

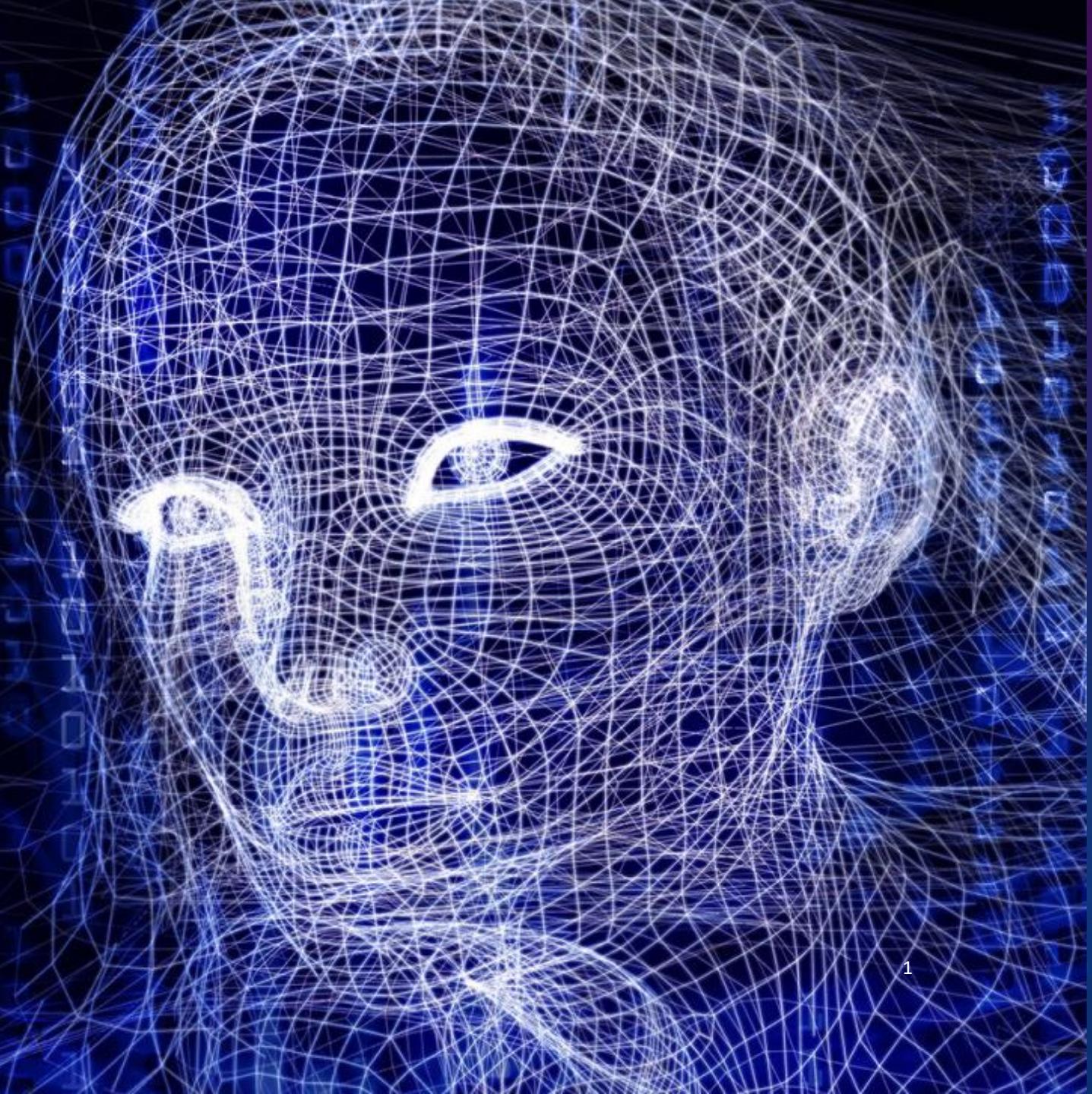
*LECTURE SERIES FOR DIGITAL  
SURVEILLANCE SYSTEMS AND  
APPLICATION*

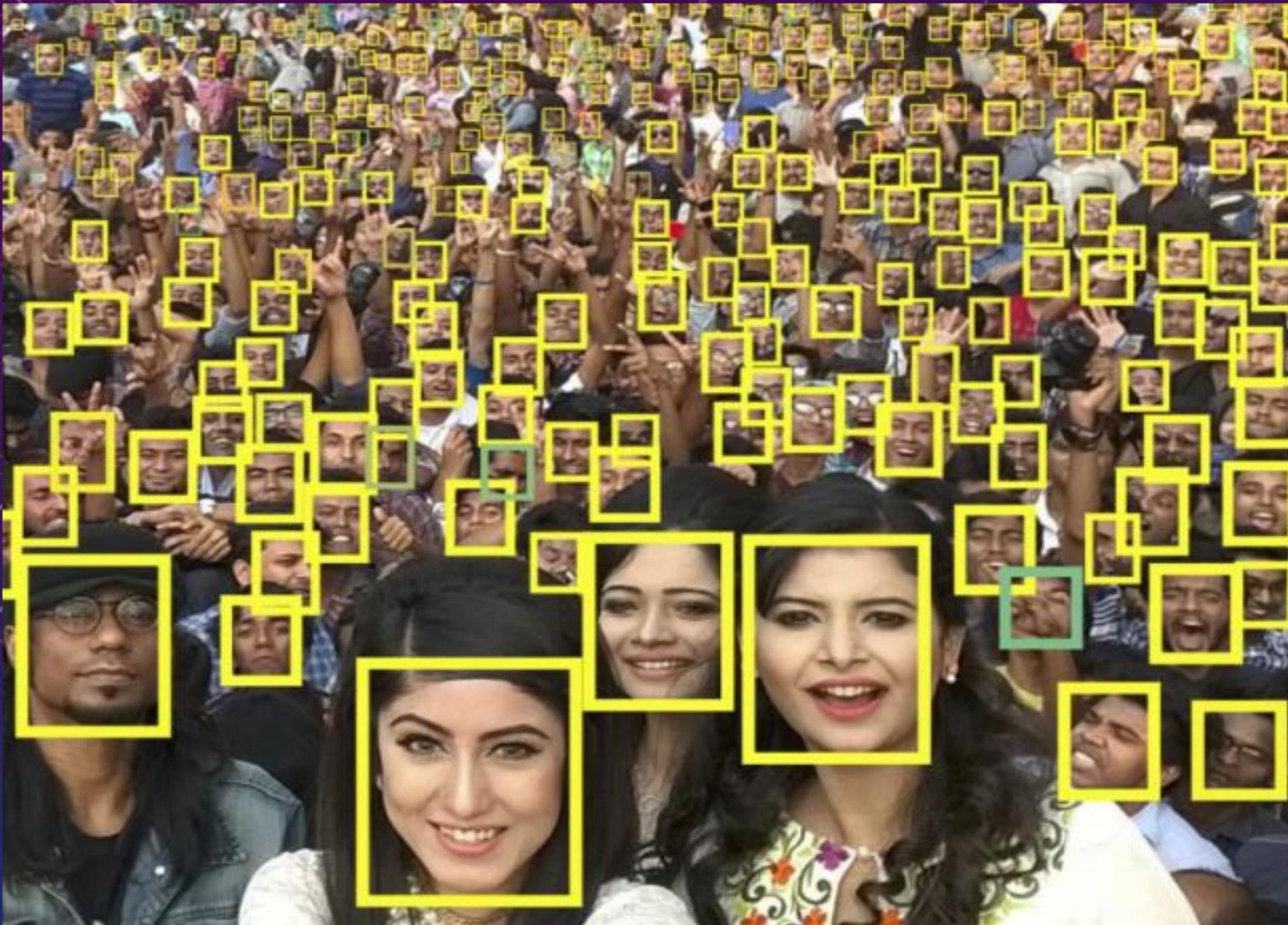
*Chapter 4*  
*Object Detection*

徐繼聖

Gee-Sern Jison Hsu

National Taiwan University of Science  
and Technology

