
Introduction to Object Detection and Image Segmentation

발표자: 조용원

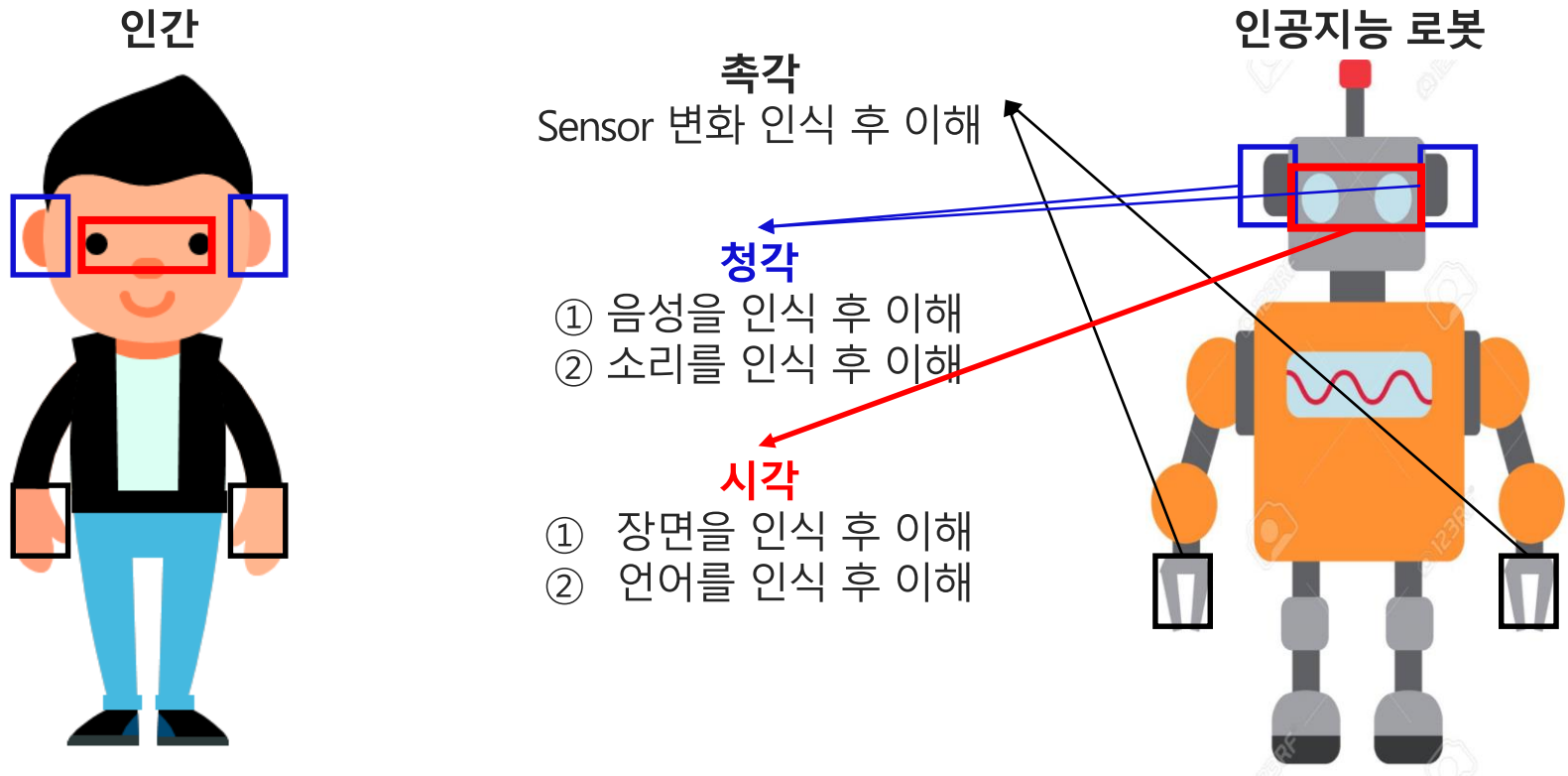
목차

1. Introduction
2. Object Detection
3. Image Segmentation
4. Mask R-CNN
5. Conclusions
6. Additional Topic

1. Introduction

❖ 인간과 인공지능

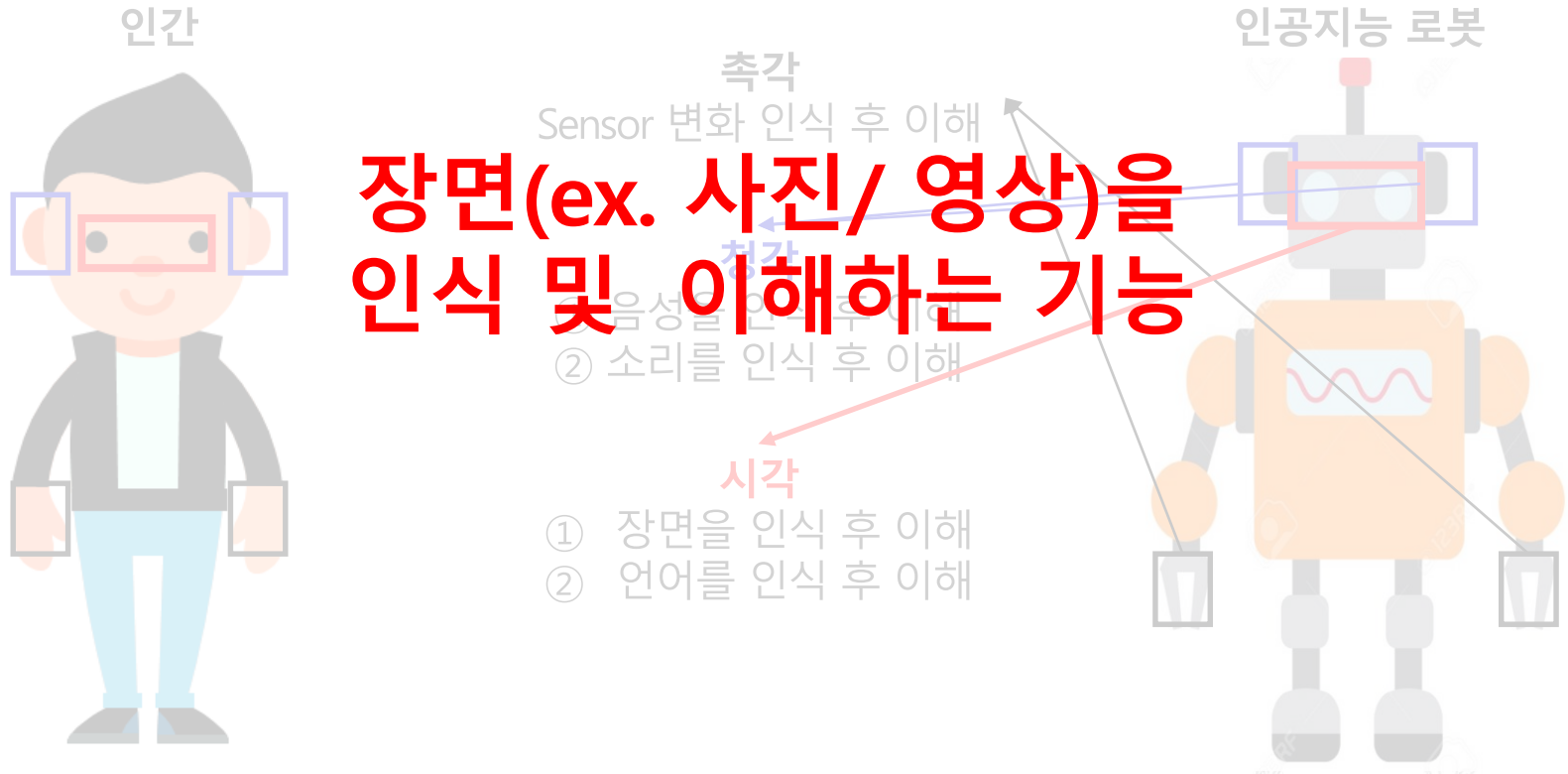
- 인간과 똑같은 인공지능 로봇을 개발한다면 어떤 기능이 있어야 할까?



1. Introduction

❖ 인간과 인공지능

- 인간과 똑같은 인공지능 로봇을 개발한다면 어떤 기능이 있어야 할까?



1. Introduction

❖ 컴퓨터 비전(Computer Vision)

- 사진/ 영상에서 정보를 추출 및 처리하고, 이해하고, 분석하는 알고리즘을 의미
- 인공지능 분야에서는 이미지 분류, 객체 탐지, 이미지 분할 등이 존재
- 대부분 지도학습 기반의 알고리즘으로 사진/ 영상에 대한 정답(Label)이 존재 해야 함

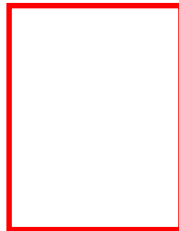
단일 객체(Single Object)

이미지 분류
(Image Classification)



올라프

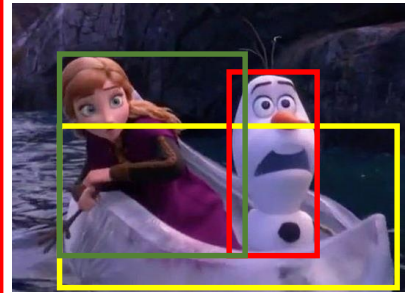
이미지 분류 및 위치 파악
(Image Localization)



올라프

다중 객체(Multiple Object)

객체 탐지
(Object Detection)

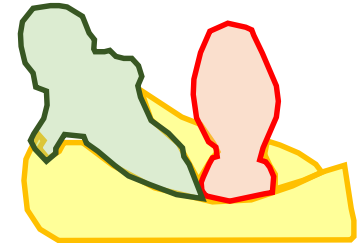


올라프

안나

보트

이미지 분할
(Image Segmentation)



올라프

안나

보트

목차

1. Introduction
- 2. Object Detection**
3. Image Segmentation
4. Mask R-CNN
5. Conclusions
6. Additional Topic

2. Object Detection

❖ 객체 탐지(Object Detection)

- 사전에 정의한 범주가 사진에 존재하는지 결정하기
- 사전에 정의한 범주가 사진에 **존재한다면**, 사진 속에서 위치(**Bounding Box**)를 찾기

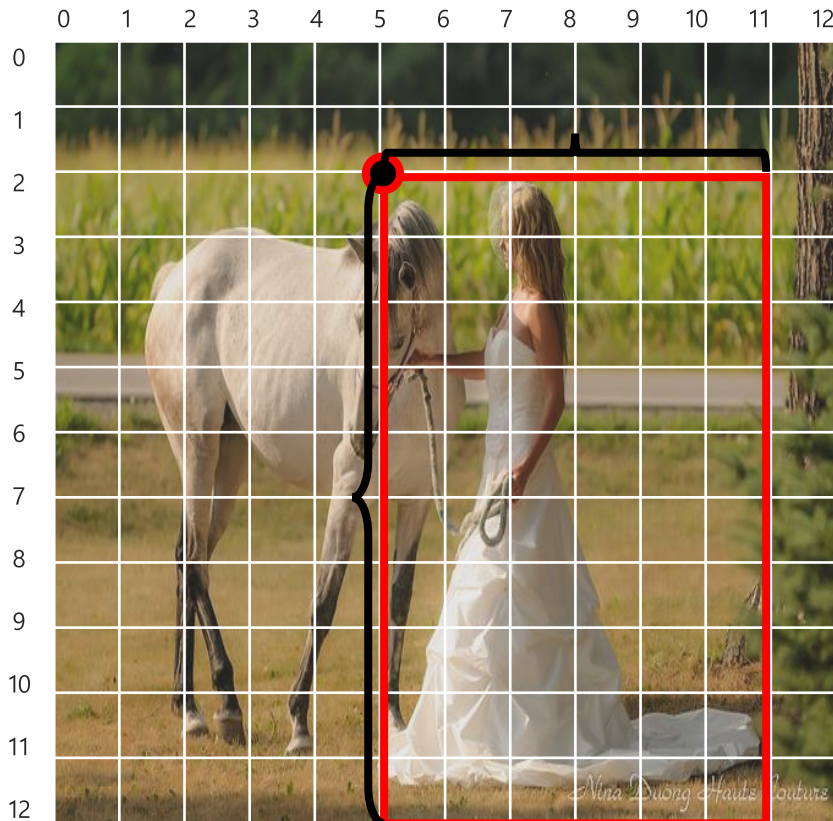


- Deep Learning for Generic Object Detection: A Survey(2018), Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, Matti Pietikäinen

2. Object Detection

❖ 객체 탐지(Object Detection) 레이블링(Labeling)

- **Labeling:** 사진에 대해, 사전에 정의한 범주와 해당 범주의 위치를 지정해주는 과정
- 지도 학습 기반의 Object Detection에서 레이블이 없다면, 예측 모델 구축이 불가능



□ 사람

$(X, Y, W, H) = ?$

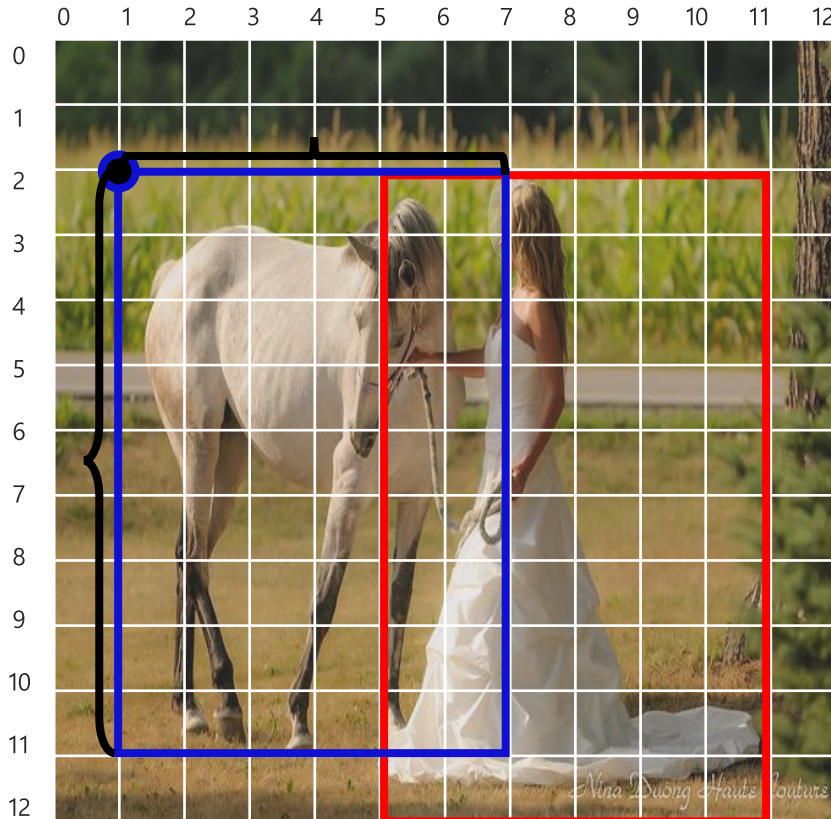
● $(X, Y) = (5, 2)$

— $(W, H) = (6, 10)$

2. Object Detection


❖ 객체 탐지(Object Detection) 레이블링(Labeling)


- **Labeling:** 사진에 대해, 사전에 정의한 범주와 해당 범주의 위치를 지정해주는 과정
- 지도 학습 기반의 Object Detection에서 레이블이 없다면, 예측 모델 구축이 불가능



 **사람**


$(X, Y, W, H, \text{범주}) = (2, 5, 6, 10, \text{사람})$

 $(X, Y) = (5, 2)$

 $(W, H) = (6, 10)$

 **말**

$(X, Y, W, H) = ?$

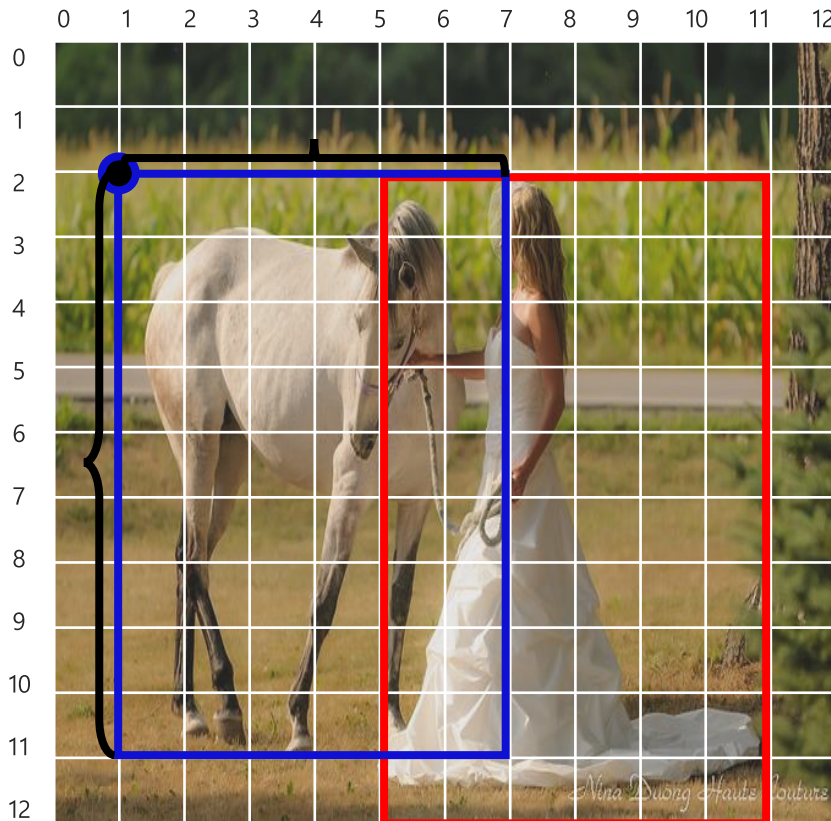
 $(X, Y) = (1, 2)$

 $(W, H) = (6, 9)$

2. Object Detection


❖ 객체 탐지(Object Detection) 레이블링(Labeling)


- **Labeling:** 사진에 대해, 사전에 정의한 범주와 해당 범주의 위치를 지정해주는 과정
- 지도 학습 기반의 Object Detection에서 레이블이 없다면, 예측 모델 구축이 불가능



 **사람**


$(X, Y, W, H, \text{범주}) = (2, 5, 6, 10, \text{사람})$

 $(X, Y) = (5, 2)$

 $(W, H) = (6, 10)$

 **말**

$(X, Y, W, H, \text{범주}) = (1, 2, 6, 9, \text{말})$

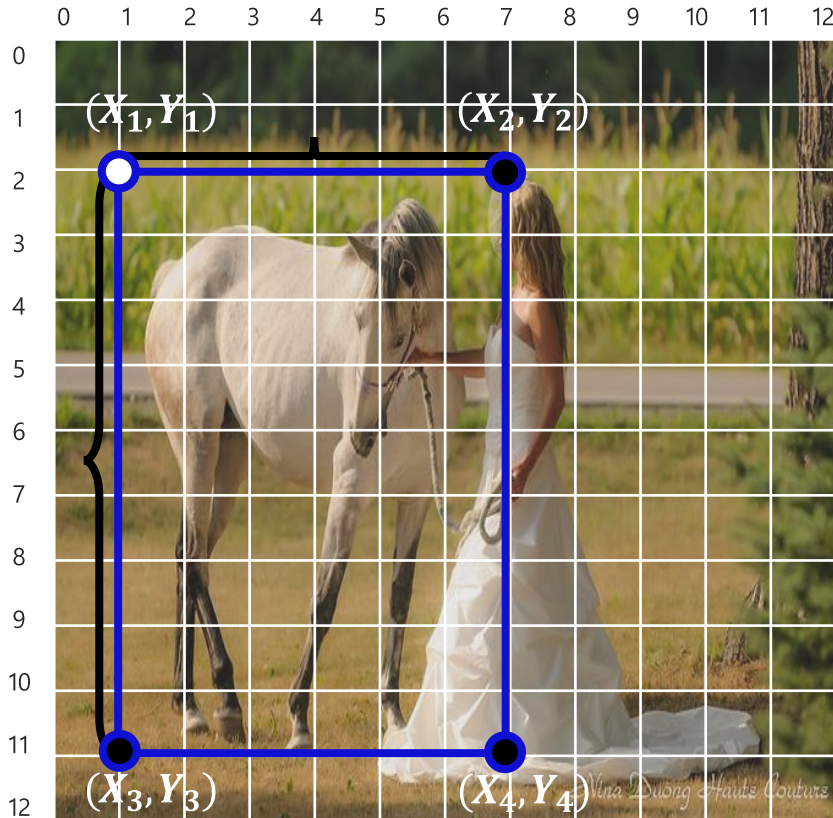
 $(X, Y) = (1, 2)$

 $(W, H) = (6, 9)$

2. Object Detection

❖ 객체 탐지(Object Detection) 레이블

- Bounding Box에 대한 예측을 왜 X, Y, W, H 로 진행하는가?



