**One Stage OD vs Two Stage OD**

**1 Stage OD vs 2 Stage OD 비교**

Reference Link : Click [Here](https://velog.io/@cha-suyeon/%EB%94%A5%EB%9F%AC%EB%8B%9D-Object-Detection-Architecture-1-stage-detector%EC%99%80-2-stage-detector)!

**2-Stage Object Detecting**

1. **Region Proposal** = Object가 있을 법한 위치의 후보를 선정하는 단계
2. **Classification** = 실제로 Object가 있는지를 Classification하고,

정확한 Bounding Box를 구하는 Regression을 수행하는 단계

즉, 2-Stage Object Detecting은 Regional Proposal과 Classification/Regression이 순차적으로 이루어진다.

다시 말하면, Proposed Region에 한해서 Classification/Regression이 진행되기 때문에

전반적인 정확도는 좋지만, 속도가 느린 단점이 있다.

대표적인 **2-Stage Object Detecting Model: R-CNN계열**(R-CNN, Fast R-CNN, Faster R-CNN)

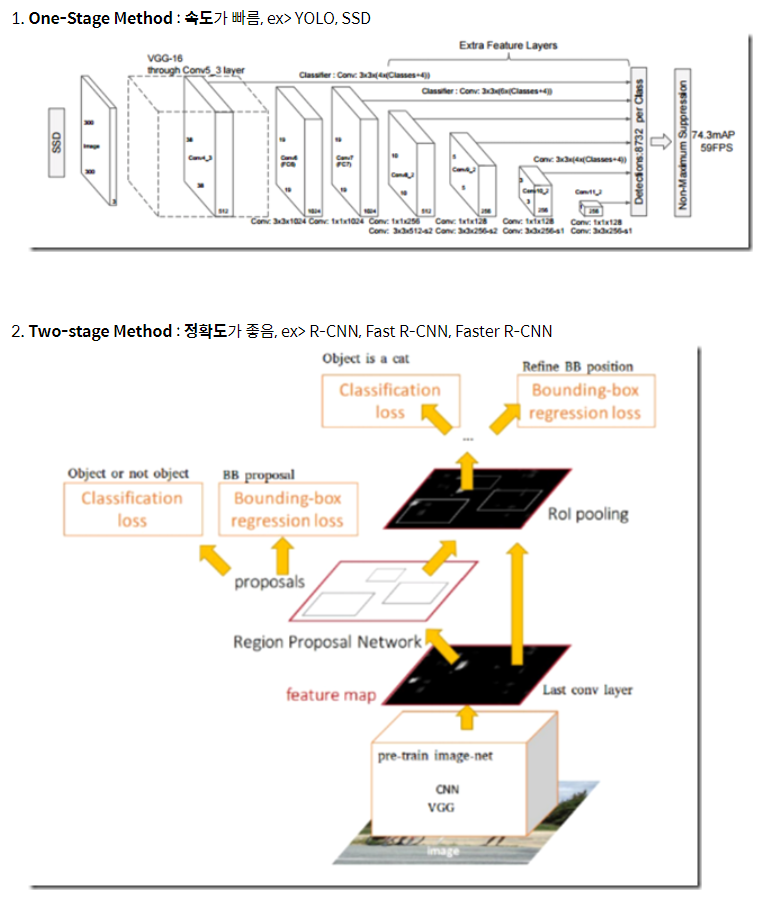
**1-Stage Object Detecting**

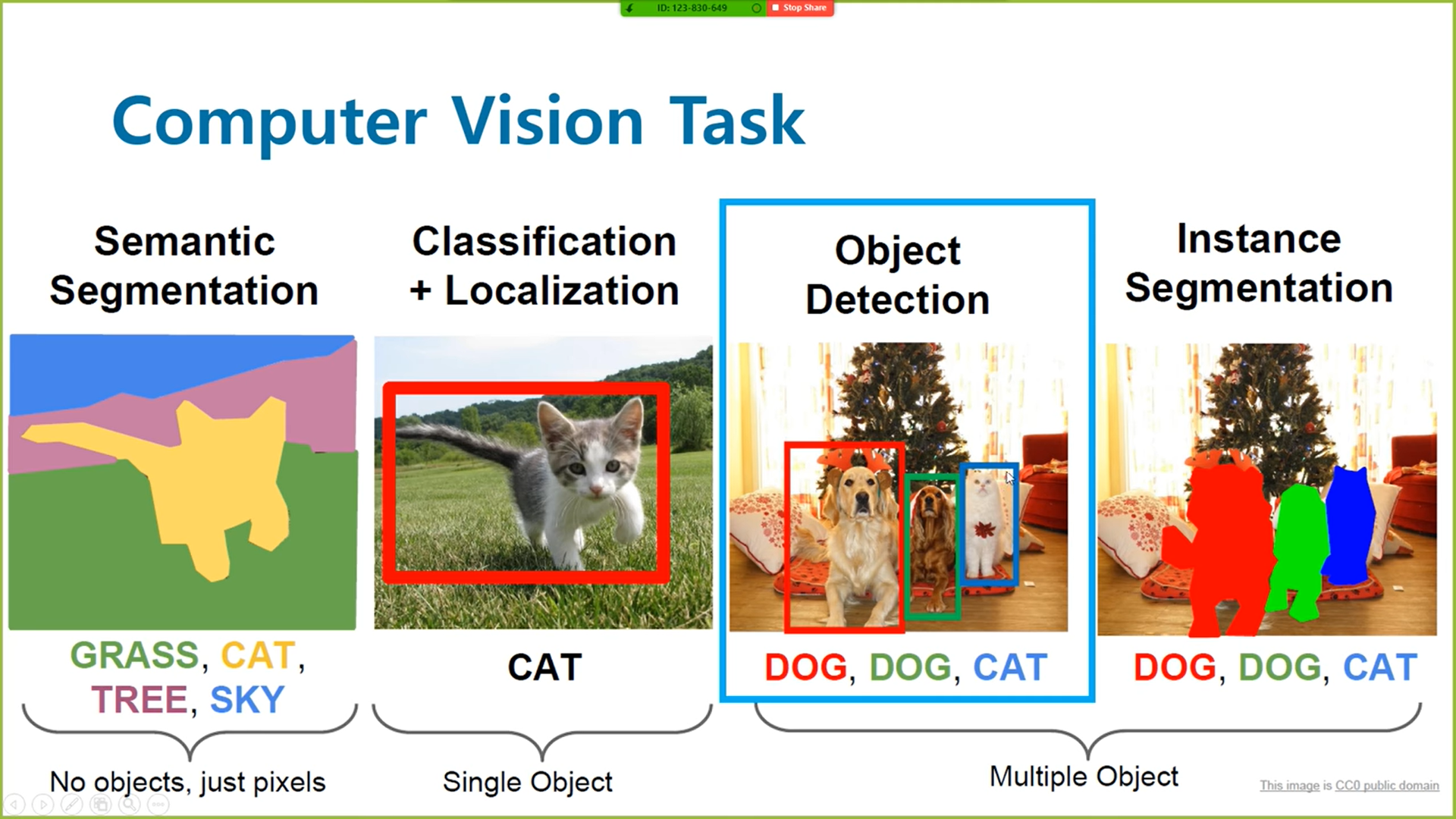
**1-Stage**에서는 Regional Proposal과 Classification이 동시에 이루어진다.

