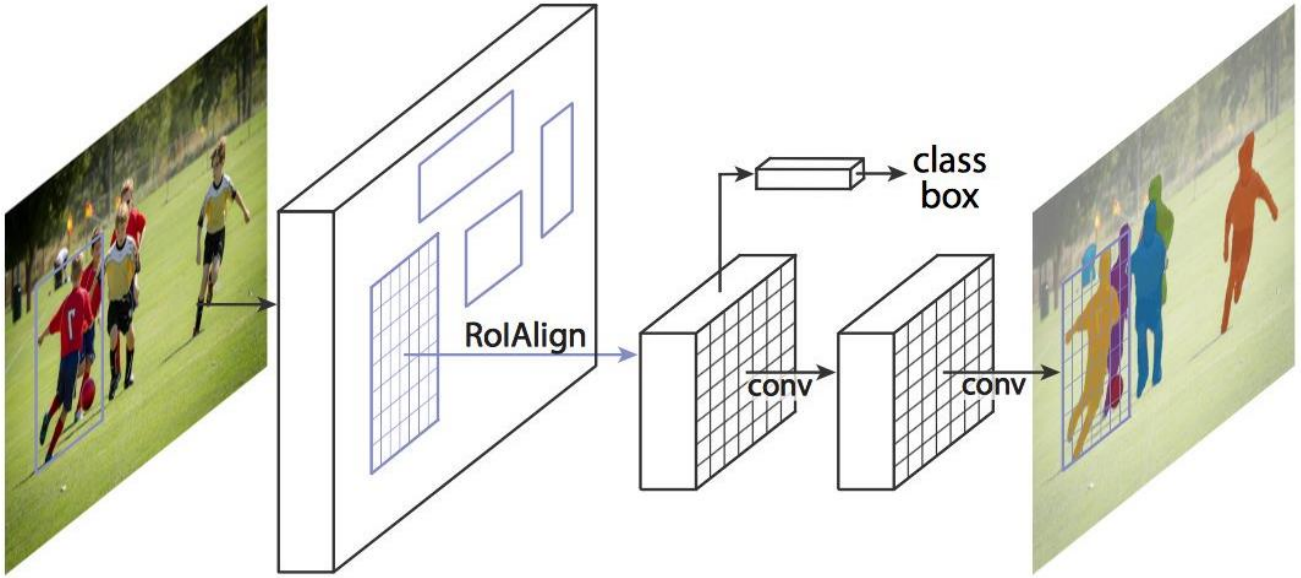
Object Detection

Mask R-CNN = Faster R-CNN + Object Mask



출처: https://arxiv.org/abs/1703.06870

Object Detection

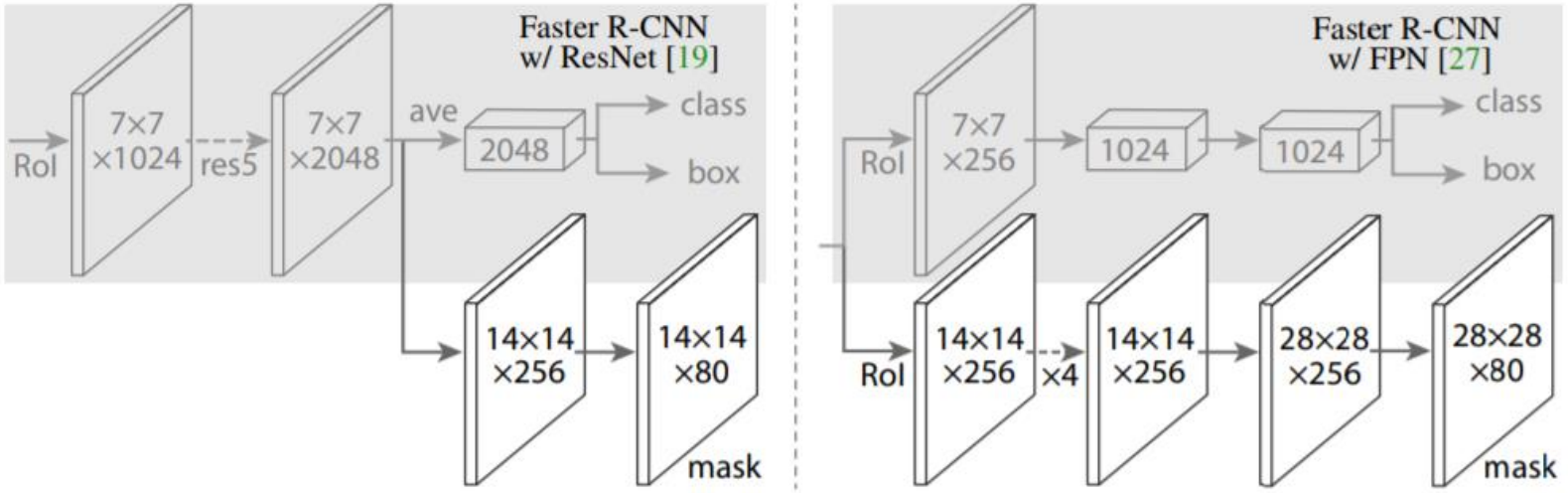


Class Score와 Bounding box와 날리 Mask는 FCN으로 이미지처럼 m x m x class 형태로 생성

각 클래스는 기존의 방법처럼 클래스 별 softmax가 아닌 class independent 한 binary mask를 사용함

출처: https://arxiv.org/abs/1703.06870

Object Detection

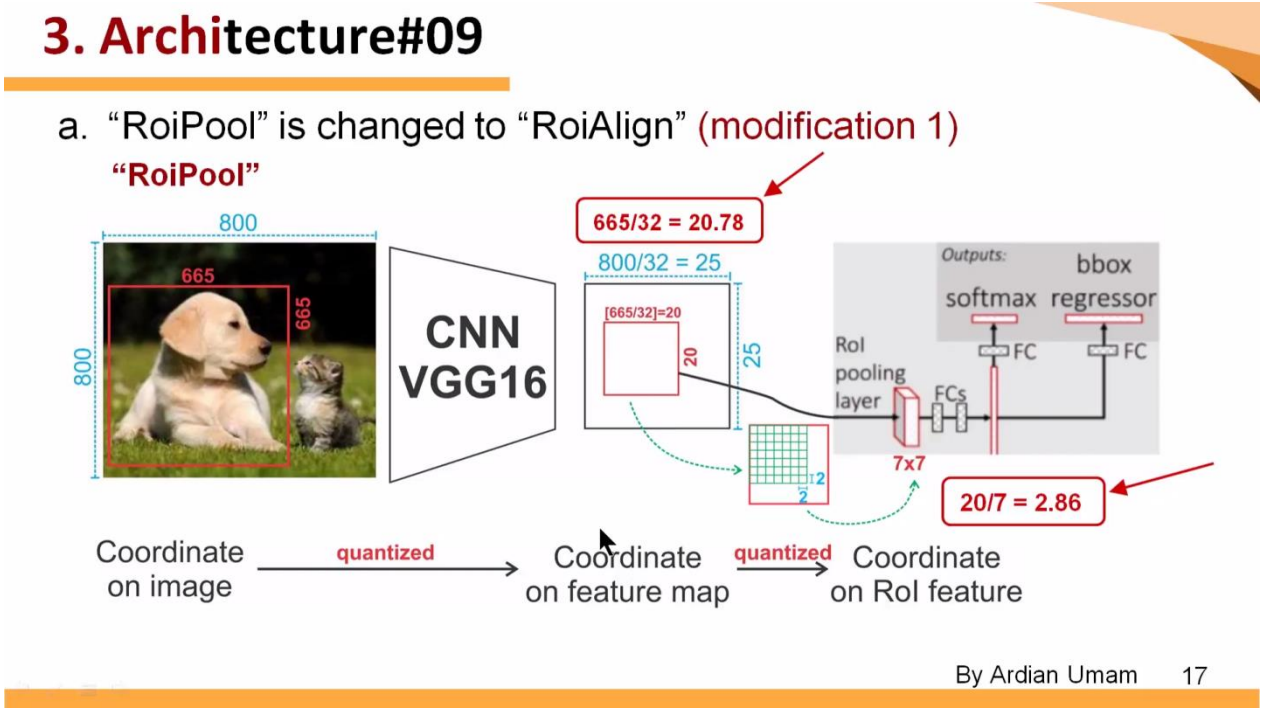


m x m x class의 텐서에서 어떻게 원본 크기로 늘릴까?

Roi Align

출처: https://arxiv.org/abs/1703.06870

Object Detection



ROI Pooling은 위치 정보에 대한 손실이 많음
2번이나 quantization이 발생함

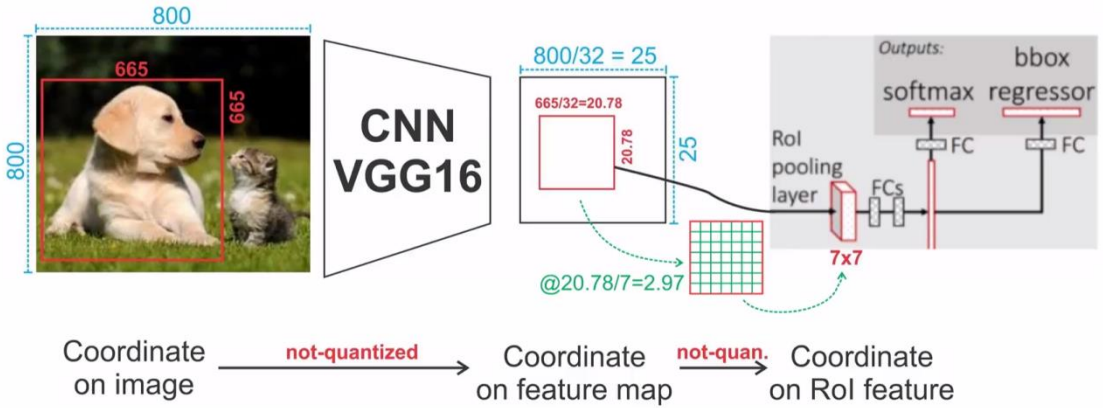
출처: https://www.youtube.com/watch?v=XGi-Mz3do2s

Object Detection

3. Architecture#10

a. "RoiPool" is changed to "RoiAlign"

"RoiAlign"

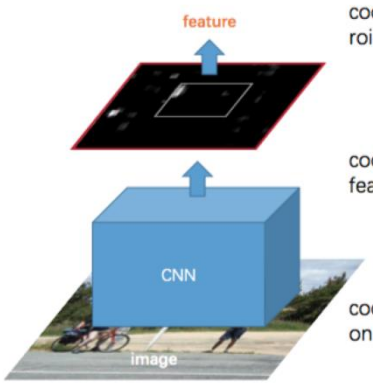


By Ardian Umam 18

ROI Align은 대신 Bilinear Interpolation을 사용하여 정보의 손실을 최소화 함

출처: https://www.youtube.com/watch?v=XGi-Mz3do2s

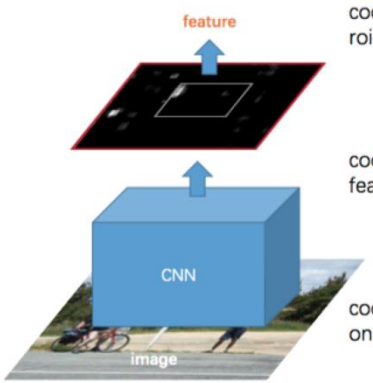
Object Detection



	RoI pooling	RoI warp	RoI align
coordinates on roi feature			
quantization	√	×	×
coordinates on feature map			
quantization	√	√	×
coordinates on image			

출처: https://arxiv.org/abs/1703.06870

Object Detection



	AP	AP_{50}	AP_{75}
softmax	24.8	44.1	25.1
sigmoid	**30.3**	**51.2**	**31.5**
	+5.5	*+7.1*	*+6.4*

(b) **Multinomial *vs.* Independent Masks** (ResNet-50-C4): *Decoupling* via per-class binary masks (sigmoid) gives large gains over multinomial masks (softmax).

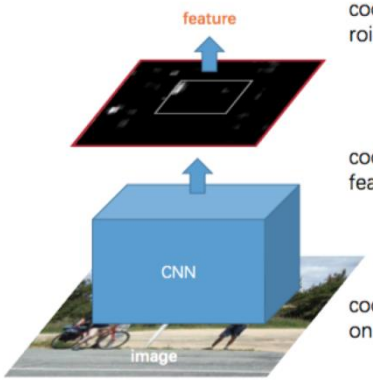
	align?	bilinear?	agg.	AP	AP_{50}	AP_{75}
RoIPool [12]			max	26.9	48.8	26.4
RoIWarp [10]		✓	max	27.2	49.2	27.1
		✓	ave	27.1	48.9	27.1
RoIAlign	✓	✓	max	**30.2**	**51.0**	**31.8**
	✓	✓	ave	**30.3**	**51.2**	**31.5**

(c) **RoIAlign** (ResNet-50-C4): Mask results with various RoI layers. Our RoIAlign layer improves AP by ~3 points and AP_{75} by ~5 points. Using proper alignment is the only factor that contributes to the large gap between RoI layers.