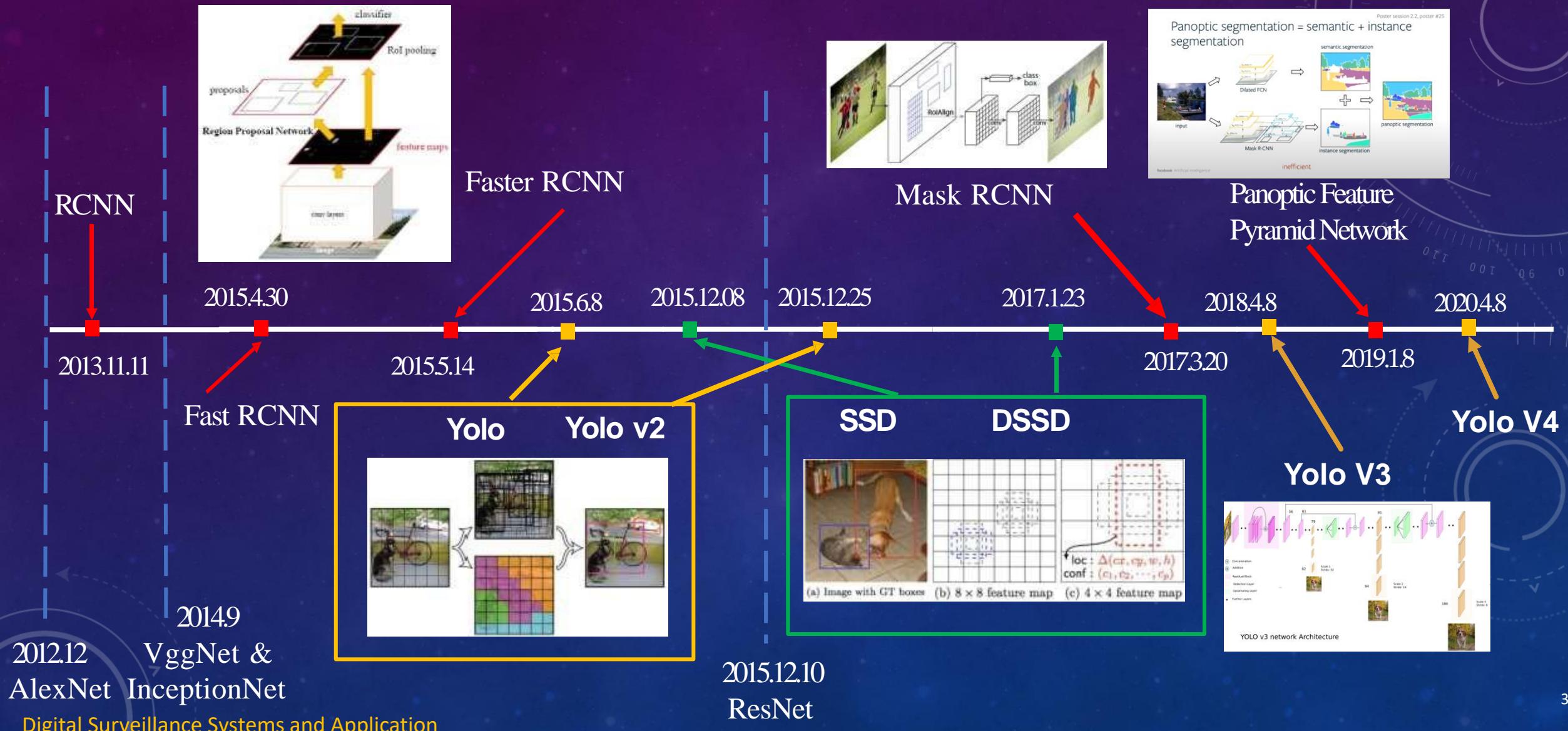




Face detection by Finding Tiny Faces

2

# The History of Object Detection in Deep Learning



# Contents

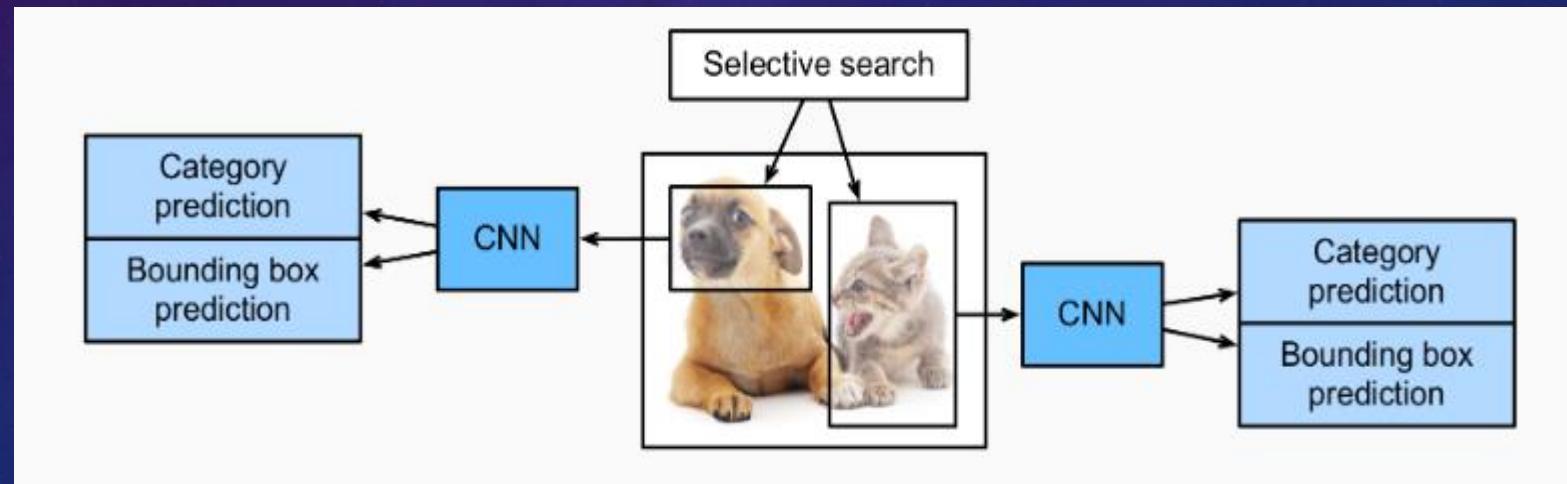
- Region-based CNN
- Fast R-CNN
- Faster R-CNN
- Mask R-CNN