말

(범주) = (말)

● $(X_1, Y_1), (X_2, Y_2),$

$(X_3, Y_3), (X_4, Y_4)$

→ 9개



말

$(X, Y, W, H, \text{범주}) = (1, 2, 6, 9, \text{말})$

● $(X, Y) = (1, 2)$

→ 5개

(X, Y, W, H) 를 예측하는 것이 효율적

2. Object Detection

❖ R-CNN

- 2014년 **Computer Vision and Pattern Recognition(CVPR)**에서 발표된 논문
- Object Detection을 수행하기 위해, 처음 합성곱 신경망(CNN)을 사용한 방법론
- 2020년 1월 28일 기준으로 **11511회** 인용

Rich feature hierarchies for accurate object detection and semantic segmentation

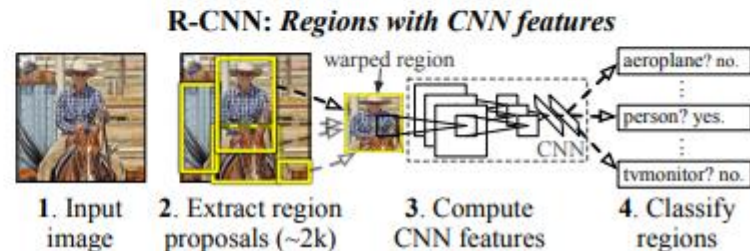
Tech report (v5)

Ross Girshick Jeff Donahue Trevor Darrell Jitendra Malik
UC Berkeley

{rbg, jdonahue, trevor, malik}@eecs.berkeley.edu

Abstract

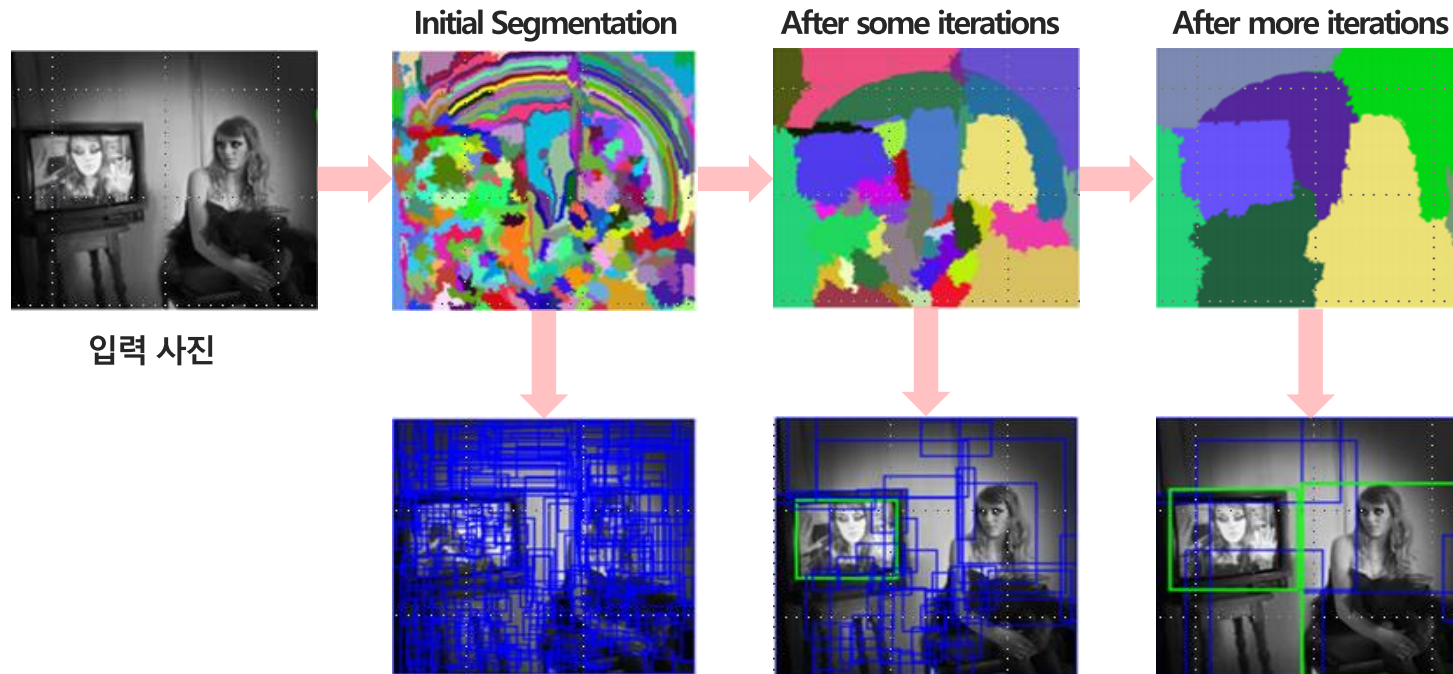
Object detection performance, as measured on the canonical PASCAL VOC dataset, has plateaued in the last few years. The best-performing methods are complex ensemble systems that typically combine multiple low-level image features with high-level context. In this paper, we



2. Object Detection

❖ Proposal Region

- 사진 내 객체가 존재할만한 영역
- Selective Search 방법을 이용해 입력 이미지 상에서 Proposal Region 추출
- 픽셀을 군집화 후, 군집마다 유사한 경우 이들을 통합해가며 Proposal Region 생성

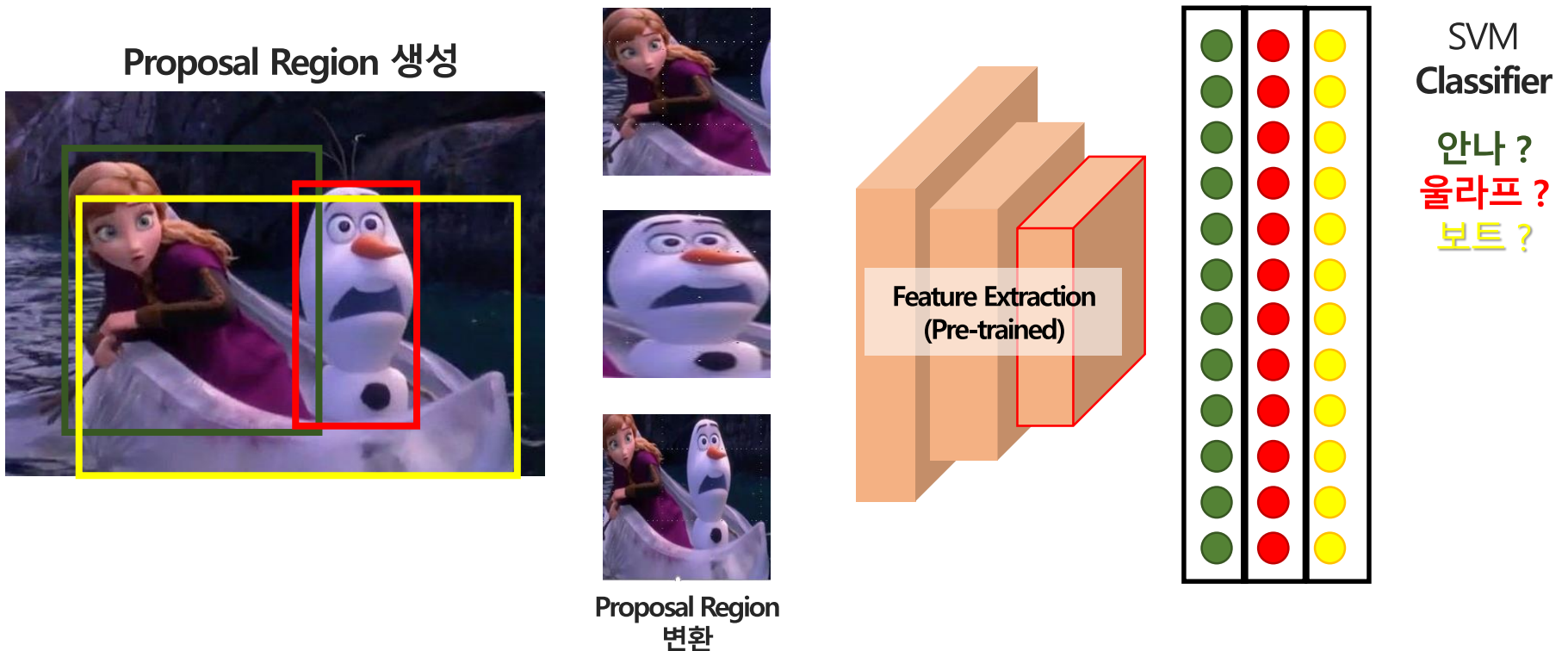


- Felzenszwalb, P. F., & Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. International journal of computer vision, 59(2), 167-181.

2. Object Detection

❖ R-CNN - Classifier

- Selective Search를 이용해 2,000개의 Proposal Region을 입력 이미지에서 생성
- Proposal Region을 고정된 크기의 이미지로 변환 후, Pre-trained CNN에 입력
- Feature를 저장했다가, SVM Classifier 학습할 때 사용

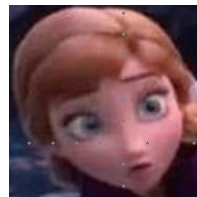
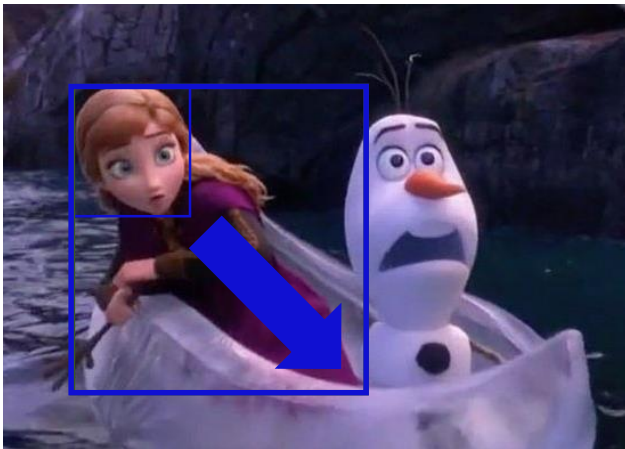


2. Object Detection

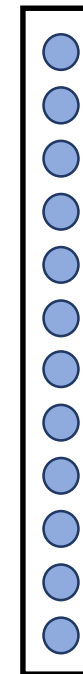
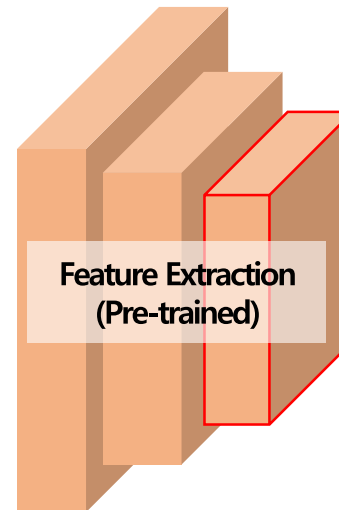
❖ R-CNN – Bounding Box Regression

- Feature 저장 시, Proposal Region-X, Y, W, H를 저장하고, 이를 **예측 Bounding Box**라고 함
- **예측 Bounding Box**와 **실제 Bounding Box**를 일치시키고자 하는 Linear Regression 구축
- Bounding Box를 예측하는 모델을 Bounding Box Regression이라 함

Proposal Region 추출



Proposal Region
변환



SVM
Classifier

안나!
올라프?
보트?

Ridge
Regression

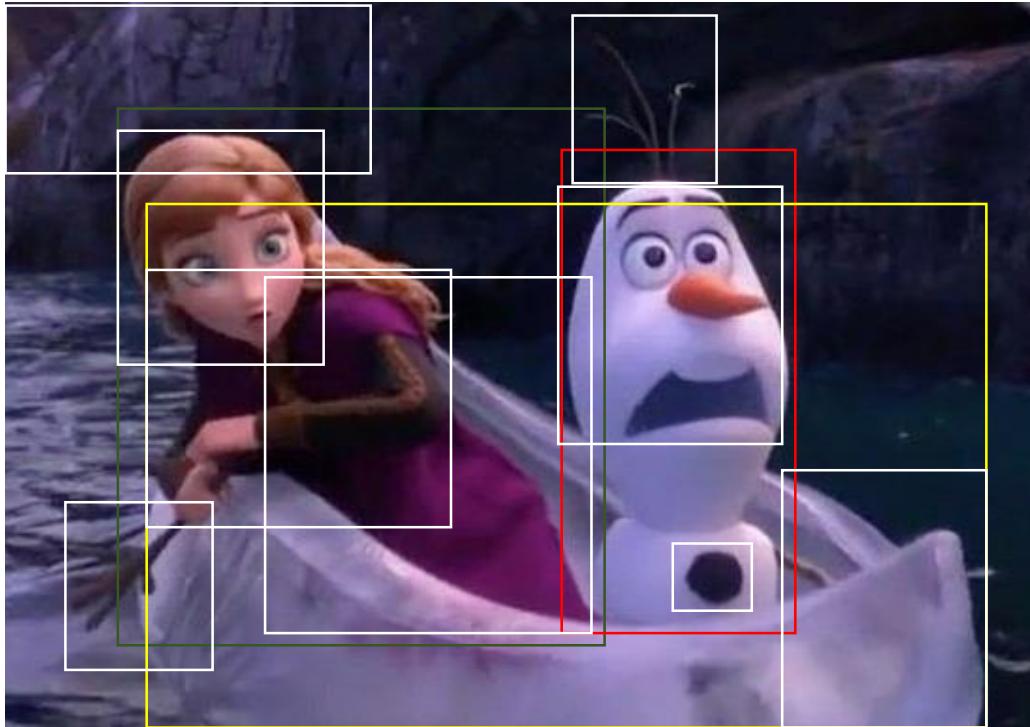


**Bounding Box
Regression**

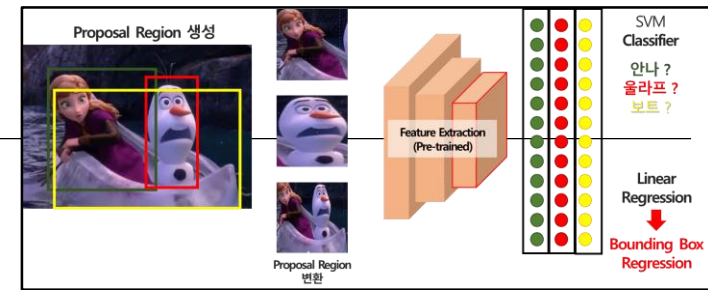
2. Object Detection

❖ R-CNN 문제점

- Selective Search 방법은 CPU를 사용하기에 많은 시간 필요
- 또한 Proposal Region이 탐지하고자 하는 객체가 없는 경우도 존재
- 모든 Proposal Region마다 CNN에 입력하는 것은 불 필요함



2. Object Detection



❖ Fast R-CNN

- 2015년 Computer Vision and Pattern Recognition(CVPR)에서 발표된 논문
- R-CNN을 개발했던 저자(Ross Girshick)가 본인의 방법론을 발전시킴
- 2020년 1월 28일 기준으로 **8347회** 인용

Fast R-CNN

Ross Girshick
Microsoft Research

rbg@microsoft.com

Abstract

This paper proposes a Fast Region-based Convolutional Network method (Fast R-CNN) for object detection. Fast R-CNN builds on previous work to efficiently classify object proposals using deep convolutional networks. Compared to previous work, Fast R-CNN employs several innovations to improve training and testing speed while also increasing detection accuracy. Fast R-CNN trains the very deep VGG16 network 9× faster than R-CNN, is 213× faster at test-time, and achieves a higher mAP on PASCAL VOC 2012. Compared to SPPnet, Fast R-CNN trains VGG16 3× faster, tests 10× faster, and is more accurate. Fast R-CNN is implemented in Python and C++ (using Caffe) and is available under the open-source MIT License at <https://github.com/rbgirshick/fast-rcnn>.

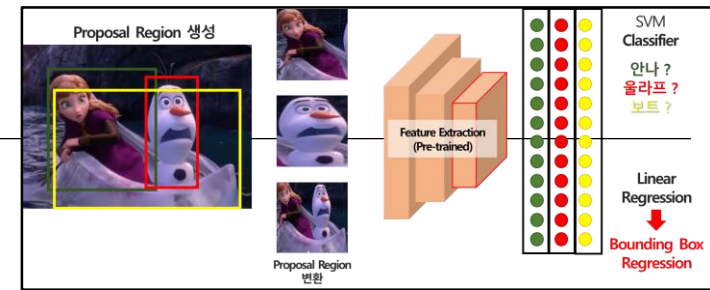
while achieving top accuracy on PASCAL VOC 2012 [7] with a mAP of 66% (vs. 62% for R-CNN).¹

1.1. R-CNN and SPPnet

The Region-based Convolutional Network method (R-CNN) [9] achieves excellent object detection accuracy by using a deep ConvNet to classify object proposals. R-CNN, however, has notable drawbacks:

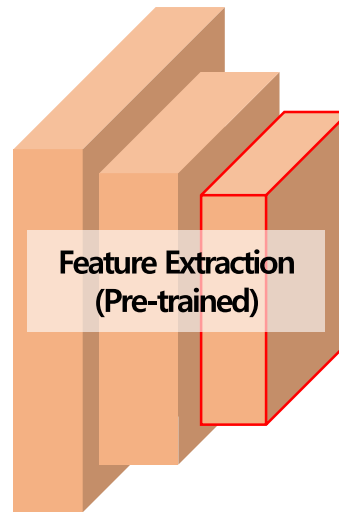
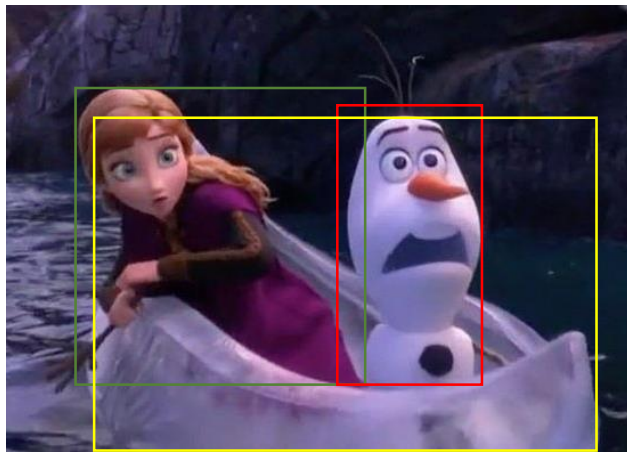
1. **Training is a multi-stage pipeline.** R-CNN first fine-tunes a ConvNet on object proposals using log loss. Then, it fits SVMs to ConvNet features. These SVMs act as object detectors, replacing the softmax classifier learnt by fine-tuning. In the third training stage, bounding-box regressors are learned.

