

Image Processing

2021. 10

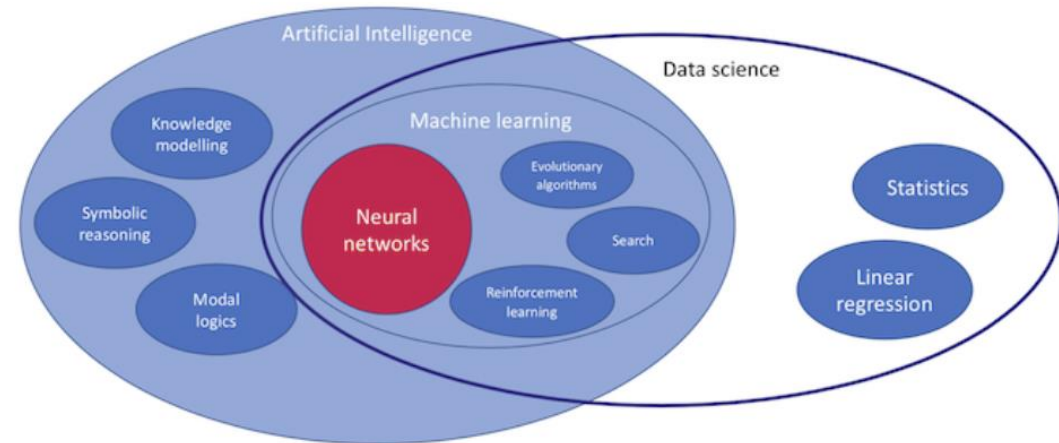
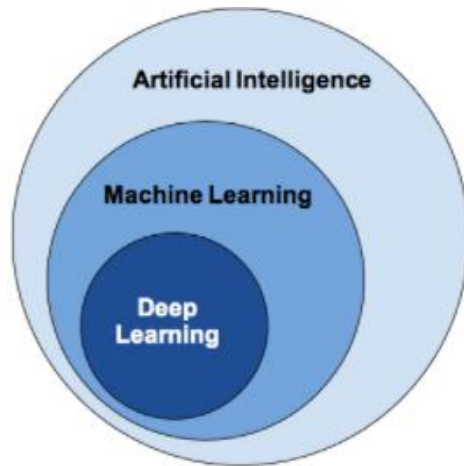
Yongjin Jeong, KwangWoon University

[참고] 본 자료에는 인터넷에서 다운받아 사용한 그림이나 수식들이 일부
있으니 다른 용도로 사용하거나 외부로 유출을 금해 주시기 바랍니다.

Machine Learning

- **What is ML**

- the study of computer algorithms that improve automatically **through experience and by the use of data**. It is seen as a part of artificial intelligence. [wikipedia]
- ML algorithms build a model based on sample data (training data) in order to make **predictions** or **decisions** without being explicitly programmed to do so.



[ref] <https://ictinstitute.nl/ai-machine-learning-and-neural-networks-explained/>

Datasets for Machine Learning

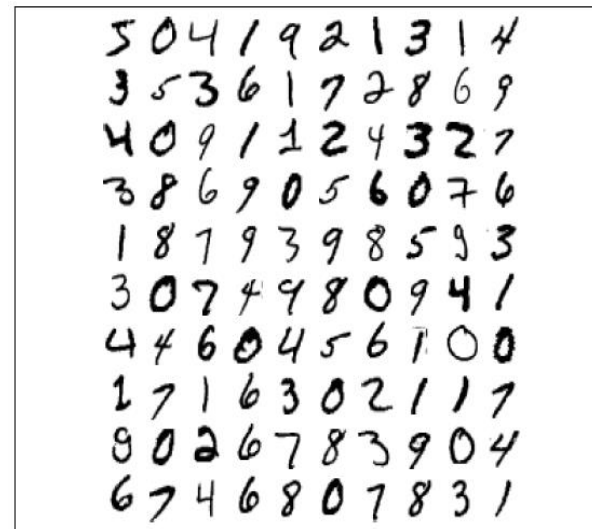
- **Iris** dataset
 - 4-column features, 150 samples
 - Target: 3 classes
- **MNIST**(Modified National Institute of Standards and Technology dataset)
 - 28*28 grayscale pixels, 60,000 train, 10,000 test set
 - Target: single digits 0 ~ 9
- **Fashion MNIST**
 - 28*28 pixels, 60,000 train set, 10,000 test set
 - Target: 10 categories
- Many more in Kaggle.com



Iris Versicolor

Iris Setosa

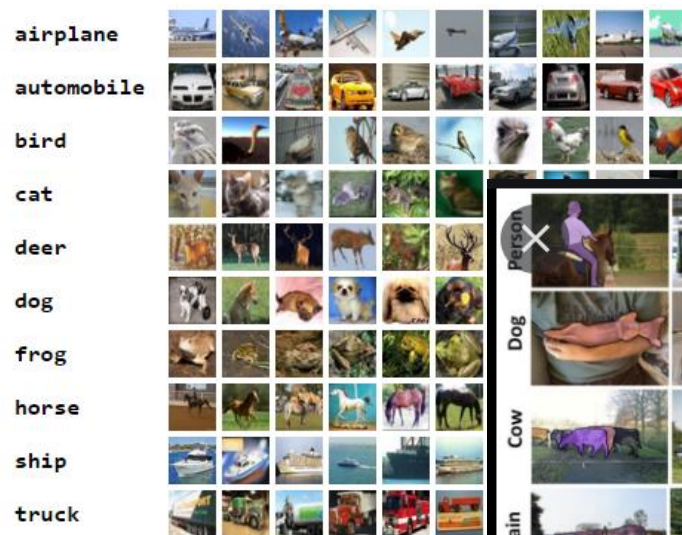
Iris Virginica



Fashion MNIST

Datasets for Machine Learning

- **Cifar-10 (Canadian institute for advanced research)**
 - 32*32 color images in 10 classes, with 6,000 per class
 - 50,000 training, and 10,000 test images
 - Bigger dataset (Cifar-100)
- **Microsoft COCO** (common objects in Context)
 - Large image recognition/classification, object detection, [segmentation](#), and captioning dataset
 - 330K images (200K+ annotated), more than 2M instances in 80 object categories
 - 5 captions per image, and 250,000 people with key points



Datasets for Machine Learning

- **ImageNet**

- largest image dataset for computer vision, used in [ILSVRC](#)(ImageNet Large Scale Visual Recognition Challenge) for image classification and object detection (150 GB)
- 469*387 resolution in average (usually cropped to 256*256 or 224*224 before usage)
- More than 14 million images with more than 21,000 groups(classes)
- More than 1 million images have bounding box annotations

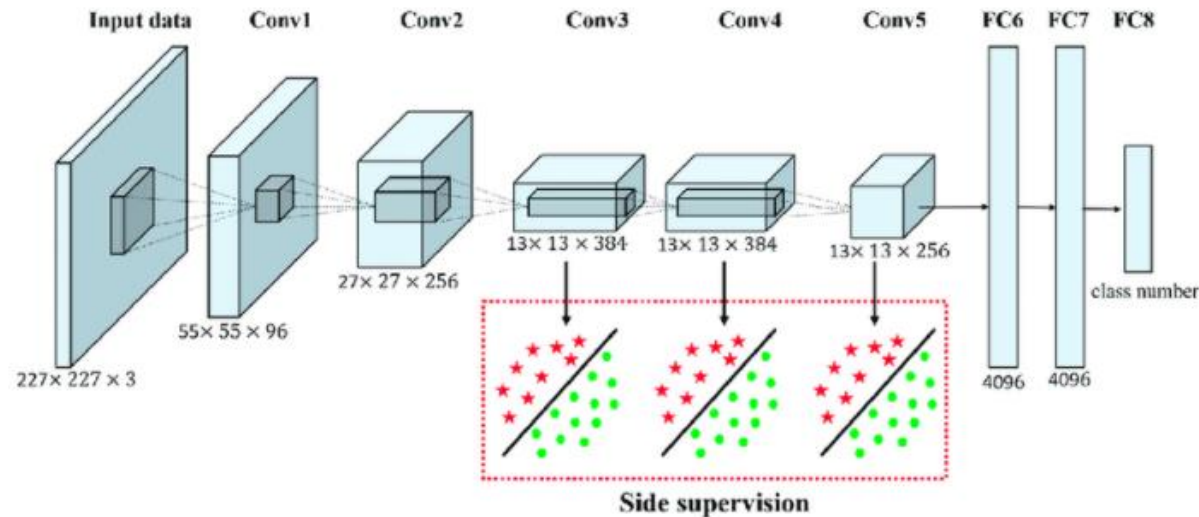
- **ILSVRC**

- To evaluate algorithms for object detection and image classification (and localization) at large scale
- To measure the progress of computer vision for large scale image indexing for retrieval and annotation
- Uses smaller portion of the ImageNet **(1,000 categories with 1.3 million train images**, 50,000 validation images, and 100,000 test images
- Available in Kaggle
- 2010 ~ 2017



ILSVRC Winners

- **AlexNet – 2012 Winner (top-5 error rate 15.3%)**
 - Use CNN (prior to 2012, the classification model error was 25%)
 - Regarded as a Pioneer of CNN and starting point of the Deep learning
 - 60 million parameters
 - Used ReLU activation function, data augmentation, dropout, and overlapped pooling layers



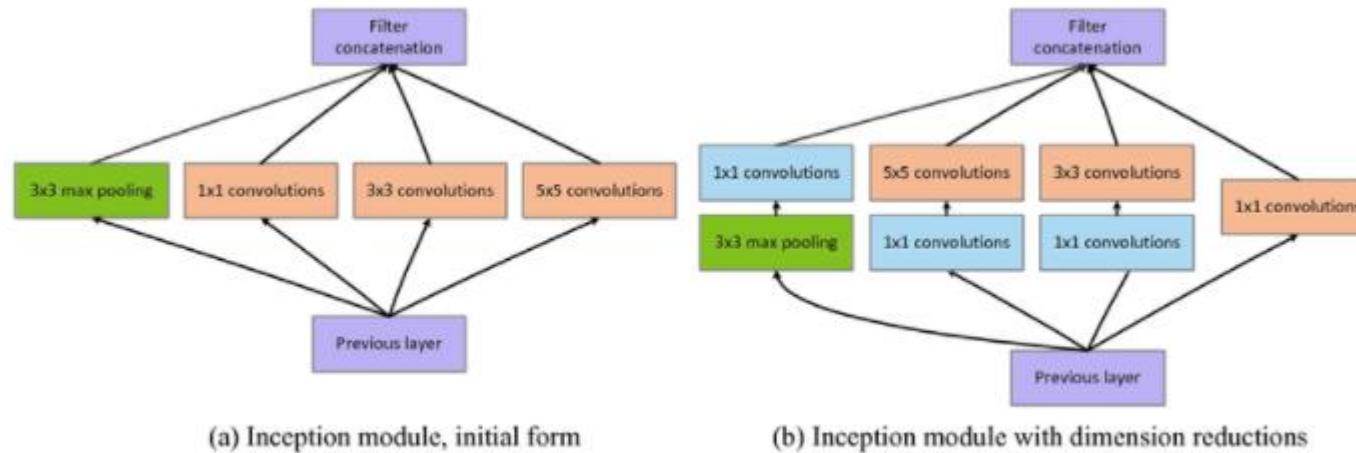
ImageNet Challenge (2012) – AlexNet ([Source](#))

(*) top-5 error: The model is considered to have classified a given image correctly if the target label is one of the model's top 5 predictions. (image classification)

ILSVRC Winners

- **Inception V1 (GoogLeNet) – 2014 Winner (top-5 error rate 6.67%)**

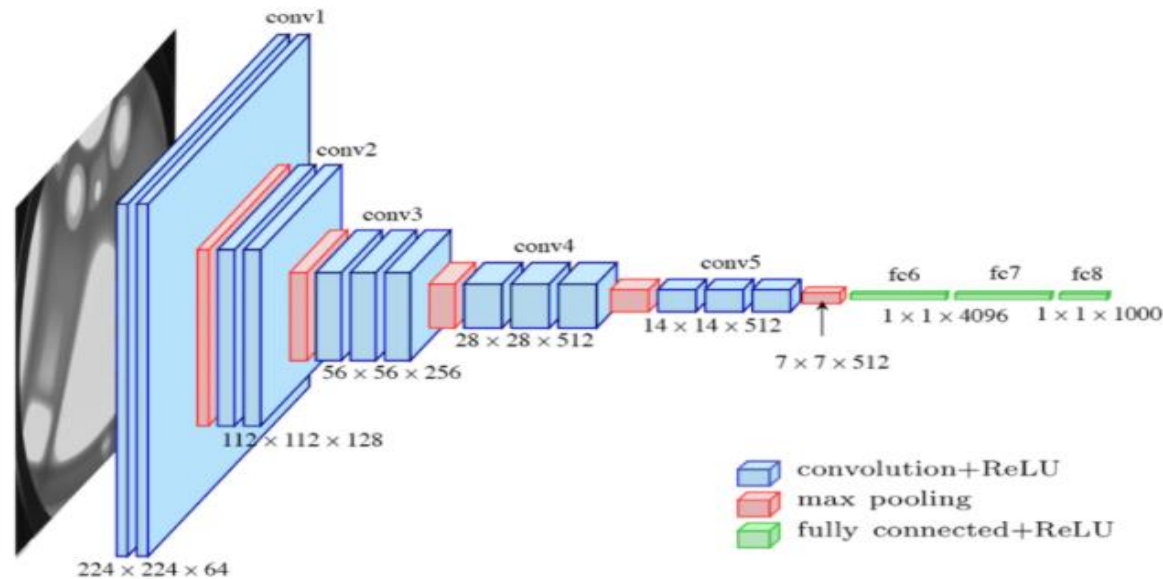
- The v1 stands for 1st version and later there were further versions v2, v3, etc. It is also popularly known as GoogLeNet.
- Deep with 22 layers.
- Used multiple types of filter size, instead of being restricted to a single filter size, in a single image block, which we then concatenate and pass onto the next layer.
- Used 1x1 convolutional with ReLU to reduce dimensions and number of operations.