따라서 정확도가 다소 떨어지지만, 비교적 빠른 편에 속한다.

대표적인 **1-Stage Object Detecting Model: YOLO, SSD계열**(YOLO, SSD, RetinaNet, RefineDet)





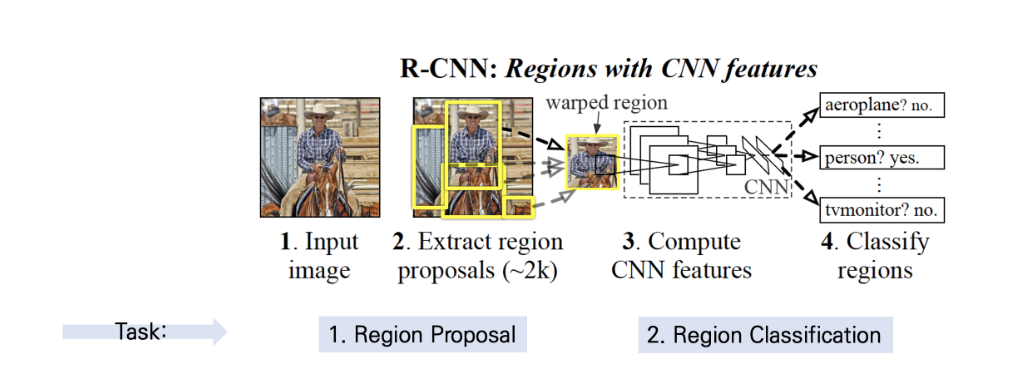


**Two-Stage Object Detection Structure**

**R-CNN**

전통적인 two-stage OD으로,

1. 객체가 있을 법한 예상 범위 추출 작업(Region Proposal)을 진행
2. 제안된 영역을 CNN에 넣어서 Classification을 진행



**# R-CNN 알고리즘의 구체적인 개요**

1. 입력 이미지에 **Selective Search** 알고리즘을 적용해 물체가 있을 만한 **박스 후보 2000개 추출**

Sliding window를 모든 window에 전부 적용하지 않고,  
몇 개의 window만 골라서 거기에만 component classifier을 적용하는 방식

즉, Region Proposal을 Selective Search로 진행하며 Bottom-Up Segmentation과 Merging과정을 통해 핵심 ROI추출

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

2. 모든 박스를 CNN에 들어갈 수 있는 사이즈인 227 x 227 사이즈로 resize(=warp). 이 때 박스의 종횡비는 무시

3. 미리 이미지셋 데이터를 통해 학습시켜 놓은 CNN을 통과시켜 4096 차원의 Feature 벡터를 추출

4. 추출된 벡터를 클래스 종류 마다 학습시켜 놓은 SVM Classifier(이진분류기)에 통과시킴

5. Bounding box Regression을 적용해 박스의 위치를 더 정확히 조정

**+) R-CNN의 구조**

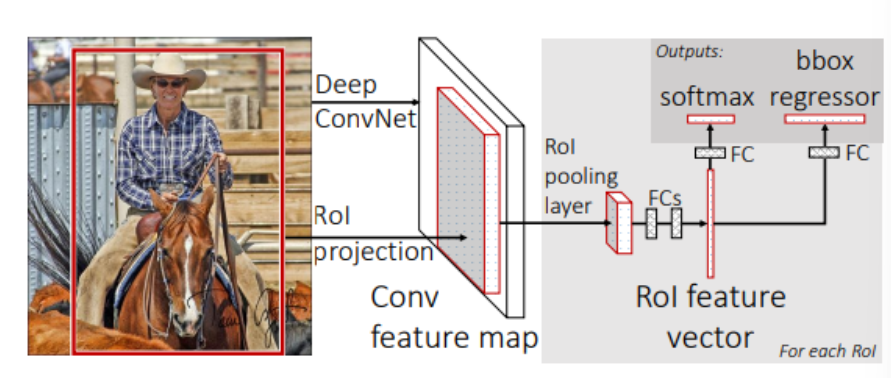
Pre-Trained ConvNet(AlexNet)사용 (ImageNet Classification Dataset으로 Pre-Trained)

