**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)

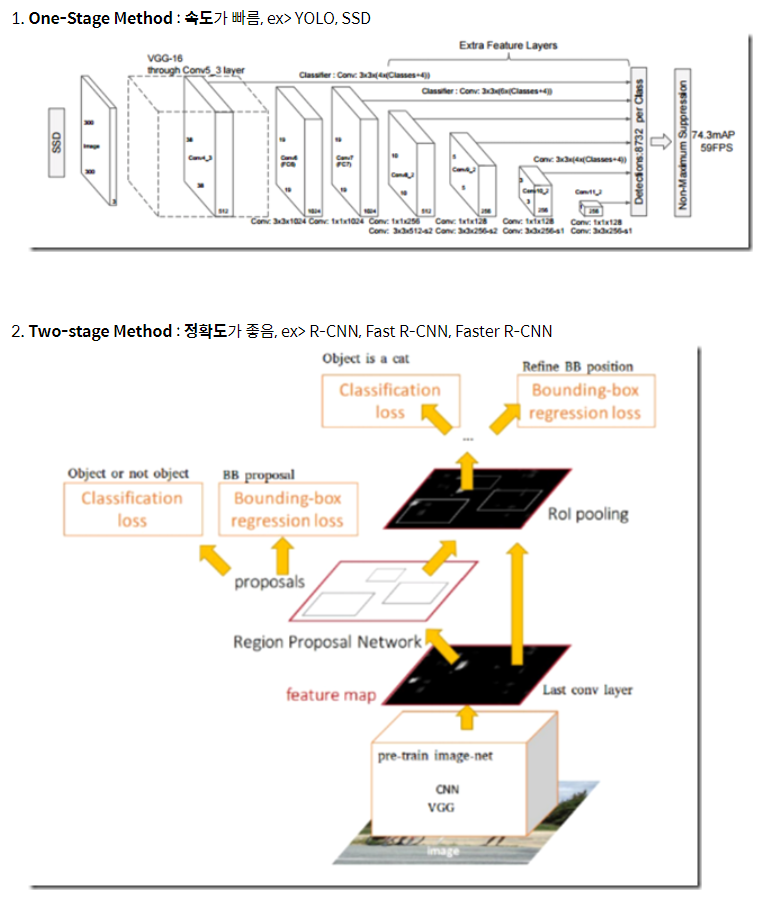
**1-Stage Object Detecting**

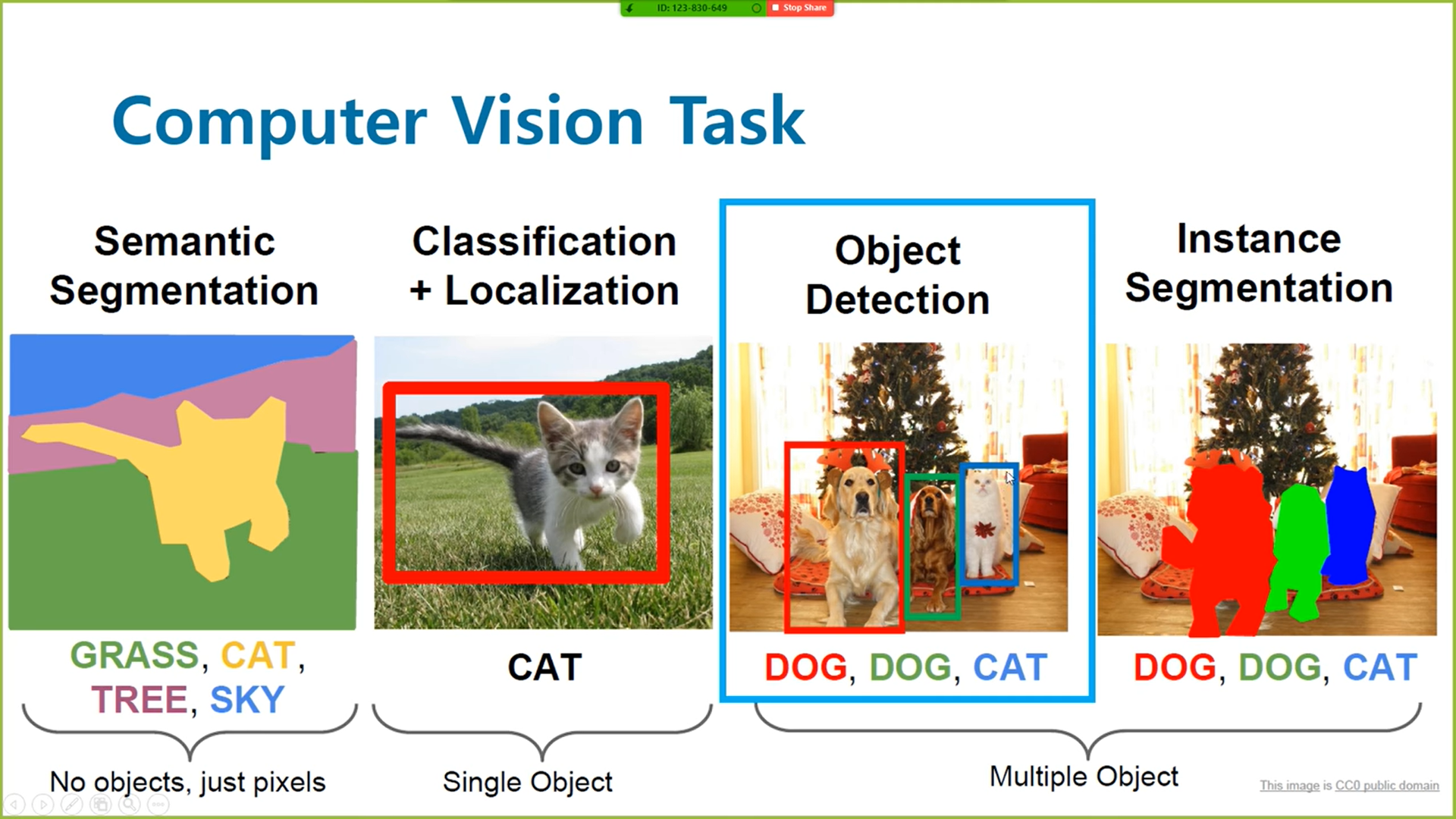
**1-Stage**에서는 Regional Proposal과 Classification이 동시에 이루어진다.