출처: https://arxiv.org/abs/1703.06870

Object Detection

클래스별 비교보다 각자 mask를 구하는 것이 더 좋았음



	RoI pooling	RoI warp	RoI align
coordinates on roi feature	√	×	×
coordinates on feature map	√	√	×

	AP	AP_{50}	AP_{75}
softmax	24.8	44.1	25.1
sigmoid	**30.3**	**51.2**	**31.5**
	+5.5	*+7.1*	*+6.4*

(b) **Multinomial *vs.* Independent Masks** (ResNet-50-C4): *Decoupling* via per-class binary masks (sigmoid) gives large gains over multinomial masks (softmax).

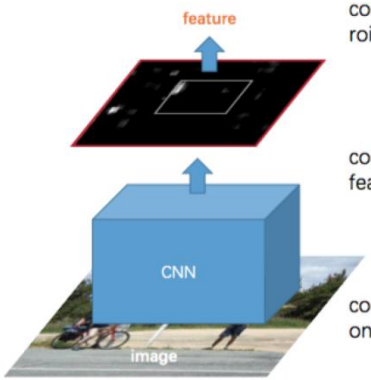
	align?	bilinear?	agg.	AP	AP_{50}	AP_{75}
RoIPool [12]			max	26.9	48.8	26.4
RoIWarp [10]		✓	max	27.2	49.2	27.1
		✓	ave	27.1	48.9	27.1
RoIAlign	✓	✓	max	**30.2**	**51.0**	**31.8**
	✓	✓	ave	**30.3**	**51.2**	**31.5**

(c) **RoIAlign** (ResNet-50-C4): Mask results with various RoI layers. Our RoIAlign layer improves AP by ~3 points and AP_{75} by ~5 points. Using proper alignment is the only factor that contributes to the large gap between RoI layers.

출처: https://arxiv.org/abs/1703.06870

Object Detection

클래스별 비교보다 각자 mask를 구하는 것이 더 좋았음



	RoI pooling	RoI warp	RoI align
coordinates on roi feature	√	×	×
coordinates on feature map	√	√	×

ROI Align이 제일 좋았음

	AP	AP_{50}	AP_{75}
softmax	24.8	44.1	25.1
sigmoid	**30.3**	**51.2**	**31.5**
	+5.5	*+7.1*	*+6.4*

(b) **Multinomial *vs.* Independent Masks** (ResNet-50-C4): *Decoupling* via per-class binary masks (sigmoid) gives large gains over multinomial masks (softmax).

	align?	bilinear?	agg.	AP	AP_{50}	AP_{75}
RoIPool [12]			max	26.9	48.8	26.4
RoIWarp [10]		✓	max	27.2	49.2	27.1
		✓	ave	27.1	48.9	27.1
RoIAlign	✓	✓	max	**30.2**	**51.0**	**31.8**
	✓	✓	ave	**30.3**	**51.2**	**31.5**

(c) **RoIAlign** (ResNet-50-C4): Mask results with various RoI layers. Our RoIAlign layer improves AP by ~3 points and AP_{75} by ~5 points. Using proper alignment is the only factor that contributes to the large gap between RoI layers.

출처: https://arxiv.org/abs/1703.06870

Image Captioning

Show and Tell

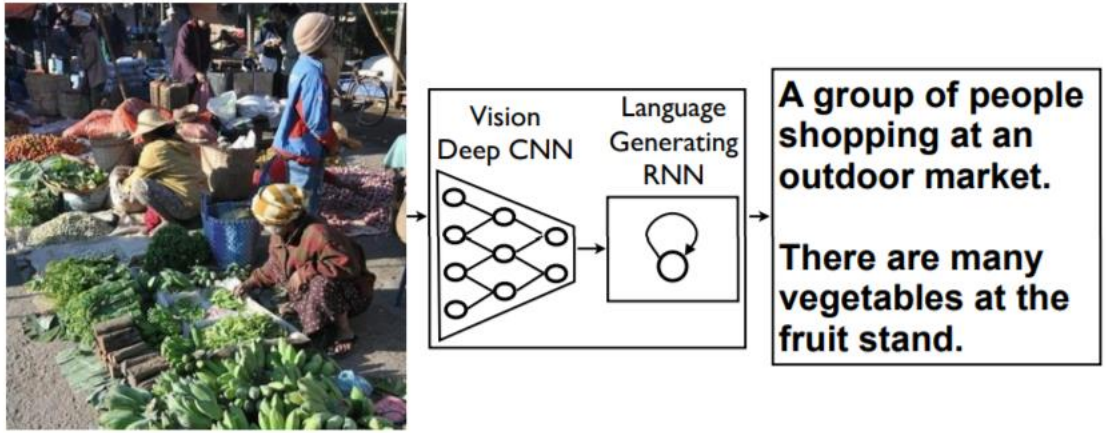
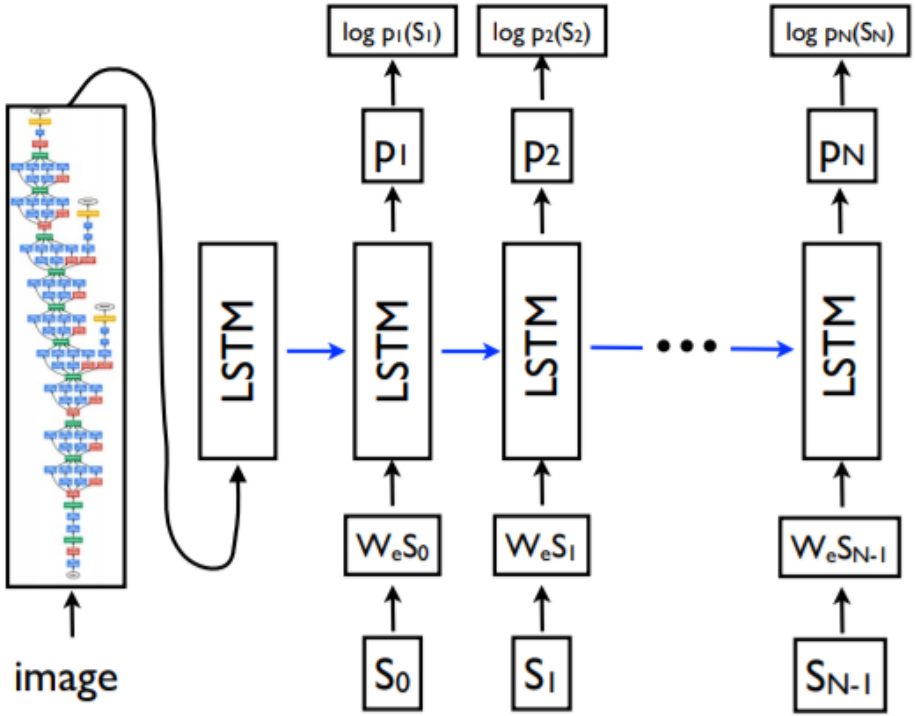


Figure 1. NIC, our model, is based end-to-end on a neural network consisting of a vision CNN followed by a language generating RNN. It generates complete sentences in natural language from an input image, as shown on the example above.