**# R-CNN 알고리즘의 단점**

1. region proposal을 selective search로 수행한 뒤 약 2,000개에 달하는 후보 이미지 각각에 대해서 convolution 연산을 수행하게 된다. 한 이미지에서 feature을 반복해서 추출하기에 비효율적이고 느리다.
2. SVM과정과 Bounding Box Regression Training 과정에서 추출된 Feature들을 Disk에 저장해서 공간과 시간이 많이 필요하다.

**Fast R-CNN**

* **Convolution Neural Network로 들어가는 입력의 수를 1개**로 줄임으로써, R-CNN 알고리즘의 단점 극복!
* Feature들을 저장하는 데에 별도의 공간이 필요하지 않다!



**# Fast R-CNN 알고리즘의 개요**

1. 이미지에서 RCNN처럼 Region Proposal 과정을 거친다.

2. 전체 이미지를 CNN모델의 입력으로 하고, MaxPooling을 활용하여 Feature Map을 만든다.

3. 전체 이미지의 Feature Map으로부터 기존에 Region Proposal과정에서 나온 영역의 fixed-length feature vector를 추출

4. 각각의 feature vector를 Fully Connected Layer층에 넣는다.

5. 출력에서 2가지가 있는데, 하나는 softmax층으로 들어가서 해당 객체의 유사도(확률 값)를 출력하고,  
다른 하나는 해당 class의 bounding box regression을 수행한다.

**이렇게 된 경우의 장점**

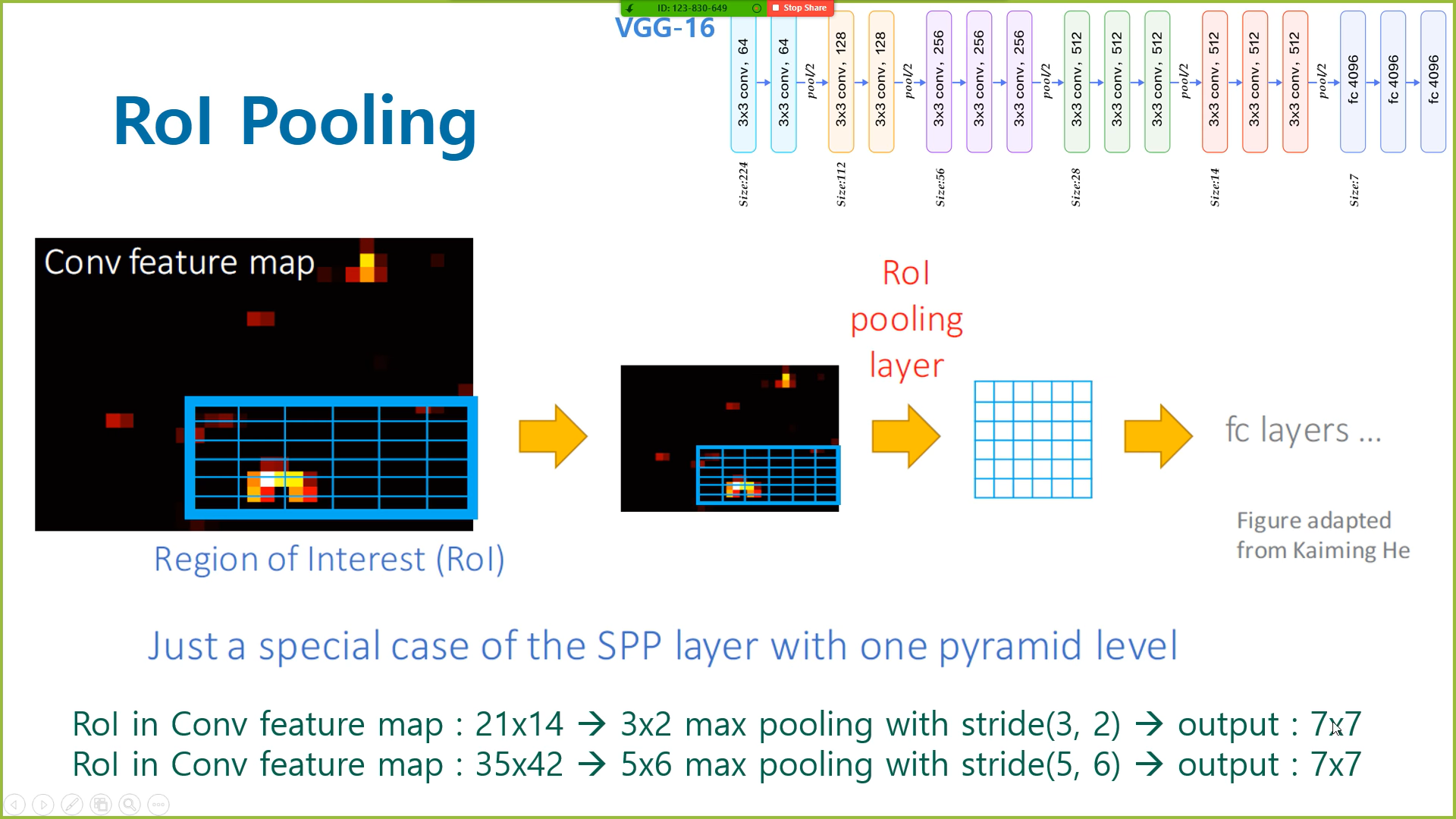
Fast R-CNN에서는 R-CNN처럼 2000개에 달하는 객체 후보영역을 하나씩 집어넣는 것이 아니라, Region Proposal을 추출하고 원본 이미지 자체를 CNN에 넣는다. 그 후에 검출된 Feature Map에서 각각의 Region Proposal에서 Fixed-length feature vector를 추출한다.