2. Object Detection



❖ 변경된 Selective Search 결과 사용법

- Fast R-CNN에서는 입력 이미지에서 Selective Search 시행하는 것은 동일
- Proposal Region이 CNN을 통과해서 Feature map에서 어떤 부분에 있는지를 계산
- 즉 CNN은 1번만 통과함



| | | | | | | | | | |
|------|------|------|------|------|------|------|------|------|------|
| 0.71 | 0.99 | 0.74 | 0.93 | 0.94 | 0.83 | 0.82 | 0.08 | 0.84 | 0.75 |
| 0.82 | 0.41 | 0.61 | 0.57 | 0.32 | 0.56 | 0.76 | 0.90 | 0.13 | 0.74 |
| 0.41 | 0.88 | 0.88 | 0.44 | 0.14 | 0.16 | 0.37 | 0.77 | 0.96 | 0.29 |
| 0.43 | 0.21 | 0.19 | 0.45 | 0.57 | 0.16 | 0.63 | 0.29 | 0.71 | 0.94 |
| 0.59 | 0.04 | 0.66 | 0.26 | 0.82 | 0.64 | 0.54 | 0.73 | 0.59 | 0.73 |
| 0.92 | 0.66 | 0.85 | 0.34 | 0.76 | 0.84 | 0.29 | 0.75 | 0.62 | 0.80 |
| 0.76 | 0.59 | 0.32 | 0.74 | 0.21 | 0.39 | 0.34 | 0.03 | 0.33 | 0.43 |
| 0.04 | 0.52 | 0.20 | 0.14 | 0.16 | 0.13 | 0.73 | 0.65 | 0.96 | 0.45 |
| 0.08 | 0.19 | 0.19 | 0.69 | 0.09 | 0.86 | 0.88 | 0.07 | 0.01 | 0.52 |
| 0.41 | 0.28 | 0.68 | 0.30 | 0.55 | 0.94 | 0.48 | 0.08 | 0.10 | 0.92 |

2. Object Detection

❖ RoI(Region of Interest) Pooling

- 서로 다른 크기의 Proposal region을 동일한 크기로 변경하기 위함

| | | | | | | | | | |
|------|------|------|------|------|------|------|------|------|------|
| 0.71 | 0.99 | 0.74 | 0.93 | 0.94 | 0.83 | 0.82 | 0.08 | 0.84 | 0.75 |
| 0.82 | 0.41 | 0.61 | 0.57 | 0.32 | 0.56 | 0.76 | 0.90 | 0.13 | 0.74 |
| 0.41 | 0.88 | 0.88 | 0.44 | 0.14 | 0.16 | 0.37 | 0.77 | 0.96 | 0.29 |
| 0.43 | 0.21 | 0.19 | 0.45 | 0.57 | 0.16 | 0.63 | 0.29 | 0.71 | 0.94 |
| 0.59 | 0.04 | 0.66 | 0.26 | 0.82 | 0.64 | 0.54 | 0.73 | 0.59 | 0.73 |
| 0.92 | 0.66 | 0.85 | 0.34 | 0.76 | 0.84 | 0.29 | 0.75 | 0.62 | 0.80 |
| 0.76 | 0.59 | 0.32 | 0.74 | 0.21 | 0.39 | 0.34 | 0.03 | 0.33 | 0.43 |
| 0.04 | 0.52 | 0.20 | 0.14 | 0.16 | 0.13 | 0.73 | 0.65 | 0.96 | 0.45 |
| 0.08 | 0.19 | 0.19 | 0.69 | 0.09 | 0.86 | 0.88 | 0.07 | 0.01 | 0.52 |
| 0.41 | 0.28 | 0.68 | 0.30 | 0.55 | 0.94 | 0.48 | 0.08 | 0.10 | 0.92 |

□: Proposal region 중 하나

□: RoI pooling에 사용

| | | | | | | |
|------|------|------|------|------|------|------|
| 0.88 | 0.44 | 0.14 | 0.16 | 0.37 | 0.77 | 0.96 |
| 0.19 | 0.45 | 0.57 | 0.16 | 0.63 | 0.29 | 0.71 |
| 0.66 | 0.26 | 0.82 | 0.64 | 0.54 | 0.73 | 0.59 |
| 0.85 | 0.34 | 0.76 | 0.84 | 0.29 | 0.75 | 0.62 |
| 0.32 | 0.74 | 0.21 | 0.39 | 0.34 | 0.03 | 0.33 |
| 0.20 | 0.14 | 0.16 | 0.13 | 0.73 | 0.65 | 0.96 |
| 0.19 | 0.69 | 0.09 | 0.86 | 0.88 | 0.07 | 0.01 |

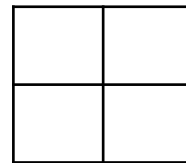
2. Object Detection

❖ RoI(Region of Interest) Pooling

- 서로 다른 크기의 Proposal region을 동일한 크기로 변경하기 위함

Proposal region 내 Feature map 값

| | | | | | | |
|------|------|------|------|------|------|------|
| 0.88 | 0.44 | 0.14 | 0.16 | 0.37 | 0.77 | 0.96 |
| 0.19 | 0.45 | 0.57 | 0.16 | 0.63 | 0.29 | 0.71 |
| 0.66 | 0.26 | 0.82 | 0.64 | 0.54 | 0.73 | 0.59 |
| 0.85 | 0.34 | 0.76 | 0.84 | 0.29 | 0.75 | 0.62 |
| 0.32 | 0.74 | 0.21 | 0.39 | 0.34 | 0.03 | 0.33 |
| 0.20 | 0.14 | 0.16 | 0.13 | 0.73 | 0.65 | 0.96 |
| 0.19 | 0.69 | 0.09 | 0.86 | 0.88 | 0.07 | 0.01 |



2. Object Detection

❖ RoI(Region of Interest) Pooling

- 서로 다른 크기의 Proposal region을 동일한 크기로 변경하기 위함

Proposal region 내 Feature map 값

| | | | | | | |
|------|------|------|------|------|------|------|
| 0.88 | 0.44 | 0.14 | 0.16 | 0.37 | 0.77 | 0.96 |
| 0.19 | 0.45 | 0.57 | 0.16 | 0.63 | 0.29 | 0.71 |
| 0.66 | 0.26 | 0.82 | 0.64 | 0.54 | 0.73 | 0.59 |
| 0.85 | 0.34 | 0.76 | 0.84 | 0.29 | 0.75 | 0.62 |
| 0.32 | 0.74 | 0.21 | 0.39 | 0.34 | 0.03 | 0.33 |
| 0.20 | 0.14 | 0.16 | 0.13 | 0.73 | 0.65 | 0.96 |
| 0.19 | 0.69 | 0.09 | 0.86 | 0.88 | 0.07 | 0.01 |

| | |
|------|--|
| 0.88 | |
| | |

2. Object Detection

❖ RoI(Region of Interest) Pooling

- 서로 다른 크기의 Proposal region을 동일한 크기로 변경하기 위함

Proposal region 내 Feature map 값

| | | | | | | |
|------|------|------|------|------|------|------|
| 0.88 | 0.44 | 0.14 | 0.16 | 0.37 | 0.77 | 0.96 |
| 0.19 | 0.45 | 0.57 | 0.16 | 0.63 | 0.29 | 0.71 |
| 0.66 | 0.26 | 0.82 | 0.64 | 0.54 | 0.73 | 0.59 |
| 0.85 | 0.34 | 0.76 | 0.84 | 0.29 | 0.75 | 0.62 |
| 0.32 | 0.74 | 0.21 | 0.39 | 0.34 | 0.03 | 0.33 |
| 0.20 | 0.14 | 0.16 | 0.13 | 0.73 | 0.65 | 0.96 |
| 0.19 | 0.69 | 0.09 | 0.86 | 0.88 | 0.07 | 0.01 |

| | |
|------|------|
| 0.88 | 0.96 |
| | |

2. Object Detection

❖ RoI(Region of Interest) Pooling

- 서로 다른 크기의 Proposal region을 동일한 크기로 변경하기 위함

Proposal region 내 Feature map 값

| | | | | | | |
|------|------|------|------|------|------|------|
| 0.88 | 0.44 | 0.14 | 0.16 | 0.37 | 0.77 | 0.96 |
| 0.19 | 0.45 | 0.57 | 0.16 | 0.63 | 0.29 | 0.71 |
| 0.66 | 0.26 | 0.82 | 0.64 | 0.54 | 0.73 | 0.59 |
| 0.85 | 0.34 | 0.76 | 0.84 | 0.29 | 0.75 | 0.62 |
| 0.32 | 0.74 | 0.21 | 0.39 | 0.34 | 0.03 | 0.33 |
| 0.20 | 0.14 | 0.16 | 0.13 | 0.73 | 0.65 | 0.96 |
| 0.19 | 0.69 | 0.09 | 0.86 | 0.88 | 0.07 | 0.01 |

| | |
|------|------|
| 0.88 | 0.96 |
| 0.85 | |

2. Object Detection

❖ RoI(Region of Interest) Pooling

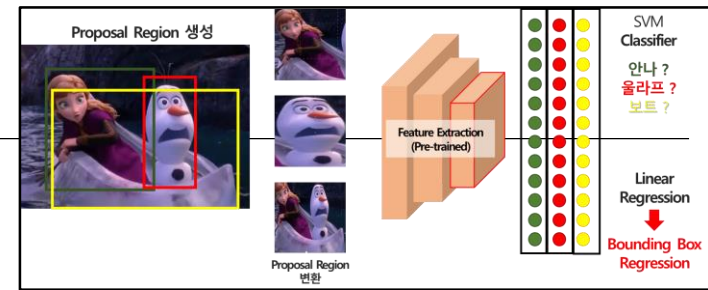
- 서로 다른 크기의 Proposal region을 동일한 크기로 변경하기 위함

Proposal region 내 Feature map 값

| | | | | | | |
|------|------|------|------|------|------|------|
| 0.88 | 0.44 | 0.14 | 0.16 | 0.37 | 0.77 | 0.96 |
| 0.19 | 0.45 | 0.57 | 0.16 | 0.63 | 0.29 | 0.71 |
| 0.66 | 0.26 | 0.82 | 0.64 | 0.54 | 0.73 | 0.59 |
| 0.85 | 0.34 | 0.76 | 0.84 | 0.29 | 0.75 | 0.62 |
| 0.32 | 0.74 | 0.21 | 0.39 | 0.34 | 0.03 | 0.33 |
| 0.20 | 0.14 | 0.16 | 0.13 | 0.73 | 0.65 | 0.96 |
| 0.19 | 0.69 | 0.09 | 0.86 | 0.88 | 0.07 | 0.01 |

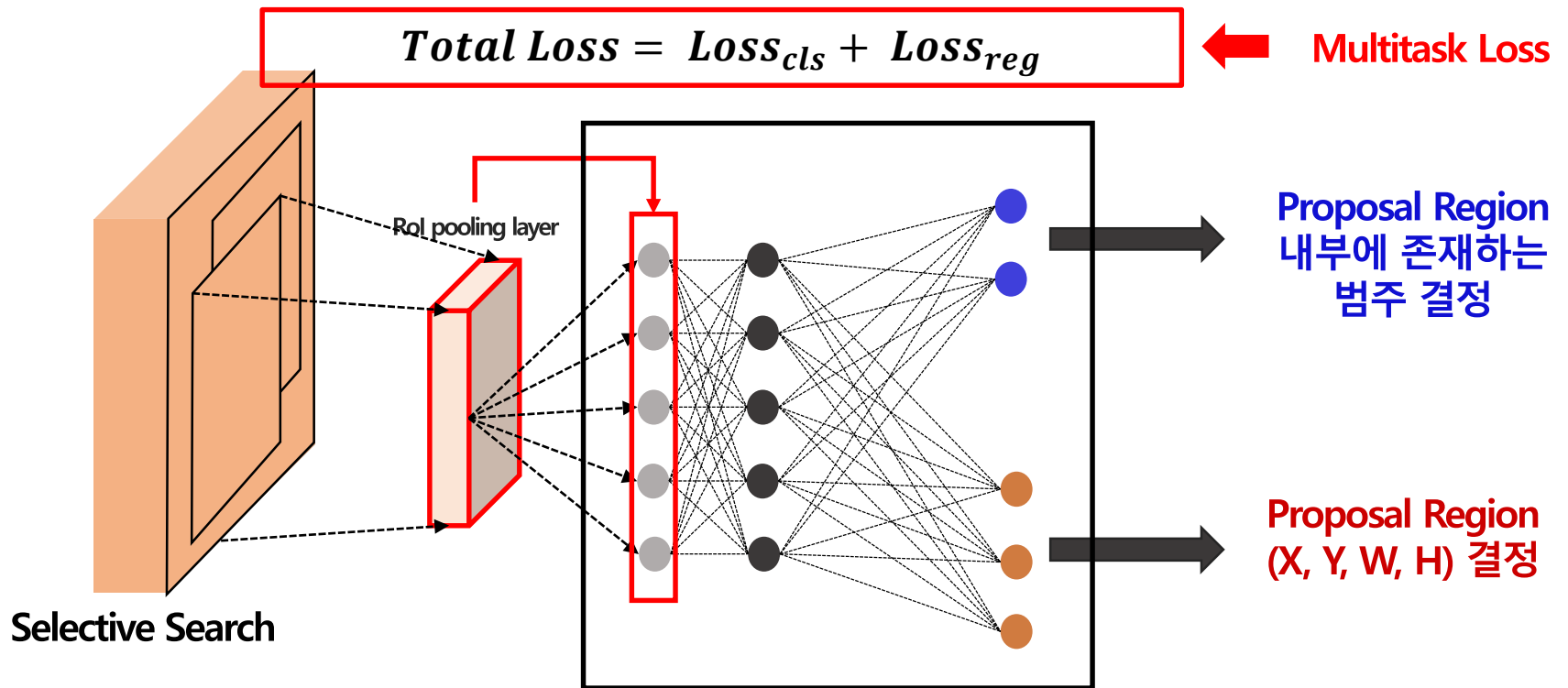
| | |
|------|------|
| 0.88 | 0.96 |
| 0.85 | 0.88 |

2. Object Detection



❖ Fast R-CNN 손실(Loss)함수

- R-CNN에서는 Classifier와 Bounding Box Regressor를 각각 학습
- Fast R-CNN에서는 Classification Loss와 Regression Loss를 합해서 최종 Loss 계산
- 오차 역전파(Backpropagation)을 이용해 Fully Connected Layer 학습

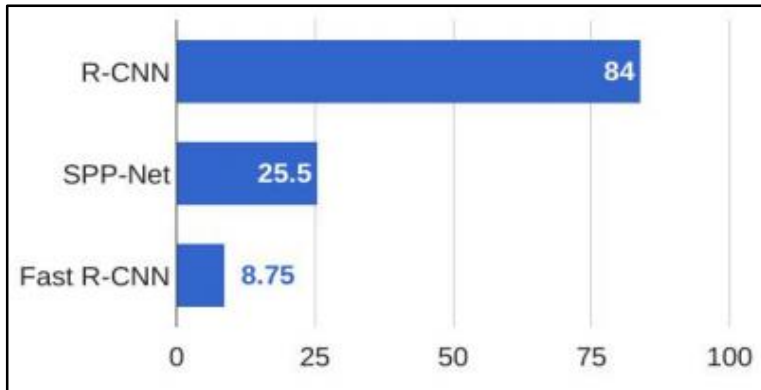


2. Object Detection

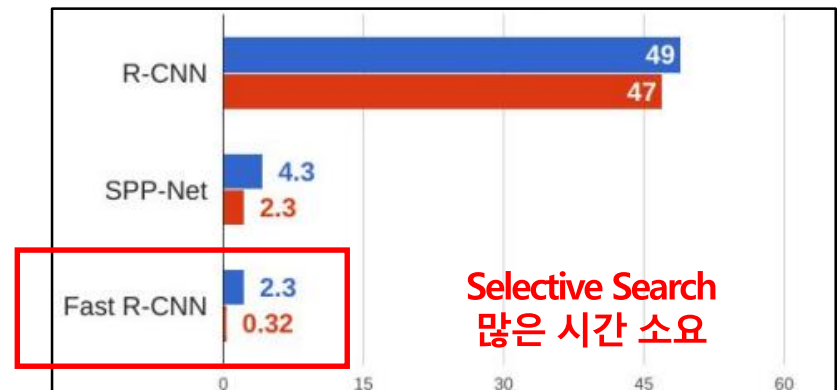
❖ R-CNN과 Fast R-CNN 시간 비교

- R-CNN의 경우 학습에 84시간 소요, Fast R-CNN의 경우 8.75시간 소요
- 이미지 1장에 대해 검증 시간이 49초에서 2.3초로 급감
- 여전히 Proposal Region 추출 시, 많은 시간 소요

학습 시간(Hours)



검증 시간(Seconds)



Selective Search
많은 시간 소요

- 이미지 1장 검증에 걸린 시간
- 이미지 1장 검증에 걸린 시간(Selective Search 제외)

2. Object Detection

❖ Faster R-CNN

- 2015년 **Neural Information Processing Systems(Neural IPS)**에서 발표된 논문
- 해당 논문은 **Facebook AI Research(FAIR)** 그룹에서 발표한 논문
- 2020년 1월 28일 기준으로 **15514회** 인용

Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun

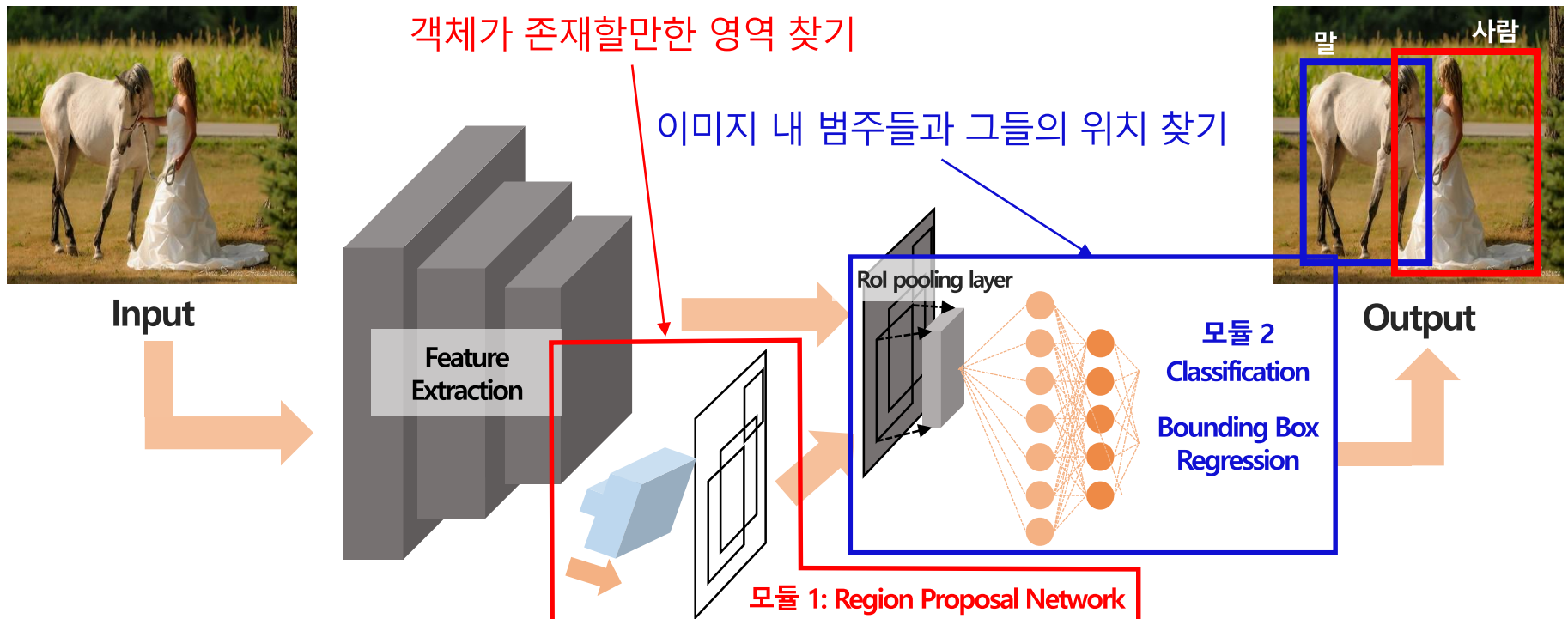
Abstract—State-of-the-art object detection networks depend on region proposal algorithms to hypothesize object locations. Advances like SPPnet [1] and Fast R-CNN [2] have reduced the running time of these detection networks, exposing region proposal computation as a bottleneck. In this work, we introduce a *Region Proposal Network* (RPN) that shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals. An RPN is a fully convolutional network that simultaneously predicts object bounds and objectness scores at each position. The RPN is trained end-to-end to generate high-quality region proposals, which are used by Fast R-CNN for detection. We further merge RPN and Fast R-CNN into a single network by sharing their convolutional features—using the recently popular terminology of neural networks with “attention” mechanisms, the RPN component tells the unified network where to look. For the very deep VGG-16 model [3], our detection system has a frame rate of 5fps (including all steps) on a GPU, while achieving state-of-the-art object detection accuracy on PASCAL VOC 2007, 2012, and MS COCO datasets with only 300 proposals per image. In ILSVRC and COCO 2015 competitions, Faster R-CNN and RPN are the foundations of the 1st-place winning entries in several tracks. Code has been made publicly available.