따라서 정확도가 다소 떨어지지만, 비교적 빠른 편에 속한다.

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





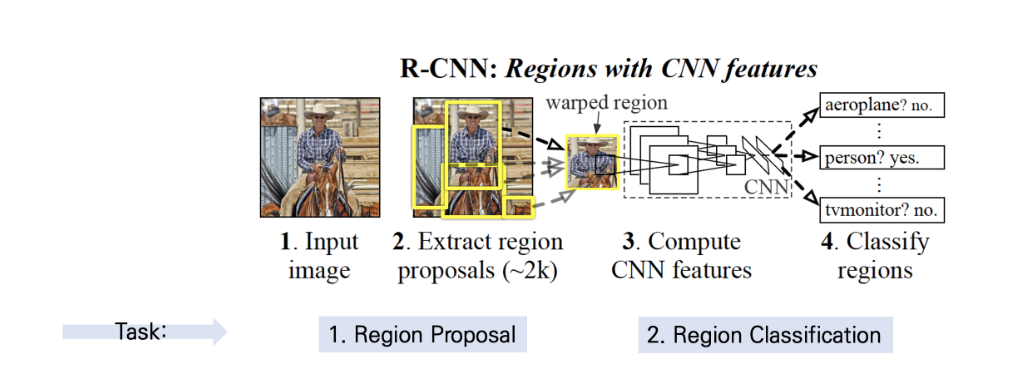


**Two-Stage Object Detection Structure**

**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**

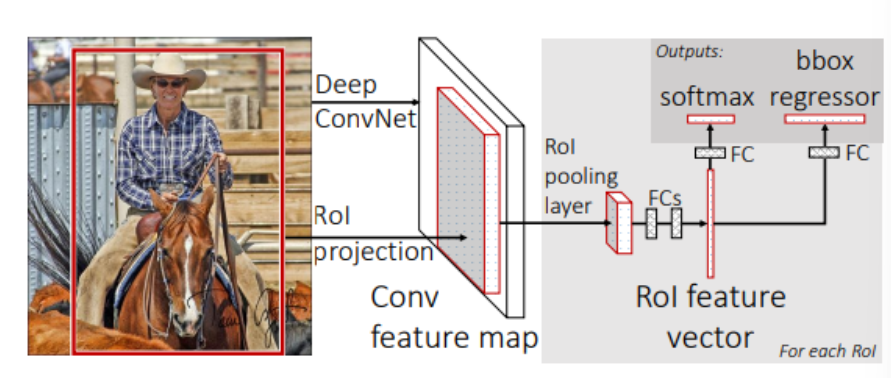
Pre-Trained ConvNet(AlexNet)사용 (ImageNet Classification Dataset으로 Pre-Trained)