CNN + RNN

Image Captioning

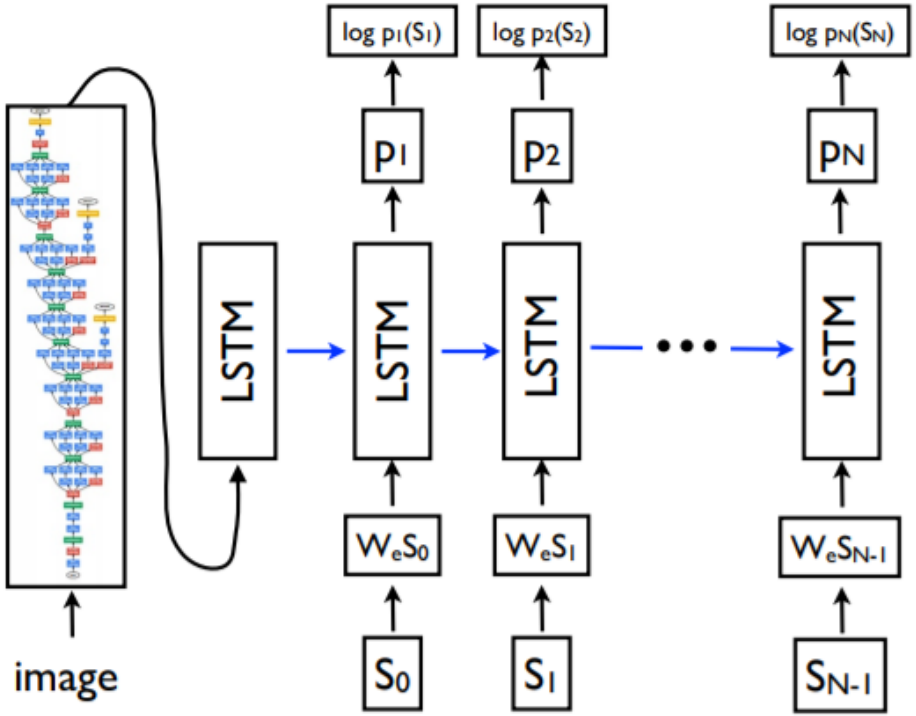
Show and Tell



Show and Tell: A Neural Image Caption Generator

Image Captioning

Show and Tell



Show and Tell: A Neural Image Caption Generator

Image Captioning

Image에 대한 정보를
Hidden state로 압축
ex) 4096 vector

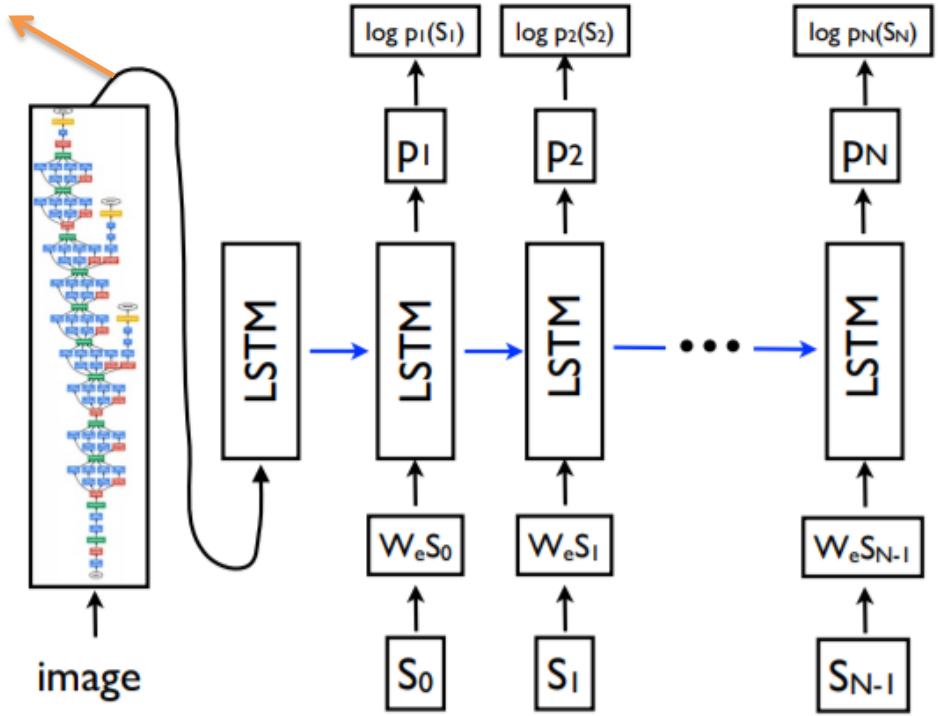


Image Captioning

Show and Tell

Image에 대한 정보를
Hidden state로 압축
ex) 4096 vector

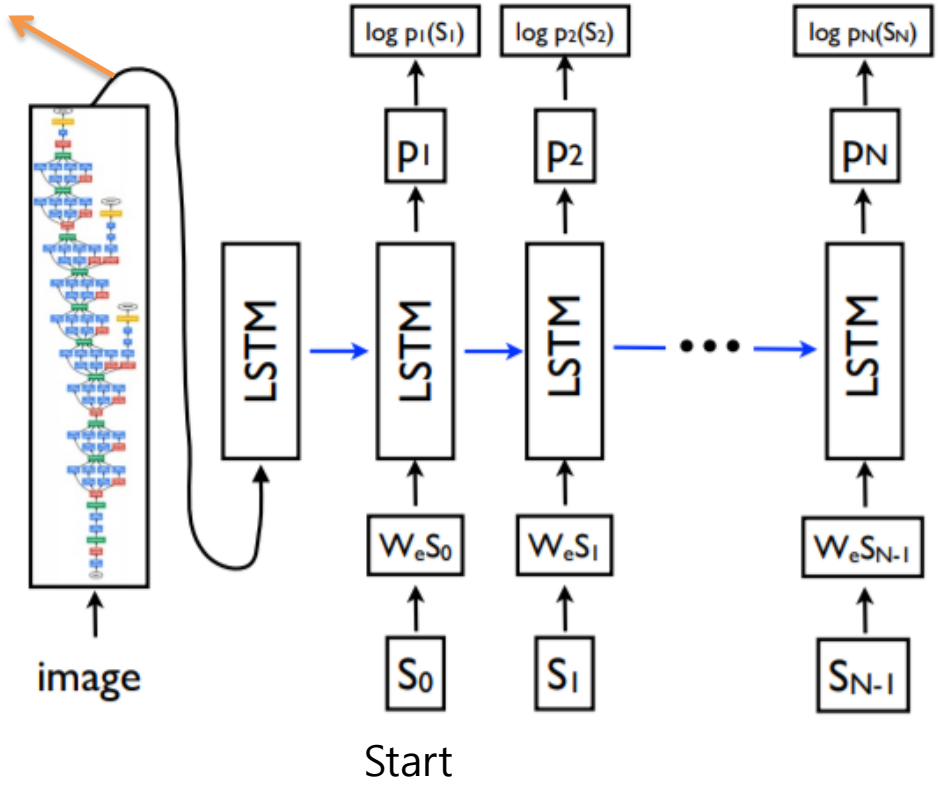


Image Captioning

Show and Tell

Image에 대한 정보를
Hidden state로 압축
ex) 4096 vector

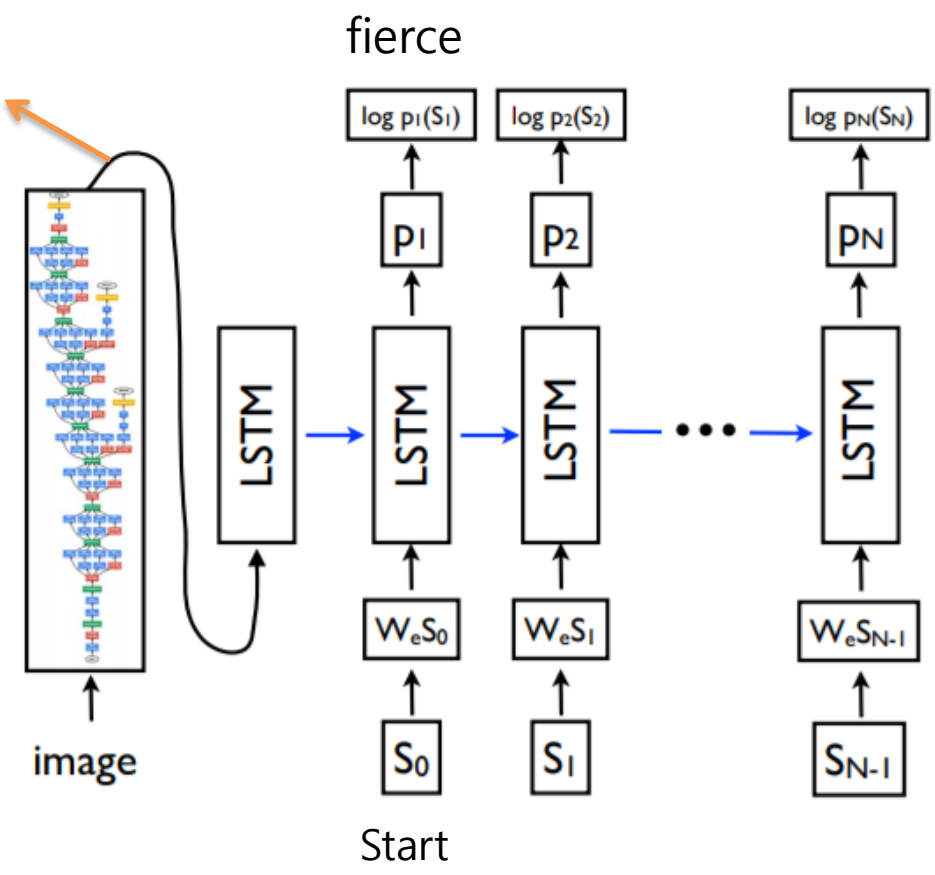
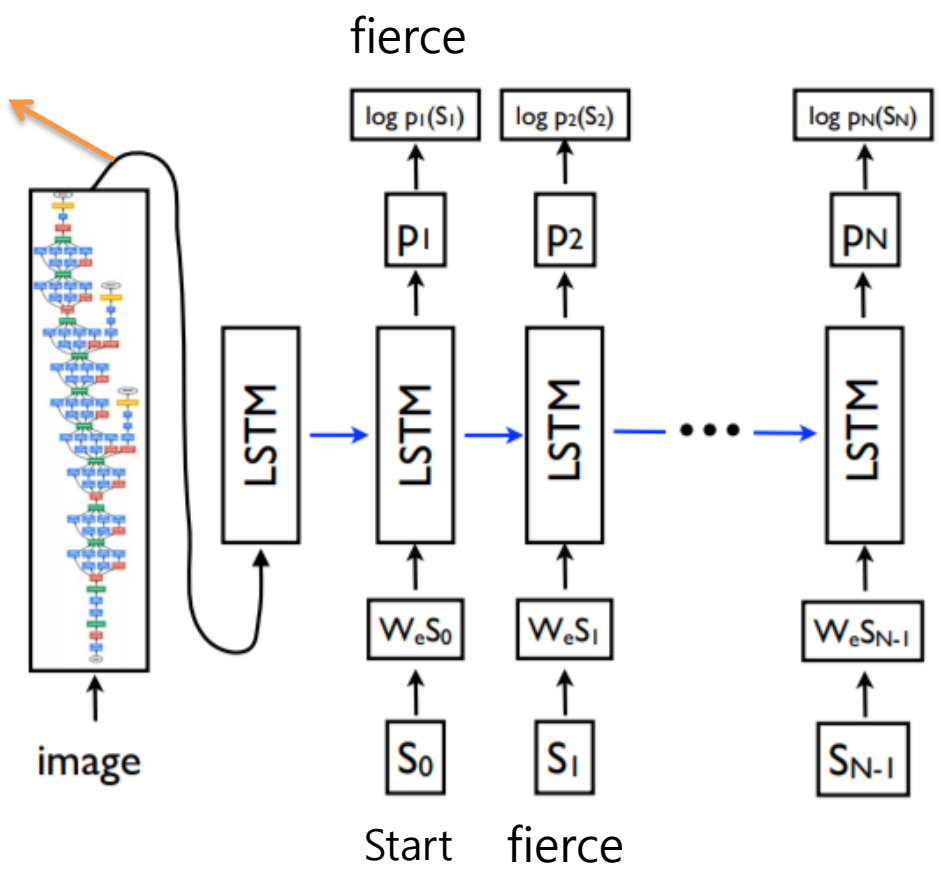


Image Captioning

Show and Tell

Image에 대한 정보를
Hidden state로 압축
ex) 4096 vector



Show and Tell: A Neural Image Caption Generator

Image Captioning

Show and Tell

Image에 대한 정보를
Hidden state로 압축
ex) 4096 vector

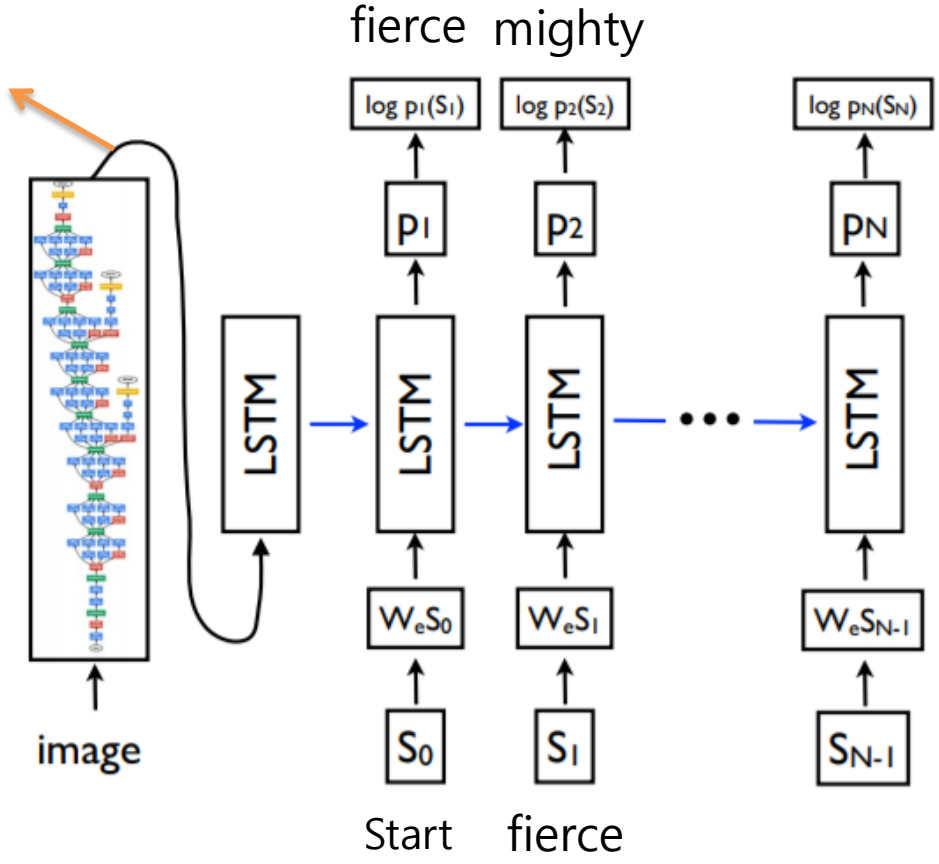
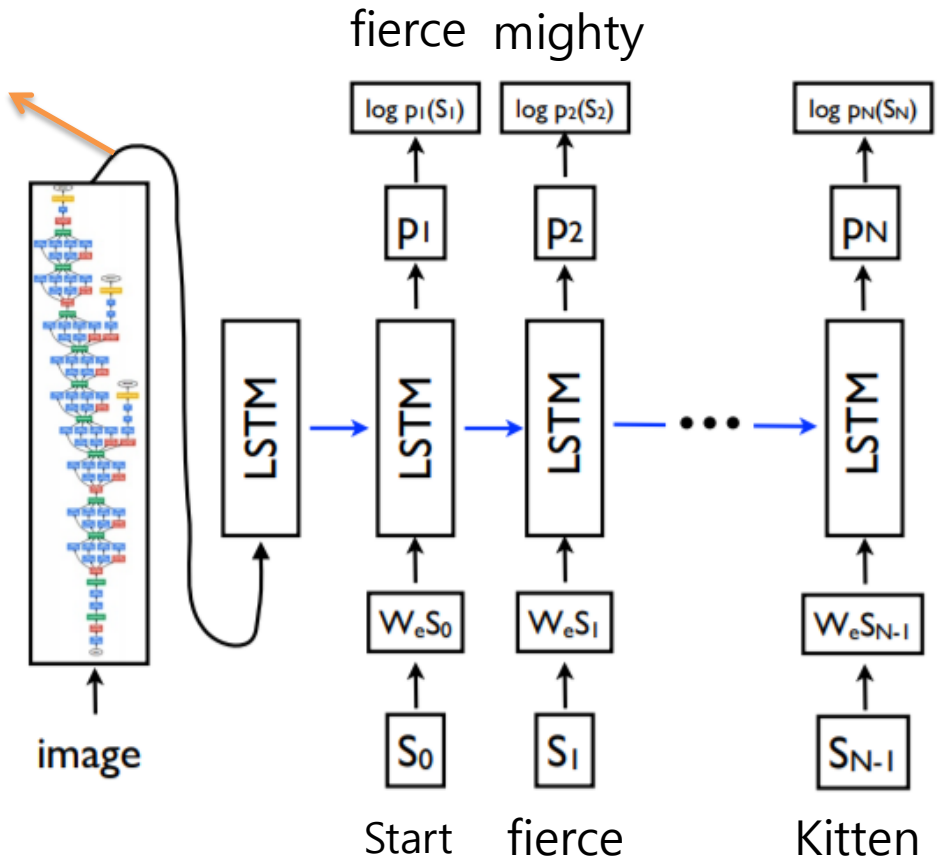