Index Terms—Object Detection, Region Proposal, Convolutional Neural Network.

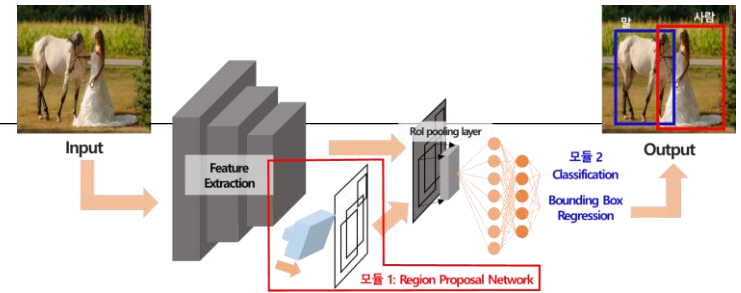
2. Object Detection

❖ Faster R-CNN

- 해당 모형은 End-to-End 모형이며 두 가지 모듈로 구성되어 있음
- **모듈 1: Region Proposal Network(RPN)**
- **모듈 2: Classification and Bounding Box Regression(Fast R-CNN)**

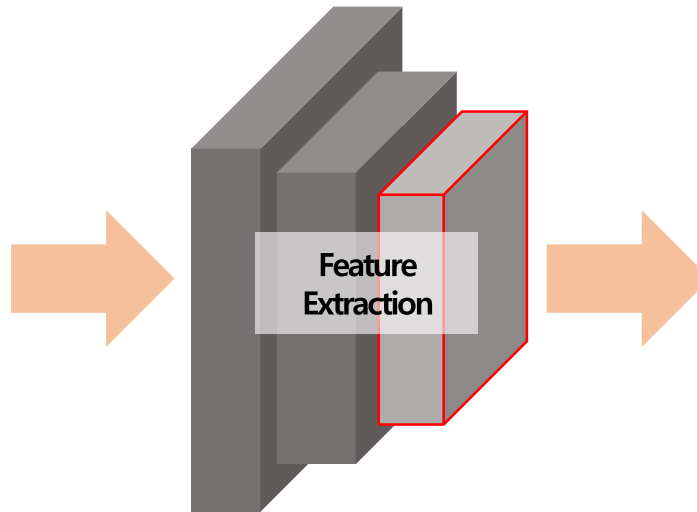
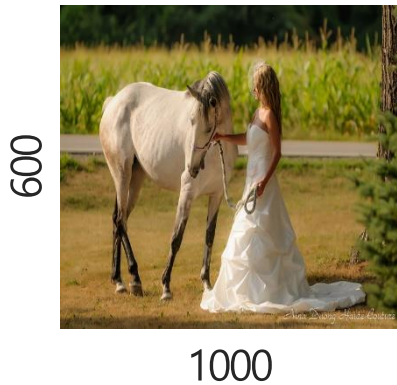


2. Object Detection



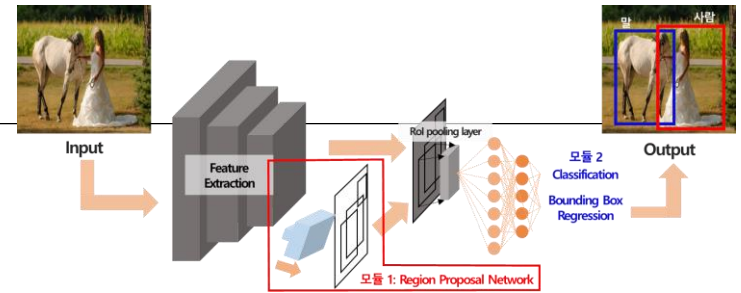
❖ 모듈 1: Region Proposal Network(RPN)

- RPN을 이용해 Selective Search 방법 대체
- Feature Map 상에서 Proposal Region을 추출하고자 함



Feature Map(60x40x256)

2. Object Detection



❖ 모듈 1: Region Proposal Network(RPN)

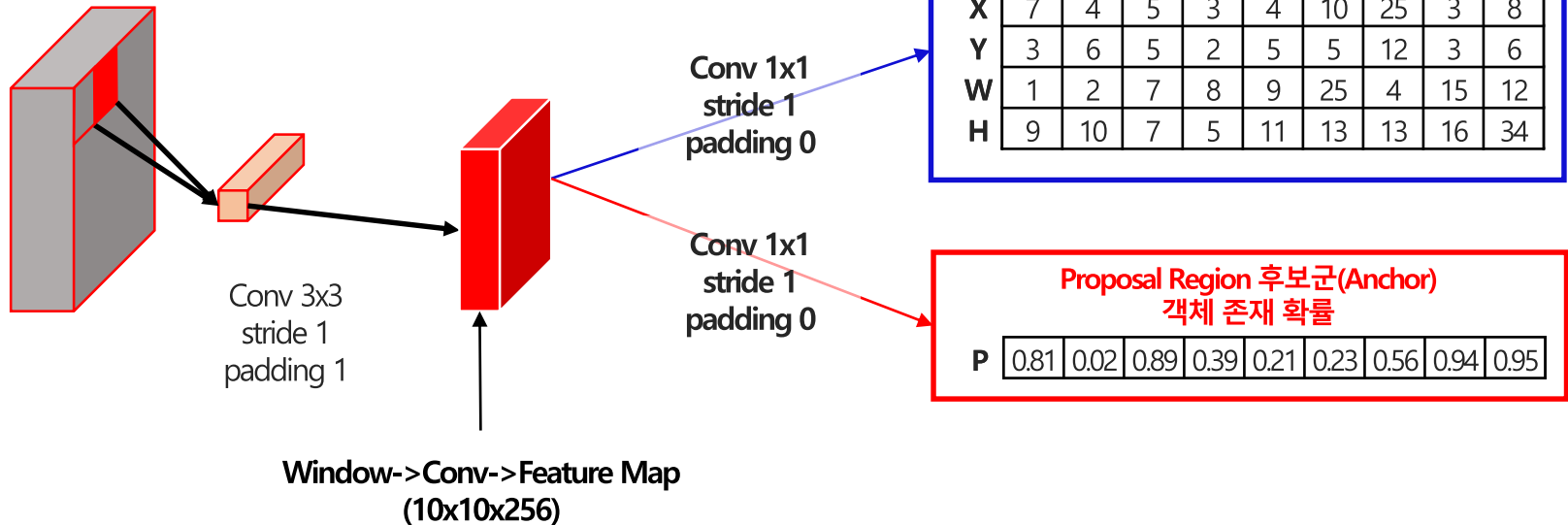
- Feature Map들을 사용해 Proposal Region 후보군(Anchors)을 생성하고자 함
- 합성곱 연산 후, Feature map의 한 픽셀마다 K개의 Proposal Region의 후보군 생성
- 논문에서는 K값을 9로 지정하였고, 20,000개의 후보군 생성

RPN의 Input

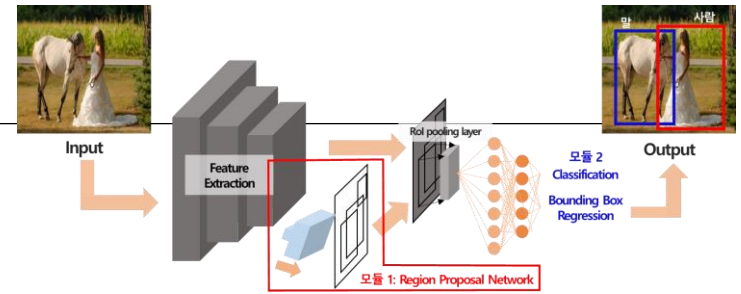
(10x10x256)

(Feature Map 내 하나의 Window)

Feature Map(60x40x256)

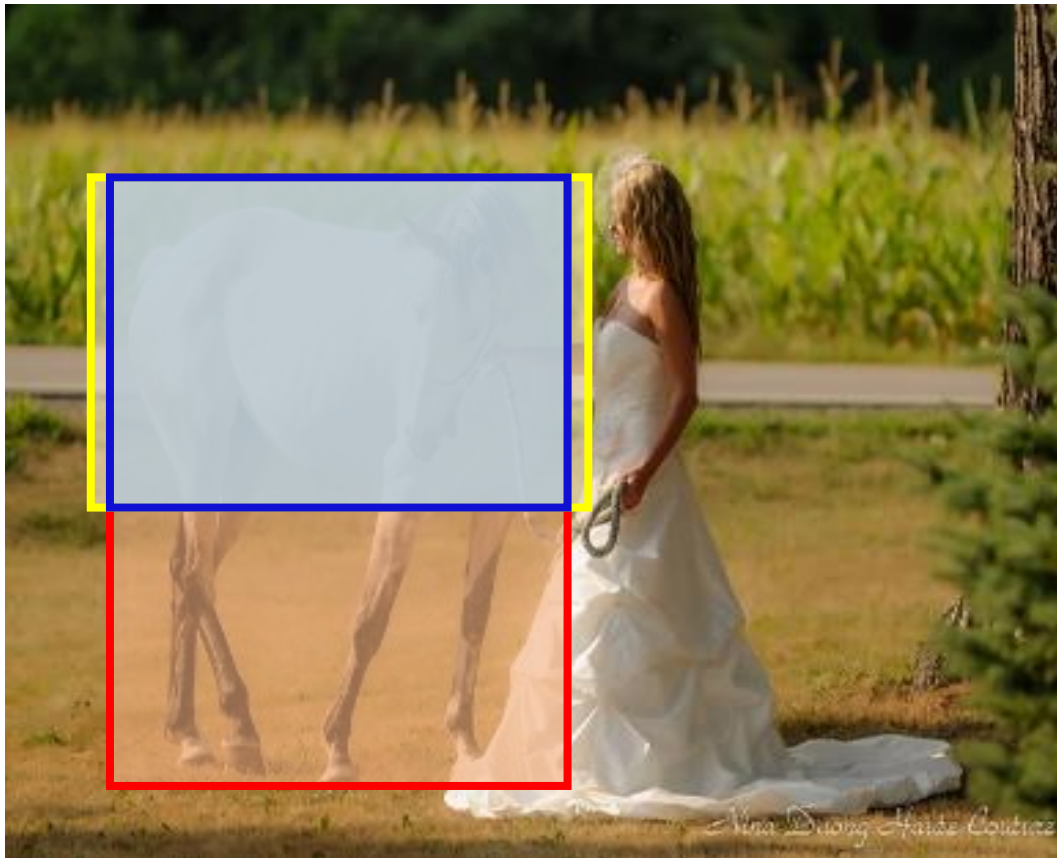


2. Object Detection

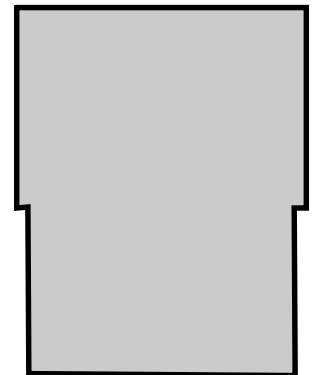


❖ 모듈 1: Region Proposal Network(RPN)

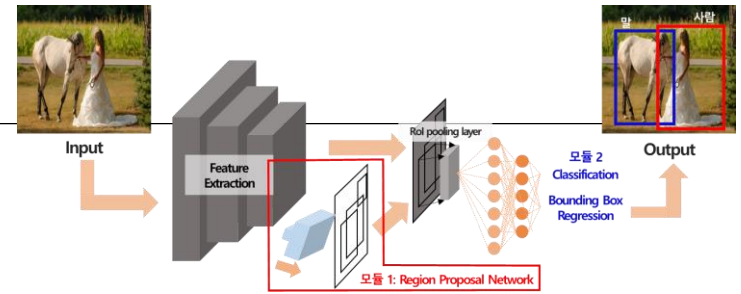
- Intersection over Union(IoU)



IoU

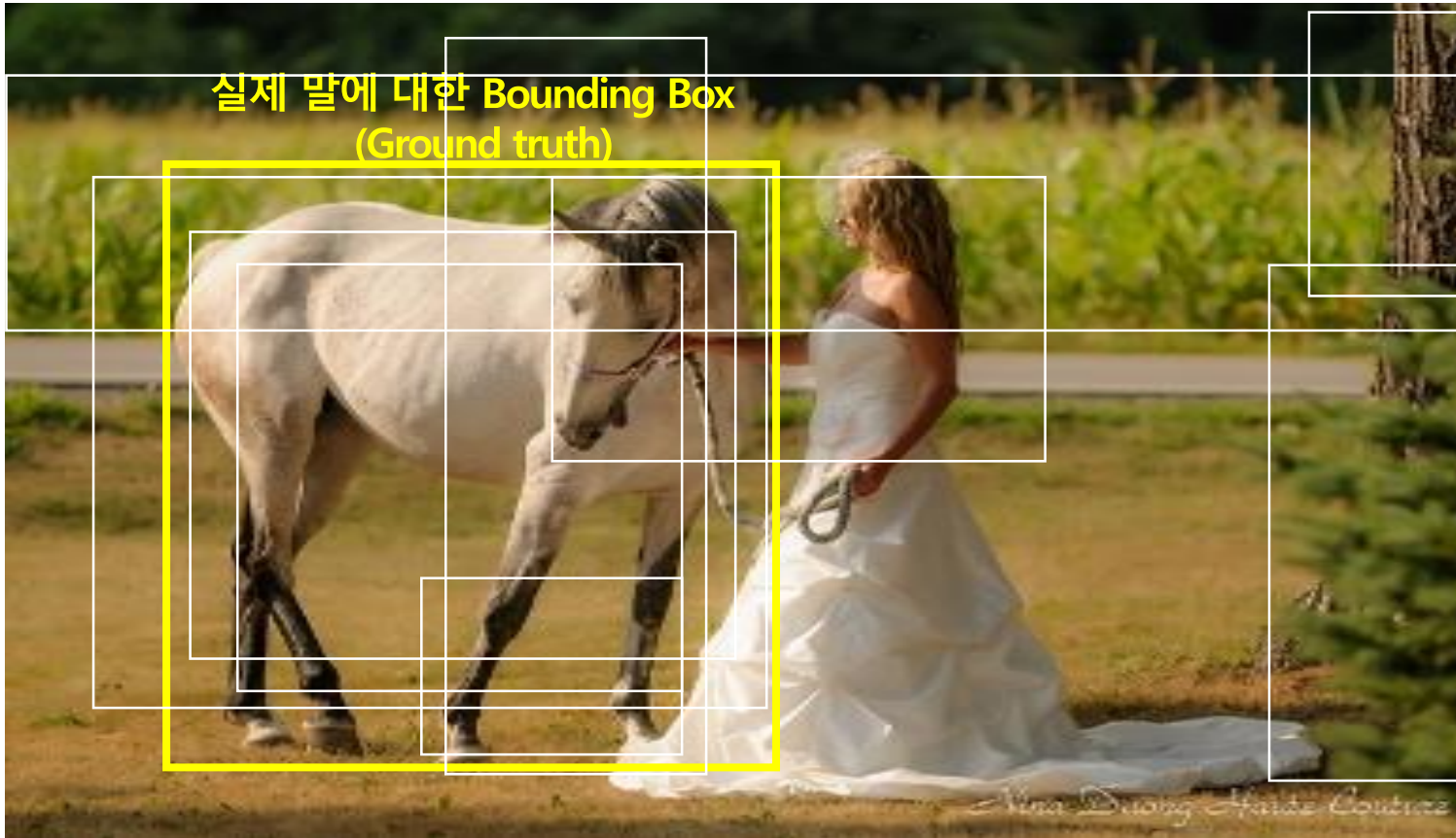


2. Object Detection

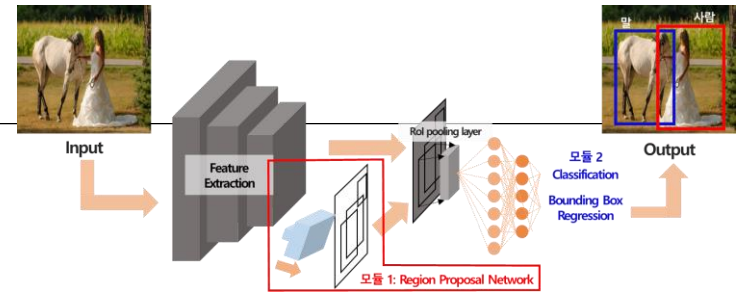


❖ 모듈 1: Region Proposal Network(RPN)

- Proposal Region 후보군 중 Proposal Region 정하는 방법

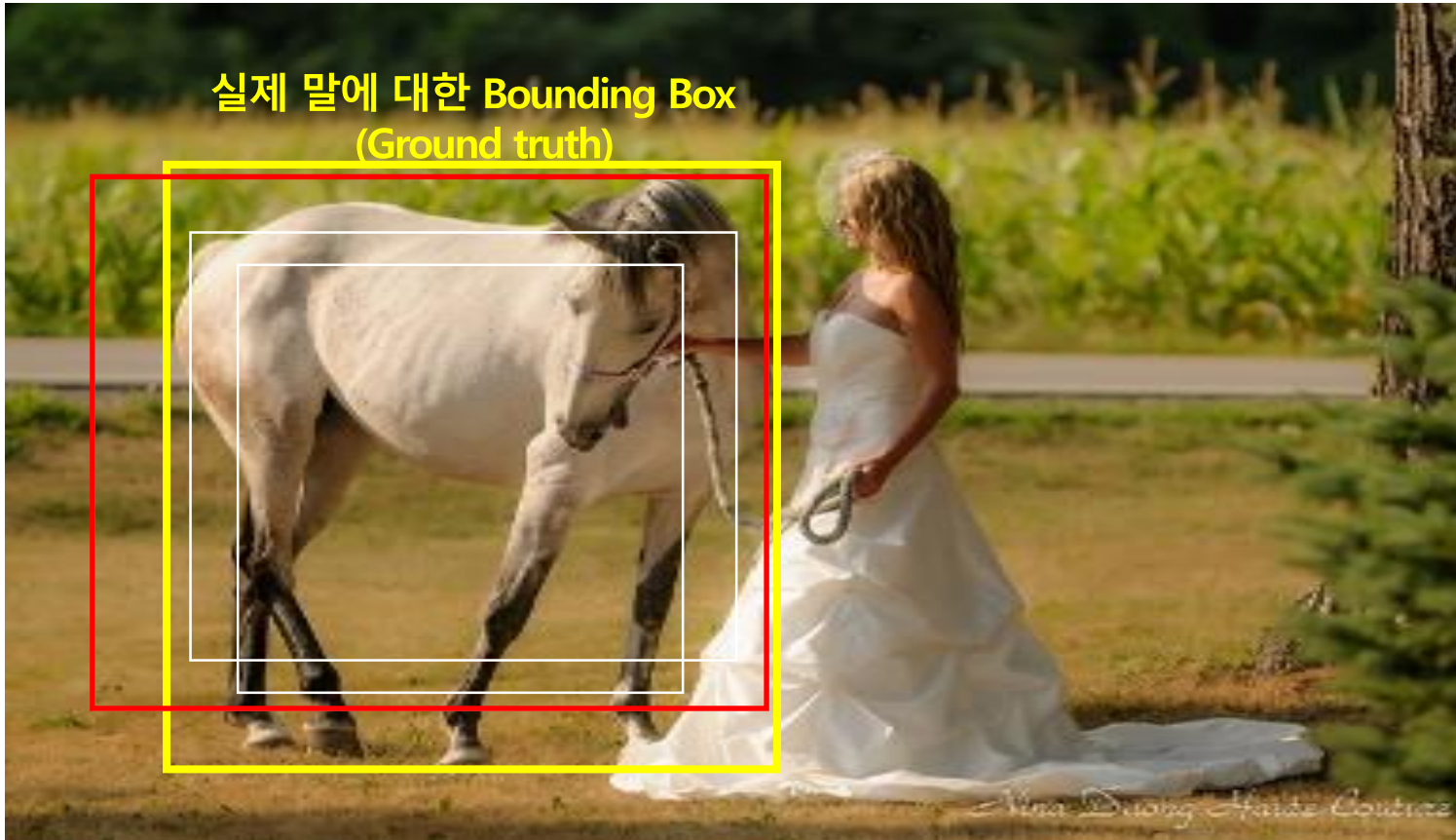


2. Object Detection



❖ 모듈 1: Region Proposal Network(RPN)