# Region-based CNN

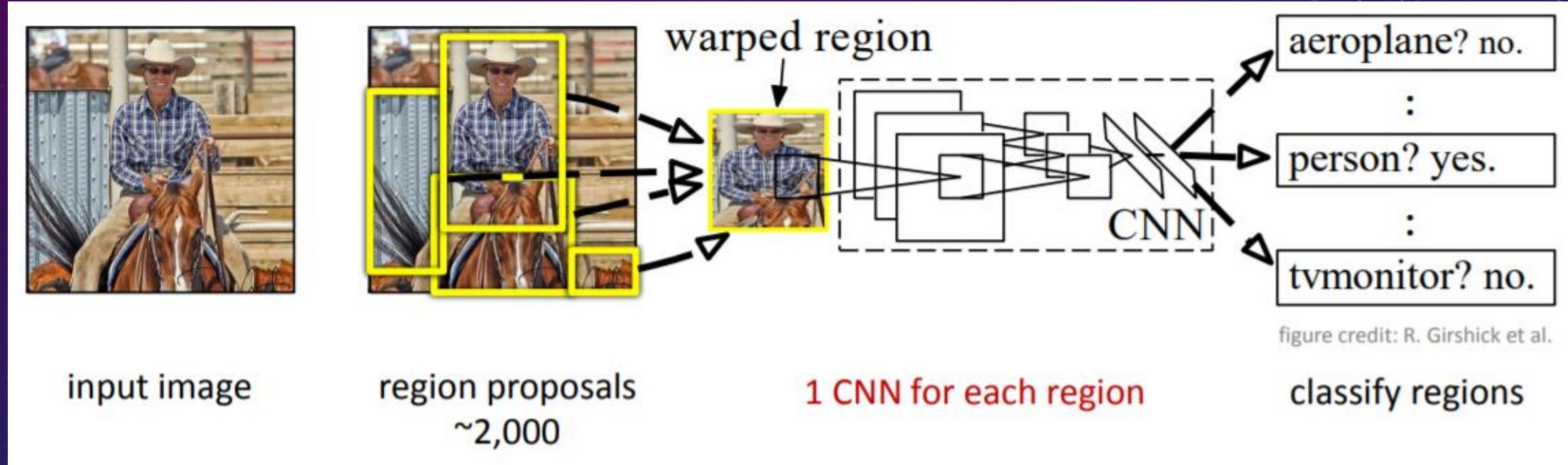
- Introduction
- Region-Based CNN
- R-CNN Algorithm
- Region Proposals – Selective Search

# Region-based CNN

- R-CNN models first select several proposed regions from an image (for example, anchor boxes are one type of selection method) and then label their categories and bounding boxes (e.g., offsets).
- They use a CNN to perform forward computation to extract features from each proposed area. Afterwards, we use the features of each proposed region to predict their categories and bounding boxes.