Image Captioning

Show and Tell

Image에 대한 정보를
Hidden state로 압축
ex) 4096 vector



Show and Tell: A Neural Image Caption Generator

Image Captioning

Show and Tell

Image에 대한 정보를
Hidden state로 압축
ex) 4096 vector

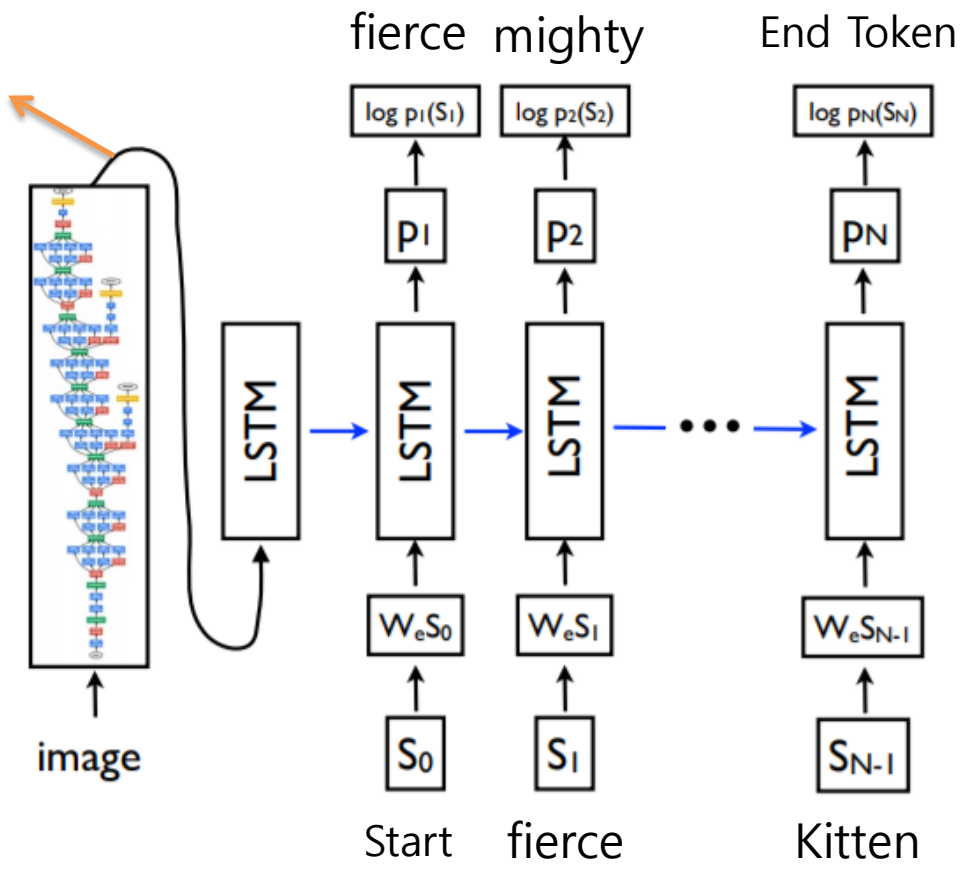
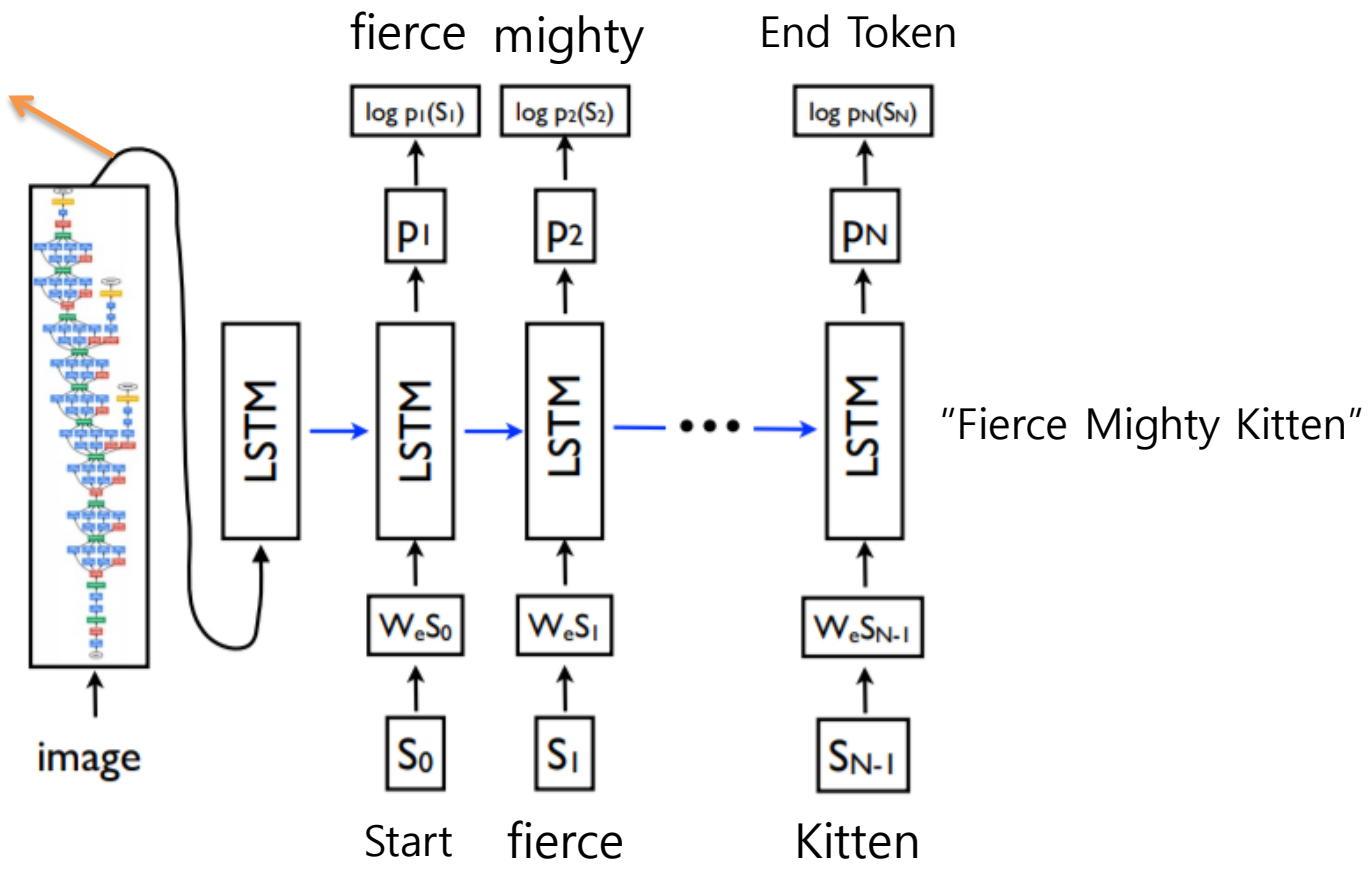


Image Captioning

Show and Tell

Image에 대한 정보를
Hidden state로 압축
ex) 4096 vector

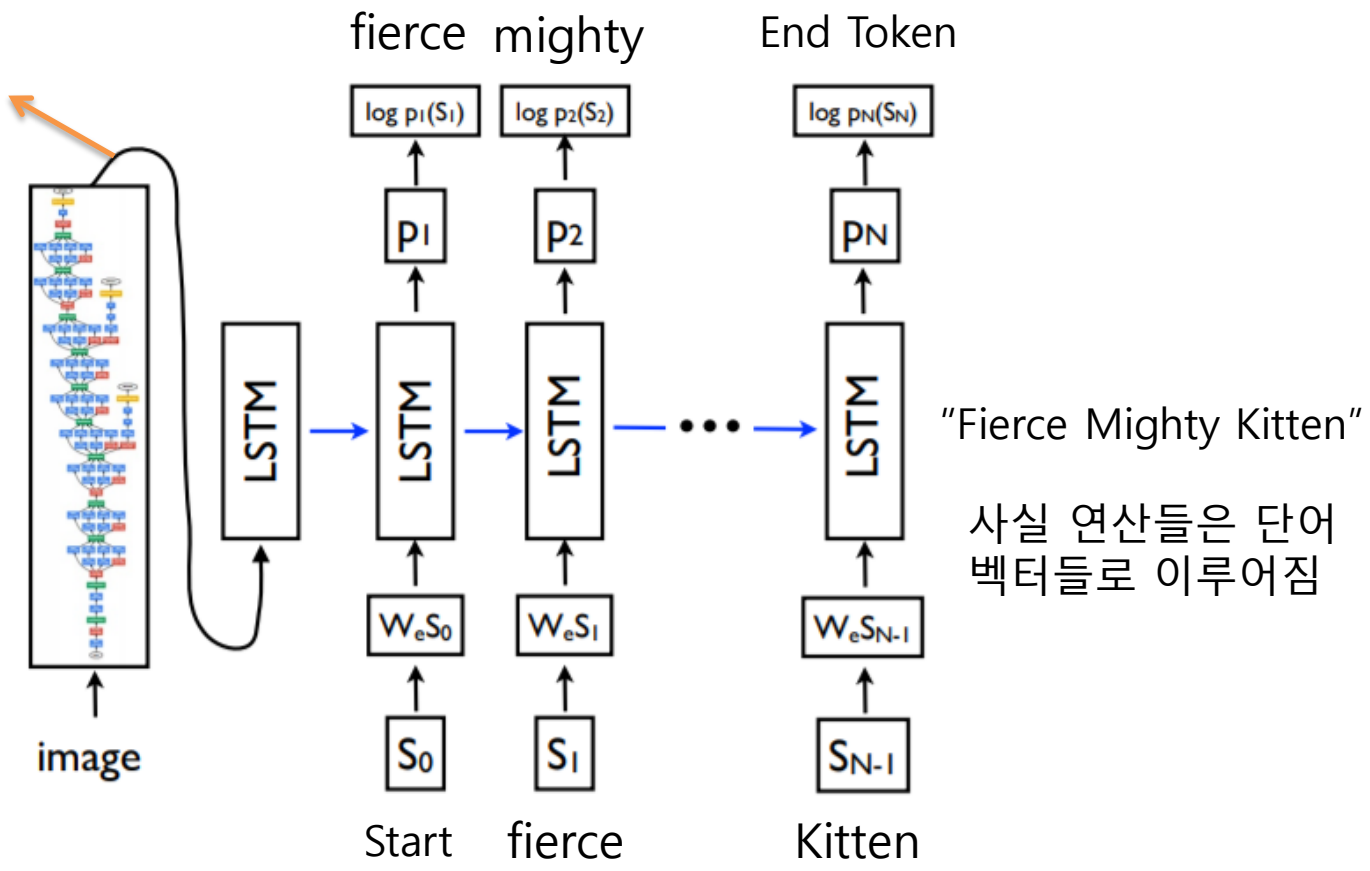


Show and Tell: A Neural Image Caption Generator

Image Captioning

Show and Tell

Image에 대한 정보를
Hidden state로 압축
ex) 4096 vector



"Fierce Mighty Kitten"

사실 연산들은 단어
벡터들로 이루어짐

Show and Tell: A Neural Image Caption Generator

Image Captioning

Neural Talk

A cat sitting on a suitcase on the floor

A cat is sitting on a tree branch

A dog is running in the grass with a frisbee

A white teddy bear sitting in the grass

Two people walking on the beach with surfboards

A tennis player in action on the court

Two giraffes standing in a grassy field

A man riding a dirt bike on a dirt track

Successful cases

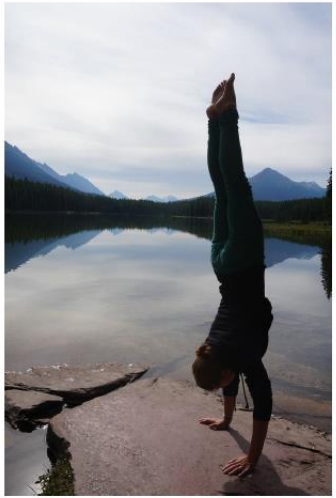
https://github.com/karpathy/neuraltalk2

Image Captioning

Neural Talk

A woman is holding a cat in her hand

A person holding a computer mouse on a desk



A woman standing on a beach holding a surfboard

A bird is perched on a tree branch

A man in a baseball uniform throwing a ball

Failure cases

Show and Tell: A Neural Image Caption Generator

Image Captioning

실제로 사람들은 어떻게 생각할까?

Image Captioning

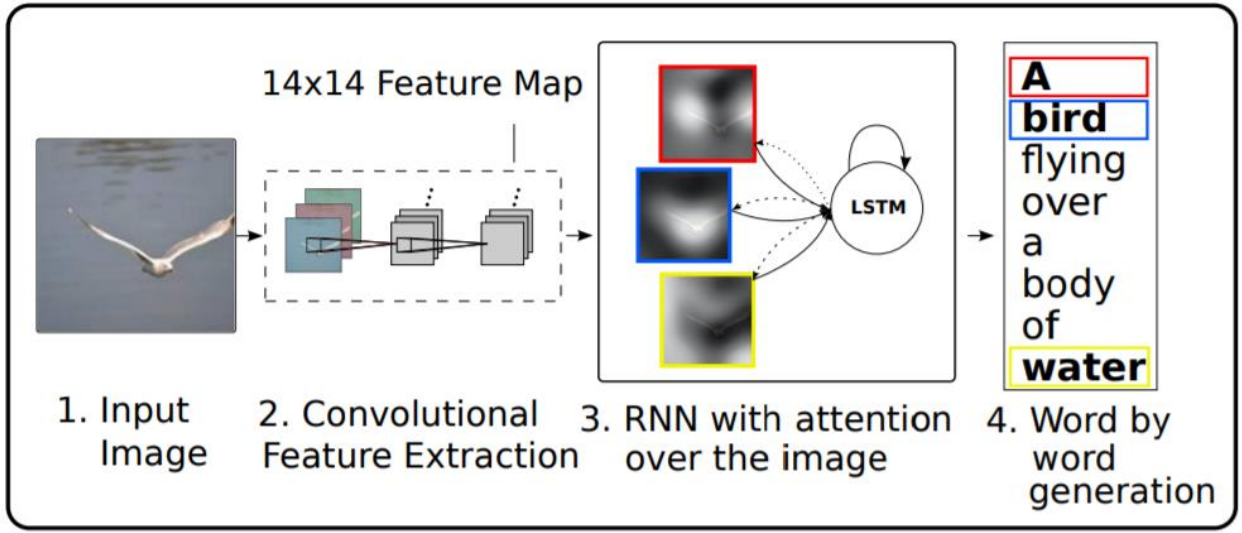
실제로 사람들은 어떻게 생각할까?