**# 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**

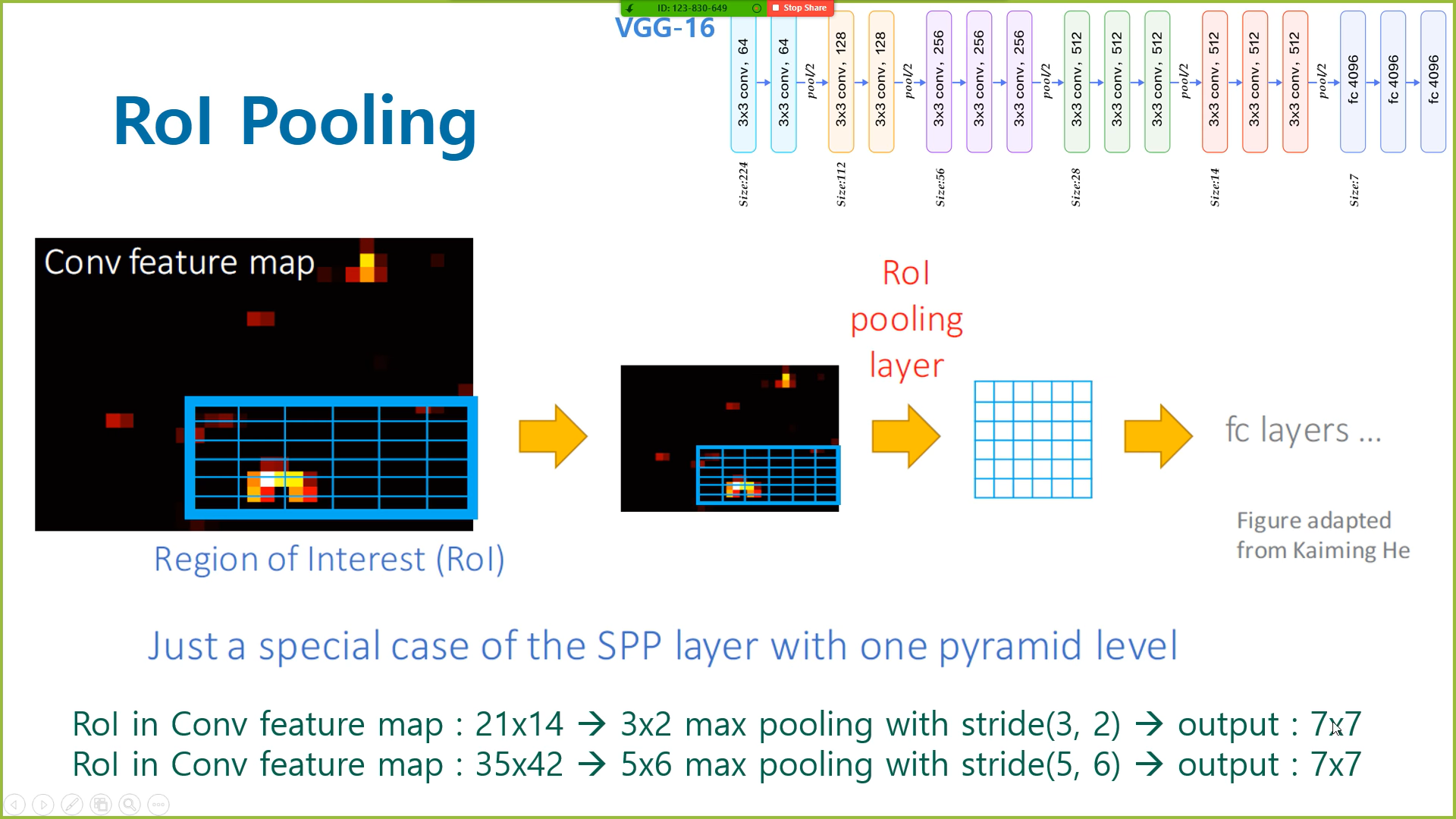
Fast R-CNN에서는 R-CNN처럼 2000개에 달하는 객체 후보영역을 하나씩 집어넣는 것이 아니라, Region Proposal을 추출하고 원본 이미지 자체를 CNN에 넣는다. 그 후에 검출된 Feature Map에서 각각의 Region Proposal에서 Fixed-length feature vector를 추출한다.