ImageNet Challenge (2014)- Inception-V1 (GoogLeNet) [Source](#)

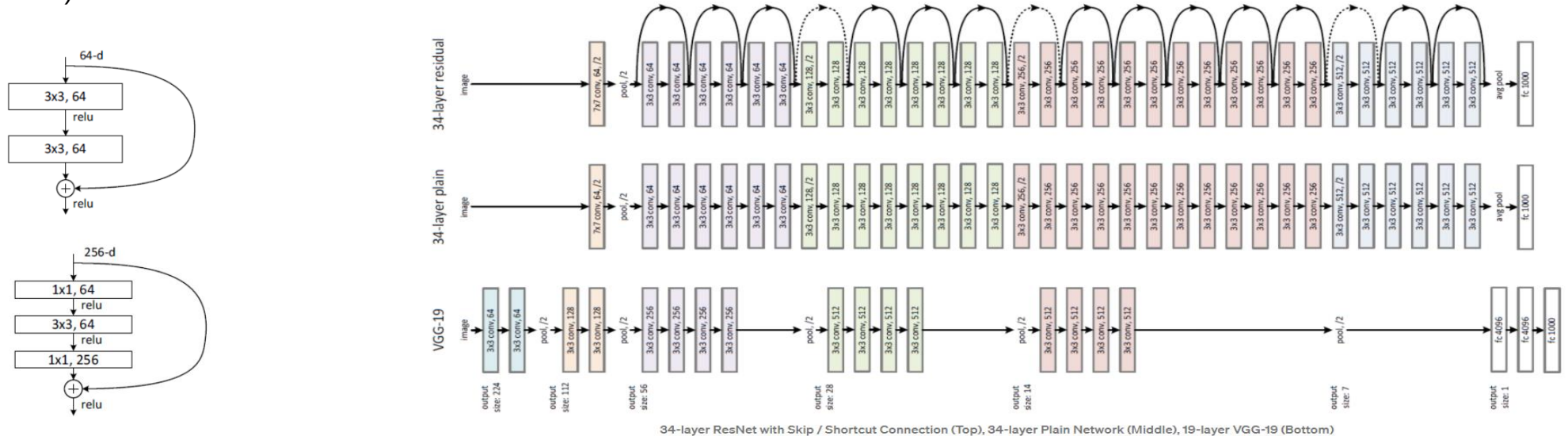
ILSVRC Winners

- **VGG-16 (University of Oxford) – 2014 Runners-Up (top-5 error rate 7.3%)**
 - Despite not winning the competition, VGG-16 architecture was appreciated and went on to become one of the most popular image classification models.
 - instead of using large-sized filters like AlexNet, it uses several 3×3 kernel-sized filters consecutively. The hidden layers of the network leverage ReLU activation functions.
 - VGG-16 is however very **slow to train** and the network weights, when saved on disk, occupy a **large** space.



ILSVRC Winners

- **ResNet – 2015 Winner (top-5 error rate 3.57%)**
 - ResNet ([Residual Network](#)) was created by the Microsoft Research team.
 - To solve the problem of vanishing/exploding gradients, a [skip/shortcut connection](#) is added to add the input x to the output after few weight layers as below:
 - 1×1 Conv can reduce the number of connections (parameters) while not degrading the performance of the network so much. (as in Inception-V1)
 - ResNet-18/34/50/101/152 has 1.8/3.6/3.8/7.6/11.3 GFLOPs (lower than VGG-16/19 with 15.3/19.6 GFLOPS)



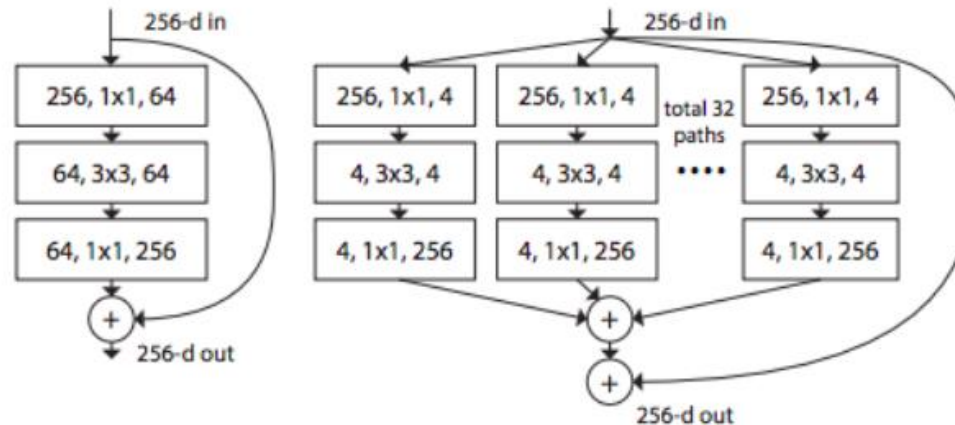
34-layer ResNet with Skip / Shortcut Connection (Top), 34-layer Plain Network (Middle), 19-layer VGG-19 (Bottom)

The Basic block (top) and the Proposed Bottleneck design (bottom)

(*) VGG-19 (bottom): state-of-the-art approach in 2014
middle: deeper network of VGG-19 (i.e. more Conv layers)

ILSVRC Winners

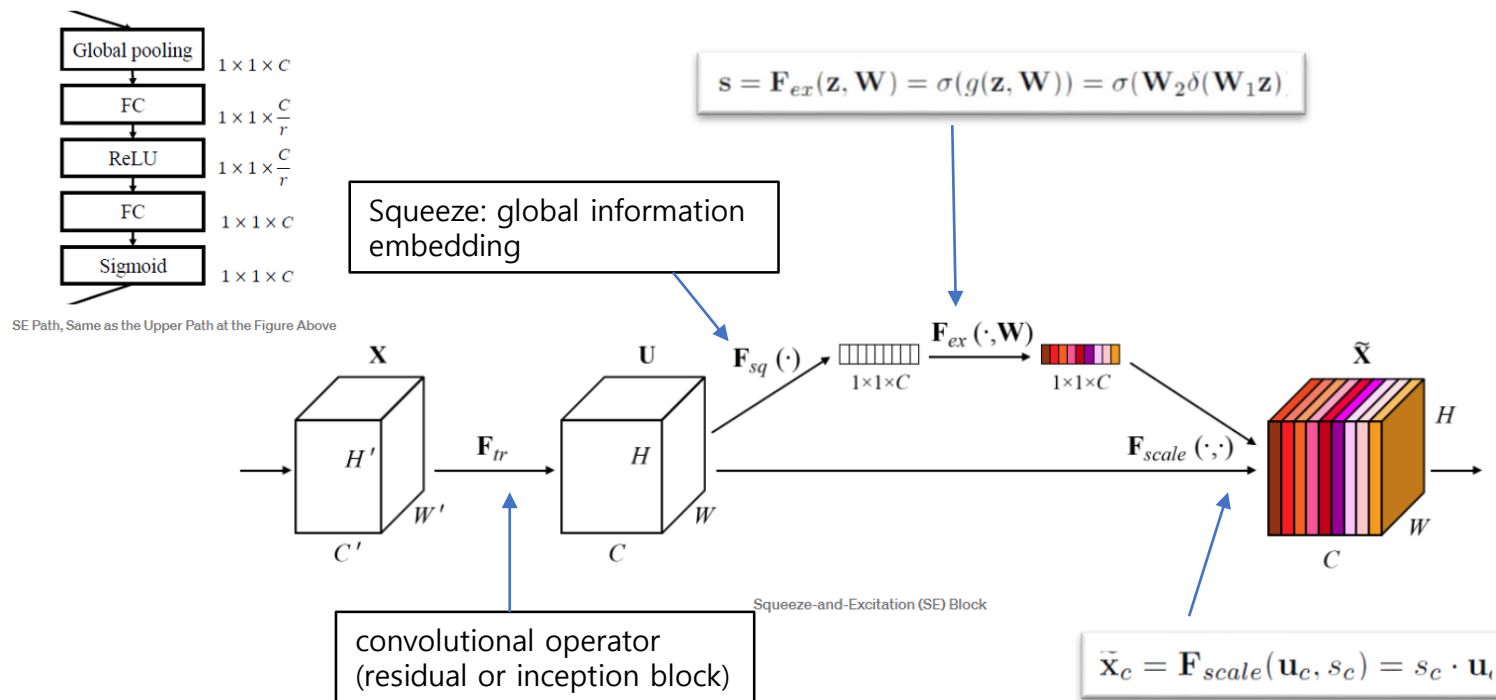
- **ResNeXt – 2016 Runners-Up (top-5 error rate 4.1%)**
 - Developed in the collaboration of the Researchers from UC San Diego and Facebook [AI](#) Research.
 - Inspired by (ResNet + VGG + Inception).
 - Still it became a popular model.
 - stacks the blocks and then use the ResNet approach of residual blocks. Here the hyper-parameters such as width and filters were also shared.



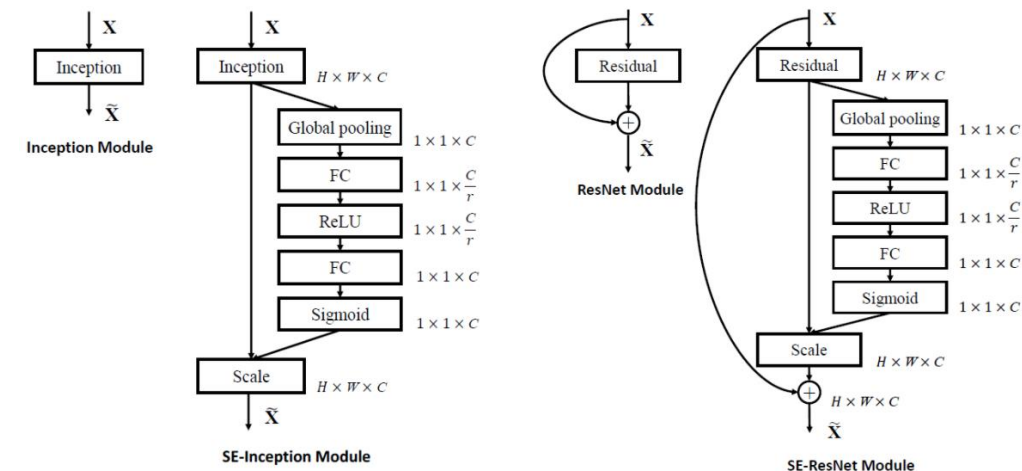
ImageNet Challenge (2016)- ResNeXt ([Source](#))

ILSVRC Winners

- **SENet(Squeeze-and-Excitation Network) – 2017 Winner (top-5 error rate 2.251%)**
 - Developed in University of Oxford.
 - With "Squeeze-and-Excitation" (SE) block that **adaptively recalibrates channel-wise feature responses by explicitly modelling interdependencies between channels**, SENet is constructed.
 - SE block can be added to both Inception and [ResNet](#) block easily as **SE-Inception** and **SE-ResNet**. (achieved the best 2.25%).