# Region-Based CNN Architecture



R-CNN pipeline

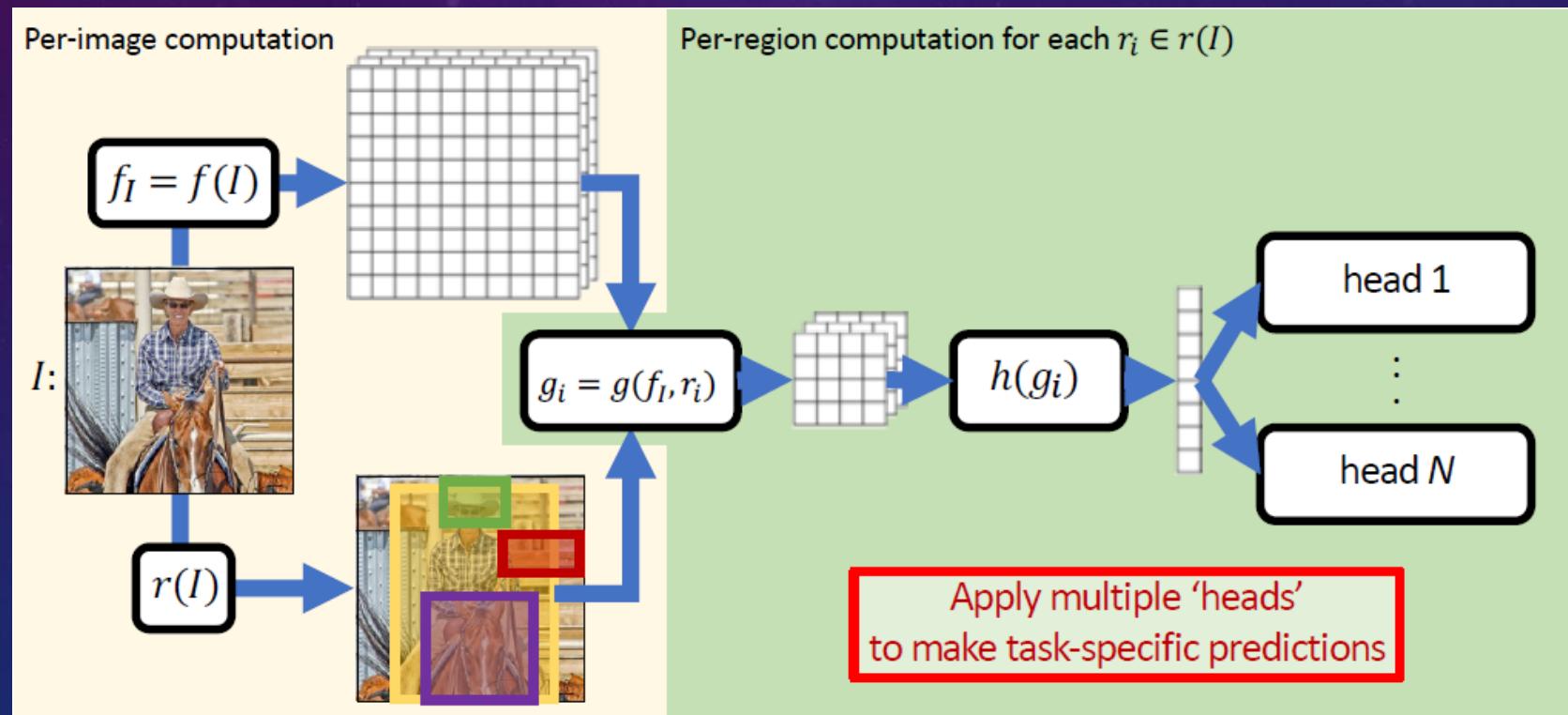
# R-CNN Algorithm

## Per-Image computation

Extract features by  $f_I$   
Region detection proposal  $r(I)$

## Per-Region computation

Extract region of Interest  $g_i$   
Pass each region to CovNet  $h(g_i)$   
multiple heads to make prediction