푸른 하늘과 들판의 나무 한 그루

Image Captioning

Show, Attend and Tell

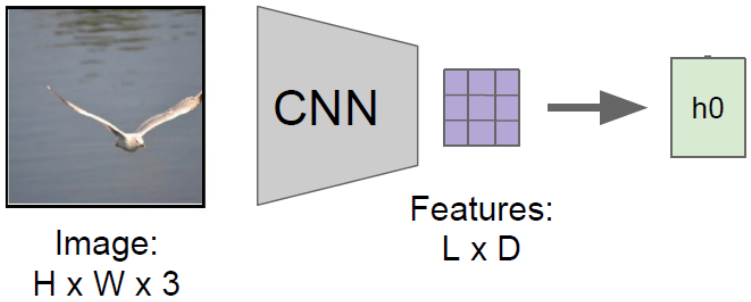
Figure 1. Our model learns a words/image alignment. The visualized attentional maps (3) are explained in section 3.1 & 5.4



기존의 Captioning model + Attention

Image Captioning

Show, Attend and Tell

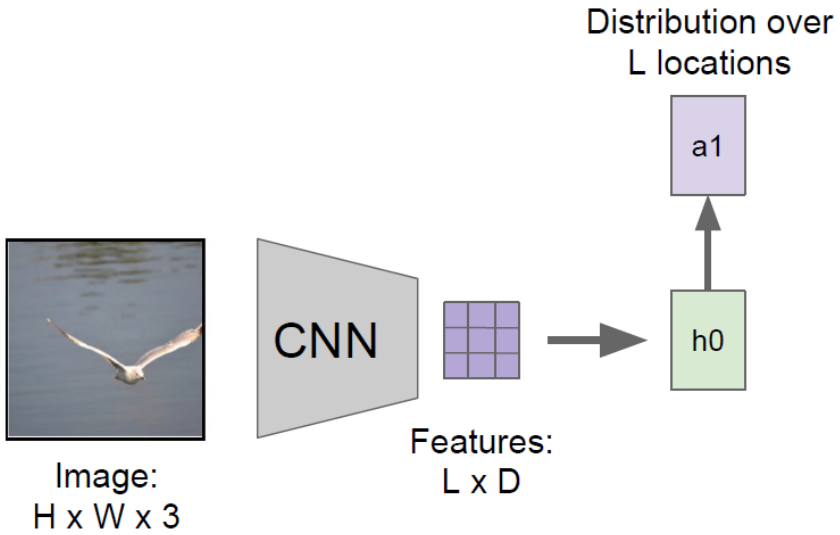


Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

출처: cs231n_2017_lecture10_Recurrent Neural Networks

Image Captioning

Show, Attend and Tell

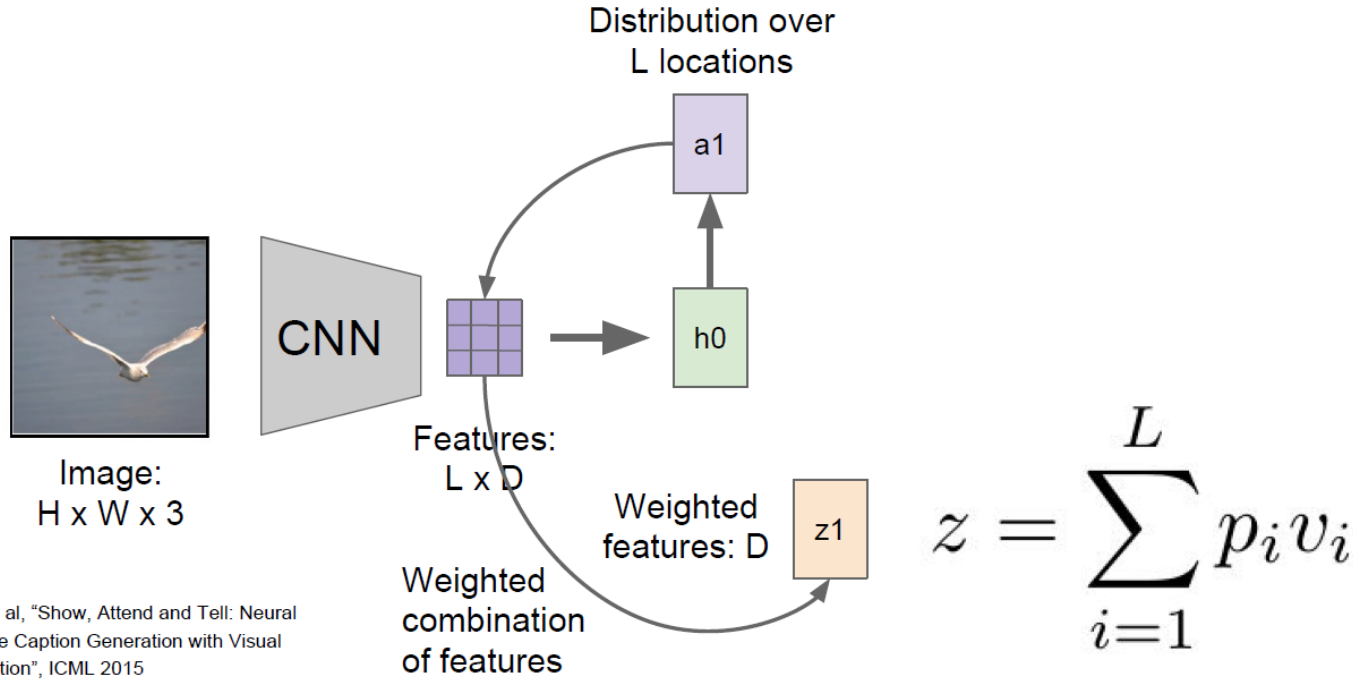


Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

출처: cs231n_2017_lecture10_Recurrent Neural Networks

Image Captioning

Show, Attend and Tell



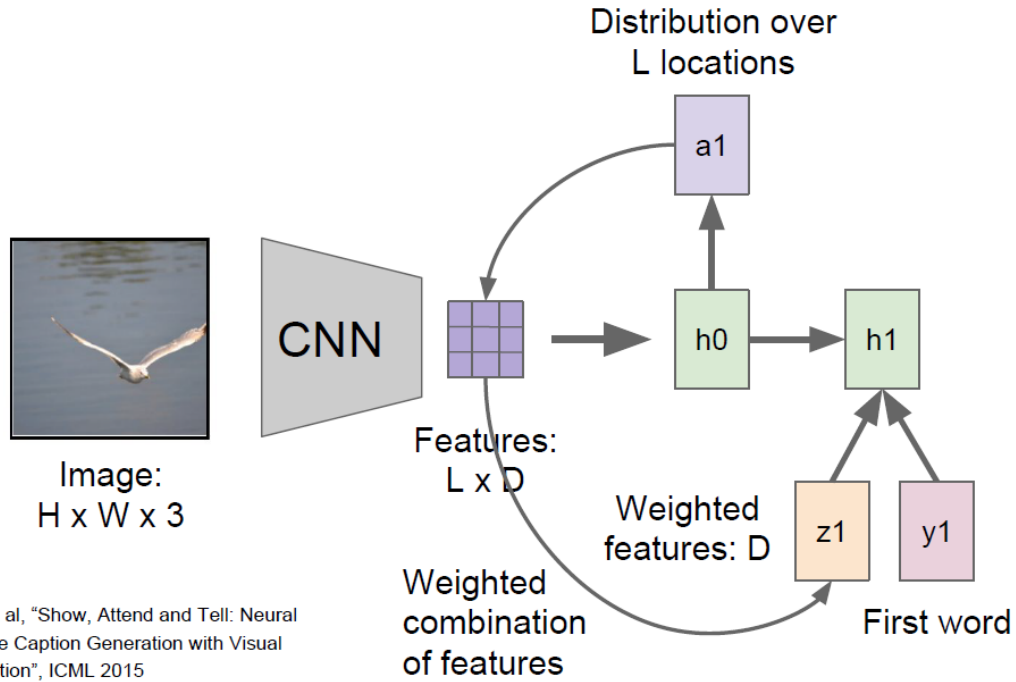
$$z = \sum_{i=1}^{L} p_i v_i$$

Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

출처: cs231n_2017_lecture10_Recurrent Neural Networks

Image Captioning

Show, Attend and Tell

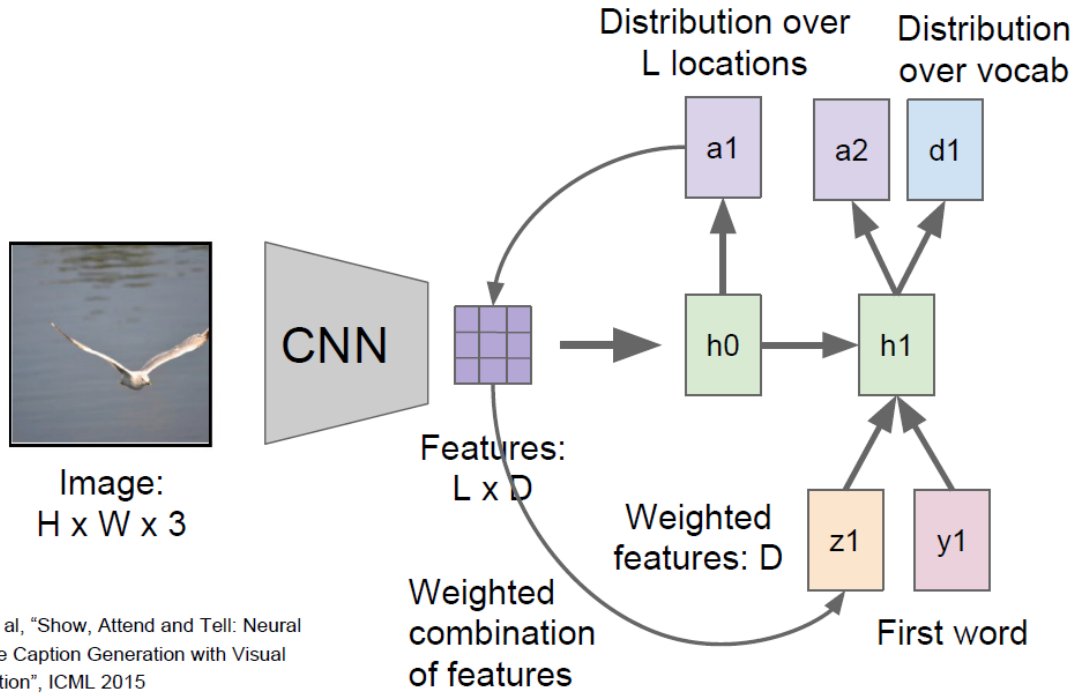


Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

출처: cs231n_2017_lecture10_Recurrent Neural Networks

Image Captioning

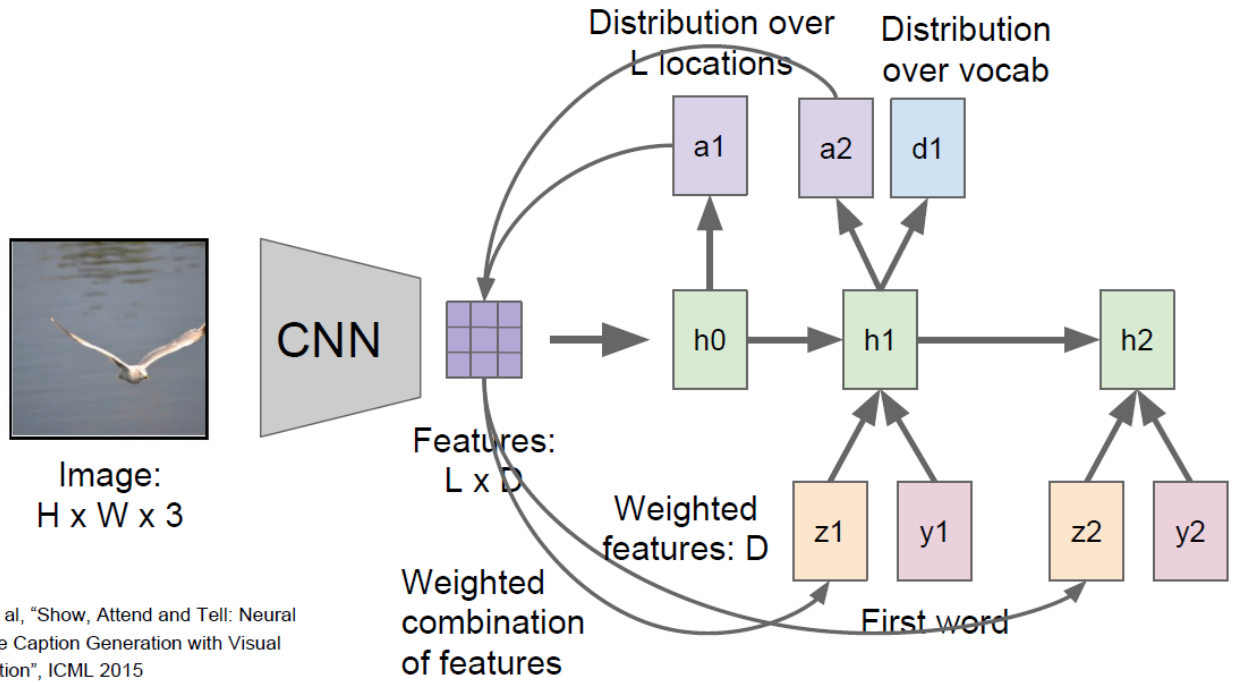
Show, Attend and Tell



Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Image Captioning

Show, Attend and Tell

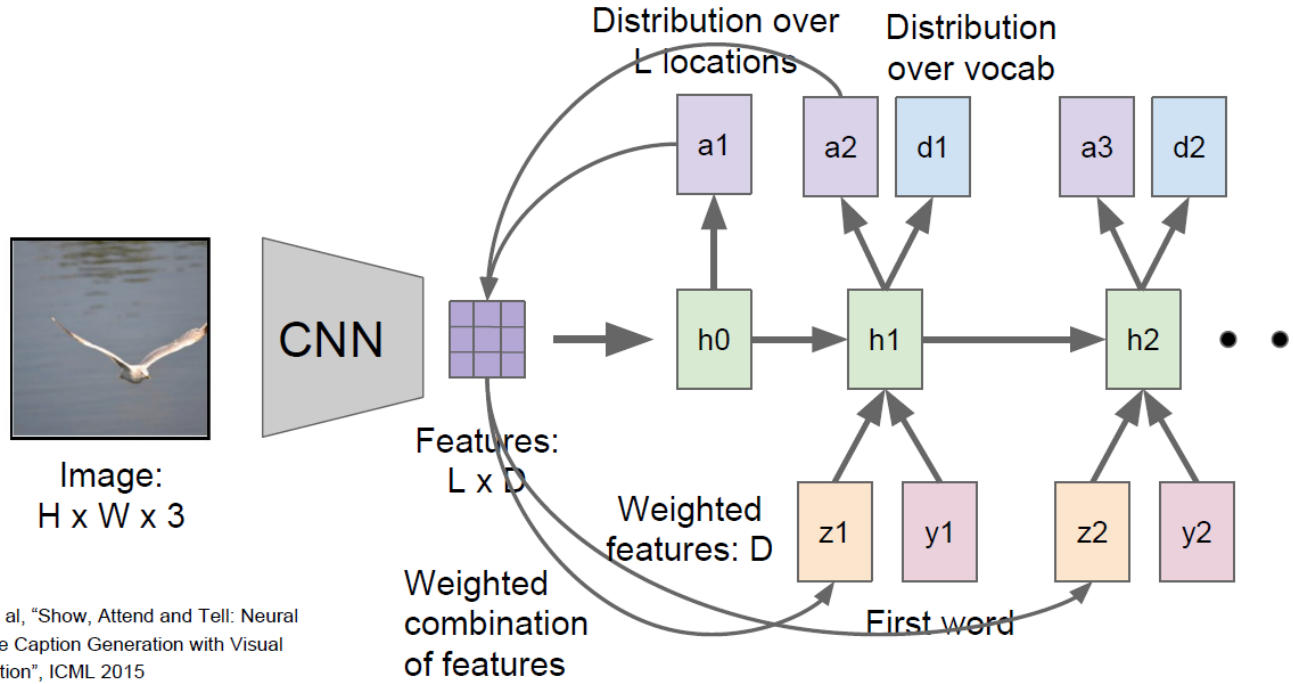


Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

출처: cs231n_2017_lecture10_Recurrent Neural Networks

Image Captioning

Show, Attend and Tell



Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

출처: cs231n_2017_lecture10_Recurrent Neural Networks

Image Captioning

Show, Attend and Tell

Figure 2. Attention over time. As the model generates each word, its attention changes to reflect the relevant parts of the image. "soft" (top row) vs "hard" (bottom row) attention. (Note that both models generated the same captions in this example.)

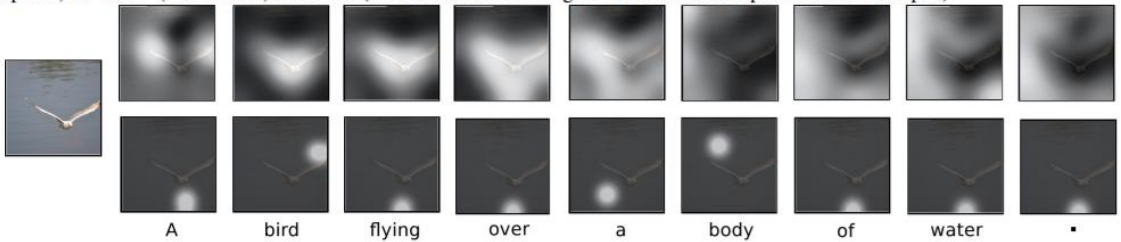
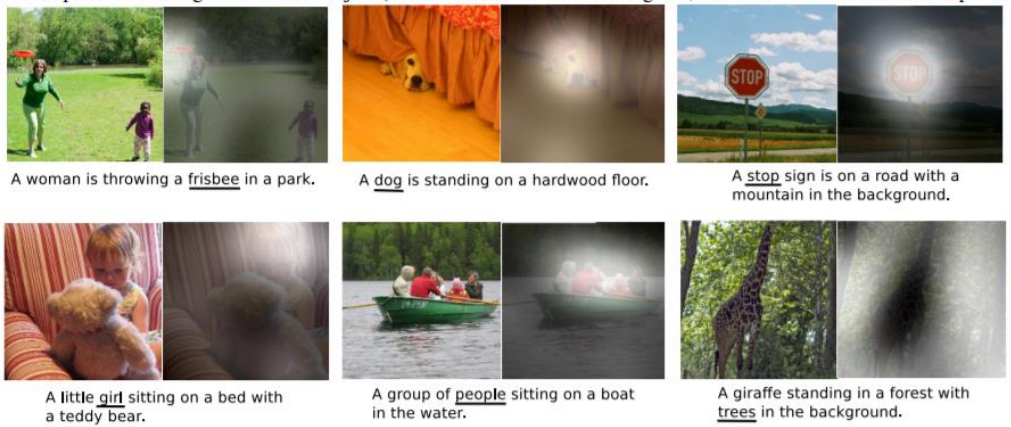


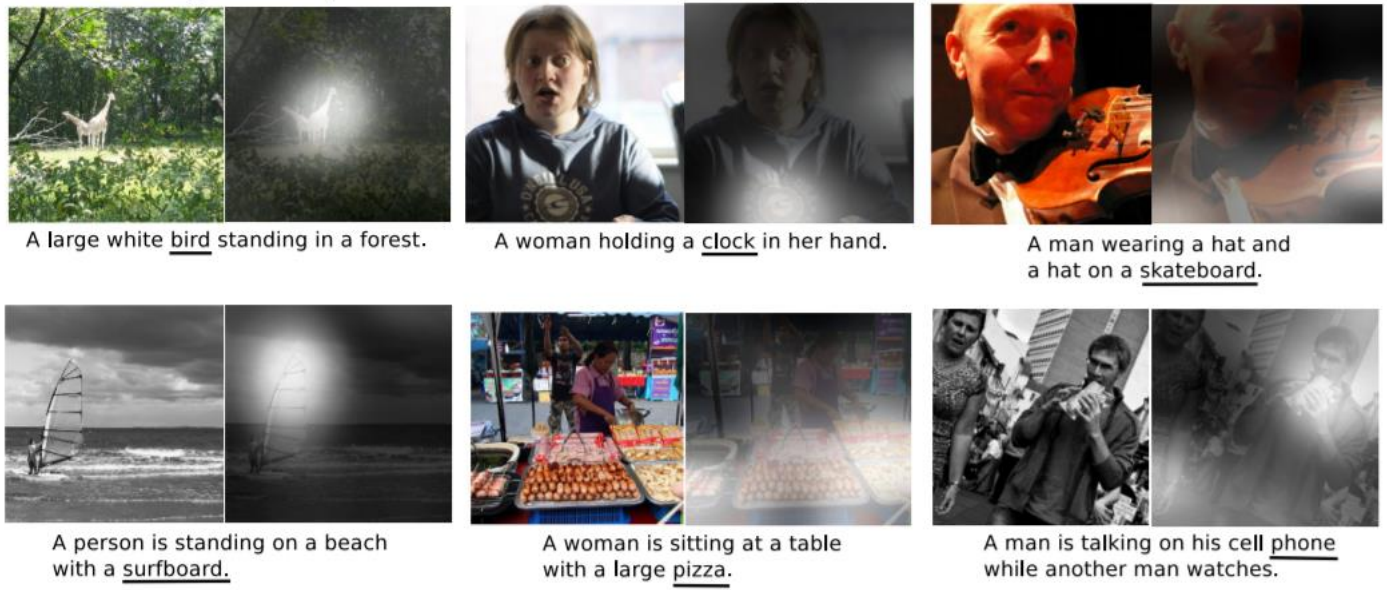
Figure 3. Examples of attending to the correct object (*white* indicates the attended regions, *underlines* indicated the corresponding word)



이 모델이 어디를 보고 해당 단어를 생성했는지 알 수 있음

Image Captioning

Figure 5. Examples of mistakes where we can use attention to gain intuition into what the model saw.



또한 어디를 잘못 보고 해당 단어를 생성했는지 알 수 있음

Image Captioning

Q: What did he just activated?

Image Captioning

Q: What did he just activated?
A: A trap card

Image Captioning

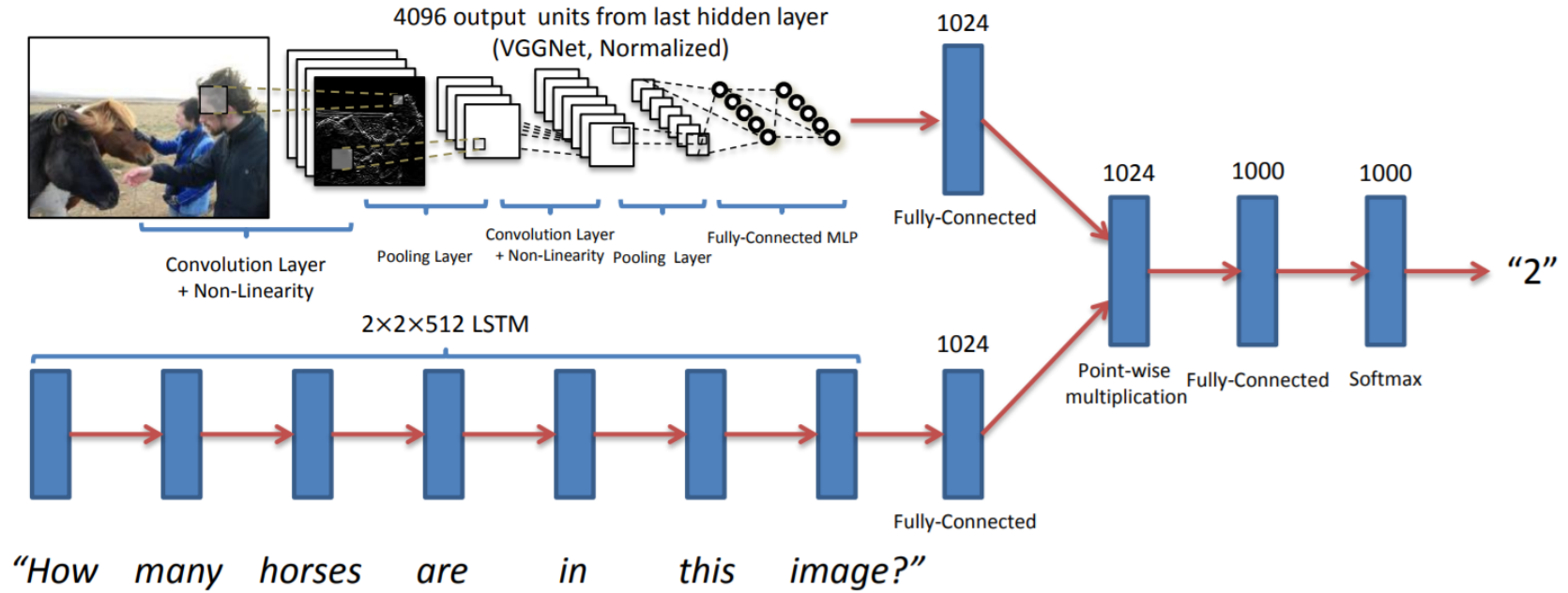
Q: What did he just activated?
A: A trap card

Does it appear to be rainy?
Does this person have 20/20 vision?