2. SE-Inception & SE-ResNet



(*) Human Beings – Top-5 Error Rate – 5.1%

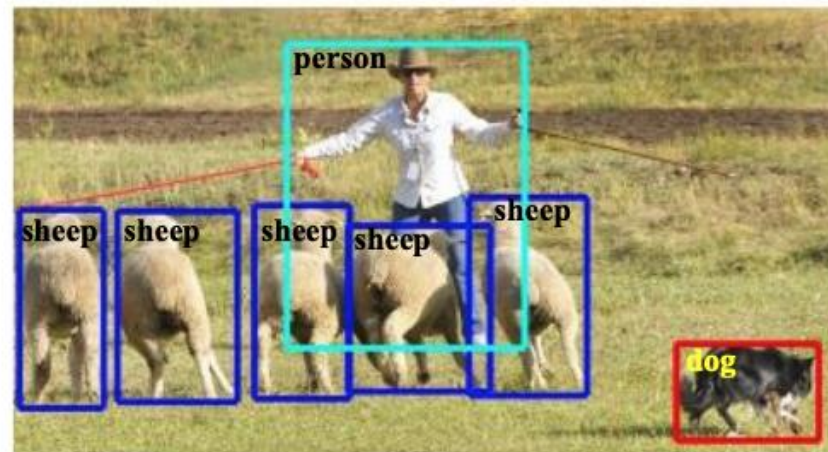
Computer Vision Tasks

- **Image Classification (이미지 분류)**
 - 이미지에 어떤 객체들이 들어 있는지만 알아내는 것
 - 여기서는 이미지에 사람, 양, 개가 있다는 것을 찾아낸다
- **Object Detection (객체 검출)**
 - 찾아낸 객체의 위치까지 알아내는 것
 - 일반적으로는 객체가 있는 위치를 박스 형태로 찾아낸다
- **Segmentation (세그멘테이션 or 분할)**
 - 객체의 위치를 박스형태가 아니라 비트 단위로 찾아낸다
 - 즉, 각 비트가 어떤 객체에 속하는지를 분류해내는 것
- **Object Instance Segmentation (객체 인스턴스 세그멘테이션)**
 - 이미지를 비트 단위로 어느 객체에 속하는지를 찾아낼 뿐 아니라 각 객체를 서로 다른 객체로 구분하는 기능까지 수행

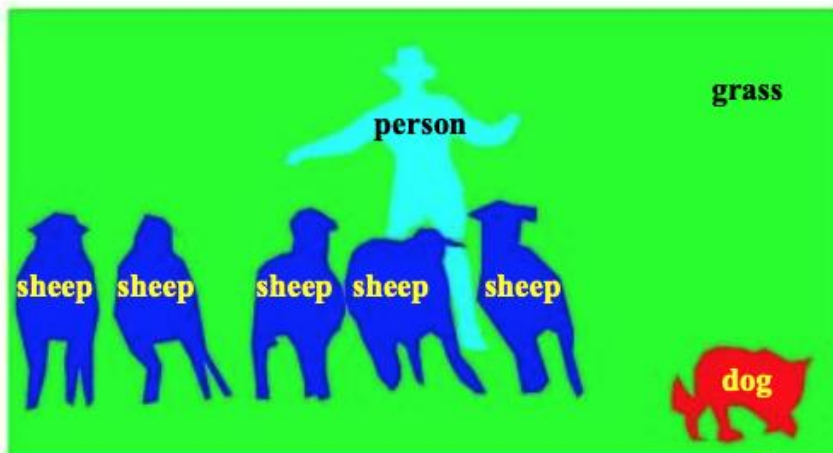
Computer Vision Tasks



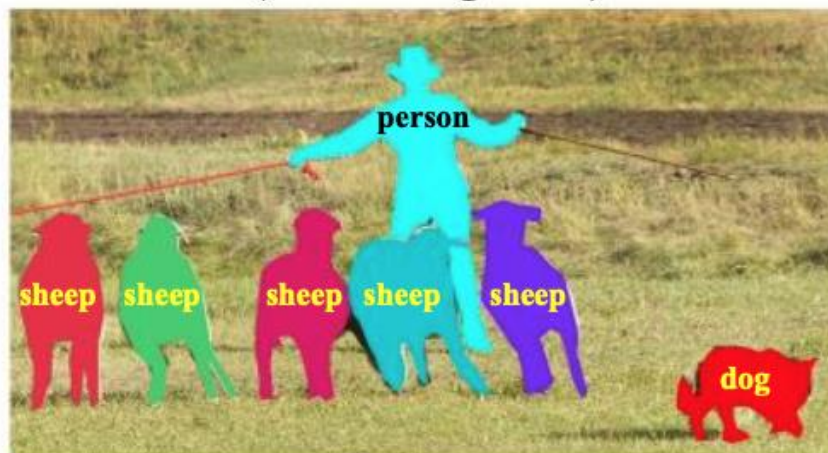
(a) Object Classification



(b) Generic Object Detection
(Bounding Box)



(c) Semantic Segmentation



(d) Object Instance Segmentation

Computer Vision Tasks (another view)

Classification



This image is CC0 public domain

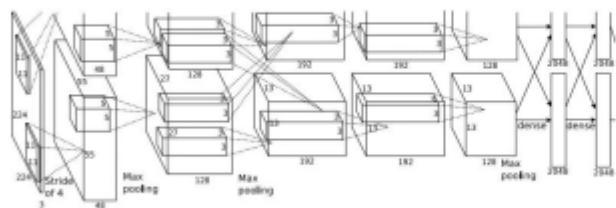


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

Vector:
4096

Fully-Connected:
4096 to 1000

Class Scores

Cat: 0.9
Dog: 0.05
Car: 0.01
...

Semantic Segmentation



GRASS, CAT,
TREE, SKY

No objects, just pixels

Classification + Localization



CAT

Single Object

Object Detection



DOG, DOG, CAT

Multiple Object

Instance Segmentation

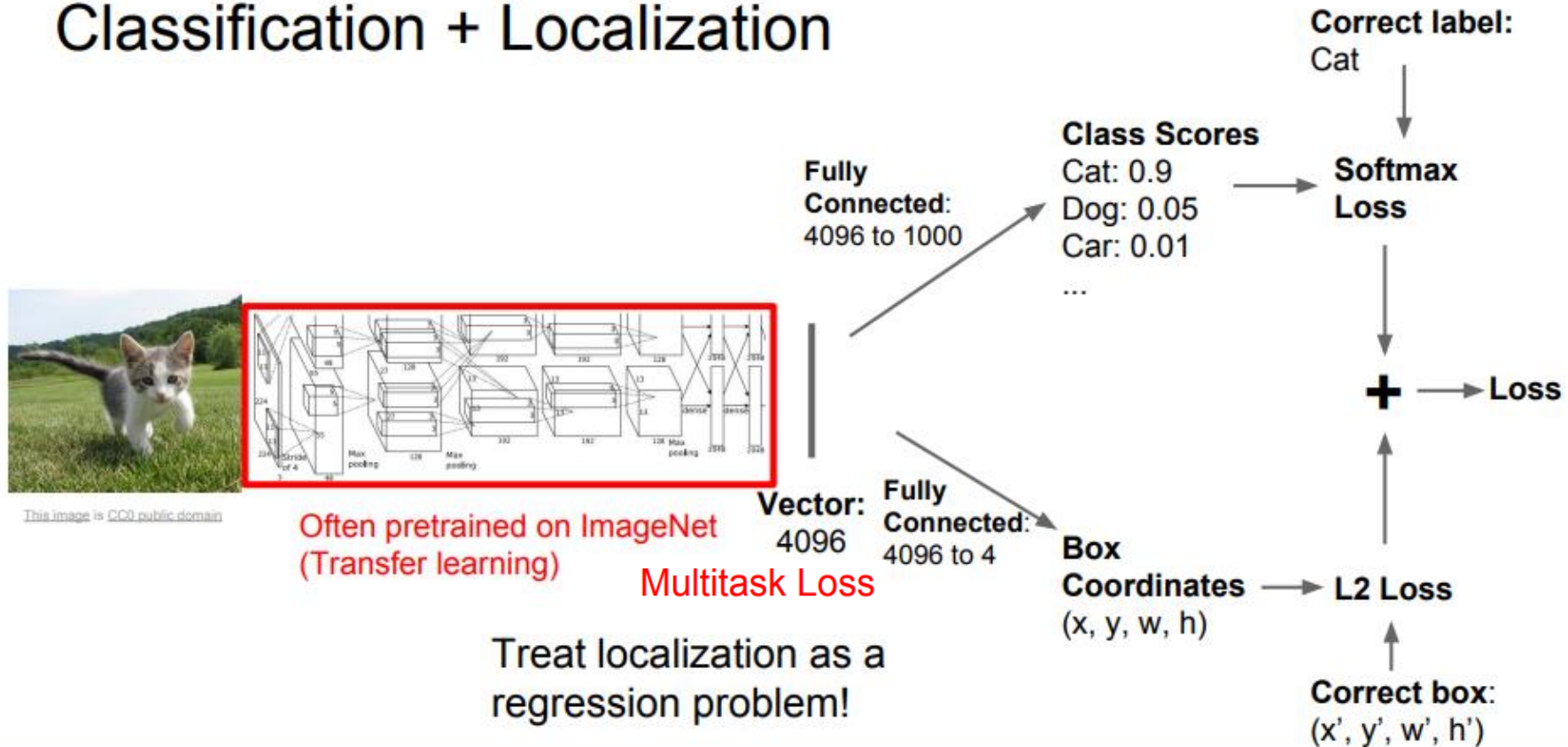


DOG, DOG, CAT

This image is CC0 public domain

Classification + Localization

Classification + Localization

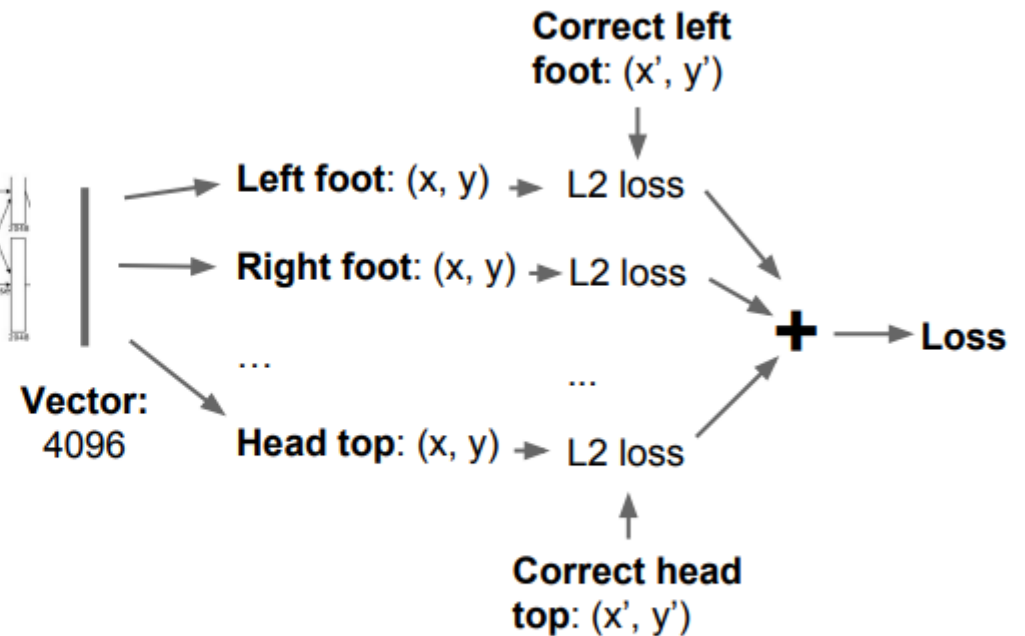
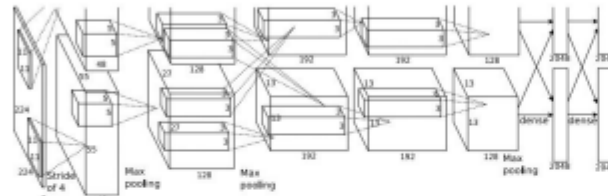


Aside: Human Pose Estimation



Represent pose as a set of 14 joint positions:

- Left / right foot
- Left / right knee
- Left / right hip
- Left / right shoulder
- Left / right elbow
- Left / right hand
- Neck
- Head top

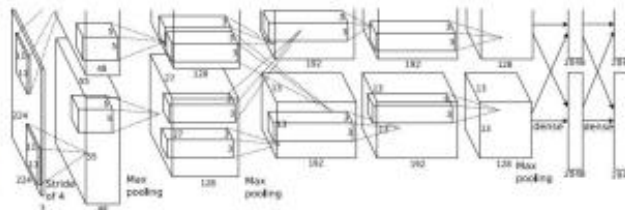


Toshev and Szegedy, "DeepPose: Human Pose Estimation via Deep Neural Networks", CVPR 2014

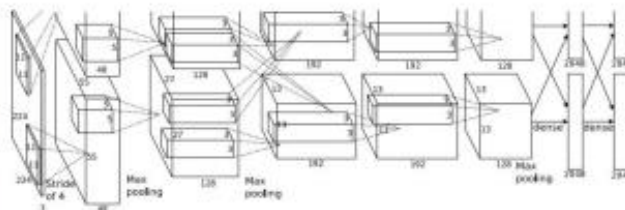
Object Detection (객체검출)

Object Detection as Regression?

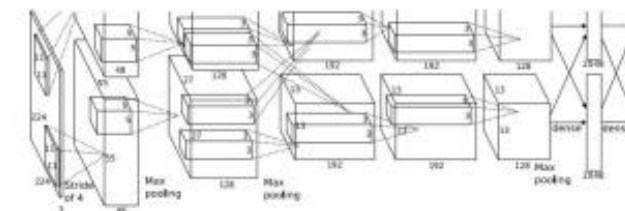
Each image needs a different number of outputs!