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

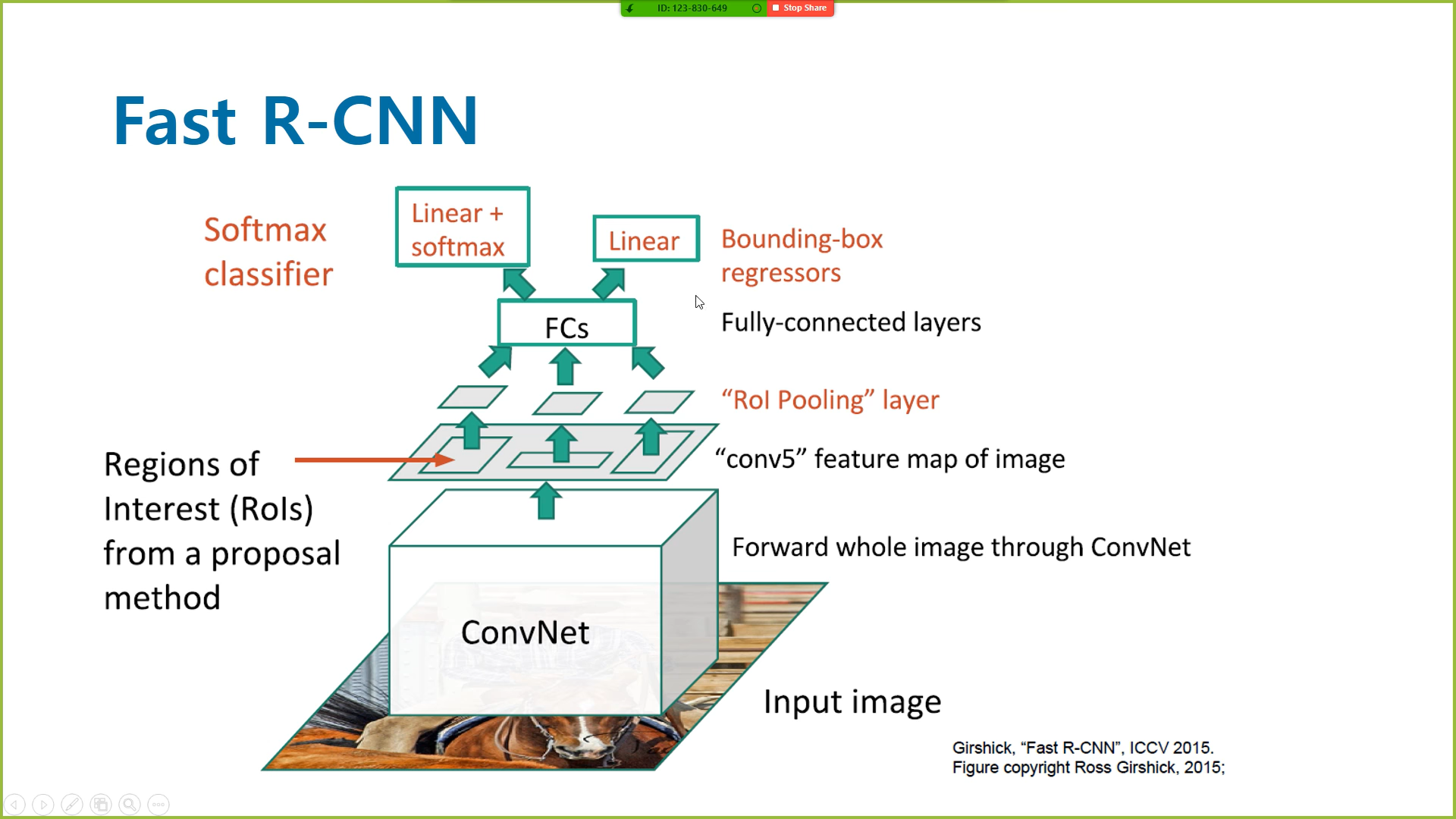
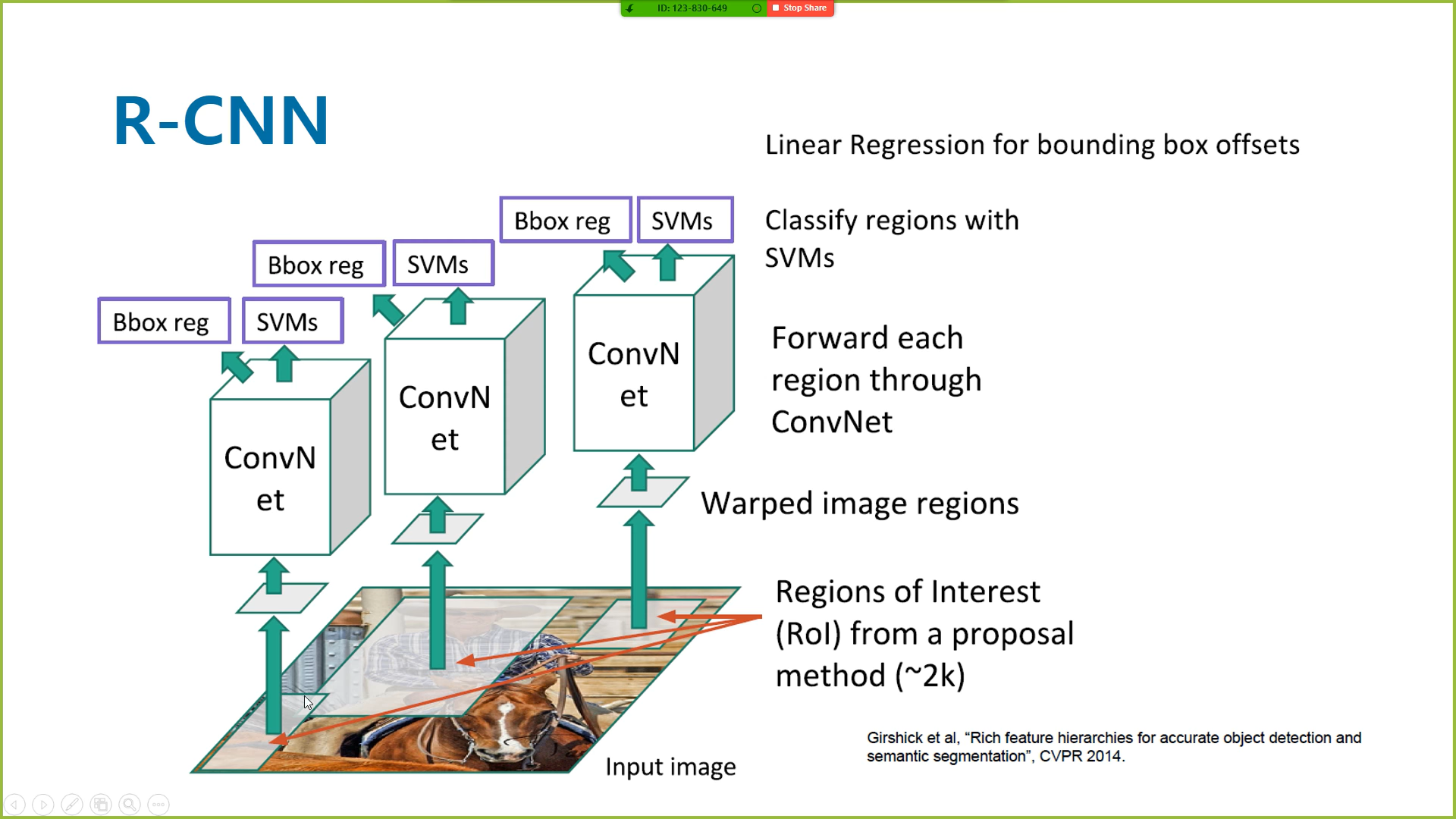
**Region Proposal을 하고 CNN에 입력으로 주느냐(=R-CNN),  
CNN을 먼저 하고 Region Proposal을 하느냐(=Fast R-CNN)** 이 차이