Visual Question and Answering

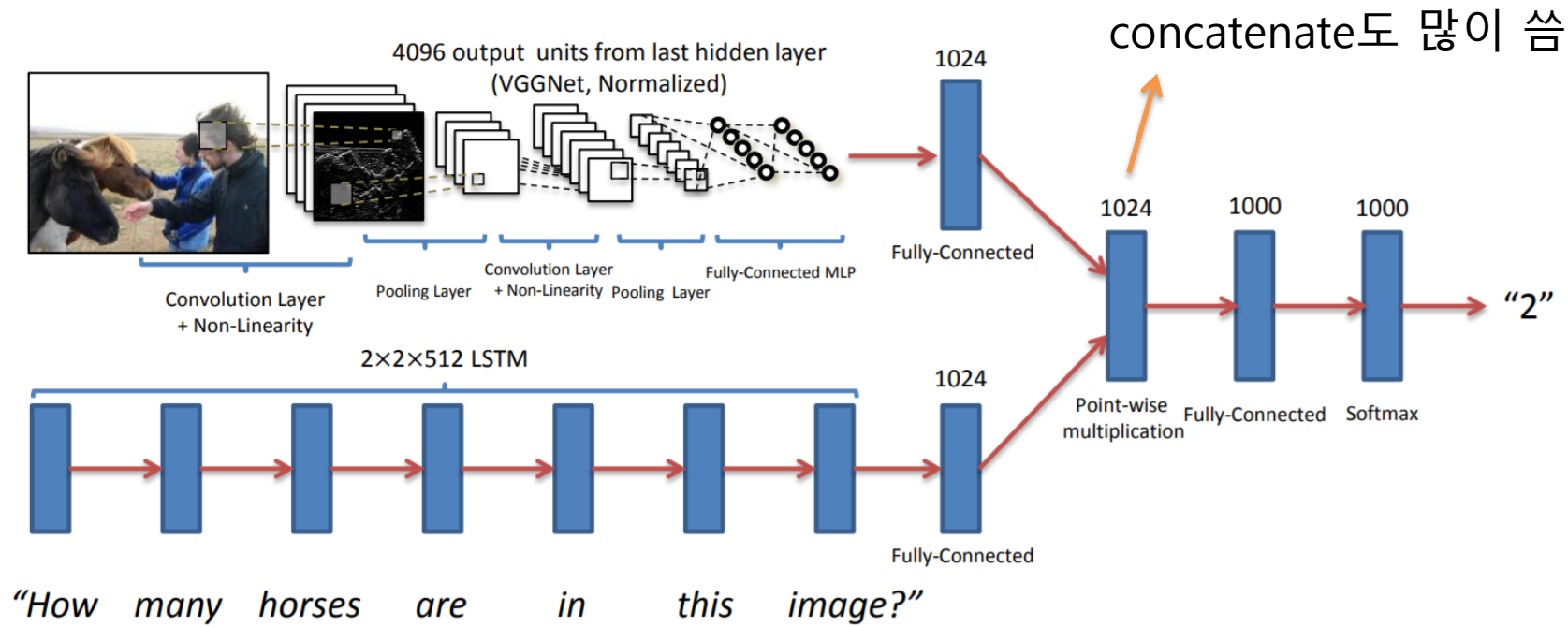
어떠한 이미지와 질문이
주어졌을 때 답을 맞추는 task

Image Captioning



기본적으로 이미지에 대한 embedding과 질문에 대한 embedding을 결합하여 답을 도출해 내는 구조를 가짐

Image Captioning



기본적으로 이미지에 대한 embedding과 질문에 대한 embedding을 결합하여 답을 도출해 내는 구조를 가짐

Image Captioning

VQA

CloudCV: Visual Question Answering (VQA)

CloudCV can answer questions you ask about an image

More details about the VQA dataset can be found here. Torch code for VQA is available here.

Browsers currently supported by the demo: Google Chrome, Mozilla Firefox.

Try VQA on Sample Images

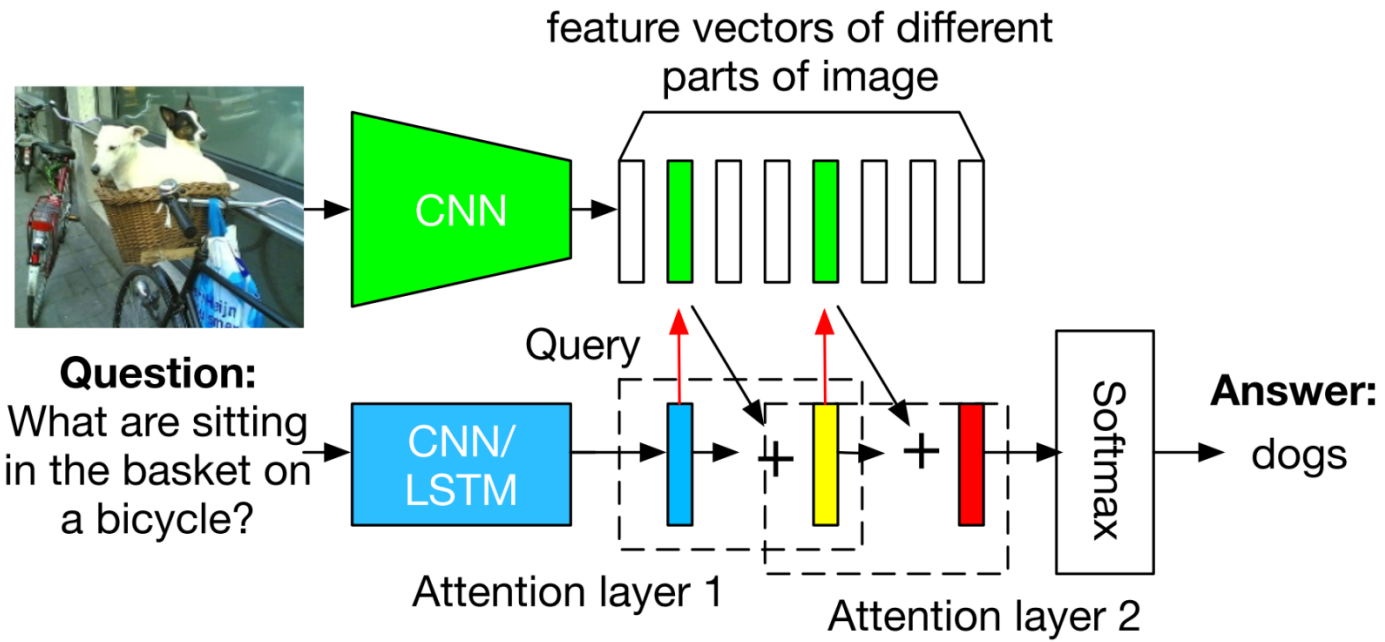
Click on one of these images to send it to our servers (Or upload your own images below)

Show Demo Images



https://vqa.cloudcv.org/

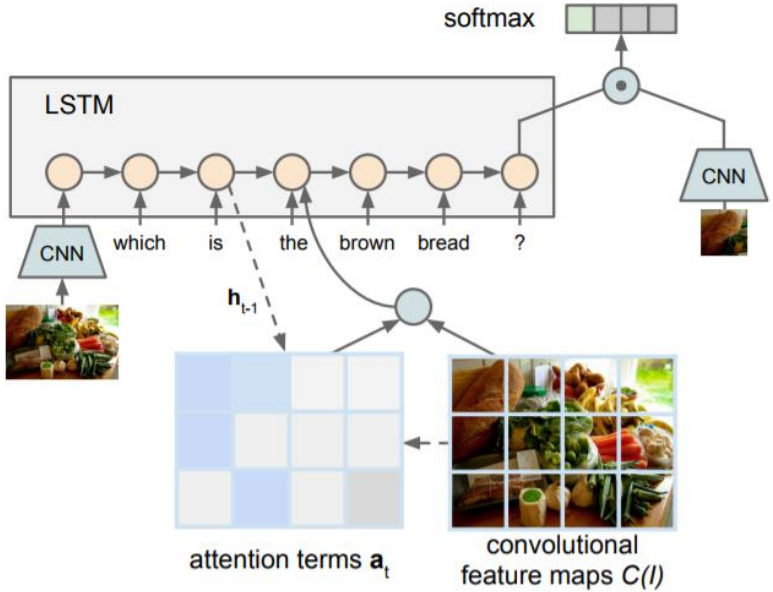
Image Captioning



기본 아키텍쳐 + Attention

https://github.com/abhshkdz/neural-vqa-attention

Image Captioning



Q: Which is the brown bread?

A1:  A2:  A3:  A4: 

문제 유형에 따라 embedding한
벡터끼리 비교를 통해 답을 구하기도 함

Image Captioning

Who is wearing glasses?

man woman

Where is the child sitting?

fridge arms

Is the umbrella upside down?

yes no

How many children are in the bed?

2 1

간단한 문제들

http://visualqa.org/static/img/vqa_examples.jpg

Image Captioning

VQA

Q: What endangered animal is featured on the truck?

A: **A bald eagle.**
A: A sparrow.
A: A humming bird.
A: A raven.

Q: Where will the driver go if turning right?

A: **Onto 24 ¾ Rd.**
A: Onto 25 ¾ Rd.
A: Onto 23 ¾ Rd.
A: Onto Main Street.

Q: When was the picture taken?

A: **During a wedding.**
A: During a bar mitzvah.
A: During a funeral.
A: During a Sunday church service.

Q: Who is under the umbrella?

A: **Two women.**
A: A child.
A: An old man.
A: A husband and a wife.

Q: Why was the hand of the woman over the left shoulder of the man?

A: **They were together and engaging in affection.**
A: The woman was trying to get the man's attention.
A: The woman was trying to scare the man.
A: The woman was holding on to the man for balance.

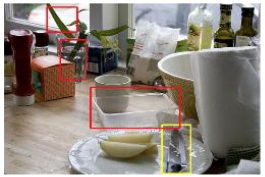
Q: How many magnets are on the bottom of the fridge?

A: **5.**
A: 2.
A: 3.
A: 4.

Q: Which pillow is farther from the window?

Q: Which step leads to the tub?

Q: Which is the small computer in the corner?



Q: Which item is used to cut items?

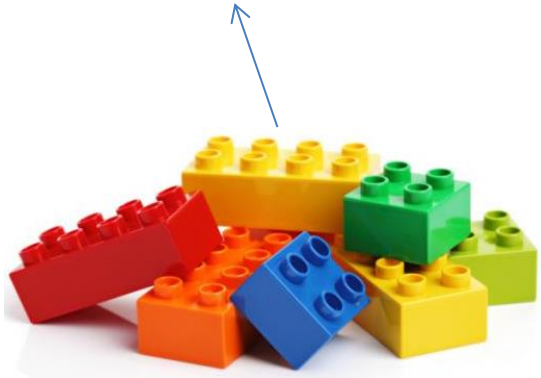
Q: Which doughnut has multicolored sprinkles?

Q: Which man is wearing the red tie?