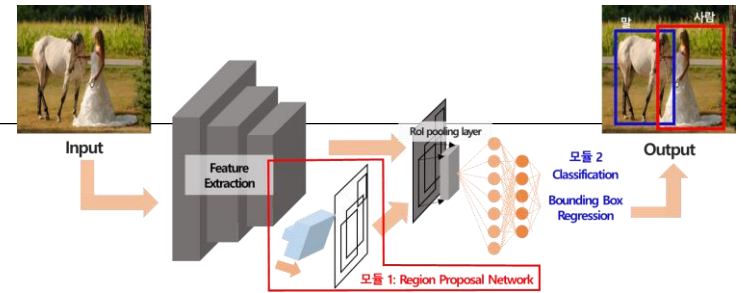
- Proposal Region 후보군 중 Proposal Region 정하는 방법



객체 존재 확률

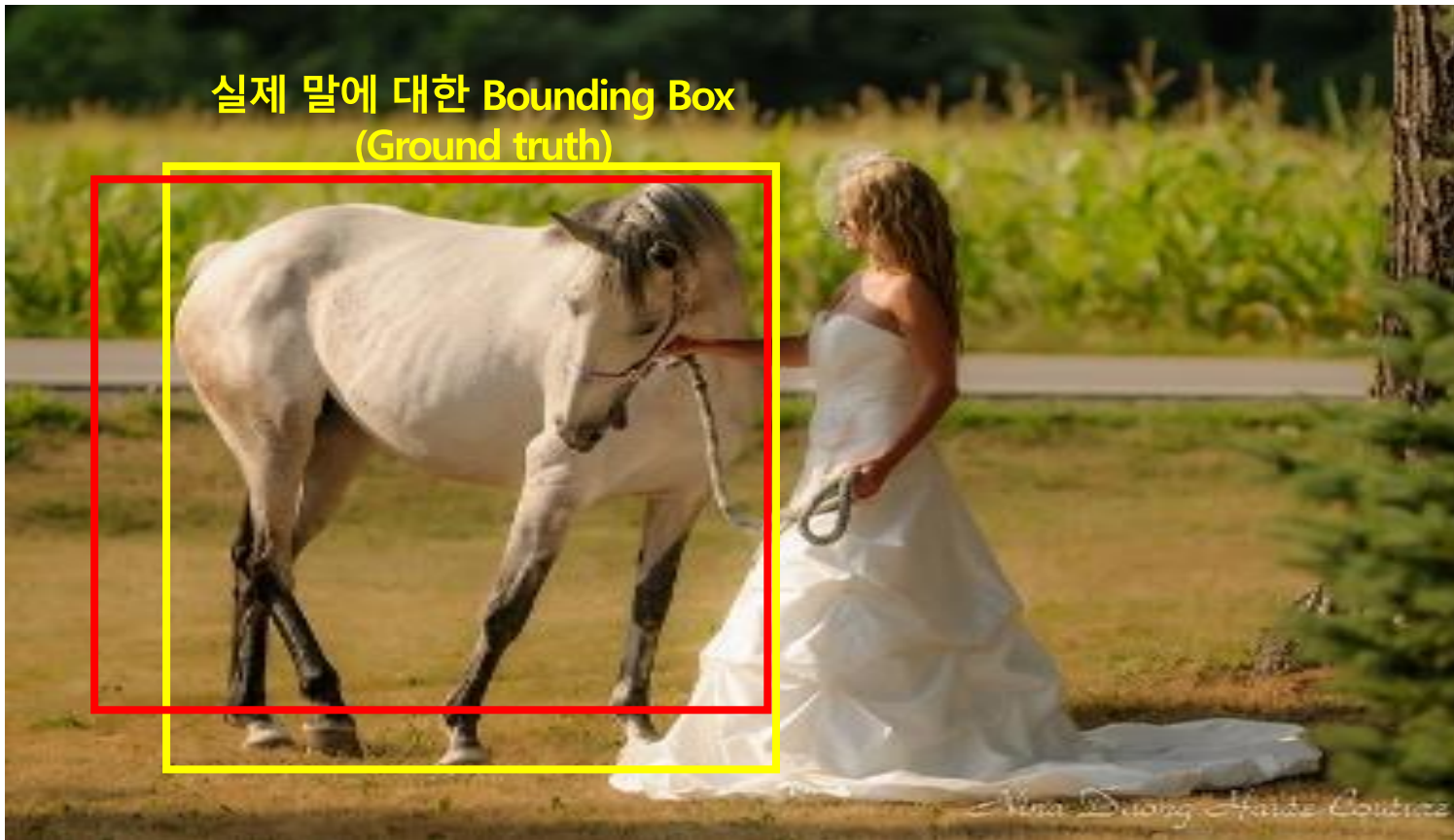
확률: 0.91

2. Object Detection



❖ 모듈 1: Region Proposal Network(RPN)

- Proposal Region 후보군 중 Proposal Region 정하는 방법
- 빨간색 상자를 Proposal Region으로 지정하고 이에 대해 RoI Pooling 진행



객체 존재 확률

확률: 0.97

2. Object Detection

❖ 모듈 1: Region Proposal Network(RPN)

- RPN 손실 함수

객체 존재 여부에 관한 손실 함수

객체가 존재할만한 위치에 관한 손실 함수

$$L(p_i, t_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \frac{\lambda}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*)$$

i : mini-batch 내 B.B 박스 후보군(Anchor)의 Index

p_i : Anchor가 사전에 정의한 객체인지의 예측 값

사전에 정의한 객체인 경우 $\rightarrow p_i^*=1$

N_{cls} : Batch size 내 B.B 박스 후보군(Anchor) 개수

$p_i^*=1$ 인 경우에 대해서만 진행

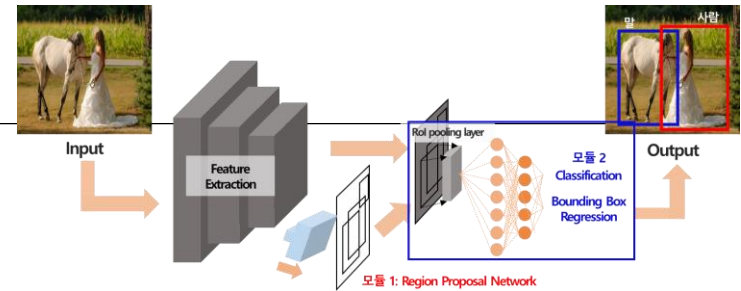
t_i : RPN을 이용해 예측한 Proposal region의 (X, Y) 좌표 + 너비와 높이

t_i^* : 실제 B.B의 (X, Y) 좌표 + 너비와 높이

N_{reg} : mini-batch 내 B.B 박스 후보군(Anchor)의 (X, Y) 좌표와 너비와 높이의 총 개수

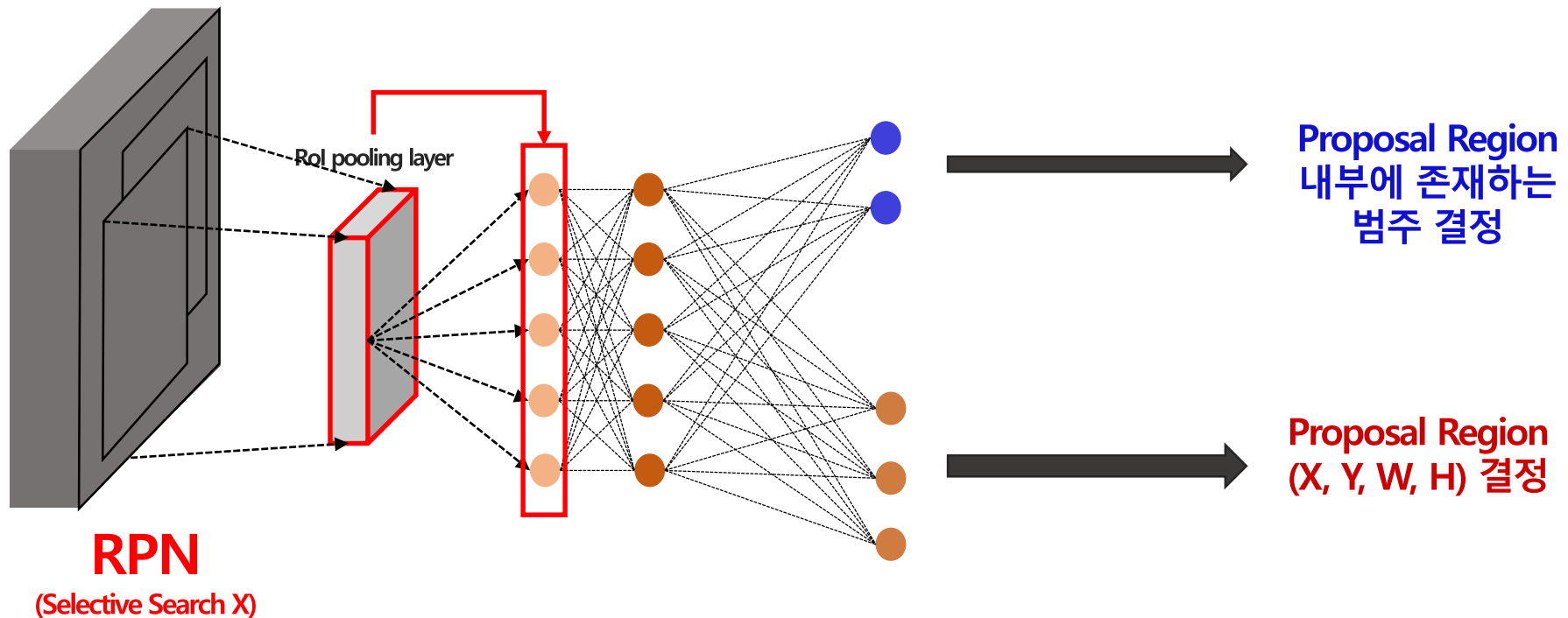
λ : 정규화 상수(논문에서는 10으로 설정)

2. Object Detection



❖ 모듈 2: Classification + Bounding Box Regression

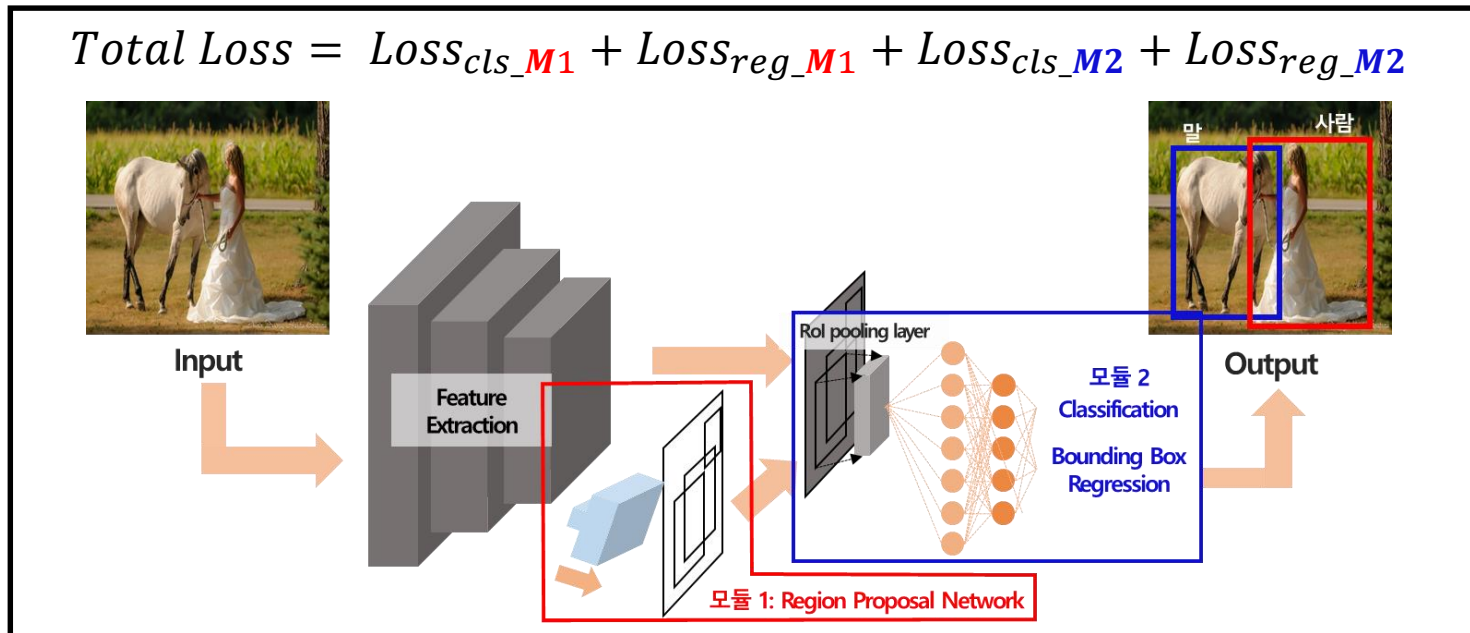
- RoI(Region of Interest) pooling을 통과한 Feature Map을 열 벡터로 변경
- Fully Connected Layer 를 사용해 사전에 정의한 범주 존재 여부 예측
- Bounding Box의 구성 요소(X, Y, W, H) 역시 Fully Connected Layer 사용해 예측



2. Object Detection

❖ Faster R-CNN

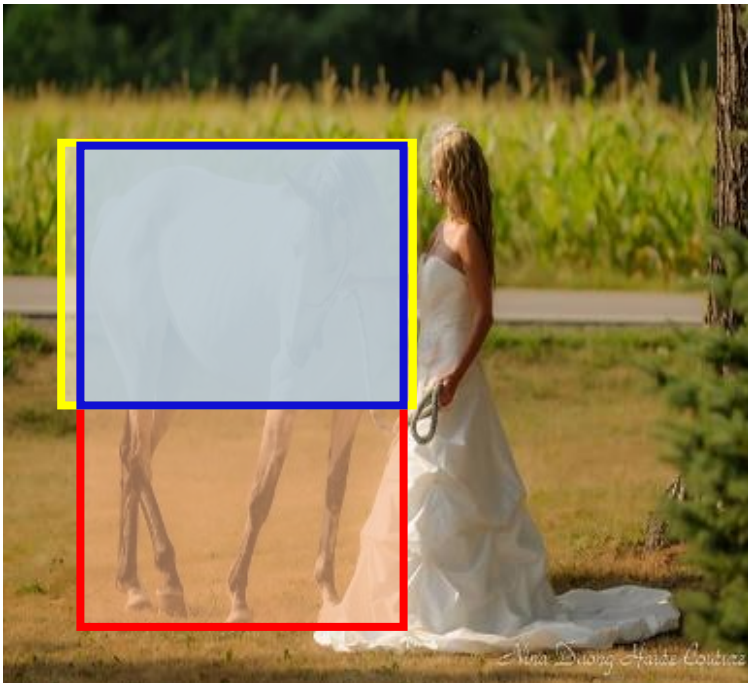
- 모듈 1: Region Proposal Network(RPN)
 - 객체가 존재할만한 영역(Proposal Region) 찾기
- 모듈 2: Classification and Bounding Box Regression
 - Proposal Region 내 존재하는 범주들과 그들의 위치 찾기
- 전체 손실 함수 값을 사용해 오차 역전파(Backpropagation)를 이용해 학습



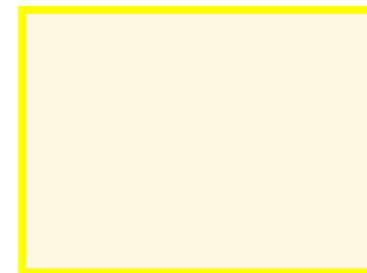
2. Object Detection

❖ Object Detection 성능 평가 지표

- **Precision:** 이미지 1장 내 예측 영역 내 실제로 말이 존재하는 영역의 비율
- **Average Precision(AP):** 데이터 셋 내 모든 말 객체에 대한 Precision 평균
- **mean Average Precision(mAP):** 모든 범주의 AP에 대한 평균



Precision



2. Object Detection

❖ Faster R-CNN 실험 결과

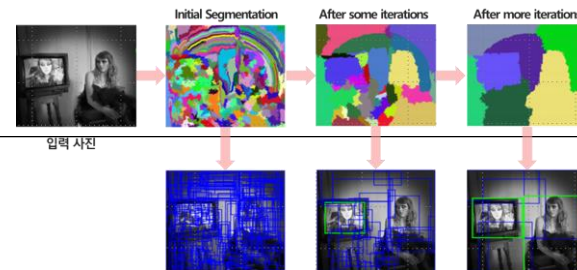
- 실험에 사용한 데이터 셋: MS COCO(**M**icrosoft **C**ommon **O**bjects in **C**ontext)
- MS COCO 데이터 셋은 마이크로소프트에서 주최하는 대회에서 사용하는 데이터 셋
- 대회 종목: Object Detection, Image Segmentation 등
- 사전에 정의한 범주 개수: 80

Dataset examples



- <http://cocodataset.org/#home>

2. Object Detection



❖ Faster R-CNN 실험 결과

- Selective Search(Fast R-CNN) 방법을 사용한 경우와 **RPN(Faster R-CNN)** 사용한 경우를 비교
- **RPN**을 사용했을 때, Object Detection 성능도 뛰어나고 속도도 빠름
- mAP 값에 100을 곱한 값을 아래 표에 표기

| | Proposal Region 탐색 방법 | mAP |
|--|--------------------------------|------|
| Fast R-CNN (기존 Object Detection 알고리즘) | Selective Search | 39.3 |
| Faster R-CNN | Region Proposal Network | 42.7 |

| | Proposal Region 탐색 방법 | fps (1초당 이미지 처리 개수) |
|--|--------------------------------|------------------------|
| Fast R-CNN (기존 Object Detection 알고리즘) | Selective Search | 0.5 |
| Faster R-CNN | Region Proposal Network | 5 |

Faster R-CNN이 기존 알고리즘보다 10배 빠름

2. Object Detection

- ❖ Faster R-CNN 최종 출력물의 형태
 - 사진 내 사전에 정의한 범주들과 이들의 위치(Bounding Box)를 함께 시각화
 - 추가로 특정 범주일 확률을 같이 표기해줄 수 있음