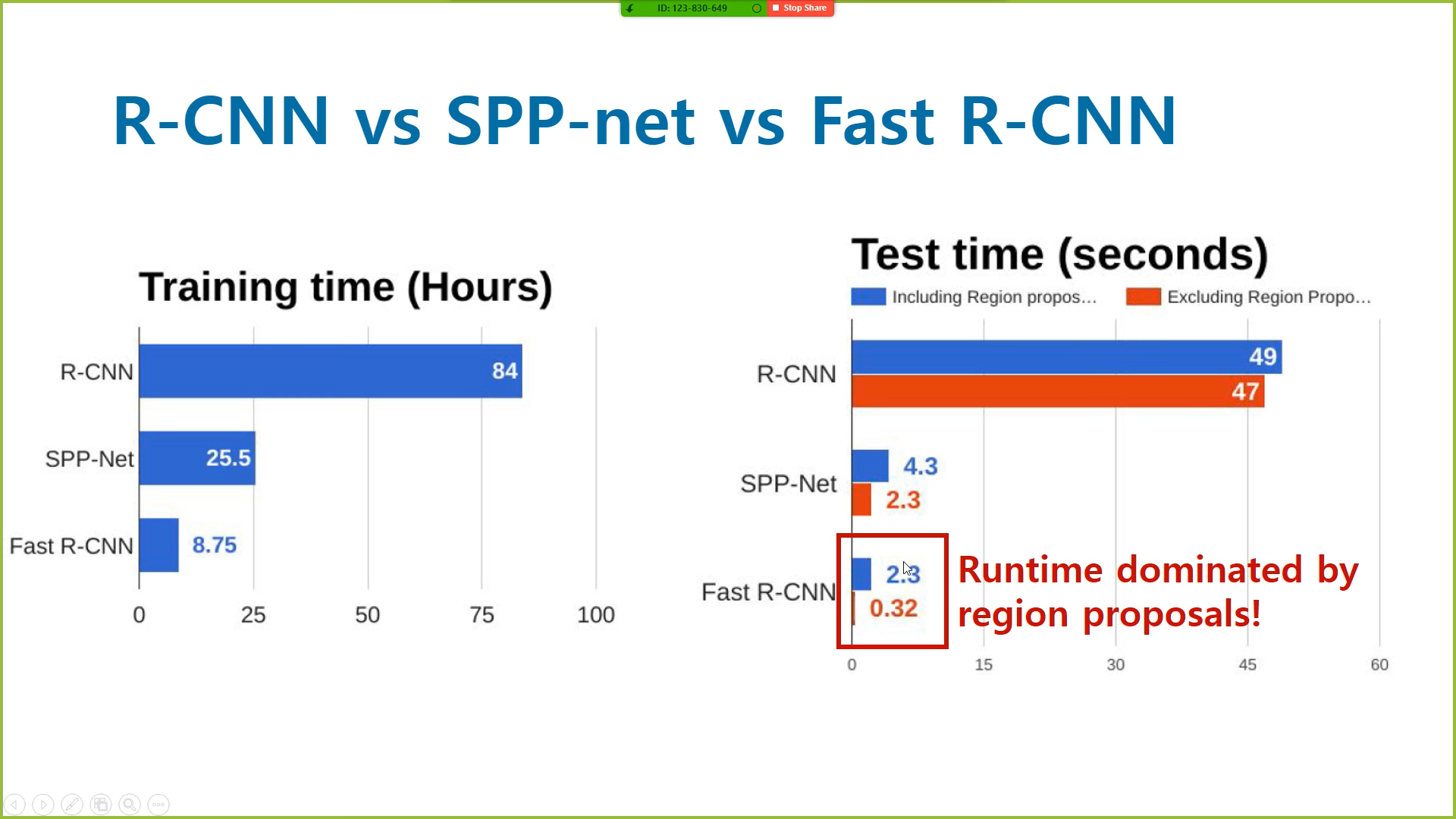


**ROI Pooling**

* 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

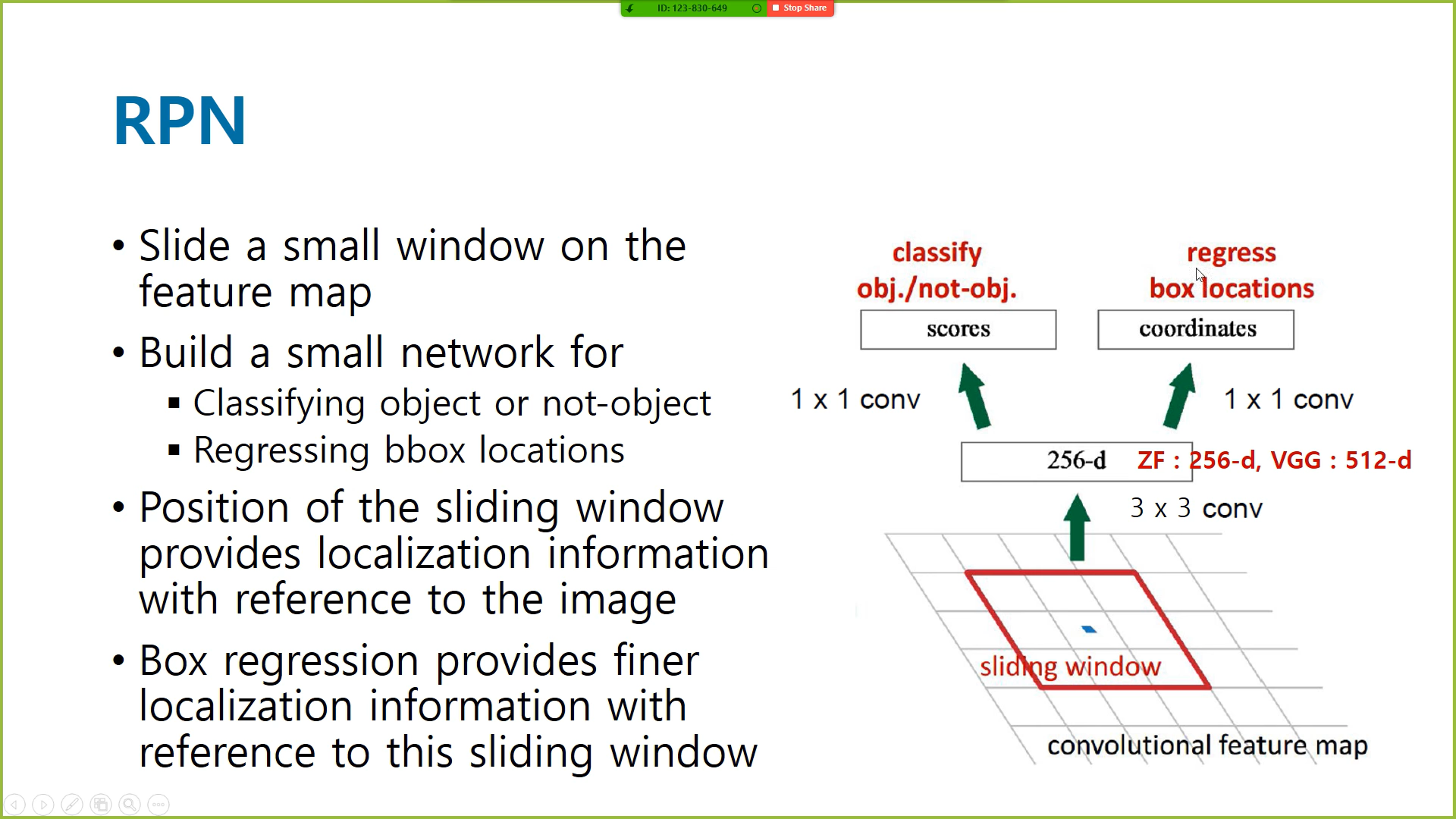
Fast R-CNN에서는 Region Proposal이 비 신경망 구조였다면,