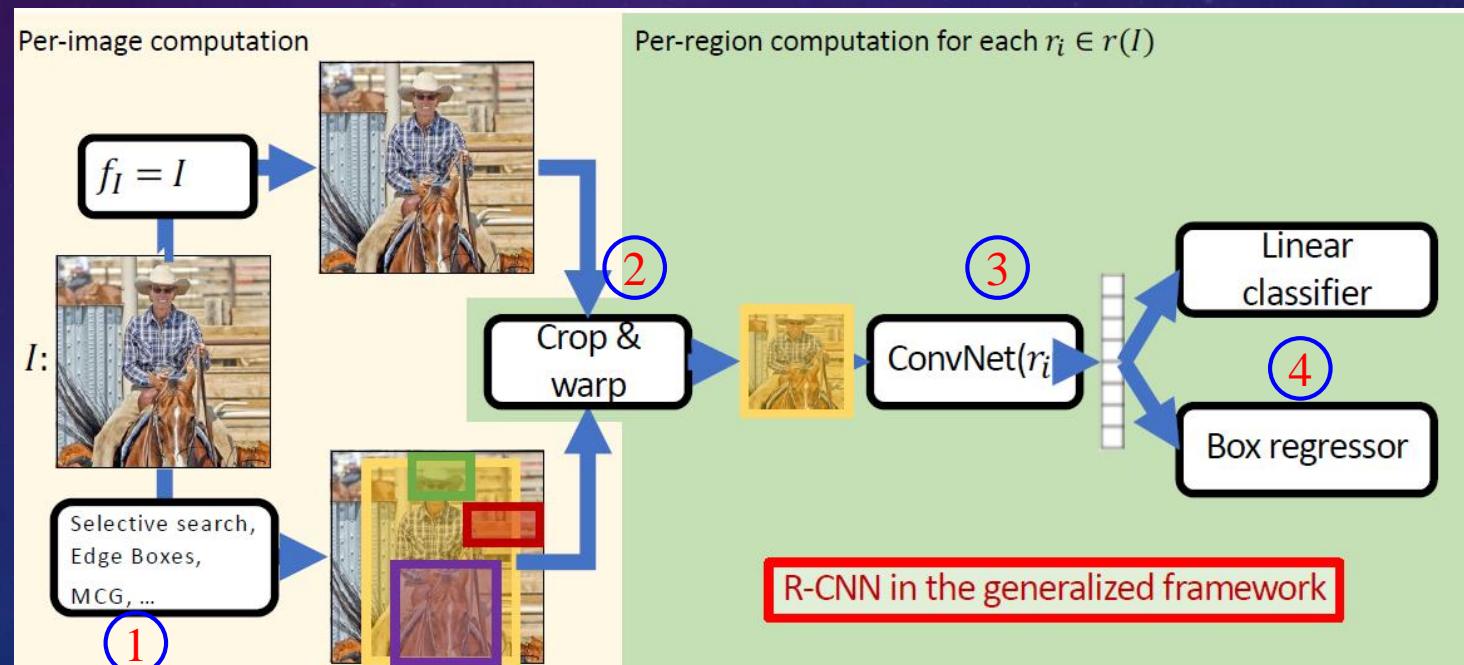


# R-CNN Algorithm

- Use selective search to find region proposal (~2k)
- Extract region of interest from original image
- Apply ConvNet to each region of interest
- Use support vector machine to find class label and linear regression to find bounding box offsets.