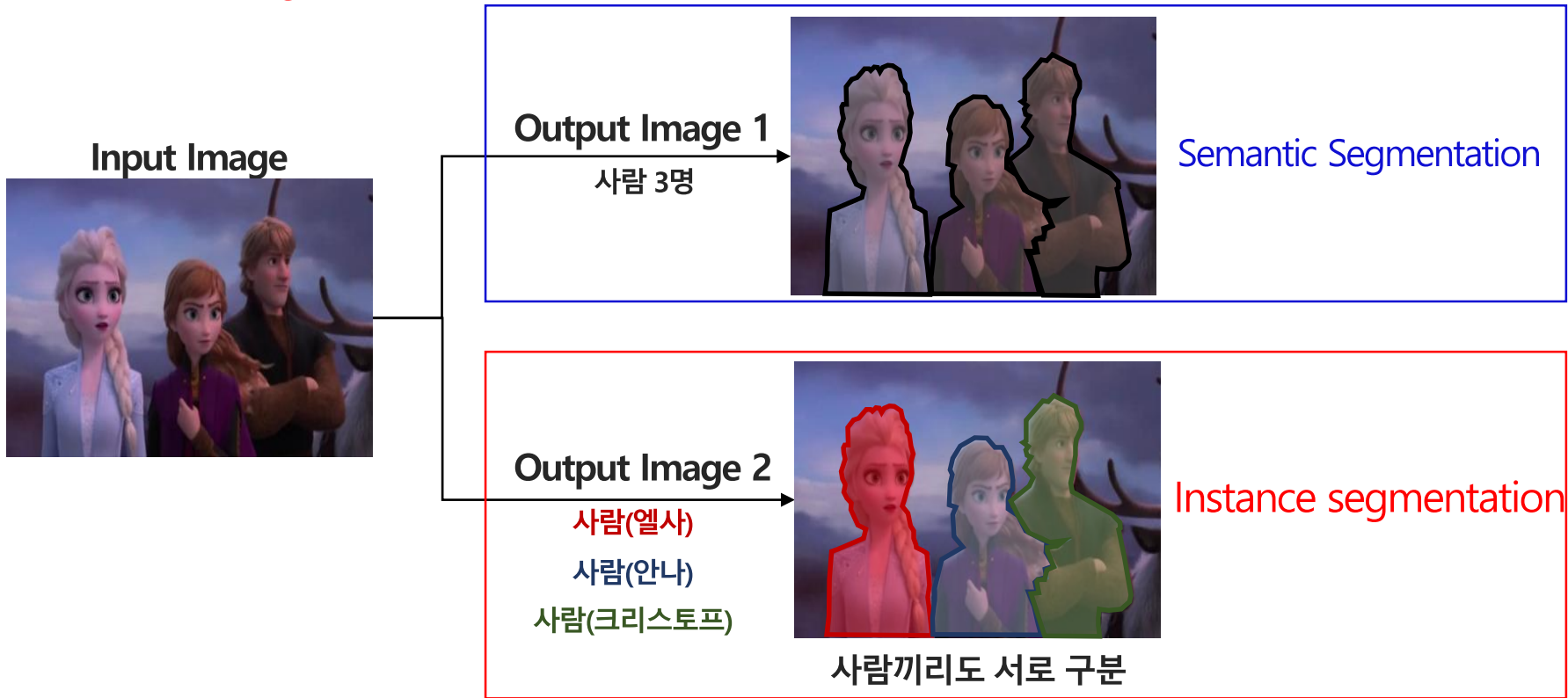


목차

1. Introduction
2. Object Detection
- 3. Image Segmentation**
4. Mask R-CNN
5. Conclusions
6. Additional Topic

3. Image Segmentation

- ❖ 이미지 분할(Image Segmentation)이란?
 - 사진 내 픽셀이 사전에 정의한 범주 중 어떤 범주에 속하는지 예측(픽셀 별 분류)
 - Semantic Segmentation과 Instance segmentation으로 구성
 - Instance segmentation에서는 인간에 대해서도 서로 구분할 수 있어야 함



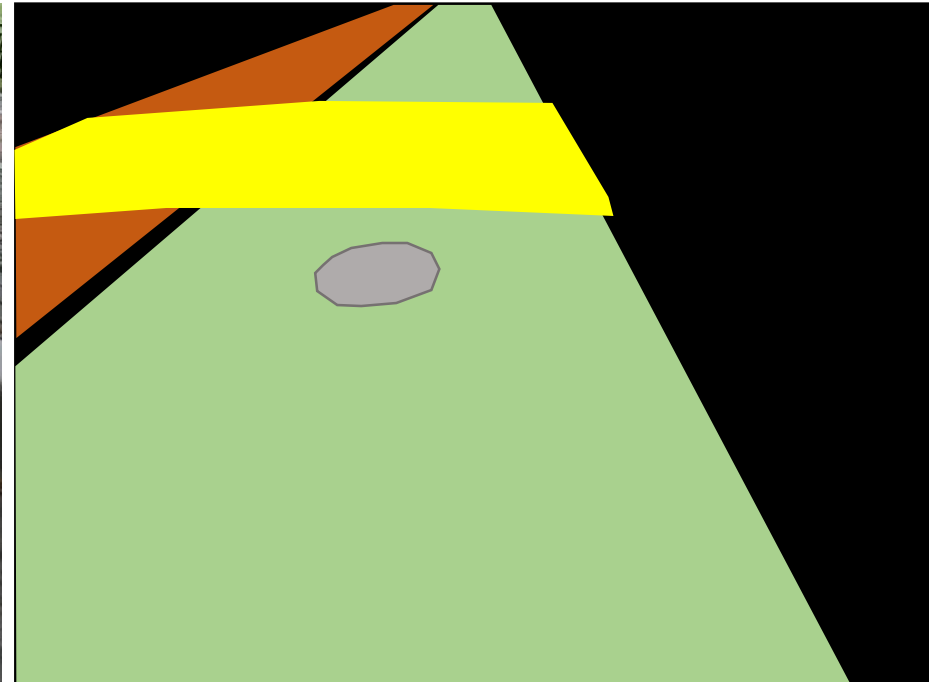
3. Image Segmentation

- ❖ Image Segmentation에서의 레이블링(Labeling)
 - Object Detection처럼 특정 사진에 대해 정답이 필요함
 - 사전에 정의한 범주: **주행 중인 차로(1)**, **나머지 차로(2)**, **방지 턱(3)**, **맨홀(4)**
 - 특정 사진에 대해 어떻게 Labeling 할까?

입력 사진



정답

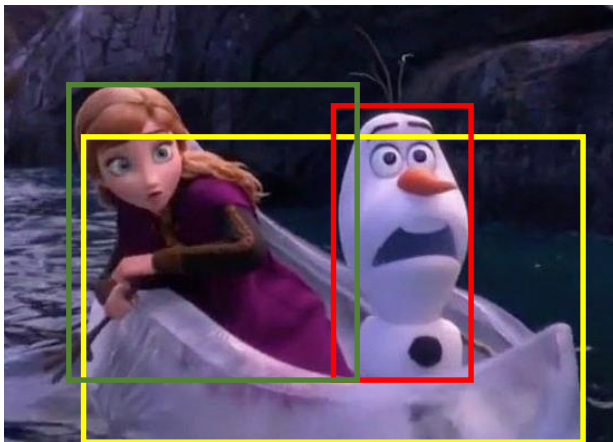


3. Image Segmentation

❖ Object Detection vs. Image Segmentation

- Object Detection: 이미지 내 어떤 범주가 있고 그들의 Bounding Box 찾기
- Image Segmentation: 픽셀 별로 어떤 범주에 속하는지 예측
- **예측해야 할 대상이 Image Segmentation이 더 많음**

Object Detection



올라프 안나 보트

$$4 + (3 \times 4) = 16$$

Image Segmentation



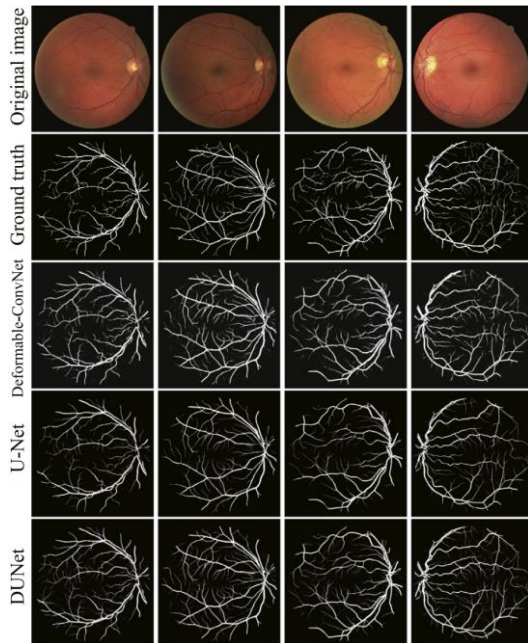
올라프 안나 보트

$$600 \times 500 \times 4 = 1,200,000$$

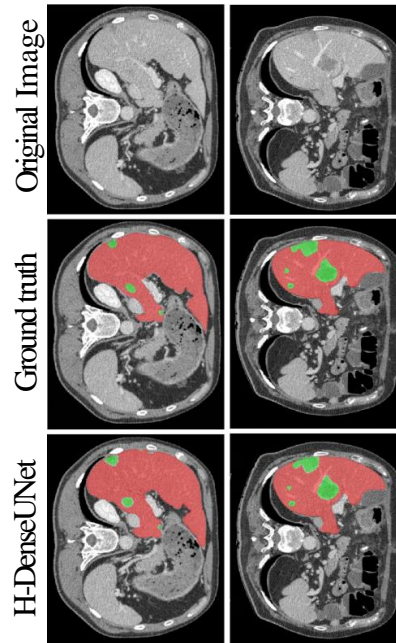
3. Image Segmentation

- ❖ Image Segmentation 사용 분야
 - 의료 영역에서 많은 응용 연구가 진행 중

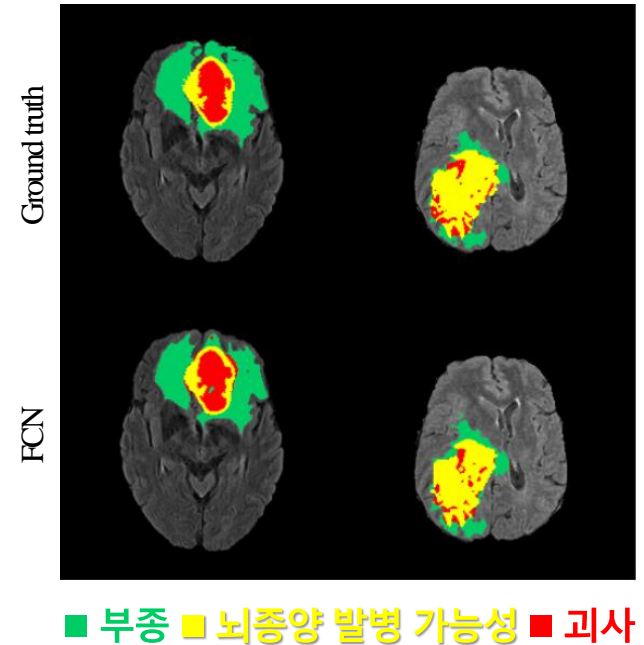
안구 내 혈관 탐지



CT 영상에서 간과 종양 탐지



뇌종양 발병 가능성 탐지



- Jin, Q., Meng, Z., Pham, T. D., Chen, Q., Wei, L., & Su, R. (2019). DUNet: A deformable network for retinal vessel segmentation. Knowledge-Based Systems, 178, 149-162.
- Li, X., Chen, H., Qi, X., Dou, Q., Fu, C. W., & Heng, P. A. (2018). H-DenseUNet hybrid densely connected UNet for liver and tumor segmentation from CT volumes. IEEE transactions on medical imaging, 37(12), 2663-2674.
- <https://software.intel.com/en-us/articles/brats-2017-glioma-segmentation-using-fully-convolutional-neural-networks>

3. Image Segmentation

❖ Fully Convolutional Networks(FCN)

- 2015년 **Computer Vision and Pattern Recognition(CVPR)**에서 발표된 논문
- 2020년 1월 28일 기준으로 **13857회** 인용되었고, 논문 저자들은 U.C. Berkeley 소속
- Semantic Segmentation을 하기 위해 수행

Fully Convolutional Networks for Semantic Segmentation

Jonathan Long* Evan Shelhamer* Trevor Darrell
UC Berkeley
{jonlong, shelhamer, trevor}@cs.berkeley.edu

Abstract

Convolutional networks are powerful visual models that yield hierarchies of features. We show that convolutional networks by themselves, trained end-to-end, pixels-to-pixels, exceed the state-of-the-art in semantic segmentation. Our key insight is to build “fully convolutional” networks that take input of arbitrary size and produce correspondingly-sized output with efficient inference and learning. We define and detail the space of fully convolutional networks, explain their application to spatially dense prediction tasks, and draw connections to prior models. We adapt contemporary classification networks (AlexNet [20], the VGG net [31], and GoogLeNet [32]) into fully convolutional networks and transfer their learned representations by fine-tuning [3] to the segmentation task. We then define a skip architecture that combines semantic information from a deep, coarse layer with appearance information from a shallow, fine layer to produce accurate and detailed segmentations. Our fully convolutional network achieves state-of-the-art segmentation of PASCAL VOC (20% relative improvement to 62.2% mean IU on 2012), NYUDv2, and SIFT Flow, while inference takes less than one fifth of a second for a typical image.

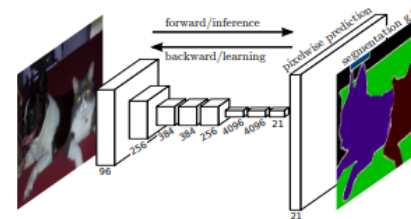


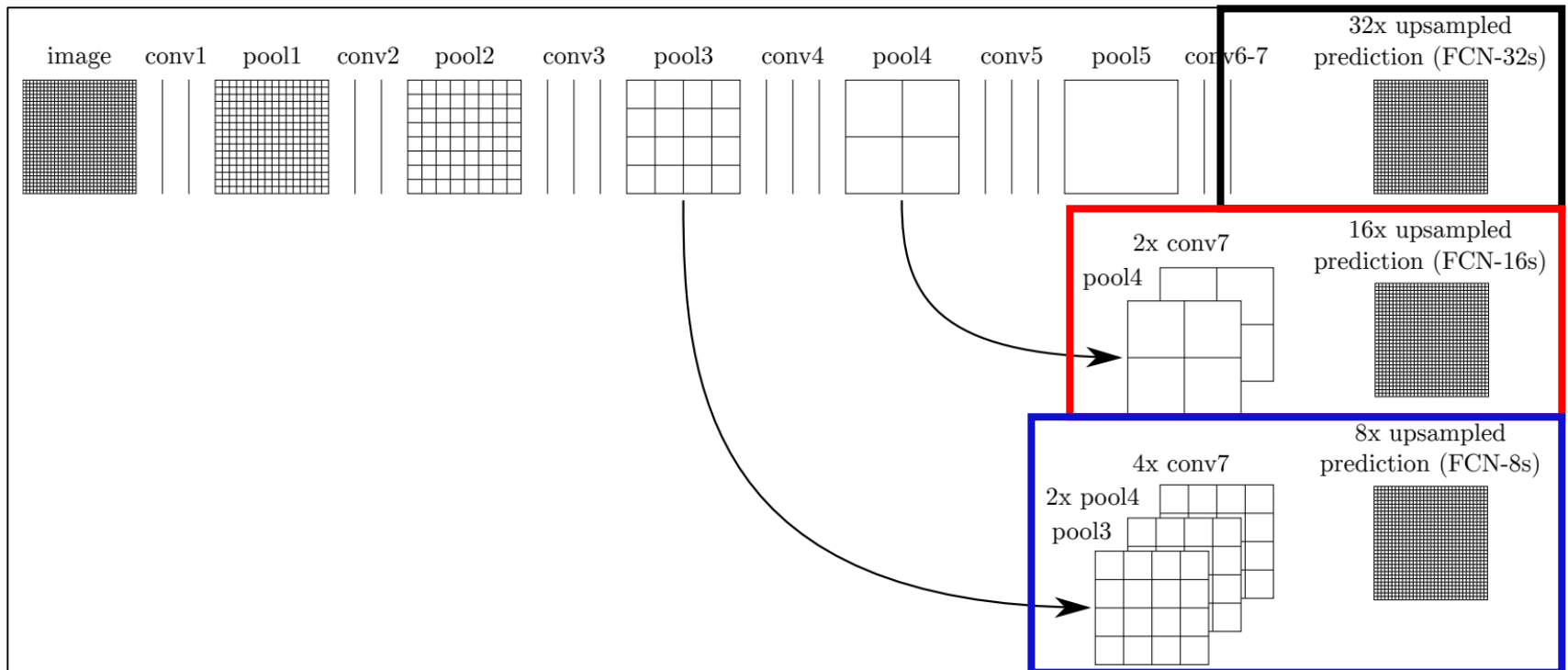
Figure 1. Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmentation.

We show that a fully convolutional network (FCN) trained end-to-end, pixels-to-pixels on semantic segmentation exceeds the state-of-the-art without further machinery. To our knowledge, this is the first work to train FCNs end-to-end (1) for pixelwise prediction and (2) from supervised pre-training. Fully convolutional versions of existing networks predict dense outputs from arbitrary-sized inputs. Both learning and inference are performed whole-image-at-a-time by dense feedforward computation and backpropagation. In-network upsampling layers enable pixelwise prediction and learning in nets with subsampled pooling.