CAT: (x, y, w, h) 4 numbers



DOG: (x, y, w, h)
DOG: (x, y, w, h) 16 numbers
CAT: (x, y, w, h)



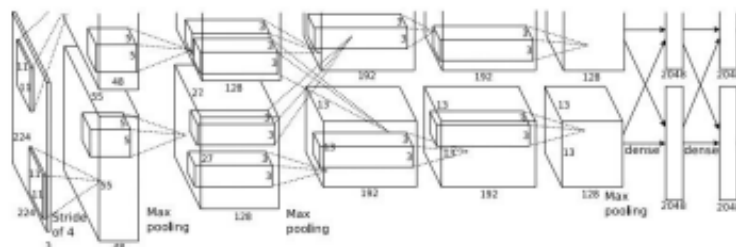
DUCK: (x, y, w, h) Many numbers!
DUCK: (x, y, w, h) numbers!

....

Object Detection

Object Detection as Classification: Sliding Window

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

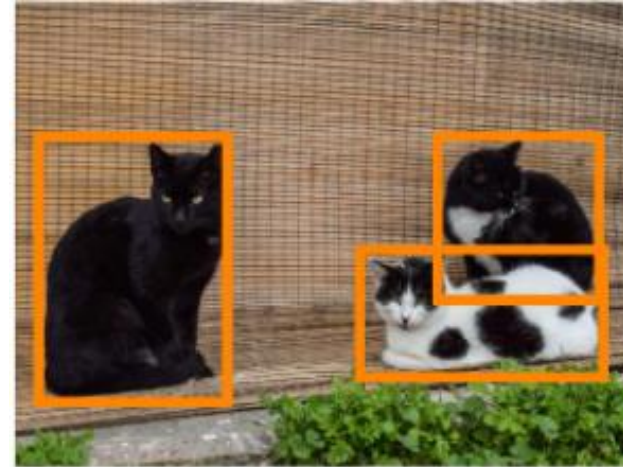


Dog? NO
Cat? YES
Background? NO

Problem: Need to apply CNN to huge number of locations and scales, very computationally expensive!

Object Detection - Goal

- **Two tasks:**
 - Object Location
 - Classification
- **Commonly solved by:**
 - Hypothesize bounding boxes
 - Re-sample selected boxes
 - Apply classifier
- **Question:** How can we combine or replace these steps to achieve higher speed or accuracy?
- **Bounding box**
 - Smallest box by some measure that fully contains the object in question
 - Typically defined by: top-left corner (\mathbf{x}, \mathbf{y}), width \mathbf{w} , height \mathbf{h} + classifier confidence



Loss Function

- 학습을 시키려면
 - 입력 이미지에 대해서 객체의 종류와 위치를 찾아야 한다
- 레이블 - 2가지 제공해야
 - 이미지 클래스
 - 이미지가 위치한 박스의 좌표
- 출력이 두 가지 이상인 멀티 출력 모델을 만들어야 하며 손실함수는 분류에 관한 손실과 회귀에 관한 손실의 합으로 구성

$$Loss = \alpha * Softmax_Loss + (1 - \alpha) * L2_Loss$$

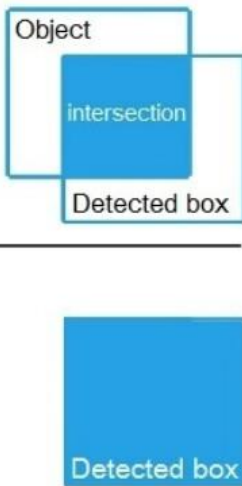
- 두 가지 성분의 손실의 상대적인 비중은 α 값으로 조정

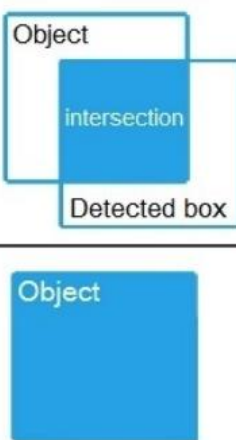
Performance

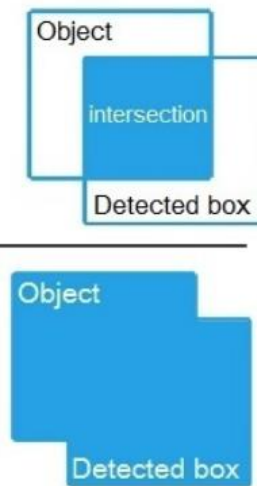
- 객체 검출에서는 단순히 어떤 객체가 있는지를 찾는 분류 문제가 아니므로 정확도 등 만으로 성능을 평가할 수 없고, **예측한 객체의 위치가 실제 객체의 위치와 얼마나 일치하는지**를 평가해야 한다
- 성능 평가척도
 - IoU (Intersection over Union)
 - mAP (mean Average Precision)
 - AR (Average Recall)

Object Detection Performance

- 객체검출 성능 평가: 객체의 위치를 얼마나 실제 위치와 겹치게 잘 찾았는지 평가
 - 정밀도 (precision) : 예측한 면적분에 겹치는 면적
 - 리콜 (recall) : 실제 이미지 면적분에 겹치는 면적
 - IoU : 두 가지 면적의 전체집합 부분에 비하여 겹치는 부분의 면적의 비율


$$\text{Precision} = \frac{\text{Area of Intersection}}{\text{Area of Detected box}}$$


$$\text{Recall} = \frac{\text{Area of Intersection}}{\text{Area of Object}}$$


$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

Performance - IoU

- 실제 이미지 예



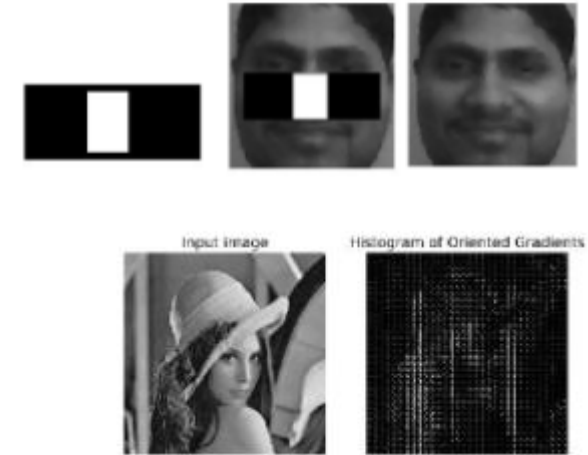
Performance – mAP

- 객체 검출에서 검출할 객체가 여러 개가 있을 때 이들의 평균 예측 성능을 표현하는데 mAP가 많이 사용
- mAP
 - 검출작업의 평균 정확도
 - 여러 객체에 대해 각각 AP를 구하고 이의 평균치를 구한 것
- 일반적으로 mAP가 0.5 이상이면 true positive라고 판단

Object Detection

- **Early approaches:**

- Viola & Jones '01:
 - Haar Features + Adaboost + cascading classifier for fast rejection
 - First competitive real-time object detection
 - Mainly used for face detection
- Dalal & Triggs '05
 - Histogram of Oriented Gradients (HOG)
 - Support Vector Machines

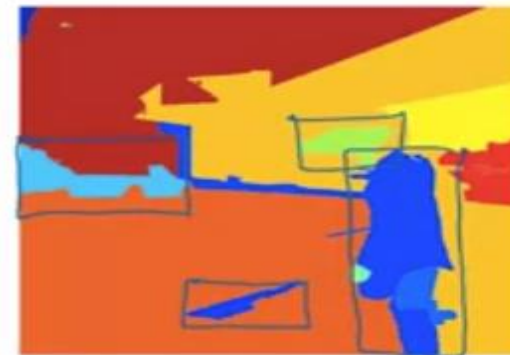


- **Modern approaches:**

- Sliding Window: classify each possible window by CNN
- [Regional proposal](#) CNN (R-CNN): find interesting image regions first, then classify by CNN
- Single-Shot Multi-box Detectors: joint detection and classification
 - You Only Look Once (YOLO)
 - Single-Shot Multi-box Detector (SSD)

Object Detection

- **슬라이딩 윈도우 방식 (초기)**
 - 객체 검출을 위해서 전체 이미지를 작은 크기로 나누고 각 부분을 대상으로 객체를 찾는 작업을 수행
 - 이 방식은 시간이 오래 걸리고 더욱이 윈도우 크기와 객체의 크기가 다를 때 찾지 못한다
- **Region Proposal (R-CNN 에 적용)**
 - Sliding window 를 전체에 대해 하지 않고 물체가 있을 것 같은 영역에 대해서만 함. (스케일링 등이 필요 없어 빠름): segmentation 이용



Segmentation algorithm
~2,000

Object Detection

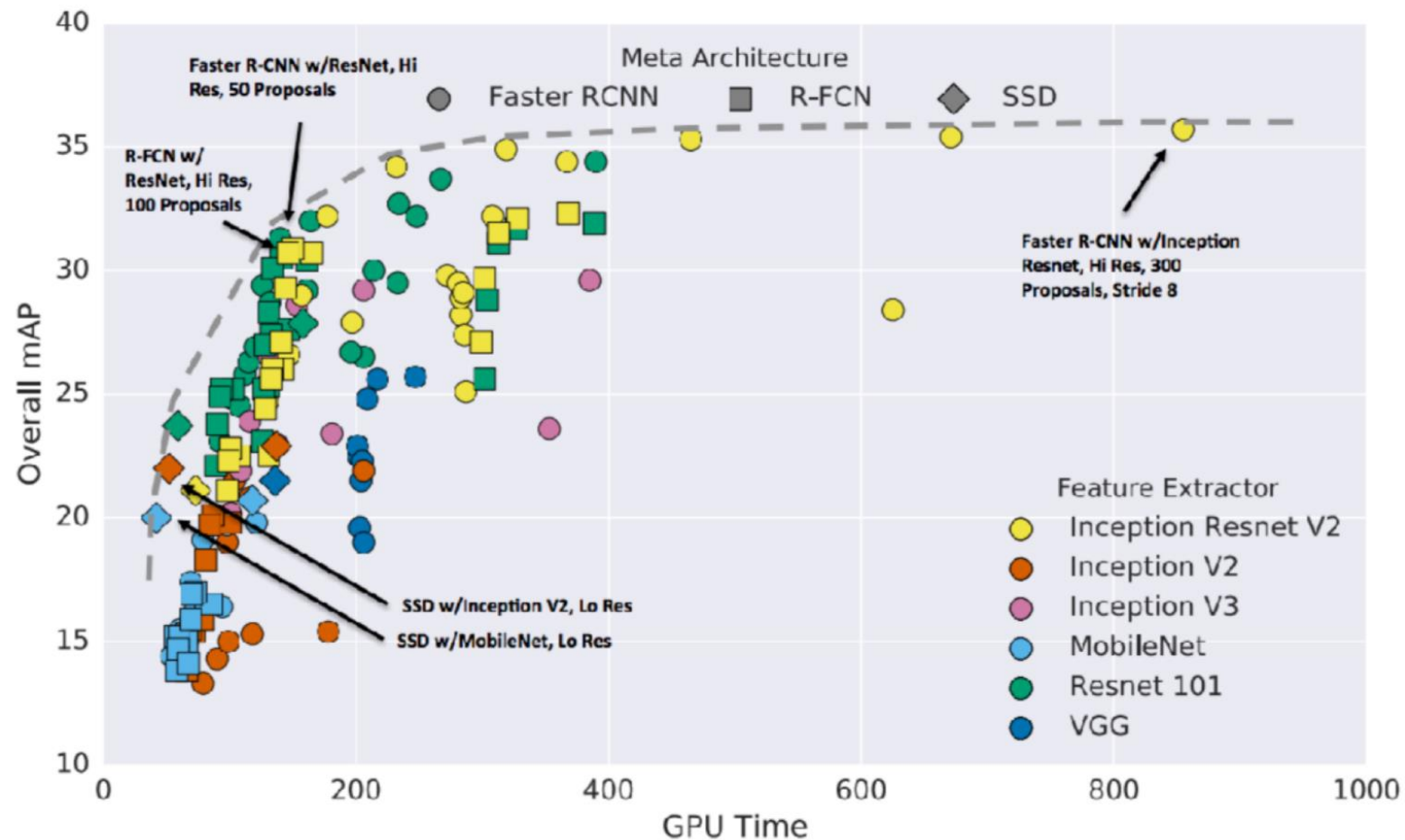
- 한편 객체의 위치를 찾는 문제
 - 분류가 아니라 회귀 문제
 - 즉, 객체의 경계 박스의 좌표 값은 회귀 문제로 예측
 - 이 방법은 사람의 자세를 예측하는 포즈 검출, 기준점 검출에서도 사용



(자세 검출의 예)