이렇게 되면 결과는 같으나, CNN에 들어가는 입력의 개수가 1개로 줄어들고, 겹쳐져 있던 ROI를 반복해서 CNN으로 계산할 필요가 없어져서 연산량이 줄어들게 된다.

+) Softmax를 마지막 층에 사용한 이유:  
Multi Task Loss를 사용할 수 있어, Classification과 Bounding Box Regression을 한 번에 계산가능!

즉, R-CNN과 Fast R-CNN의 가장 큰 차이는

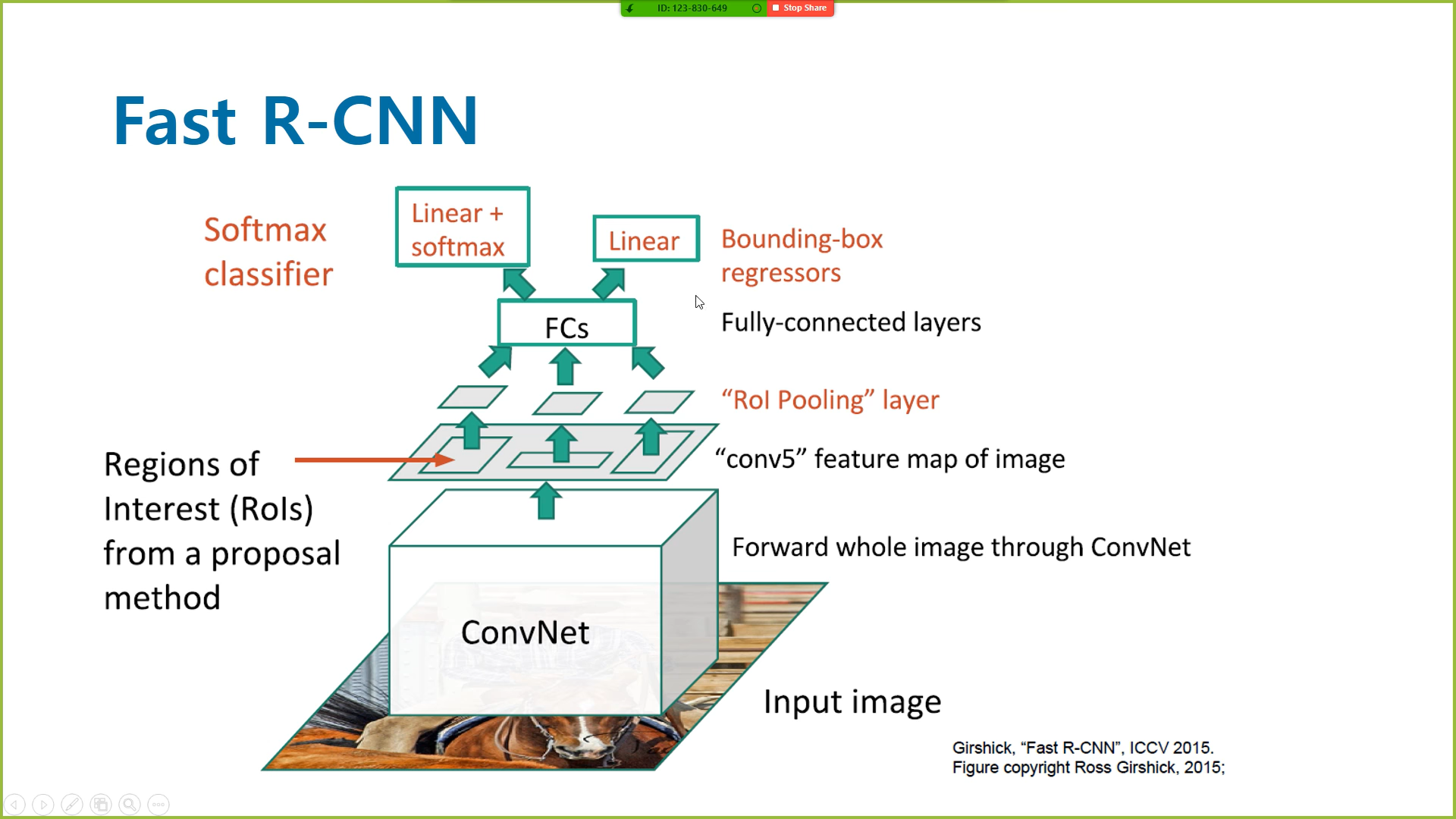
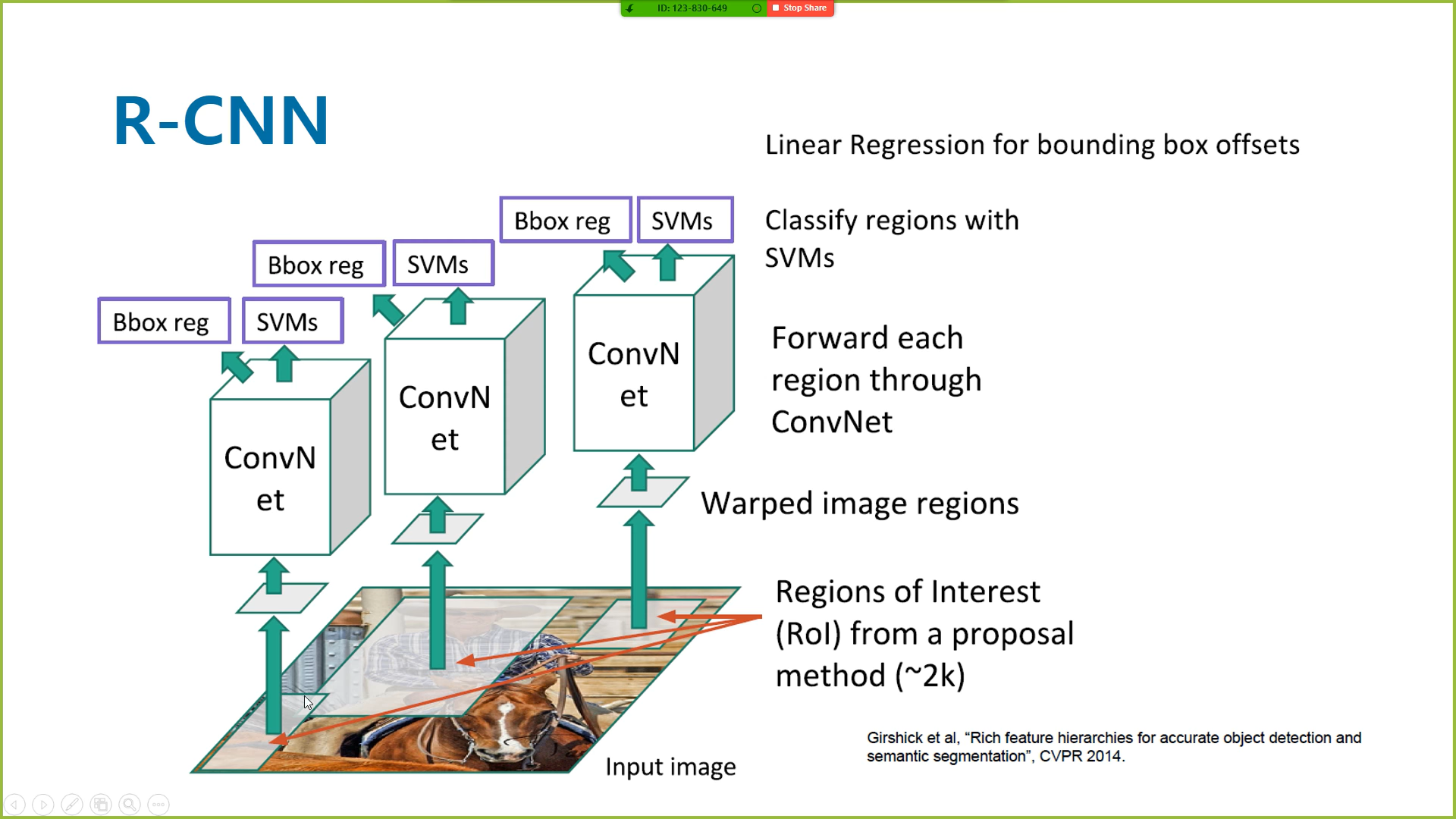
**Region Proposal을 하고 CNN에 입력으로 주느냐(=R-CNN),  
CNN을 먼저 하고 Region Proposal을 하느냐(=Fast R-CNN)** 이 차이