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,

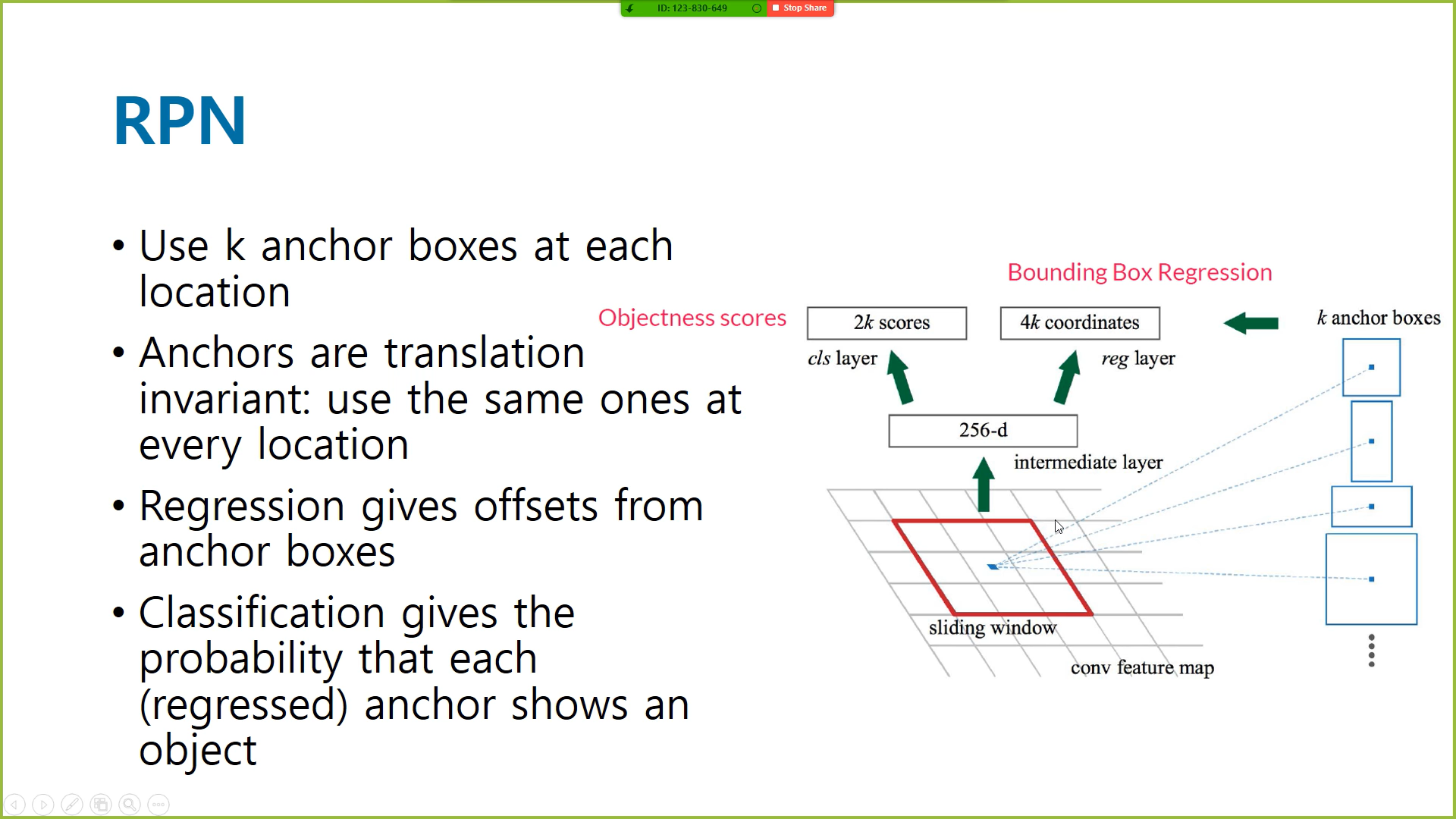
Faster R-CNN은 Fast R-CNN에 RPN만 앞에 붙인 것.