Object Detection - Algorithms

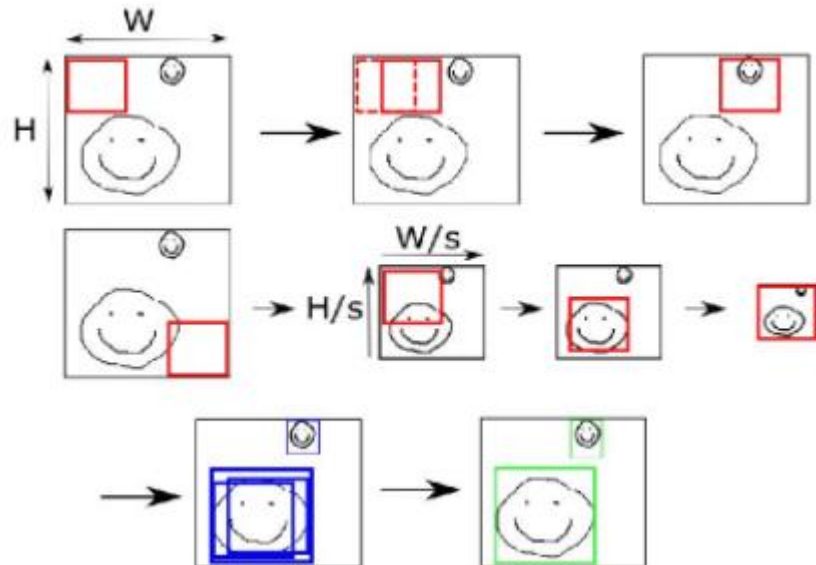
- Resnet을 사용한 Faster R-CNN등이 우수한 성능을 나타냈으나 최근 Mask R-CNN과 Yolo등 성능이 더 개선된 방식이 소개됨



Object Detection

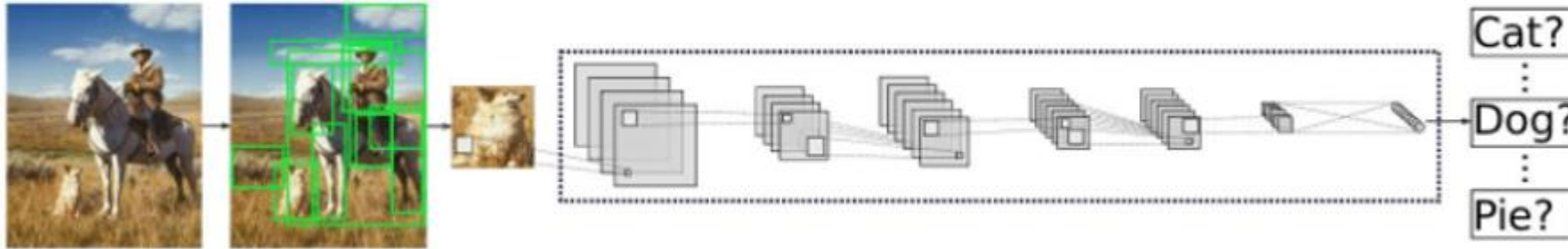
- **Single Window Approach**

- Simply take your pre-trained CNN and just move it across the image
- When you find an area of high confidence, you find it.
- Repeat this process on multiple resolutions.
- Multiple patches need be grouped to a single patch.
- Disadvantage: Large number of patches -> computationally inefficient



Object Detection

- Regional CNN (R-CNN)

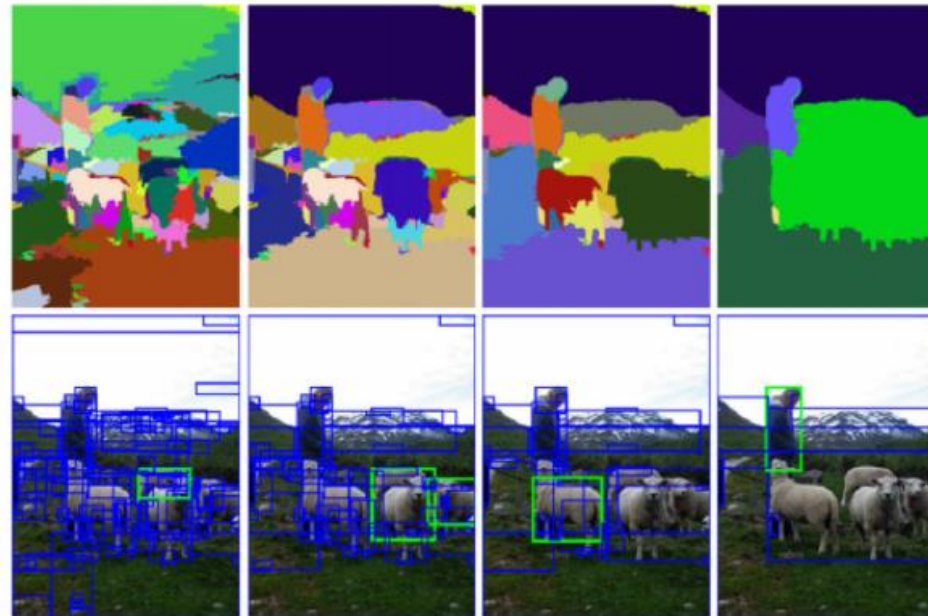


- Can we **improve efficiency** by only **considering interesting regions** (ROIs)?
- Multi-step approach in R-CNN [20]:
 - Generate **region proposals**: Selective Search
 - Classify content of region proposals + refine bounding box

Object Detection

- **Region Proposal: Selective Search**

- Candidate objects by grouping pixels of similar texture or color
- Apply for different sized windows -> produce few thousand ($\sim 2k$) object proposals per image (\ll number possible windows)
- Essentially a form of coarse segmentation



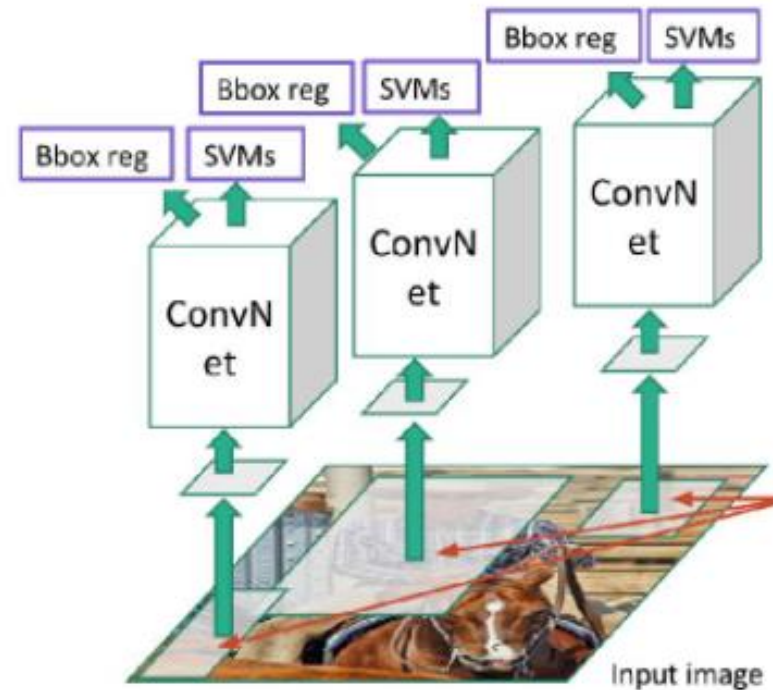
→
region size increased

R-CNN

- Regional CNN (R-CNN)

- For each region proposal window
 - Warp to standard window size
 - Pre-trained CNN for feature extraction
 - Linear SVM for object classification
 - Linear regression for bounding box refinement

- + Improved retrieval rate at that time (2013) by more than 30%
- Much faster... but still slow
- Not end-to-end



Linear Regression for bounding box offsets

Classify regions with SVMs

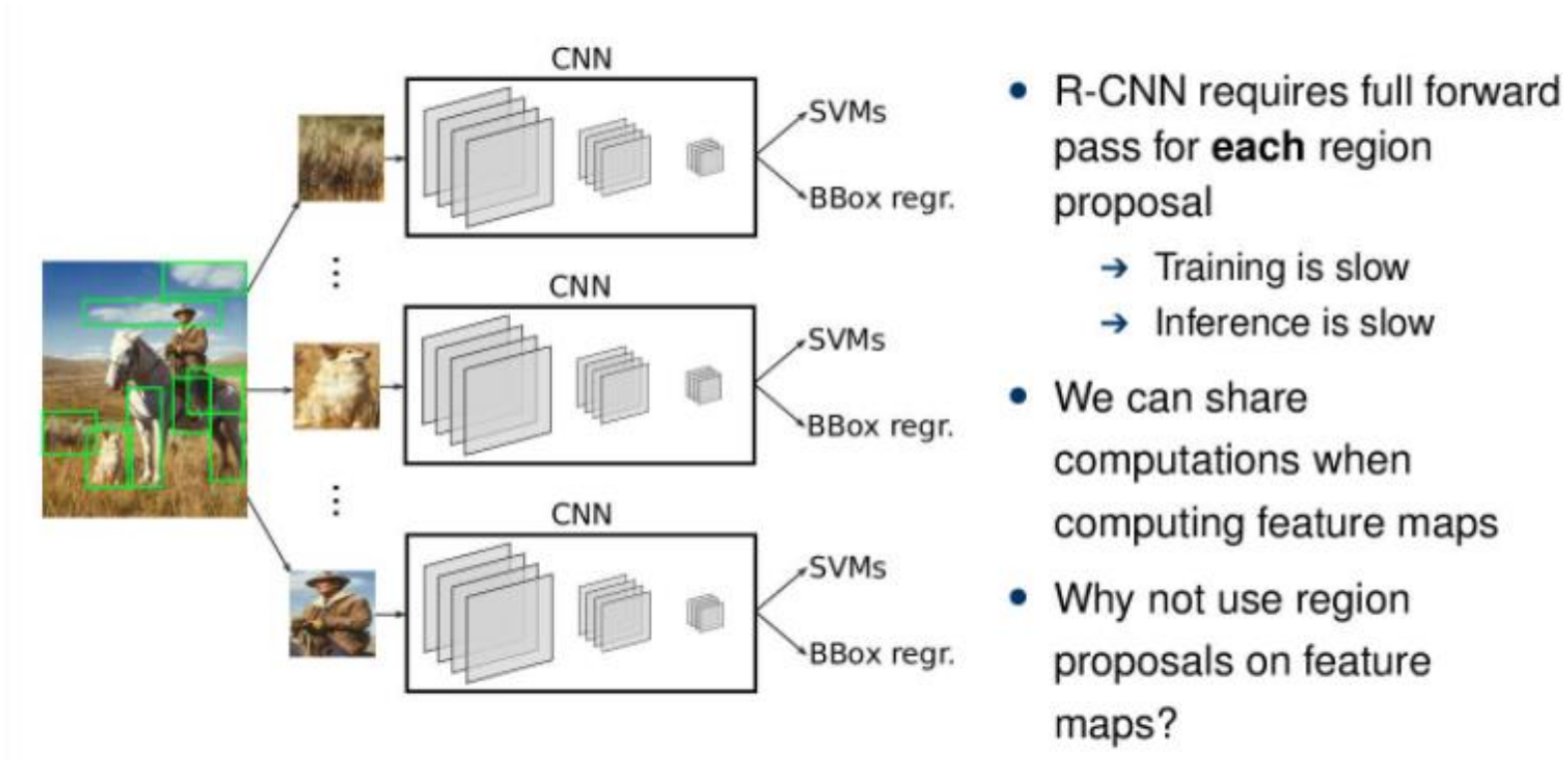
Forward each region through ConvNet

Warped image regions

Regions of Interest (RoI) from a proposal method (~2k)

Object Detection

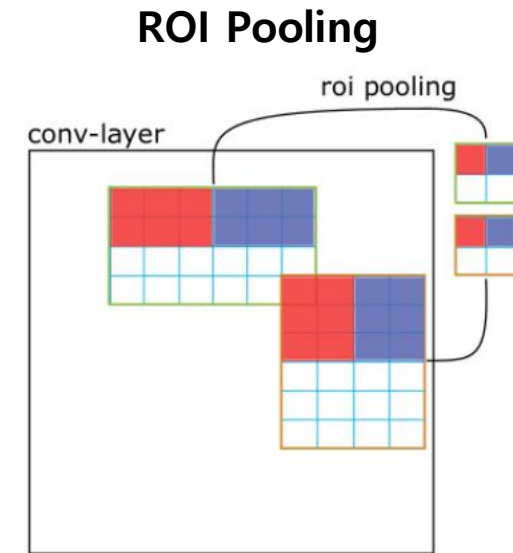
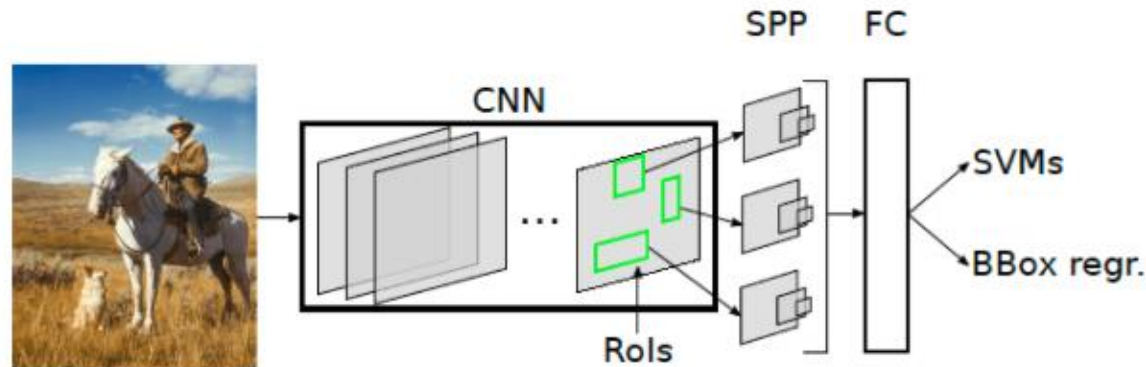
- Towards Fast R-CNN



- We can share the computations when computing the feature maps because we're doing the same or similar computations all along.
- Then, the key idea in order to improve the inference speed is that we **use the region proposals on the feature maps**.