Closing Remarks

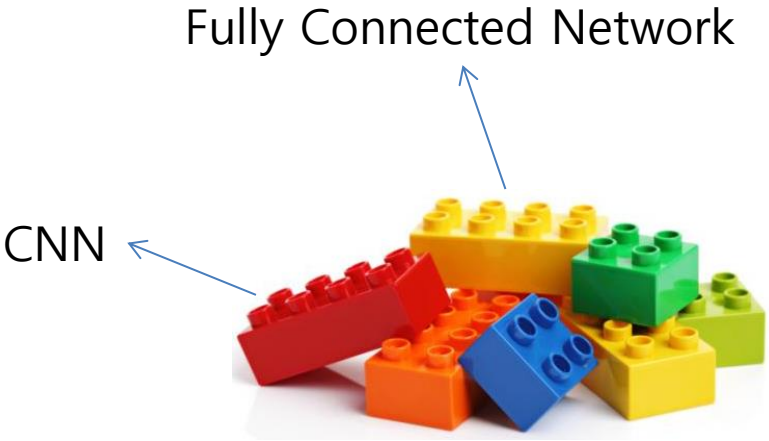
Closing Remarks

Fully Connected Network

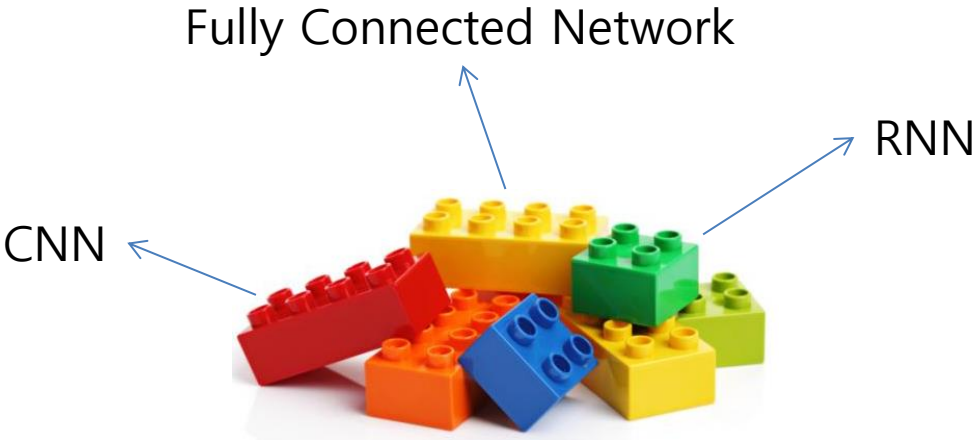


http://edpflager.com/?attachment_id=3579

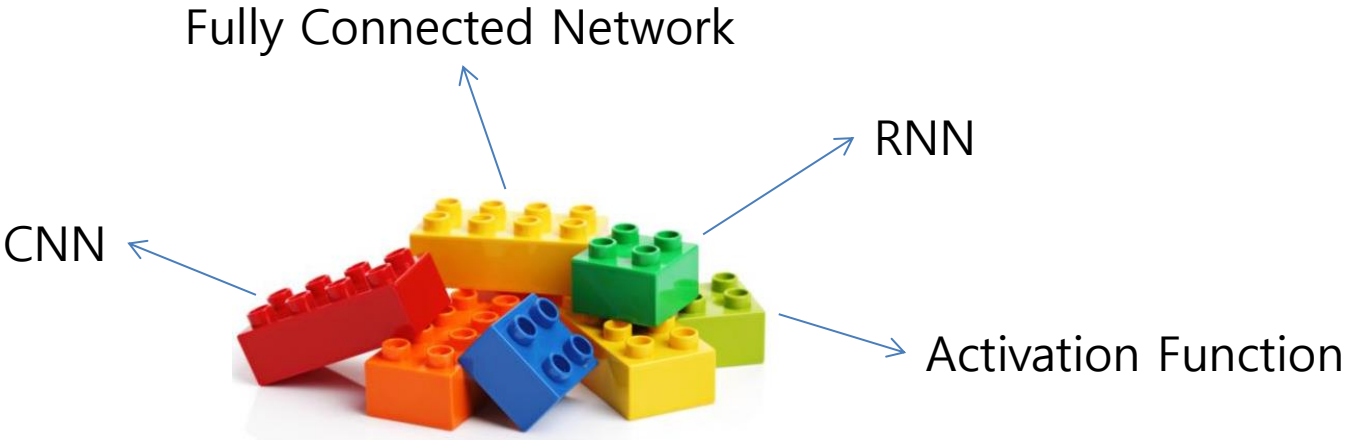
Closing Remarks



Closing Remarks

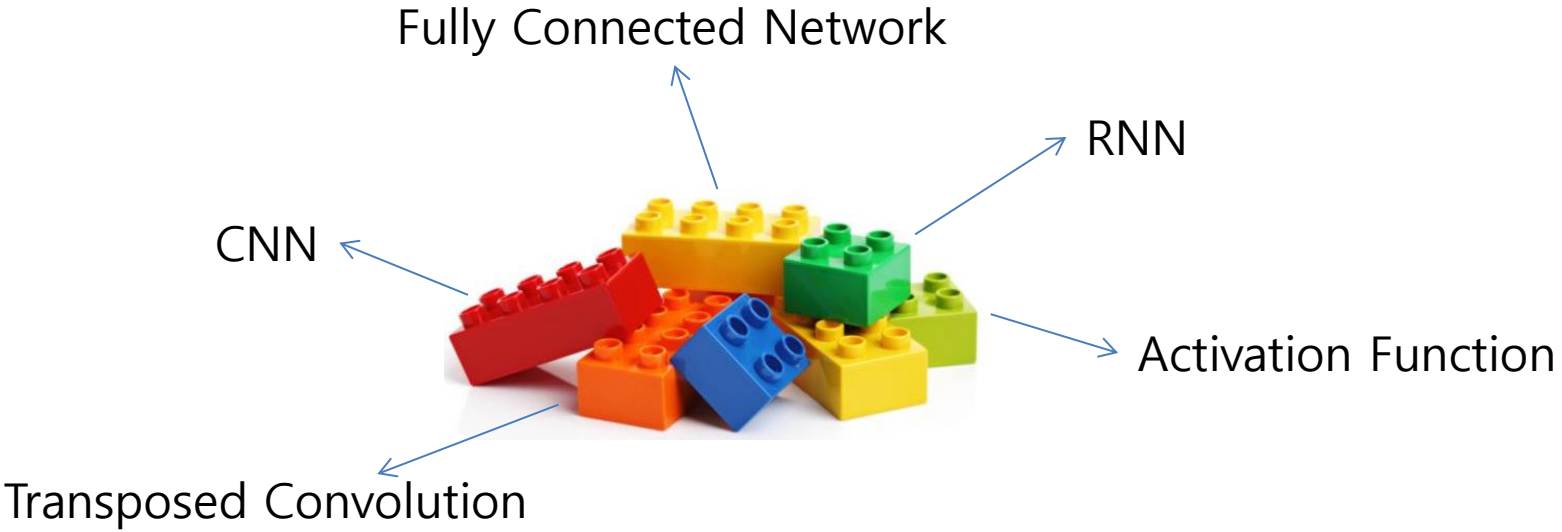


Closing Remarks

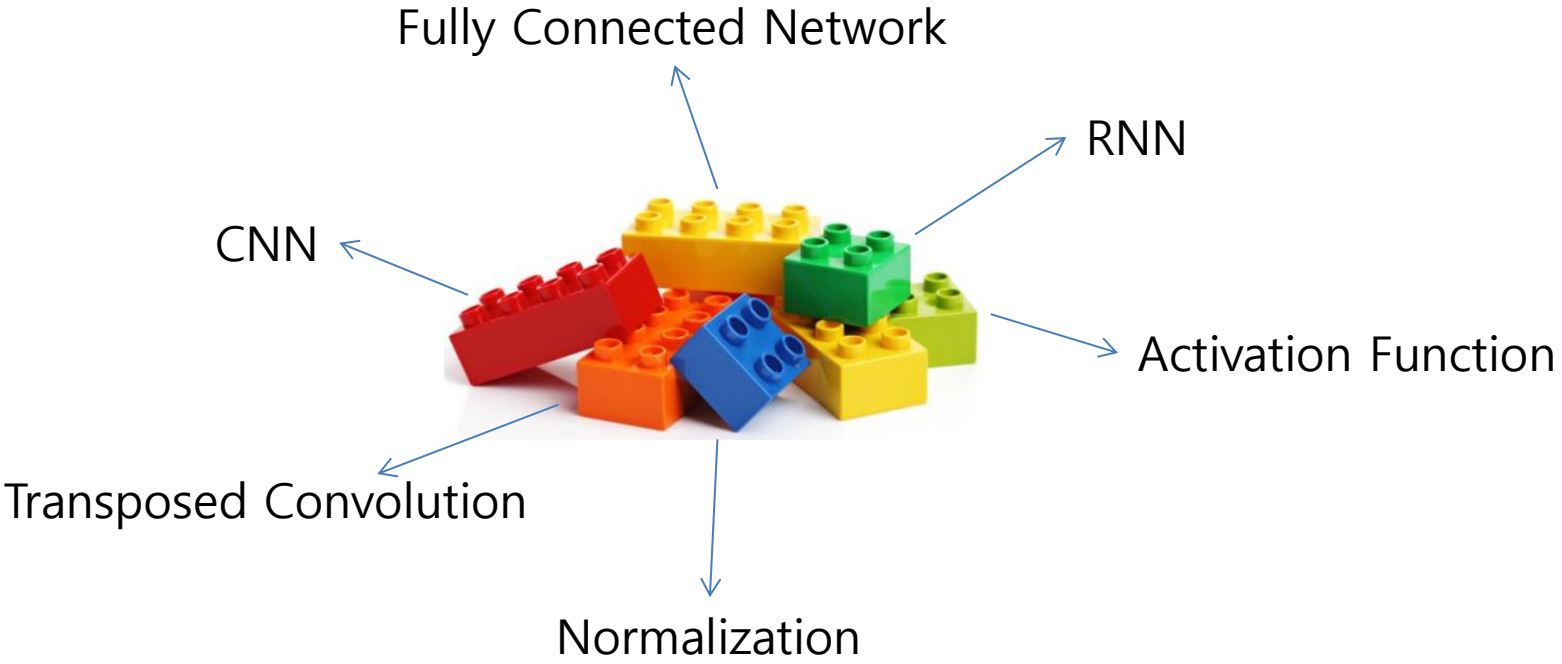


http://edpflager.com/?attachment_id=3579

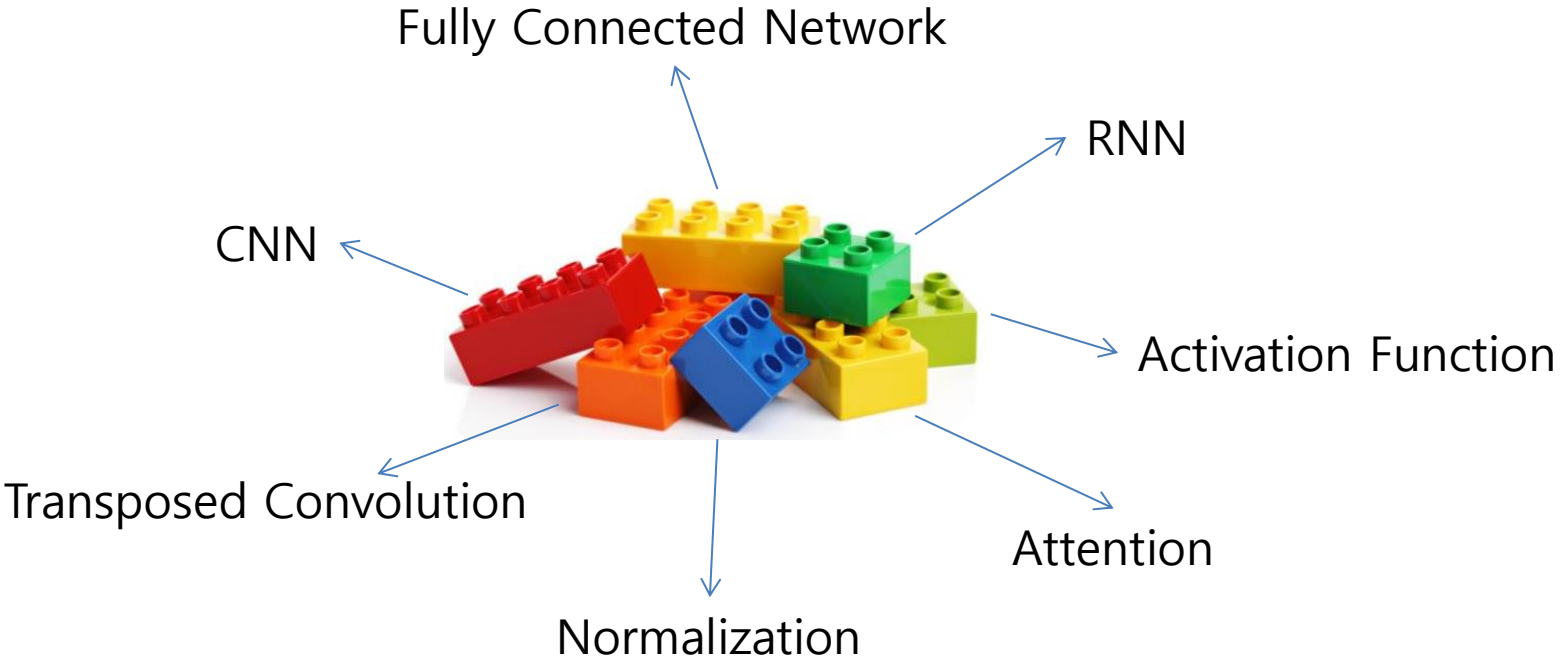
Closing Remarks



Closing Remarks



Closing Remarks



Closing Remarks

Closing Remarks

Anything Is Possible!!

http://www.cbr.com/this-life-size-lego-hulkbuster-is-made-from-1-million-bricks/

Q&A