**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)

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

**Abstract**

SPPnet, Fast R-CNN과 같이 발달된 Object Detection Network는 running time 단축은 성공하였지만,

Region proposal 자체를 별도로 Selective Search와 같은 알고리즘을 사용해서 CPU기반으로 수행했기에 Region Proposal이 굉장히 느린 문제(각 이미지당 기본 2초)가 있었습니다.

따라서, 본 논문에서는 Selective Search Algorithm을 이용하지 않고 Region Proposal Network(RPN)을 제안해서, 이러한 RPN이 CPU가 아닌 GPU에서 돌아갈 수 있도록 함으로써 Region Proposal에 걸리는 시간을 획기적으로 단축했다.

이러한 RPN은 Fully Convolutional Network 형태로 되어 있으며, 그 물체가 고양이인지 강아지인지에 대한 Class정보까지 예측하지는 않고 어떠한 위치에 무엇이 존재하는가 그렇지 않은가 정도만 판단합니다.

또한, Region Proposal까지 GPU에서 돌릴 수 있게 되면서, 모델 전체를 GPU로 처리할 수 있게 되었고, 이로 인해 end-to-end 방식으로 학습이 가능해졌다.

RPN에서 만들어진 각각의 위치에 대한 정보와 그 물체의 크기에 대한 정보를 바탕으로,

그 위치에 있는 사물이 어떤 Class로 분류가 되는지는 Fast-RCNN구조를 가져와서 사용하였다.

즉, 전체적인 구조의 경우,

Faster R-CNN은 RPN와 Fast R-CNN으로 구성이 되는데, 이 두가지 Network는 동일한 Feature를 공유하도록,

즉, 제일 앞부분에 일종의 Feature Extractor로써 CNN Layer를 두어서, Feature를 추출하는 과정이 동일하게 했다.

이렇게 해서 성능도 좋고, 시간도 단축시켰다.

한편, 본 논문에서는

RPN을 통해서 사물의 위치가 감지가 되면,

그 위치에 있는 사물이 어떤 사물인지 맞추는 방식으로 진행이 되기 때문에

이러한 RPN을 이 논문에서는 Attention Mechanism과 유사하다고 언급했다.

이 결과, 초당 5장의 이미지 처리가 가능해졌다. 상당히 Real-Time에도 적용할 정도로 빠른 수준.

사용한 이미지는 PASCAL VOC 2007, PASCAL VOC 2012, MS COCO Dataset

**Introduction & Related Work**

본 논문의 RPN는 물체가 있을 법한 위치에 대해 Regression을 이용해서 Bounding Box를 더 잘 그리도록 했고,

그 위치에 실제로 물체가 존재하는가 아닌가 예측하는 작업까지 진행하였다.

또한, 속도 상의 이점을 얻기 위해 앵커박스의 크기와 스케일을 다양하게 지정했으며,

본 논문에서는 총 9개의 앵커박스를 이용하였다.

**1. CNN을 통해 이미지의 Feature를 뽑는다.**

**2. Region Proposal Network는 어떤 위치에 물체가 존재하는지 존재하지 않는지 예측하고**

**존재하면 어떤 바운딩 박스안에 위치하는지 예측하는 역할을 한다.**

**3. RPN에서 나온 결과를 Classsifier에 넣어서 Region Proposal영역의 물체가 어떤 것인지 맞춤.**

즉, RPN를 통해 특정 영역에 대해서만 Classification을 시키기 때문에 Attention 기법과 유사하다고 함.

**Faster R-CNN**

**Region Proposal Network**

한 장의 이미지를 입력으로 받아서, 바운딩 박스 형태로 아웃풋을 낸다.

각각의 바운딩 박스는 그 위치에 물체가 존재하는지 존재하지 않는지의 정보를 담는다.

이러한 RPN은 Fast RCNN과 Faetuer Map을 공유할 ㅅ ㅜ있는 형태로 구성되며

이러한 Feature Map은 ZF net, VGGNET과 같은 다양한 CNN Architecture를 사용할 수 있다.

ZF, VGG 둘다 Backbone으로 많이 이용

한 장의 이미지를 Convolutional Layer에 넣어서 Convolutional Feature Map을 뽑고

이러한 Feature Map에 대해서 다양한 크기와 비율을 가지는 Anchor Box에 대해 Sliding을 하면서 각 위치에 대해 Prediction을 진행하는 것.

기본적인 크기는 3x3의 anchor box이며,

이러한 박스를 왼쪽 위부터 슬라이딩하면서 256 차원으로 mapping을 수행.

이후에 이러한 Feature를 토대로 regression layer와 classification layer에 전달.