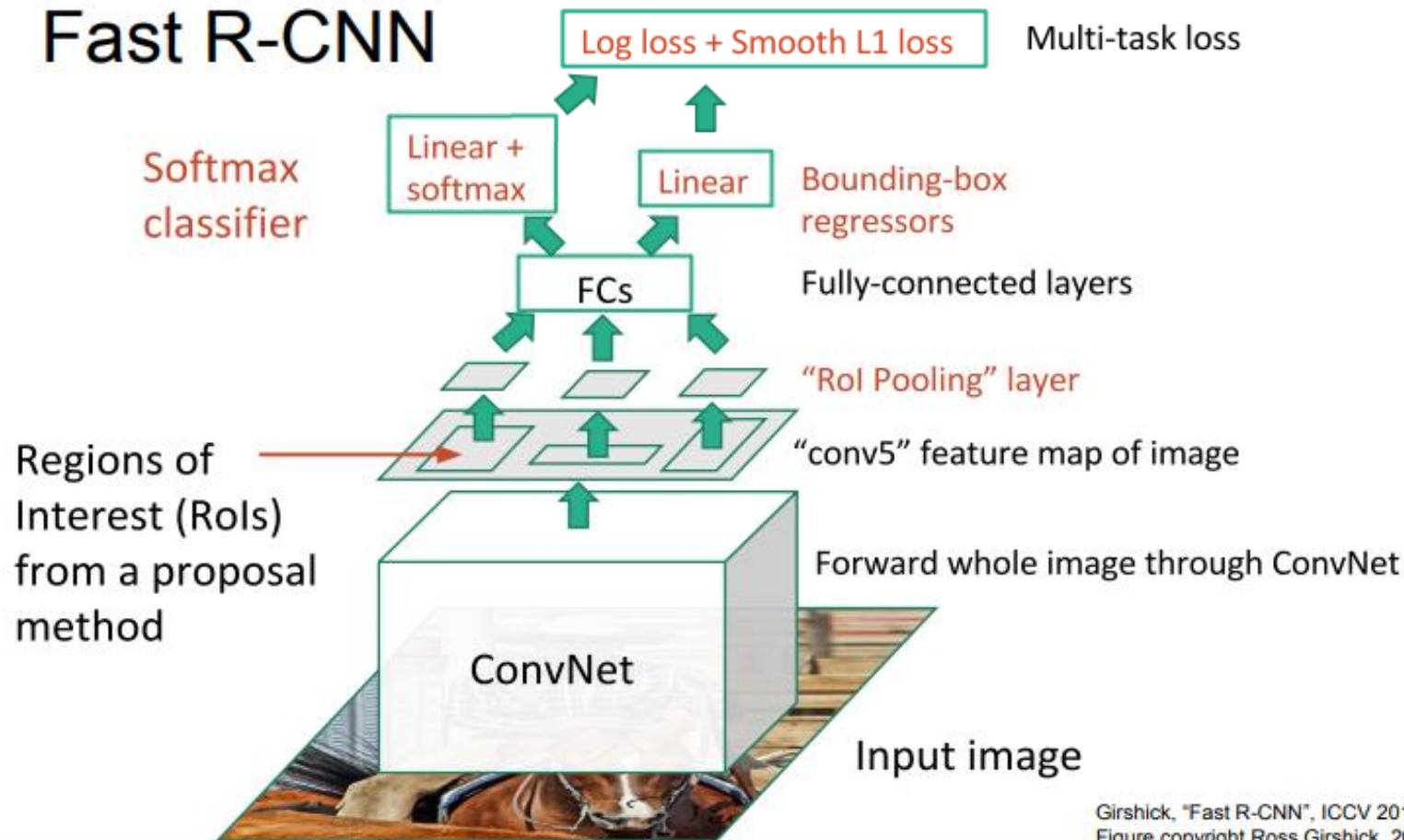
Fast R-CNN

- Towards Fast R-CNN: Spatial Pyramid Pooling (ROI pooling)



- Pass full image through the network: Feature maps
 - Apply region proposals to **last conv layer**
 - Classification CNN has **fixed input size**
 - **Spatial pyramid pooling** layer pools to fixed size using max-pooling (orig. 3x w. different window size & stride)
 - + Image-wise computation shared → Speed up by SPP during inference: $\approx 24-104\times$
 - R-CNN problems: **slow training** / not end-to-end
- Use SPP to resample detected super-pixels (patches) to a fixed size.
 - Spatial pyramid pooling allows us to run only a single pass of CNN feature extraction. It maps the detected ROIs to the max-pooled space.

Fast R-CNN

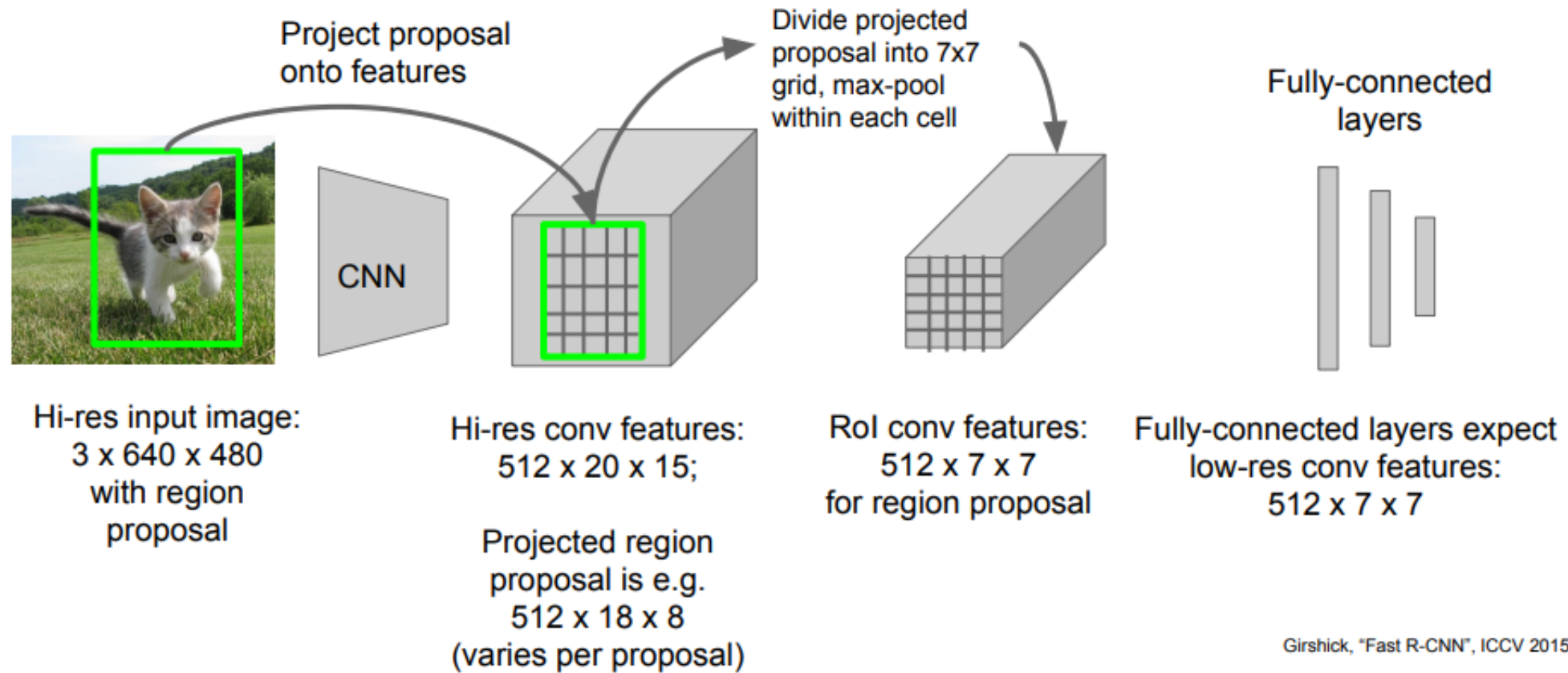


Girshick, "Fast R-CNN", ICCV 2015.

Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

Fast R-CNN

Fast R-CNN: RoI Pooling



Faster R-CNN

- Introduce RPN (region proposal network)
 - Perform region proposal directly into the architecture
 - Alleviating the need for Selective Search

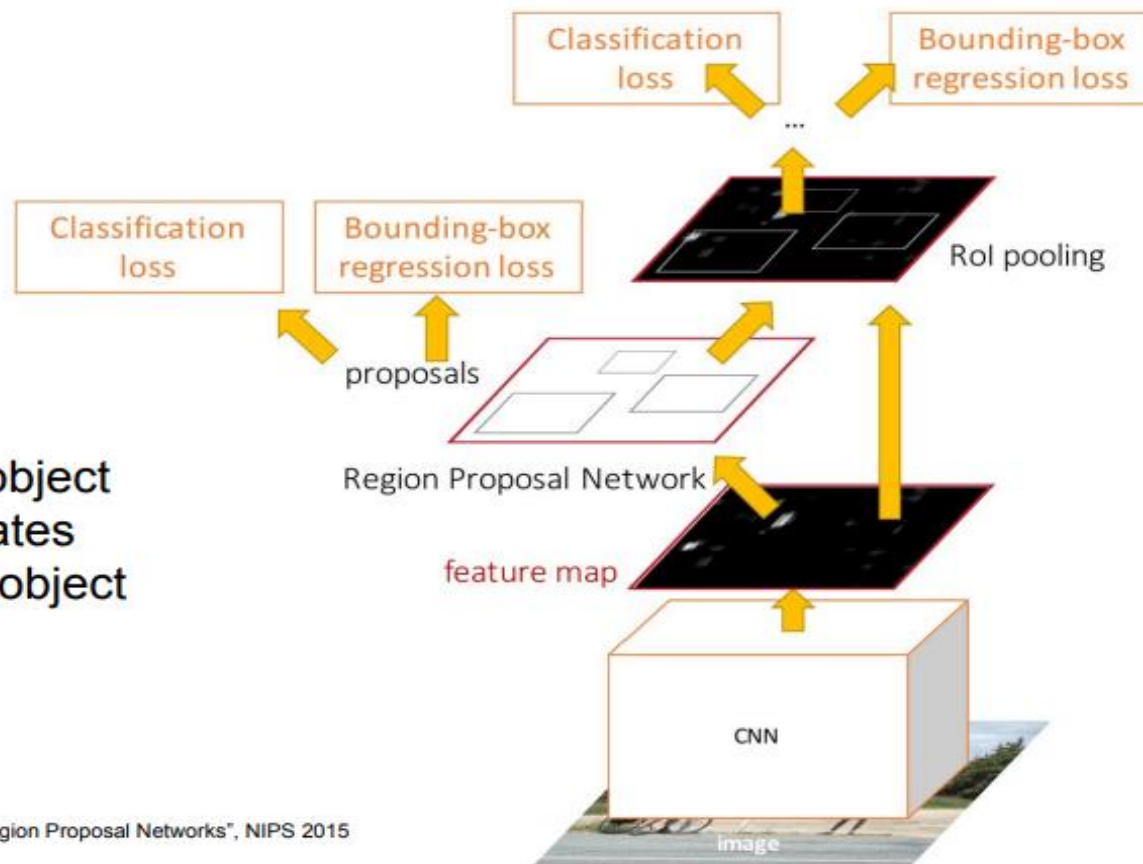
Faster R-CNN:

Make CNN do proposals!