3. Image Segmentation

❖ Fully Convolutional Network(FCN)

- 최종 Feature map을 32배, 16배, 8배 크기로 키움(Upsampling)

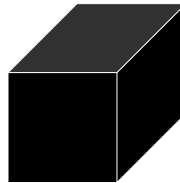
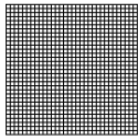


3. Image Segmentation

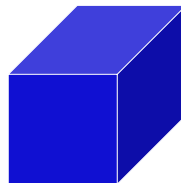
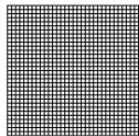
❖ Fully Convolutional Network(FCN)

- 최종 Feature map을 **32배**, **16배**, **8배** 크기로 키움(Upsampling)
- 키운 Feature map들을 합성곱 연산하여 진행해 픽셀 별로 분류

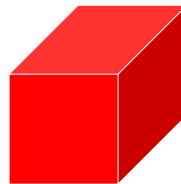
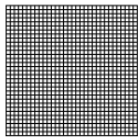
32x upsampled prediction (FCN-32s)



16x upsampled prediction (FCN-16s)

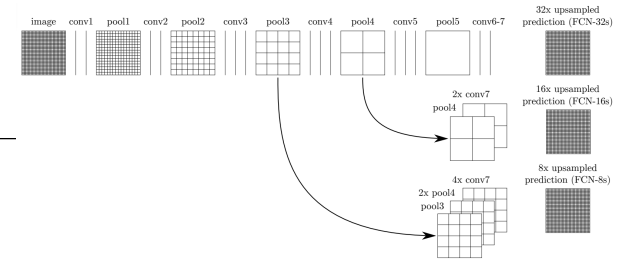


8x upsampled prediction (FCN-8s)



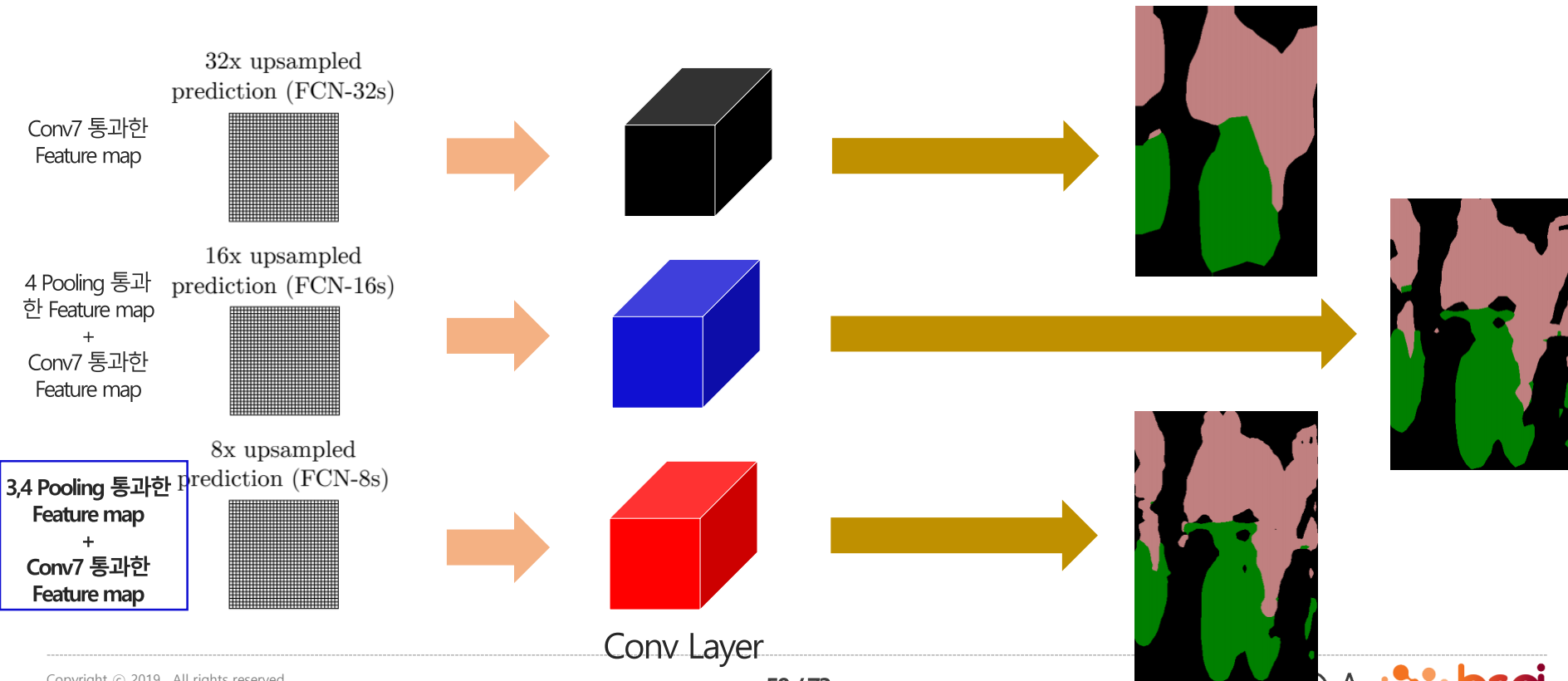
Conv Layer

3. Image Segmentation

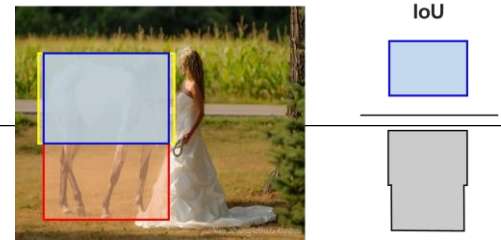


❖ Fully Convolutional Network(FCN)

- 최종 Feature map을 **32배**, **16배**, **8배** 크기로 키움(Upsampling)
- 키운 Feature map들을 합성곱 연산하여 진행해 픽셀 별로 분류
- **FCN-8s의 경우, 입력 이미지와 가까운 Feature map을 예측에 사용**



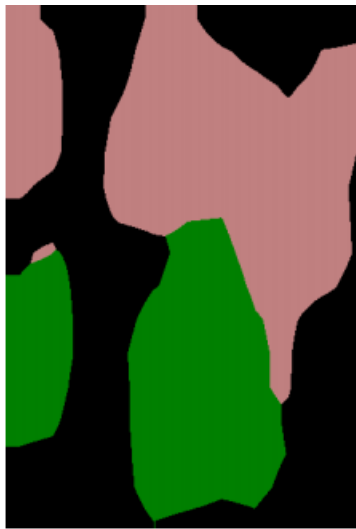
3. Image Segmentation



❖ Fully Convolutional Network(FCN) 실험 결과

- FCN-32s, FCN-16s, FCN-8s를 각각 구축하고, 예측 결과를 시각화
- 입력 이미지에 가까운 Feature map들을 예측에 사용할 수록 성능이 뛰어남

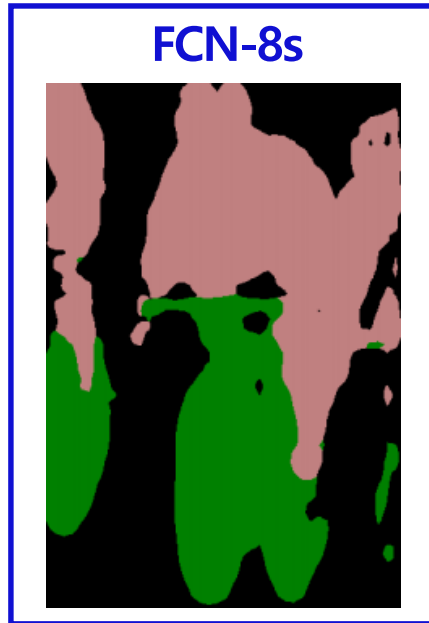
FCN-32s



FCN-16s



FCN-8s



정답(Ground truth)

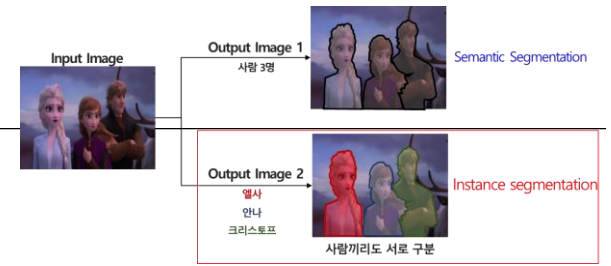


| | FCN-32s | FCN-16s | FCN-8s |
|----------------------------------|---------|---------|----------------|
| IoU (Intersection over Union) | 59.4(%) | 62.4(%) | 62.7(%) |

목차

1. Introduction
2. Object Detection
3. Image Segmentation
- 4. Mask R-CNN**
5. Conclusions
6. Additional Topic

4. Mask R-CNN



❖ Mask R-CNN

- 2017년 International Conference of Computer Vision(ICCV)에서 발표된 논문
- 2020년 1월 28일 기준으로 **4880회** 인용
- Mask R-CNN은 Instance segmentation을 수행하기 위해 개발됨

Mask R-CNN

Kaiming He

Georgia Gkioxari

Piotr Dollár

Ross Girshick

Facebook AI Research (FAIR)

Abstract

We present a conceptually simple, flexible, and general framework for object instance segmentation. Our approach efficiently detects objects in an image while simultaneously generating a high-quality segmentation mask for each instance. The method, called Mask R-CNN, extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition. Mask R-CNN is simple to train and adds only a small overhead to Faster R-CNN, running at 5 fps. Moreover,

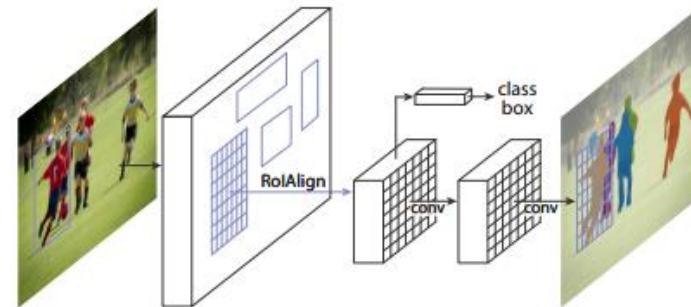


Figure 1. The **Mask R-CNN** framework for instance segmentation.

4. Mask R-CNN

❖ Mask R-CNN 기본 아이디어

- **Step1:** 객체가 있을 만한 영역 탐지(Bounding Box Regression)
- **Step2:** 탐지한 영역 내 어떠한 범주가 있을지 예측(Classification)
- **Step3:** 탐지한 영역 내 픽셀이 Step2에서 예측한 범주인지 아닌지 예측(Segmentation)

Faster R-CNN
↓
Object Detection

↑
FCN
(Fully Convolutional Network)

예시: 자동차



4. Mask R-CNN

❖ Mask R-CNN 손실 함수

- 기존 Faster R-CNN의 손실 함수에 Segmentation에 관한 손실 함수 추가
- Mask R-CNN의 손실 함수는 Multitask(Classification, Regression, Segmentation) 손실 함수

$$Total\ Loss = \boxed{LOSS_{cls_M1} + LOSS_{reg_M1} + LOSS_{cls_M2} + LOSS_{reg_M2}} + \boxed{LOSS_{segment}}$$

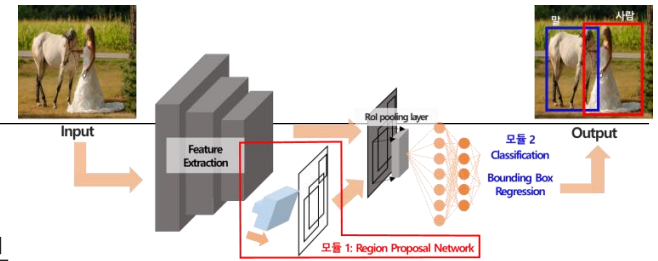


Object Detection 손실 함수
(Faster R-CNN)

Semantic Segmentation 손실 함수
(Fully Convolution Neural Net)

다중 분류 -> 이진 분류

4. Mask R-CNN



❖ Faster R-CNN 모듈 중 Mask R-CNN에서 발전된 부분

- Faster R-CNN은 Region Proposal Network(RPN) 출력 값에 대해 RoI Pooling 진행
- 기존 RoI Pooling은 RPN 출력 값을 정확히 반영하기 어려움

| | | | | | | | | | |
|------|------|------|------|------|------|------|------|------|------|
| 0.71 | 0.99 | 0.74 | 0.93 | 0.94 | 0.83 | 0.82 | 0.08 | 0.84 | 0.75 |
| 0.82 | 0.41 | 0.81 | 0.57 | 0.32 | 0.56 | 0.76 | 0.90 | 0.13 | 0.74 |
| 0.41 | 0.38 | 0.88 | 0.44 | 0.14 | 0.16 | 0.37 | 0.77 | 0.96 | 0.29 |
| 0.43 | 0.21 | 0.19 | 0.45 | 0.57 | 0.16 | 0.63 | 0.29 | 0.71 | 0.94 |
| 0.59 | 0.04 | 0.66 | 0.26 | 0.82 | 0.64 | 0.54 | 0.73 | 0.59 | 0.73 |
| 0.92 | 0.56 | 0.85 | 0.34 | 0.76 | 0.84 | 0.29 | 0.75 | 0.52 | 0.80 |
| 0.76 | 0.59 | 0.32 | 0.74 | 0.21 | 0.39 | 0.34 | 0.03 | 0.33 | 0.43 |
| 0.04 | 0.52 | 0.20 | 0.14 | 0.16 | 0.13 | 0.73 | 0.65 | 0.96 | 0.45 |
| 0.08 | 0.19 | 0.19 | 0.69 | 0.09 | 0.96 | 0.88 | 0.07 | 0.01 | 0.52 |
| 0.41 | 0.28 | 0.68 | 0.30 | 0.55 | 0.94 | 0.48 | 0.08 | 0.10 | 0.92 |

□: RPN로 생성한 Proposal Region

□: 꼭지점을 반올림 -> RoI Pooling에 사용

| | | | | | | |
|------|------|------|------|------|------|------|
| 0.88 | 0.44 | 0.14 | 0.16 | 0.37 | 0.77 | 0.96 |
| 0.19 | 0.45 | 0.57 | 0.16 | 0.63 | 0.29 | 0.71 |
| 0.66 | 0.26 | 0.82 | 0.64 | 0.54 | 0.73 | 0.59 |
| 0.85 | 0.34 | 0.76 | 0.84 | 0.29 | 0.75 | 0.62 |
| 0.32 | 0.74 | 0.21 | 0.39 | 0.34 | 0.03 | 0.33 |
| 0.20 | 0.14 | 0.16 | 0.13 | 0.73 | 0.65 | 0.96 |
| 0.19 | 0.69 | 0.09 | 0.86 | 0.88 | 0.07 | 0.01 |

4. Mask R-CNN

❖ Faster R-CNN 모듈 중 Mask R-CNN에서 발전된 부분

- Faster R-CNN은 Region Proposal Network(RPN) 출력 값에 대해 RoI Pooling 진행
- 기존 RoI Pooling은 RPN 출력 값을 정확히 반영하기 어려움

| | | | | | | | | | |
|------|------|------|------|------|------|------|------|------|------|
| 0.71 | 0.99 | 0.74 | 0.93 | 0.94 | 0.83 | 0.82 | 0.08 | 0.84 | 0.75 |
| 0.82 | 0.41 | 0.81 | 0.57 | 0.32 | 0.56 | 0.76 | 0.90 | 0.13 | 0.74 |
| 0.41 | 0.38 | 0.88 | 0.44 | 0.14 | 0.16 | 0.37 | 0.77 | 0.96 | 0.29 |
| 0.43 | 0.21 | 0.19 | 0.45 | 0.57 | 0.16 | 0.63 | 0.29 | 0.71 | 0.94 |
| 0.59 | 0.04 | 0.66 | 0.26 | 0.82 | 0.64 | 0.54 | 0.73 | 0.59 | 0.73 |
| 0.92 | 0.56 | 0.85 | 0.34 | 0.76 | 0.84 | 0.29 | 0.75 | 0.62 | 0.80 |
| 0.76 | 0.59 | 0.32 | 0.74 | 0.21 | 0.39 | 0.34 | 0.03 | 0.33 | 0.43 |
| 0.04 | 0.52 | 0.20 | 0.14 | 0.16 | 0.13 | 0.73 | 0.65 | 0.96 | 0.45 |
| 0.08 | 0.19 | 0.19 | 0.69 | 0.09 | 0.96 | 0.88 | 0.07 | 0.01 | 0.52 |
| 0.41 | 0.28 | 0.68 | 0.30 | 0.55 | 0.94 | 0.48 | 0.08 | 0.10 | 0.92 |

Region Proposal Network 출력 값 반영하지 못한 부분

| | | | | | | |
|------|------|------|------|------|------|------|
| 0.88 | 0.44 | 0.14 | 0.16 | 0.37 | 0.77 | 0.96 |
| 0.19 | 0.45 | 0.57 | 0.16 | 0.63 | 0.29 | 0.71 |
| 0.66 | 0.26 | 0.82 | 0.64 | 0.54 | 0.73 | 0.59 |
| 0.85 | 0.34 | 0.76 | 0.84 | 0.29 | 0.75 | 0.62 |
| 0.32 | 0.74 | 0.21 | 0.39 | 0.34 | 0.03 | 0.33 |
| 0.20 | 0.14 | 0.16 | 0.13 | 0.73 | 0.65 | 0.96 |
| 0.19 | 0.69 | 0.09 | 0.86 | 0.88 | 0.07 | 0.01 |

4. Mask R-CNN

- ❖ Faster R-CNN 모듈 중 Mask R-CNN에서 발전된 부분
 - RoI Pooling을 RoI Align이라는 기법으로 대체
 - RPN의 출력 값을 정확히 반영하기 위함

| | | | | | | | | | |
|------|------|------|------|------|------|------|------|------|------|
| 0.71 | 0.99 | 0.74 | 0.93 | 0.94 | 0.83 | 0.82 | 0.08 | 0.84 | 0.75 |
| 0.82 | 0.41 | 0.81 | 0.57 | 0.32 | 0.56 | 0.76 | 0.90 | 0.13 | 0.74 |
| 0.41 | 0.38 | 0.88 | 0.44 | 0.14 | 0.16 | 0.37 | 0.77 | 0.96 | 0.29 |
| 0.43 | 0.21 | 0.19 | 0.45 | 0.57 | 0.16 | 0.63 | 0.29 | 0.71 | 0.94 |
| 0.59 | 0.04 | 0.66 | 0.26 | 0.82 | 0.64 | 0.54 | 0.73 | 0.59 | 0.73 |
| 0.92 | 0.56 | 0.85 | 0.34 | 0.76 | 0.84 | 0.29 | 0.75 | 0.52 | 0.80 |
| 0.76 | 0.59 | 0.32 | 0.74 | 0.21 | 0.39 | 0.34 | 0.03 | 0.53 | 0.43 |
| 0.04 | 0.52 | 0.20 | 0.14 | 0.16 | 0.13 | 0.73 | 0.65 | 0.96 | 0.45 |
| 0.08 | 0.19 | 0.19 | 0.69 | 0.09 | 0.96 | 0.88 | 0.07 | 0.01 | 0.52 |
| 0.41 | 0.28 | 0.68 | 0.30 | 0.55 | 0.94 | 0.48 | 0.08 | 0.10 | 0.92 |

| | | | | | | | | | |
|------|------|------|------|------|------|------|------|--|--|
| 0.41 | 0.81 | 0.57 | 0.32 | 0.56 | 0.76 | 0.90 | 0.13 | | |
| 0.38 | 0.88 | 0.44 | 0.14 | 0.16 | 0.37 | 0.77 | 0.96 | | |
| 0.21 | 0.19 | 0.45 | 0.57 | 0.16 | 0.63 | 0.29 | 0.71 | | |
| 0.04 | 0.66 | 0.26 | 0.82 | 0.64 | 0.54 | 0.73 | 0.59 | | |
| 0.56 | 0.85 | 0.34 | 0.76 | 0.84 | 0.29 | 0.75 | 0.52 | | |
| 0.59 | 0.32 | 0.74 | 0.21 | 0.39 | 0.34 | 0.03 | 0.53 | | |
| 0.52 | 0.20 | 0.14 | 0.16 | 0.13 | 0.73 | 0.65 | 0.96 | | |
| 0.19 | 0.19 | 0.69 | 0.09 | 0.96 | 0.88 | 0.07 | 0.01 | | |
| | | | | | | | | | |

4. Mask R-CNN

- ❖ Faster R-CNN 모듈 중 Mask R-CNN에서 발전된 부분
 - RoI Pooling을 RoI Align이라는 기법으로 대체
 - RPN의 출력 값을 정확히 반영하기 위함

| | | | | | | | |
|------|------|------|------|------|------|------|------|
| 0.11 | 0.61 | 0.57 | 0.32 | 0.56 | 0.76 | 0.90 | 0.13 |
| 0.04 | 0.66 | 0.44 | 0.14 | 0.16 | 0.37 | 0.77 | 0.96 |
| 0.21 | 0.19 | 0.45 | 0.57 | 0.16 | 0.63 | 0.29 | 0.71 |
| 0.04 | 0.66 | 0.26 | 0.82 | 0.64 | 0.54 | 0.73 | 0.59 |
| 0.56 | 0.85 | 0.34 | 0.76 | 0.84 | 0.29 | 0.75 | 0.52 |
| 0.59 | 0.32 | 0.74 | 0.21 | 0.39 | 0.34 | 0.03 | 0.33 |
| 0.52 | 0.20 | 0.14 | 0.16 | 0.13 | 0.73 | 0.65 | 0.96 |
| 0.19 | 0.19 | 0.69 | 0.09 | 0.06 | 0.06 | 0.07 | 0.91 |

| | | |
|---|----|---|
| 4 | 8 | 2 |
| 8 | 16 | 4 |
| 2 | 4 | 1 |

0.66

$$\begin{aligned} &= \frac{1}{49} (4 \times 0.41 + 8 \times 0.61 + 2 \times 0.57 \\ &+ 8 \times 0.88 + 16 \times 0.88 + 4 \times 0.44 \\ &+ 2 \times 0.21 + 4 \times 0.19 + 1 \times 0.45) \end{aligned}$$