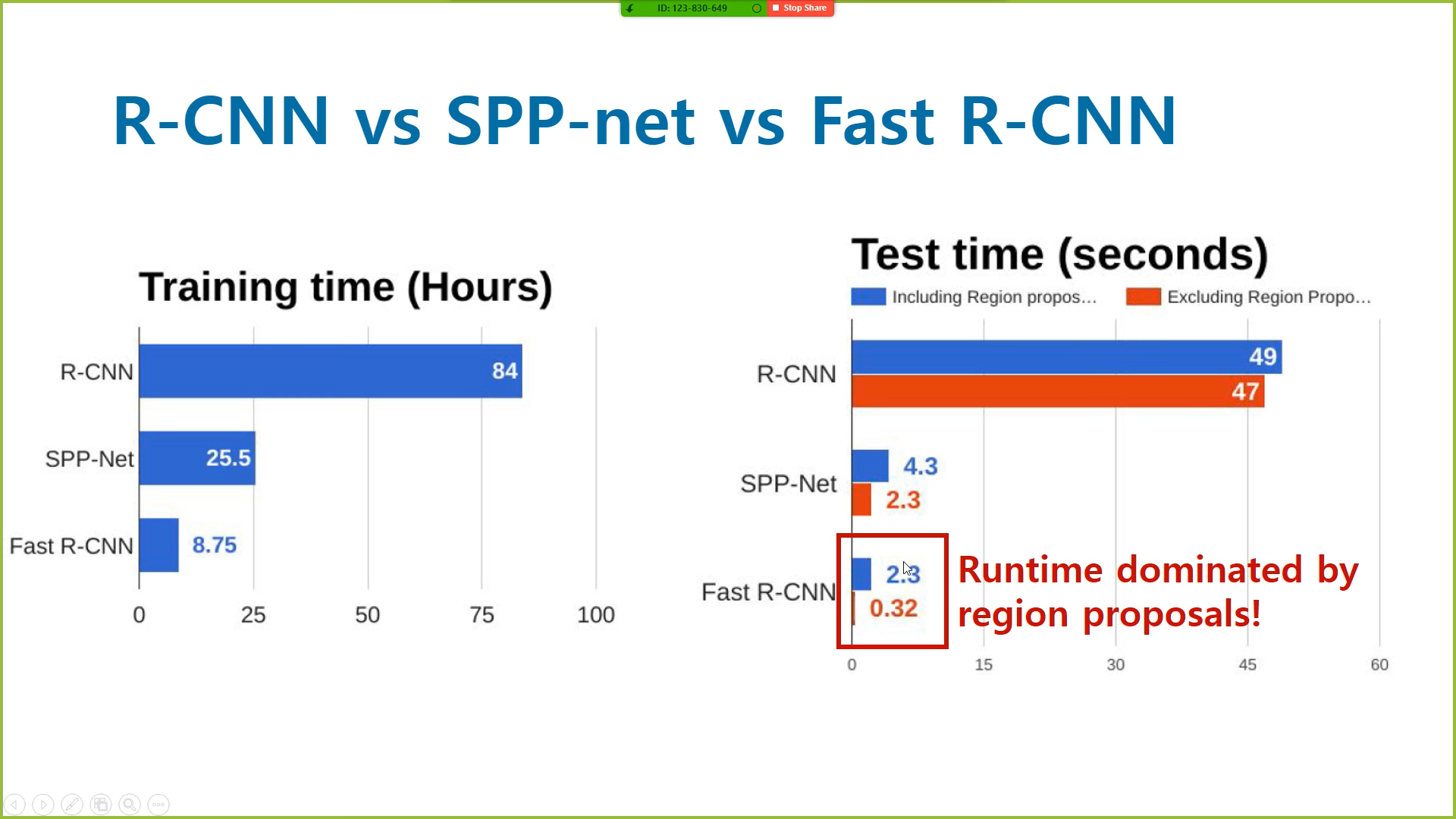


**ROI Pooling**

* 관심 영역의 크기는 그때그때 다를 수 있지만,  
  ROI Pooling을 거치면, 그 출력의 크기는 축소되지만, 같은 크기로 줄어들도록 한 것!

(Fully Connected Layer를 구성)





**Fast R-CNN의 단점에서는,**

Selective Search Algorithm은 CPU로 돌아가고, Neural Network가 GPU로 돌아가는데,  
Selective Search Algorithm이 항상 2초는 걸린다. – **치명적인 단점**

**즉, Fast R-CNN의 문제점을 정리해보면**

Region Proposal 파트가 바깥쪽에 있고 그게 Selective Search를 이용하는데  
Test Time에 Bottleneck을 일으킨다.

따라서, Selective Search Algorithm을 **Region Proposal Network라는 개념을 도입**해서  
Region Proposal 자체를 GPU에서 돌림으로써, end-to-end model을 구현한 것이 **Faster R-CNN**

즉, **Faster R-CNN =** RPN + Fast R-CNN

**Region Proposal Network**

먼저 이미지에 CNN을 적용해 feature을 뽑아내면, Feature Map만을 보고 Object가 있는지 알아낼 수 있다.

이 Feature Map을 보고 Object가 위치하고 있을 법한 후보 영역들을 얻어내는 Network가 RPN이다.

텍스트, 명함이(가) 표시된 사진

자동 생성된 설명

**즉,**

RPN은 기존의 convolutional network(ex. VGG)에 region proposal을 위한 몇 개의 컨볼루션 레이어를 추가한 형태