Insert **Region Proposal Network (RPN)** to predict proposals from features

Jointly train with 4 losses:

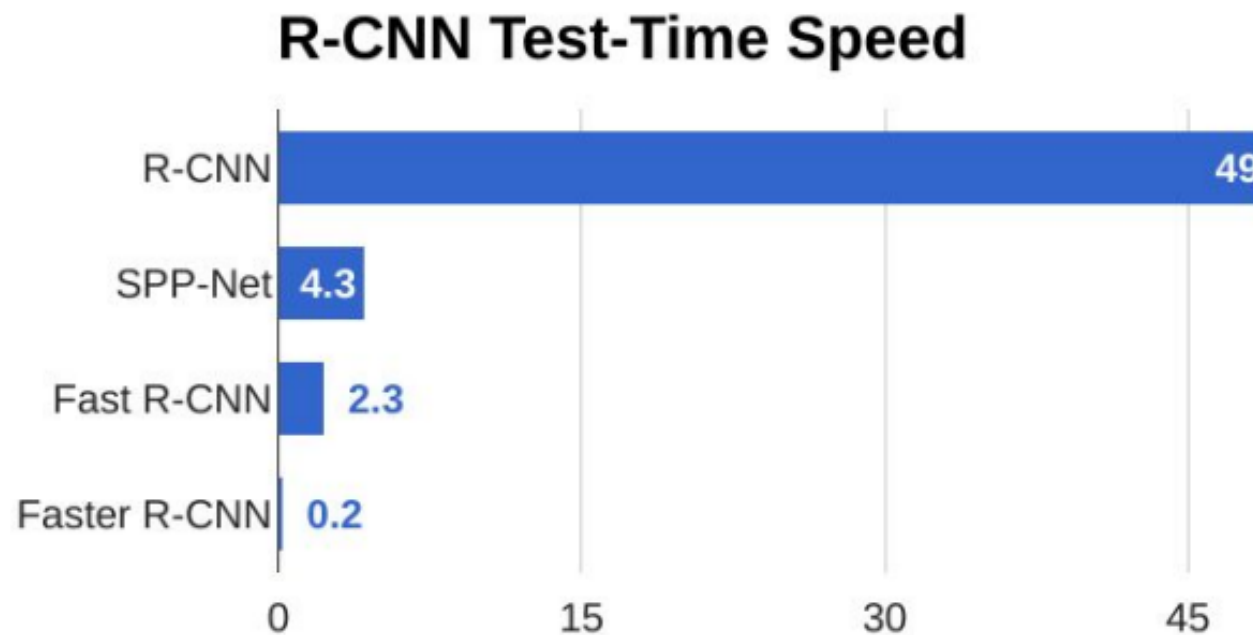
1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classification score (object classes)
4. Final box coordinates



Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015
Figure copyright 2015, Ross Girshick; reproduced with permission

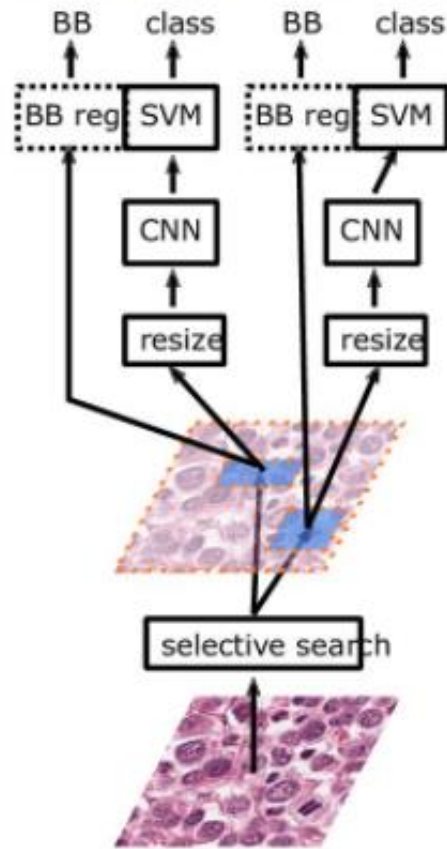
Faster R-CNN

- Make CNN do proposals!

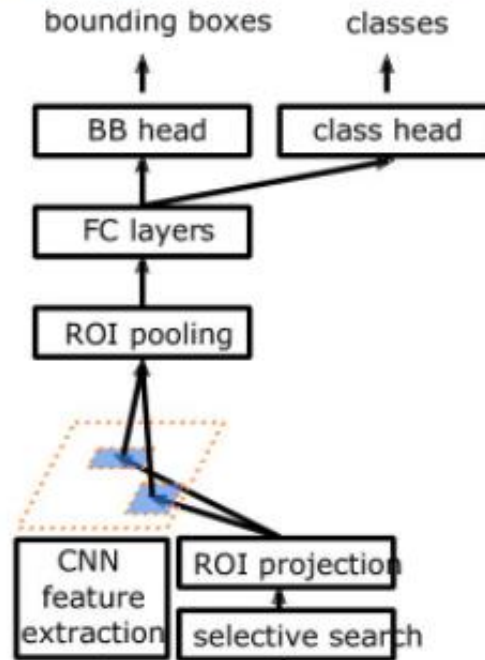


Architectures based on R-CNN

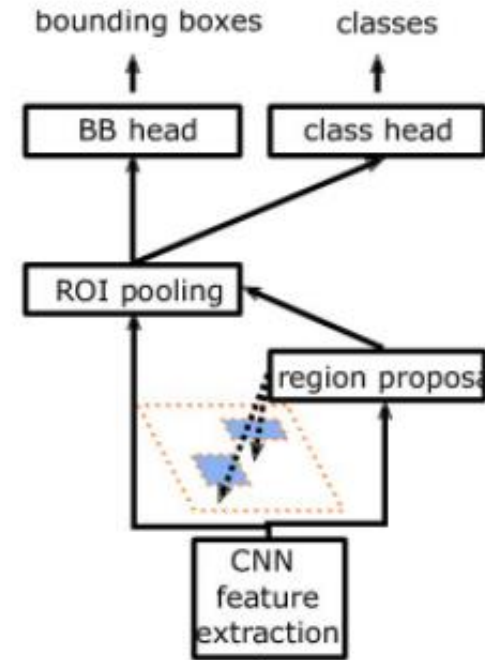
Overview: Architectures based on R-CNN



R-CNN



Fast R-CNN



Faster R-CNN

Architectures based on R-CNN

- **R-CNN 원리**

- (1) 이미지 내에 객체가 존재하는 적절한 위치를 제안
- (2) 제안된 위치의 이미지를 잘라냄
- (3) 잘라낸 이미지의 특징맵 (feature map)를 추출 – CNN
- (4) 특징 지도를 분류기에 입력하여 분류 - CNN

- **Fast R-CNN**

- (3)(4) 의 과정 개선
- 개별 ROI 대신 전체에 CNN 적용
- ROI Pooling 레이어 사용 (fixed size ROI): Classifier 모델이 간단해 짐

- **Faster R-CNN**

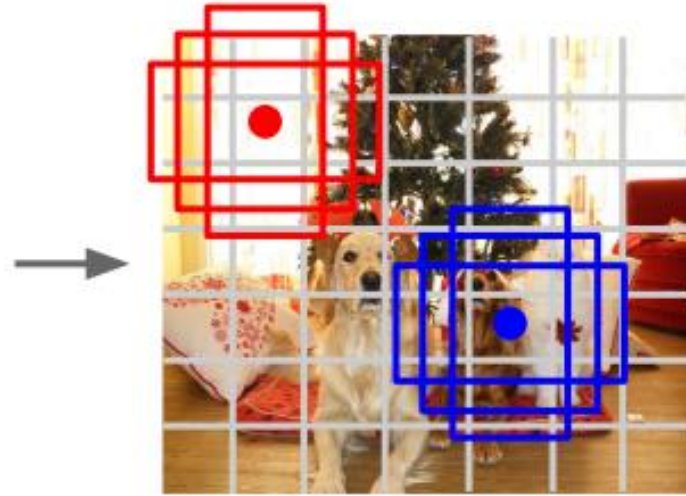
- (1) 번 과정 개선
- RPN(Region Proposal Network) 사용하여 Region 을 찾음

Detection without Proposals: YOLO / SSD

Go from input image to tensor of scores with one big convolutional network! →



Input image
 $3 \times H \times W$



Divide image into grid
 7×7

Image a set of **base boxes**
centered at each grid cell
Here $B = 3$

Within each grid cell:

- Regress from each of the B base boxes to a final box with 5 numbers:
($dx, dy, dh, dw, confidence$)
- Predict scores for each of C classes (including background as a class)

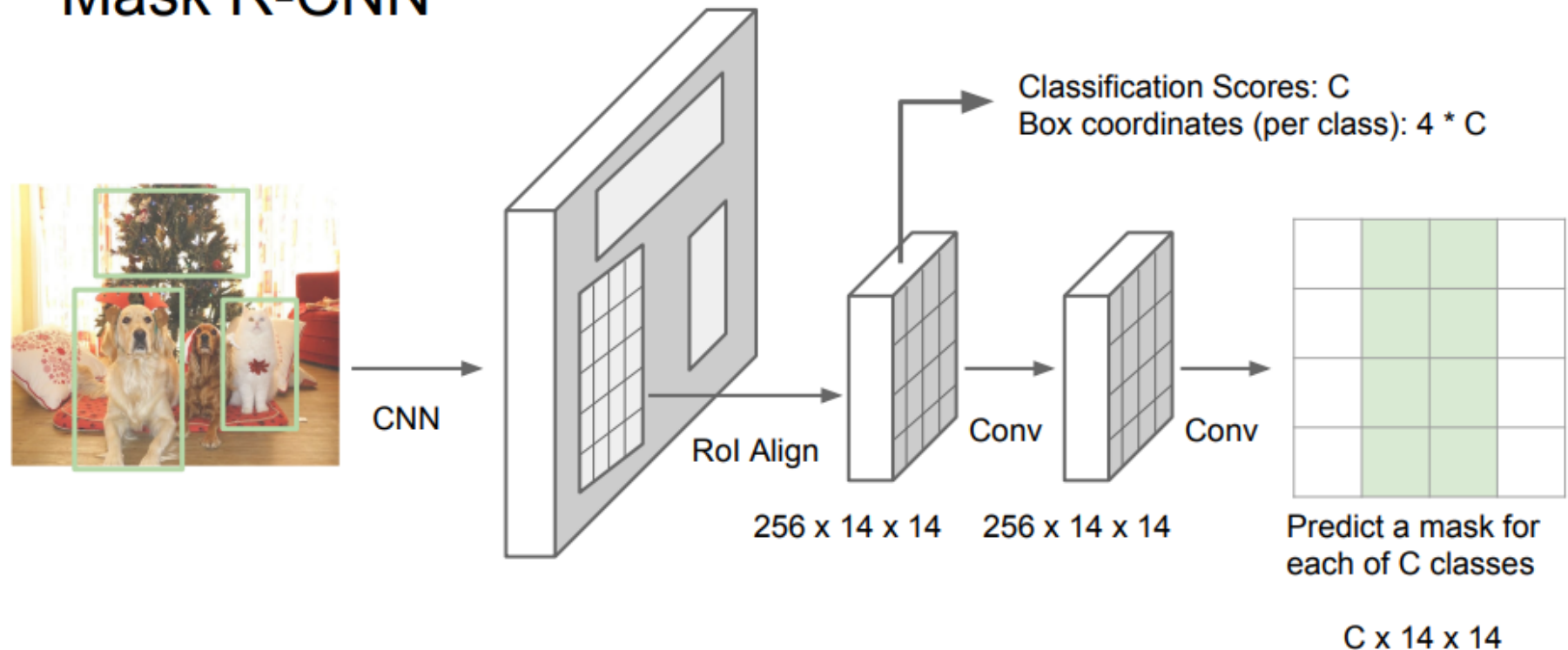
Output:
 $7 \times 7 \times (5 * B + C)$

Redmon et al, "You Only Look Once:
Unified, Real-Time Object Detection", CVPR 2016
Liu et al, "SSD: Single-Shot MultiBox Detector", ECCV 2016

Instance Segmentation (Mask R-CNN)

- object detection (Faster R-CNN) + image segmentation
- use a pre-trained model which has ResNet-50-FPN backbone