4. Mask R-CNN

- ❖ Faster R-CNN 모듈 중 Mask R-CNN에서 발전된 부분
 - RoI Pooling을 RoI Align이라는 기법으로 대체
 - RPN의 출력 값을 정확히 반영하기 위함

| | | | | | | | |
|------|------|------|------|------|------|------|------|
| 0.41 | 0.61 | 0.57 | 0.32 | 0.56 | 0.76 | 0.90 | 0.13 |
| 0.88 | 0.88 | 0.44 | 0.32 | 0.16 | 0.37 | 0.77 | 0.96 |
| 0.21 | 0.19 | 0.45 | 0.57 | 0.16 | 0.63 | 0.29 | 0.71 |
| 0.04 | 0.66 | 0.26 | 0.82 | 0.64 | 0.54 | 0.73 | 0.59 |
| 0.56 | 0.85 | 0.34 | 0.76 | 0.84 | 0.29 | 0.75 | 0.52 |
| 0.59 | 0.32 | 0.74 | 0.21 | 0.39 | 0.34 | 0.03 | 0.33 |
| 0.52 | 0.20 | 0.14 | 0.16 | 0.13 | 0.73 | 0.65 | 0.96 |
| 0.19 | 0.19 | 0.69 | 0.09 | 0.86 | 0.86 | 0.07 | 0.91 |



$$\begin{aligned} &= \frac{1}{49} (6 \times 0.57 + 8 \times 0.32 \\ &+ 12 \times 0.44 + 16 \times 0.14 \\ &+ 3 \times 0.45 + 4 \times 0.57) \end{aligned}$$

4. Mask R-CNN

- ❖ Faster R-CNN 모듈 중 Mask R-CNN에서 발전된 부분
 - RoI Pooling을 RoI Align이라는 기법으로 대체
 - RPN의 출력 값을 정확히 반영하기 위함(Bilinear Interpolation)
 - RoI Align 후, Bounding Box와 범주 예측(Object Detection)과 Semantic Segmentation 진행

| | | | | | | | |
|------|-------------|------|-------------|------|-------------|------|-------------|
| 0.41 | 0.61 | 0.57 | 0.32 | 0.56 | 0.76 | 0.90 | 0.13 |
| 0.88 | 0.66 | 0.44 | 0.35 | 0.14 | 0.38 | 0.37 | 0.50 |
| 0.21 | 0.19 | 0.45 | 0.57 | 0.16 | 0.63 | 0.29 | 0.71 |
| 0.04 | 0.34 | 0.66 | 0.55 | 0.82 | 0.50 | 0.54 | 0.58 |
| 0.56 | 0.85 | 0.34 | 0.76 | 0.84 | 0.29 | 0.75 | 0.52 |
| 0.59 | 0.59 | 0.74 | 0.51 | 0.21 | 0.50 | 0.34 | 0.44 |
| 0.52 | 0.30 | 0.14 | 0.25 | 0.16 | 0.29 | 0.73 | 0.47 |
| 0.19 | 0.19 | 0.69 | 0.09 | 0.85 | 0.66 | 0.07 | 0.01 |

| | | | |
|-------------|------|-------------|-------------|
| 0.66 | 0.35 | 0.38 | 0.50 |
| 0.34 | 0.55 | 0.50 | 0.58 |
| 0.59 | 0.51 | 0.50 | 0.44 |
| 0.30 | 0.25 | 0.29 | 0.47 |

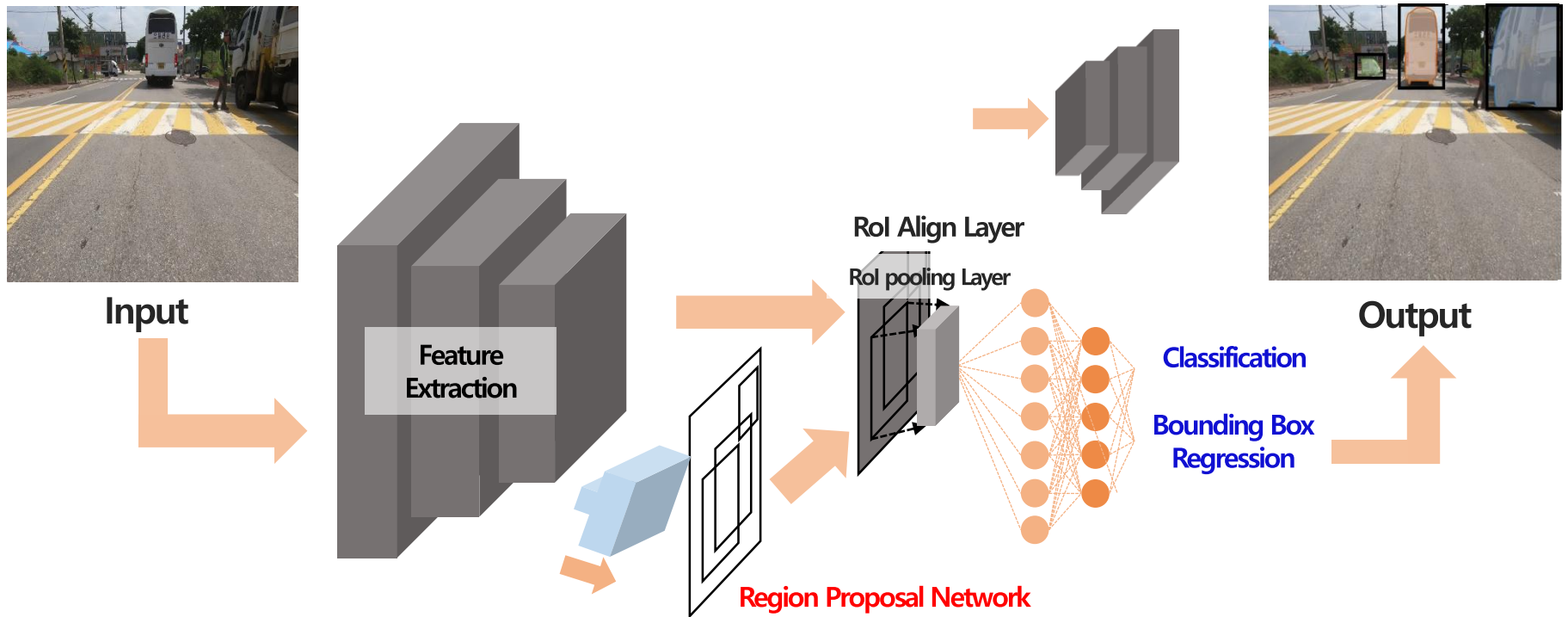
RoI Align 결과

| | |
|-------------|-------------|
| 0.66 | 0.58 |
| 0.59 | 0.50 |

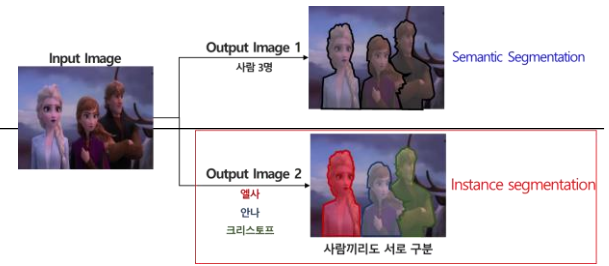
4. Mask R-CNN

❖ Mask R-CNN 정리

- Faster R-CNN에 Bounding Box 내부에서 **Semantic Segmentation**을 진행하는 모듈 추가
- Faster R-CNN의 RoI Pooling을 **RoI Align**으로 대체



4. Mask R-CNN



❖ Instance segmentation 결과 비교

- Bounding Box를 탐지 후, Segmentation을 진행했기 때문에 Mask R-CNN 성능 뛰어남
- FCIS: Mask R-CNN 이전 가장 좋은 성능을 가진 Instance segmentation 알고리즘



| | Average Precision(AP) |
|--|-----------------------|
| FCIS (기존 Instance segmentation 알고리즘) | 33.6 |
| Mask R-CNN | 37.1 |

4. Mask R-CNN

- ❖ Mask R-CNN 손실 함수에 대한 실험 결과
 - **Mask R-CNN**은 Object Detection을 진행 후 Segmentation 진행
 - **Mask R-CNN**에 대한 하이퍼파라미터는 **Faster R-CNN**과 동일
 - **Faster R-CNN**의 **RoI Pooling**을 **RoI Align**으로 대체한 것보다도 성능이 뛰어남
 - Multitask 손실 함수를 사용함에 따라 Object Detection 성능 증가를 확인

| | Average Precision(AP) |
|--------------------------------|-----------------------|
| Faster R-CNN+++ | 34.9 |
| Faster R-CNN with FPN | 36.2 |
| Faster R-CNN by G-RMI | 34.7 |
| Faster R-CNN with TDM | 36.8 |
| Faster R-CNN, RoI Align | 38.2 |
| Mask R-CNN | 39.8 |

4. Mask R-CNN

❖ Mask R-CNN 영상 적용 예시



- <https://www.youtube.com/watch?v=nGYg9hGAQQI>

목차

1. Introduction
2. Object Detection
3. Image Segmentation
4. Mask R-CNN
- 5. Conclusions**
6. Additional Topic

5. Conclusions

❖ 결론

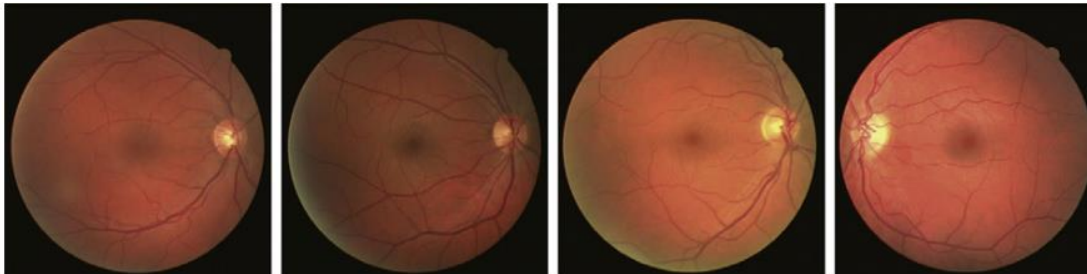
- 객체 탐지(Object Detection) -> Faster R-CNN
 - 입력 변수(정보): RGB 사진
 - 레이블: 사진 내 존재하는 범주들과 범주 별 위치 정보(X, Y, W, H)
- 이미지 분할(Image Segmentation) -> Fully Convolutional Neural Network(FCN)
 - 입력 변수(정보): RGB 사진
 - 레이블: 픽셀 별로 어느 범주에 속하는 지에 관해 레이블링 된 사진
- Mask R-CNN은 Instance Segmentation을 하는 알고리즘
 - 입력 변수(정보): RGB 사진
 - 레이블 ①: 사진 내 존재하는 범주들과 범주 별 위치 정보(X, Y, W, H)
 - 레이블 ②: 픽셀 별로 어느 범주에 속하는 지에 관해 레이블링 된 사진
- 레이블이 존재하지 않는 경우, 위의 알고리즘 사용 불가능

5. Conclusions

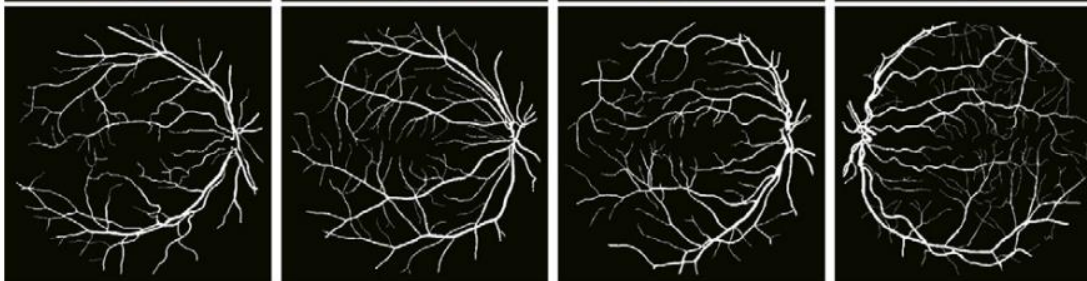
❖ 추후 계획

- 레이블은 수작업을 통해 생성 가능하기에 많은 인력, 시간이 필요하고 비용도 많이 발생
- 레이블링을 하지 않거나 소량만 해서 비용을 절감하고자 하는 연구들이 등장
 - 비지도 학습(Unsupervised Learning)기반 Computer Vision 알고리즘
 - 준지도 학습(Semi-supervised Learning)기반 Computer Vision 알고리즘
 - **Weakly Supervised Learning Computer Vision 알고리즘**
 - 사전 학습된 모델을 사용해 레이블링 도구를 개발하고자 하는 연구

입력 이미지



Ground Truth



목차

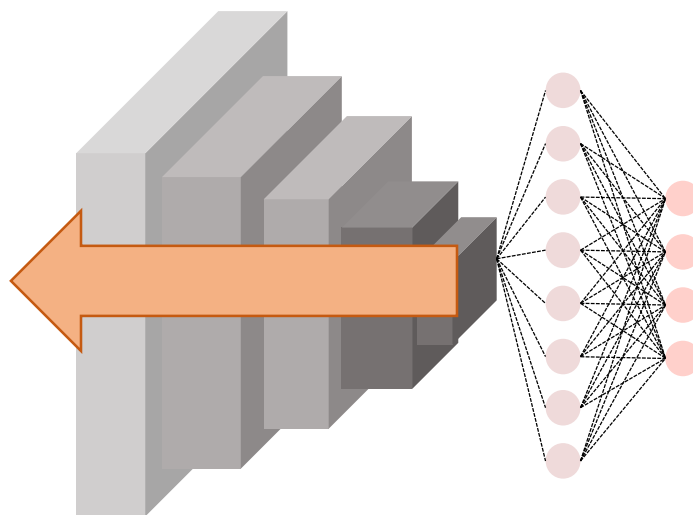
1. Introduction
2. Object Detection
3. Image Segmentation
4. Mask R-CNN
5. Conclusions
- 6. Additional Topic**

6. Additional Topic

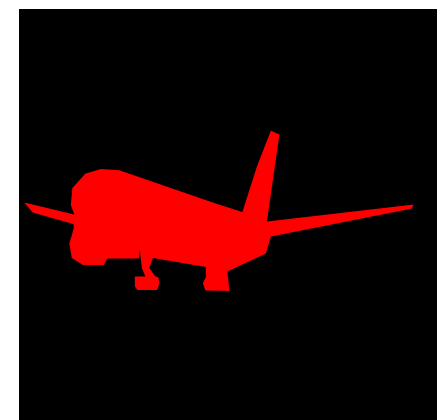


❖ Weakly Supervised Semantic Segmentation

- 입력 변수는 RGB 이미지, 출력 변수는 이미지 내 존재하는 범주
- 기존 Semantic Segmentation 레이블보다 적은 정보
- **Class Activation Map(CAM) 결과를 중심으로 픽셀 값을 군집화하여 Segmentation 수행**



비행기



CAM
(CNN 원인 분석 방법론)

- <http://dmqa.korea.ac.kr/activity/seminar/256>

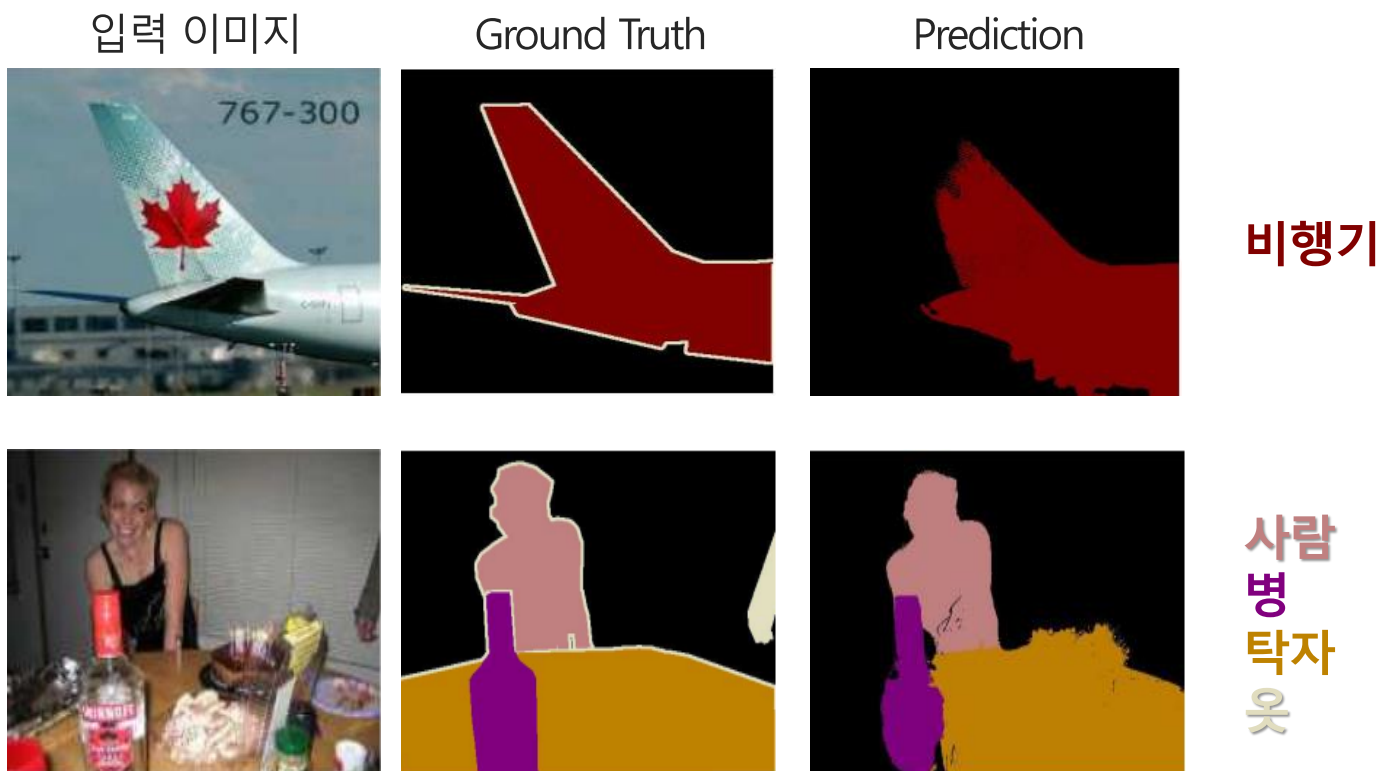
- <https://blog.lunit.io/2019/04/25/ficklenetweakly-and-semi-supervised-semantic-image-segmentation-using-stochastic-inference/>

- Lee, J., Kim, E., Lee, S., Lee, J., & Yoon, S. (2019). Ficklenet Weakly and semi-supervised semantic image segmentation using stochastic inference. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 5267-5276).

6. Additional Topic

❖ Weakly Supervised Semantic Segmentation 결과

- Mean Intersection Over Union(mIoU) : **61.9%**
- 이미지 내 존재하는 범주만 가지고 Segmentation 시행했지만 좋은 성능을 보임



Q & A

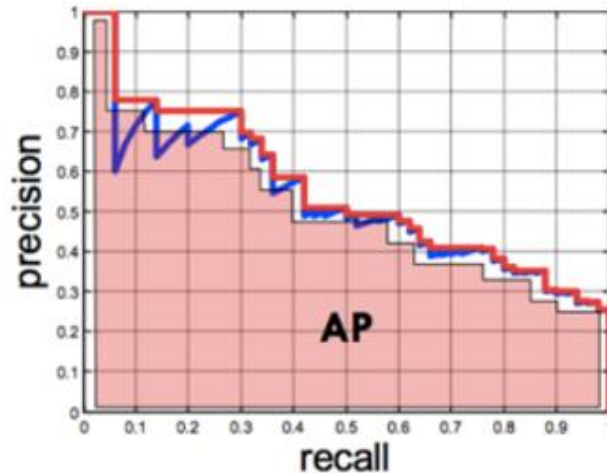
감사합니다.

Appendix. Object Detection 평가 지표

❖ Faster R-CNN 실험 결과

- **Average Precision:** precision-recall 그래프 아래 면적
- **mAP(mean Average Precision):** 모든 범주에 대한 Average Precision의 평균

범주 별로 Average Precision 산출



데이터 셋
내부
모든 말(Horse)
Precision 계산
Recall 계산

$$mAP = \frac{1}{N} \sum_{i=1}^N AP_i$$

N: 사전에 정의한 범주 개수
 AP_i : i번째 범주에 대한 Average Precision