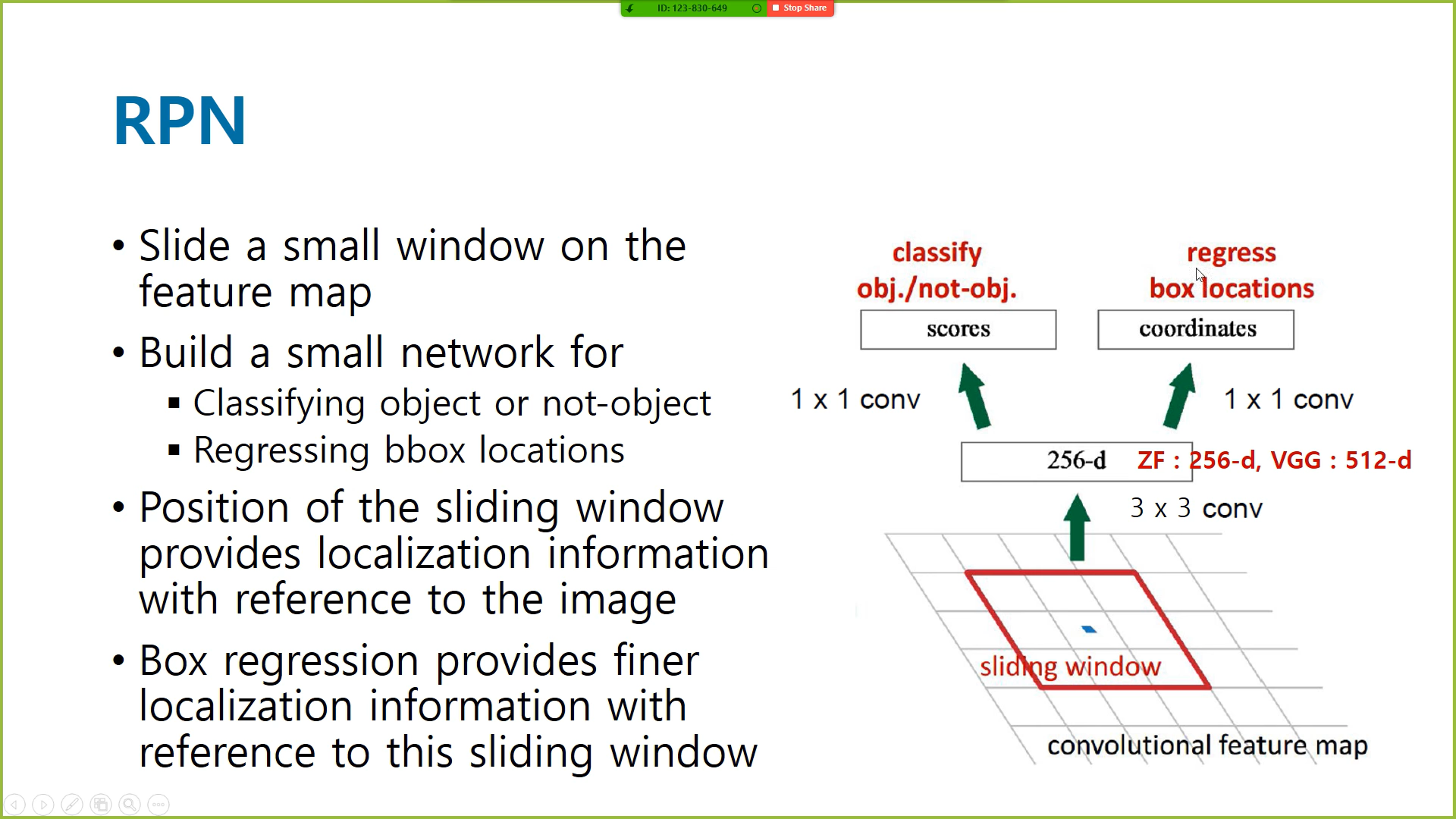
Fast R-CNN에서는 Region Proposal이 비 신경망 구조였다면,

Faster R-CNN에서는 신경망 구조(neural network)를 가지게 되는 것이다!

이와 같이, **후보 영역을 얻어내는 과정이 Fast R-CNN과 다른 것이며, 이 이후는 Fast R-CNN과 동일하다.**

Fast R-CNN에서는 region proposal은 별개의 단계라고 생각했다면,

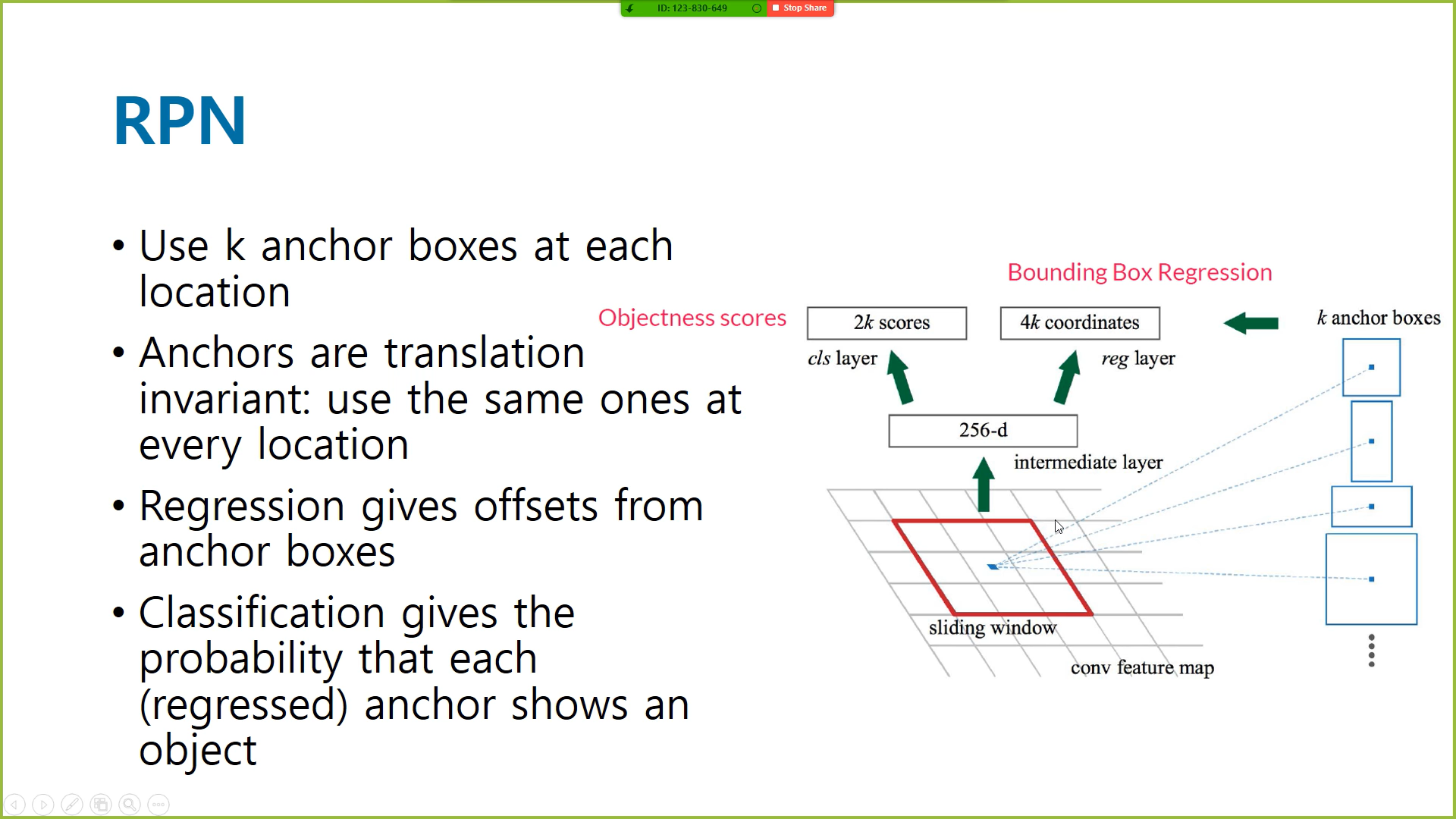
Faster R-CNN은 Fast R-CNN에 RPN만 앞에 붙인 것.