Mask R-CNN



He et al, "Mask R-CNN", arXiv 2017

Mask R-CNN

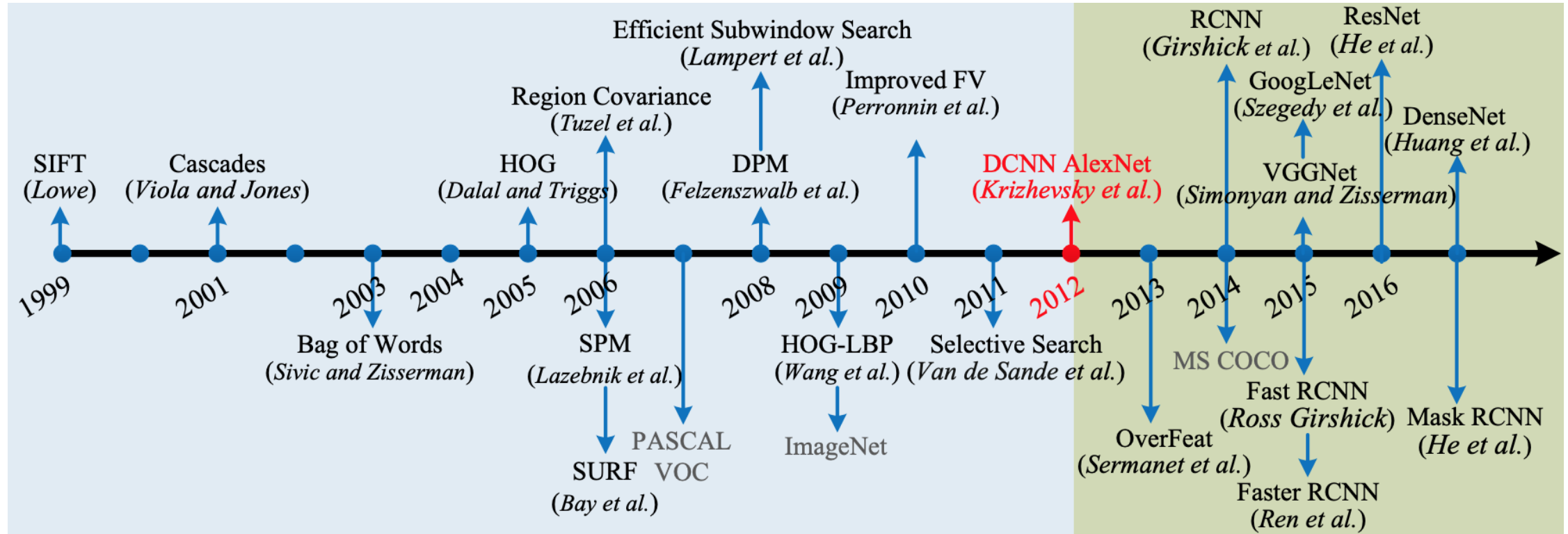
- Detectron이라는 기술로 Facebook에서 공개
- 상당히 정교하게 객체 인식과 세그멘테이션을 수행



Image Dataset for Object Detection – VOC, COCO

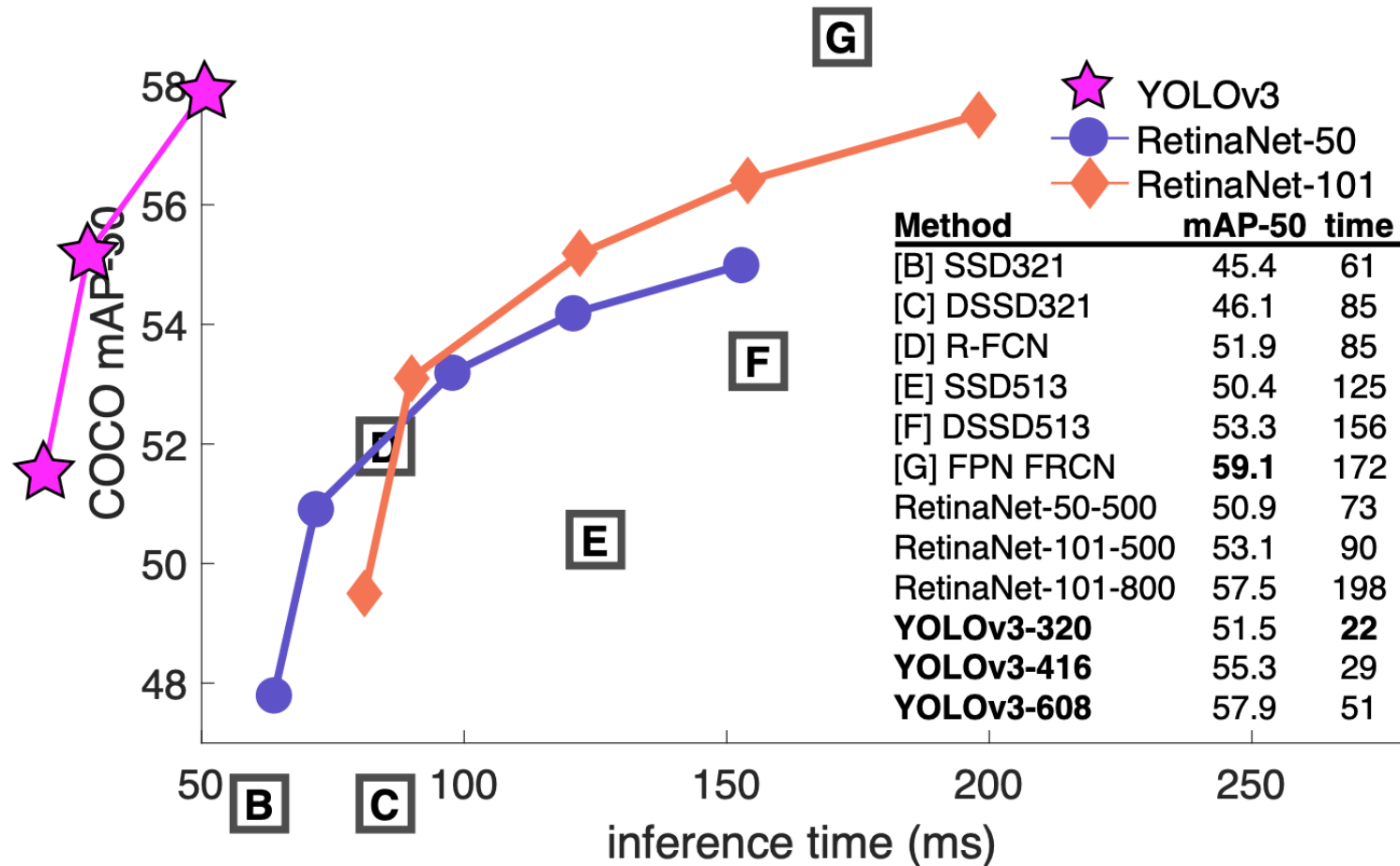
- VoC
 - 20개 클래스, 11,530 이미지, 27,450 annotation을 제공
 - image당 2.4개의 객체를 포함
- COCO
 - MS 사가 주최하는 COCO challenge 대회가 2015년부터 운영 중이며 여기서 제공되는 데이터가 COCO(common object context)
 - 91 카테고리, 20만개의 이미지, 50만개의 annotation, 32만개 영상을 제공
 - image당 평균 7.3 개의 객체를 포함

Image Dataset and Algorithms for Object Detection



YOLO (You Only Look Once)

- 객체 검출 방식 중에 가장 속도가 빠르고 성능도 우수한 방법으로 널리 사용



YOLO (You Only Look Once)

- 객체 인식(classification)과 위치예측(localization)을 동시에 하나의 신경망으로 수행
 - 즉, 객체 예측과 경계박스를 찾는 작업을 동시에 수행하는 Single Shot Detector(SSD)을 수행
 - 기존의 다른 객체 검출 알고리즘들은 별도로 수행
- 입력 이미지 전체를 여러 지역(region)으로 나누고 경계 박스와 각 지역에 대한 객체 존재 여부를 확률로 예측
 - 이 경계박스는 예측확률의 가중합으로 계산
- 98개의 검출 결과를 제공 ($7 \times 7 \times 30 = 1470$ 벡터 출력)
- GoogleLeNet 기반으로 동작, C로 구현된 DarkNet을 framework로 사용
- 참고: <https://github.com/qgwweee/keras-yolo3>
- 동작 데모 : <https://pjreddie.com/darknet/yolo/#demo>

YOLO – Tiny YOLO3

- 정식 YOLO보다 간단하면서 속도가 빠른 tiny YOLO 버전
- 이를 사용하려면 아래와 같이 해당 버전을 설치
 - !wget <https://pjreddie.com/media/files/yolov3-tiny.weights>

Semantic Segmentation (분할)

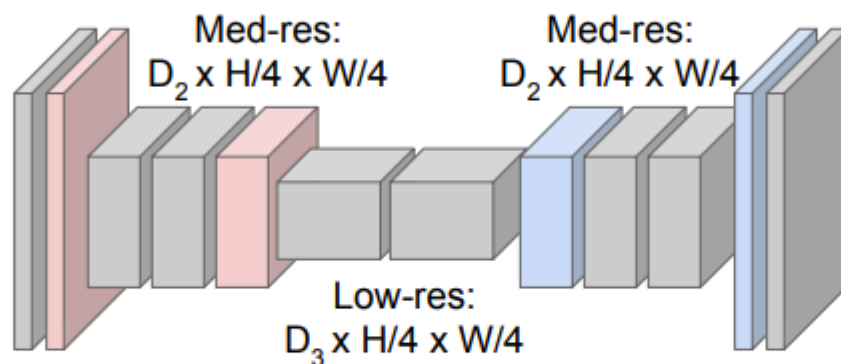
Semantic Segmentation Idea: Fully Convolutional

Downsampling:
Pooling, strided
convolution



Input:
 $3 \times H \times W$

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!



Upsampling:
Unpooling or strided
transpose convolution



Predictions:
 $H \times W$

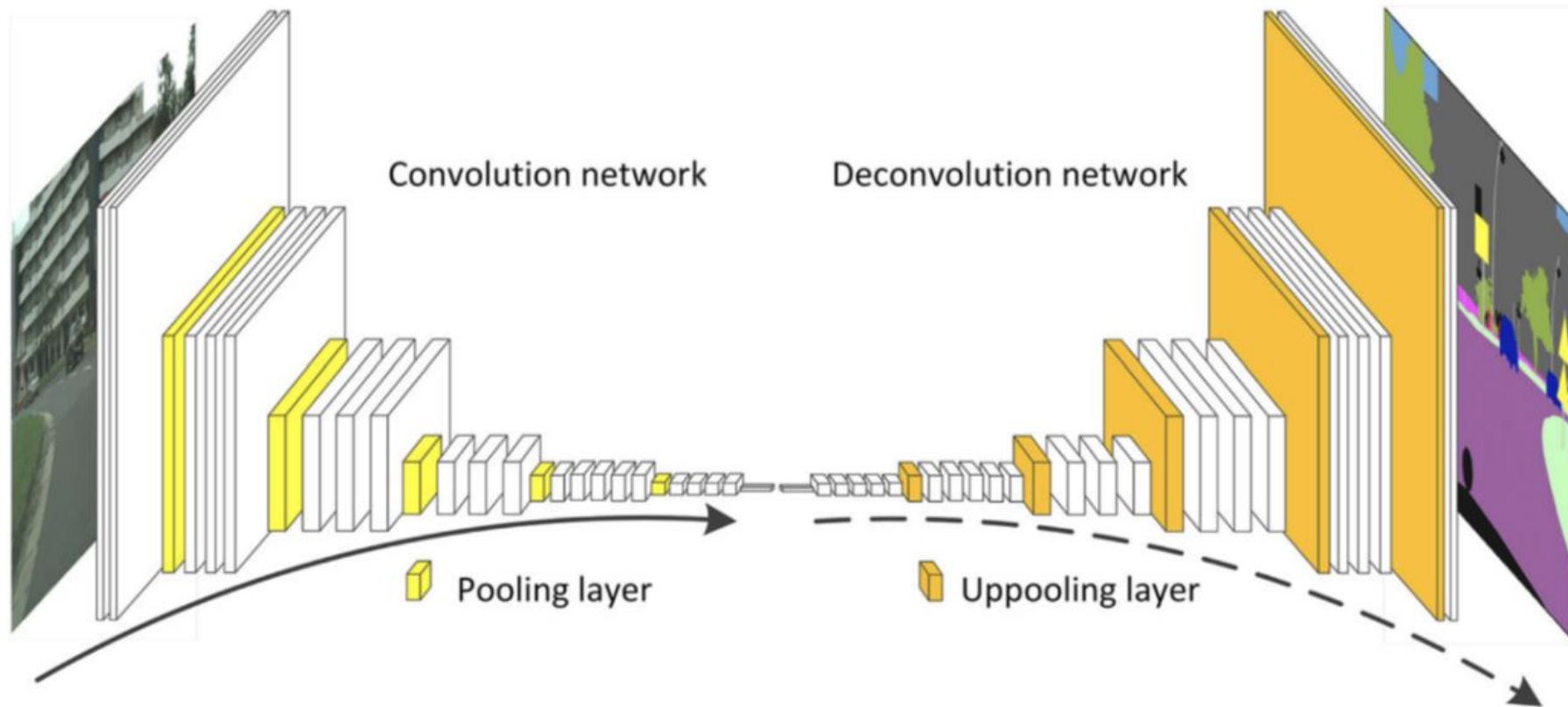
Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

Semantic Segmentation (분할)

- 픽셀 단위로 객체의 클래스를 표시하는 작업
- 머신러닝 모델
 - 정보를 계속 줄여나가는 작업을 주로 수행
 - 분류와 회귀는 모두 정보를 줄인 결과 클래스 이름이나 수치를 얻는다.
- 객체 분할
 - 원래 이미지 크기의 이미지를 다시 얻어야 한다
 - 따라서 정보를 축소한 후에 다시 정보를 늘리는 작업을 수행해야
 - Deconvolution, Up-Pooling 수행

Segmentation (분할)

- Encoder-Decoder 구조로 구성



Segmentation (분할)

- 객체 분할을 위해서는 U자형 구조를 갖는 U-Net을 주로 사용