**RPN에서는 물체 존재 여부 판단**

**Bounding Box Regression** 2가지를 RPN에서 진행

또한, 정확한 Detecting을 위해 본 논문에서는 9개의 **Anchor Box**를 정의



**k개의 anchor box,**

각각의 box에 대해 물체가 있는지 없는지 Yes/No로 판별 - 2k 개

각각의 box에 대해 좌표를 출력 - 4k 개

**즉,** Anchor = **Pre-Defined Reference Box**

Anchor has **Multi Scale and Multi Size**, Multiple Anchors **are used at each position**

**논문에서 사용한 9개의 Anchor Box**

* **3 scale (128x128, 256x256, 512x512)**
* **3 ratio (2:1, 1:1, 1:2)**

**즉, 총 9개!**

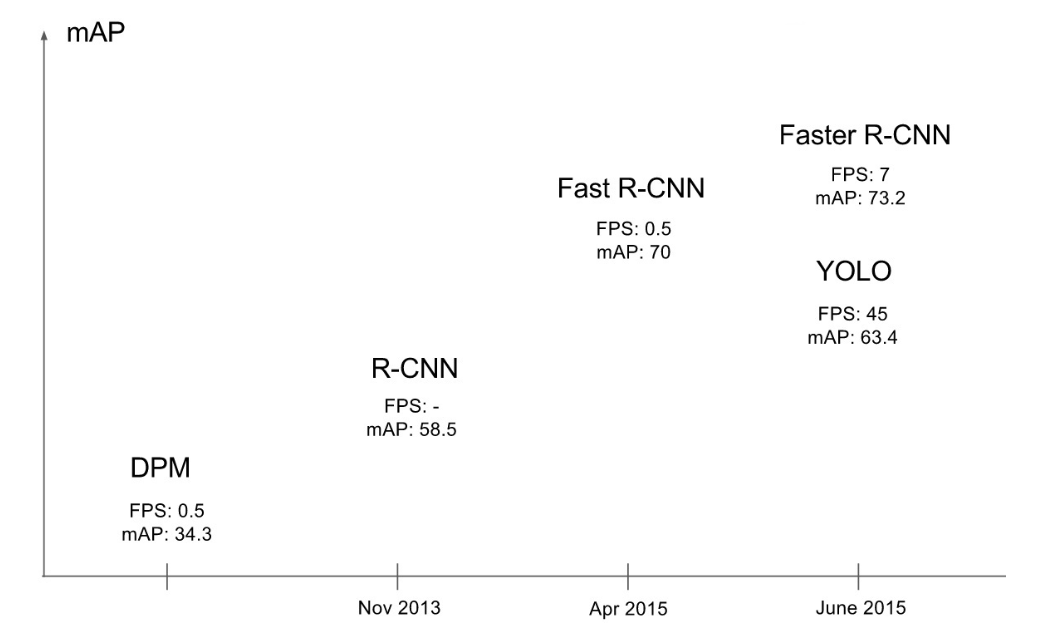
**Faster R-CNN의 특징**

파이프라인 네트워크를 사용하지 않고 하나의 신경망으로 문제를 해결하는 end to end 모델

5fps의 빠른 실행시간 (Fast R-CNN에서도 CNN연산을 1번만 진행하기 때문에 시간이 크게 감소했지만 실시간 수준은 아니었다)

**Fine Tuning 이란? Click** [**Here**](https://eehoeskrap.tistory.com/186)**!**

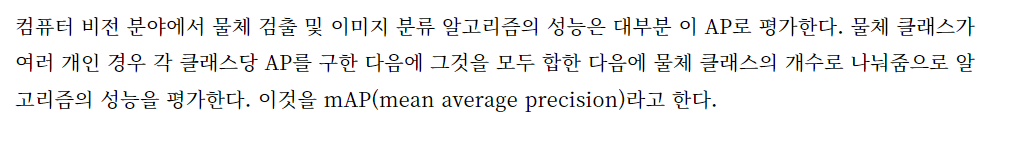
* 기존에 학습되어져 있는 모델을 기반으로 Architecture를 새로운 목적  
  (나의 이미지 데이터에 맞게) 변형하고 이미 학습된 모델 Weights로부터 학습을 업데이트하는 방법.
* 모델의 파라미터를 미세하게 조정하는 행위

**컴퓨터비전 분야의 물체 인식에 대한 CNN의 모델 성능 평가 지표 =** **mAP** ****<https://eehoeskrap.tistory.com/186>

**Click** [**Here**](https://ctkim.tistory.com/79)**!**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

****