Problem:

- ✓ Very heavy per-region computation
- ✓ Ad hoc training objective
- ✓ Inference is very slow



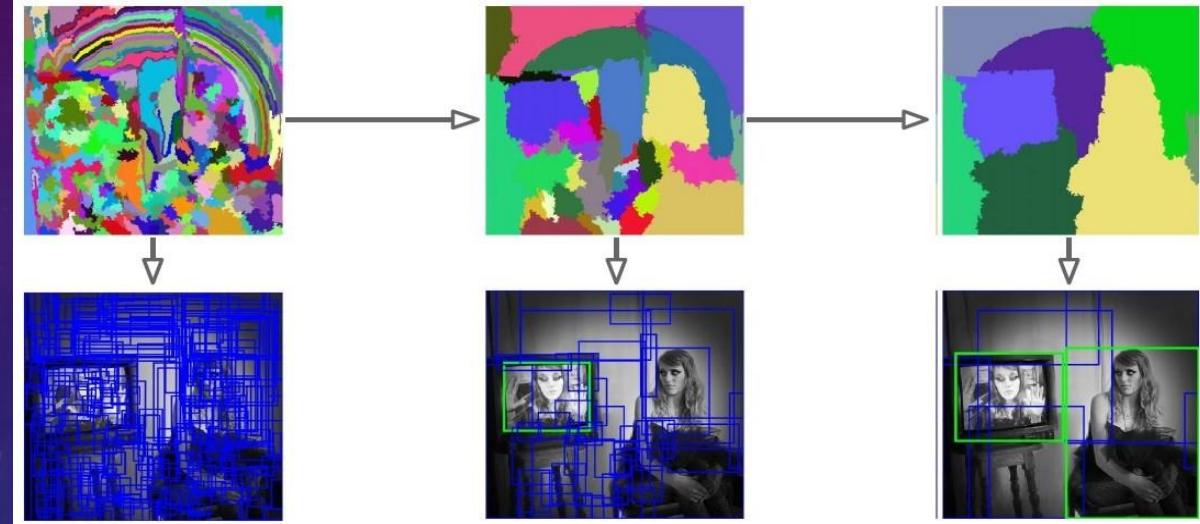
# Detection and Segmentation – Stanford University School of Engineering



<https://www.youtube.com/watch?v=nDPWywWRIRo&list=PLC1qU-LWwrF64f4QKQT-Vg5Wr4qEE1Zxk&index=11>  
[Starting from 08:04] [1:14:25]

# Region Proposals – Selective Search

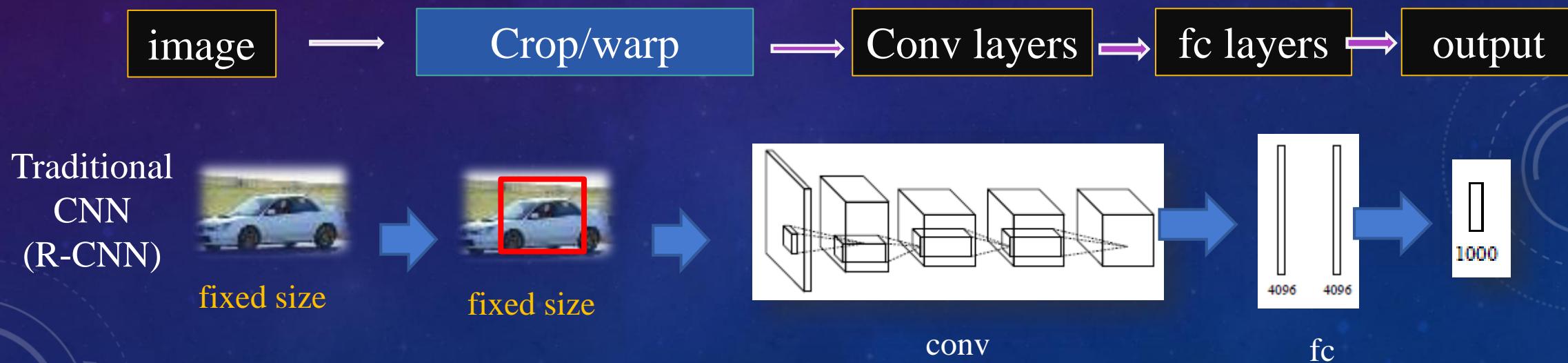
- It is based on computing hierarchical grouping of similar regions based on color, texture, size and shape compatibility.
- Selective Search starts by over-segmenting the image based on intensity of the pixels using a graph-based segmentation method by Felzenszwalb and Huttenlocher.



IJCV paper by Felzenszwalb and Huttenlocher: [http://people.cs.uchicago.edu/~pff/papers/seg\\_ijcv.pdf](http://people.cs.uchicago.edu/~pff/papers/seg_ijcv.pdf)

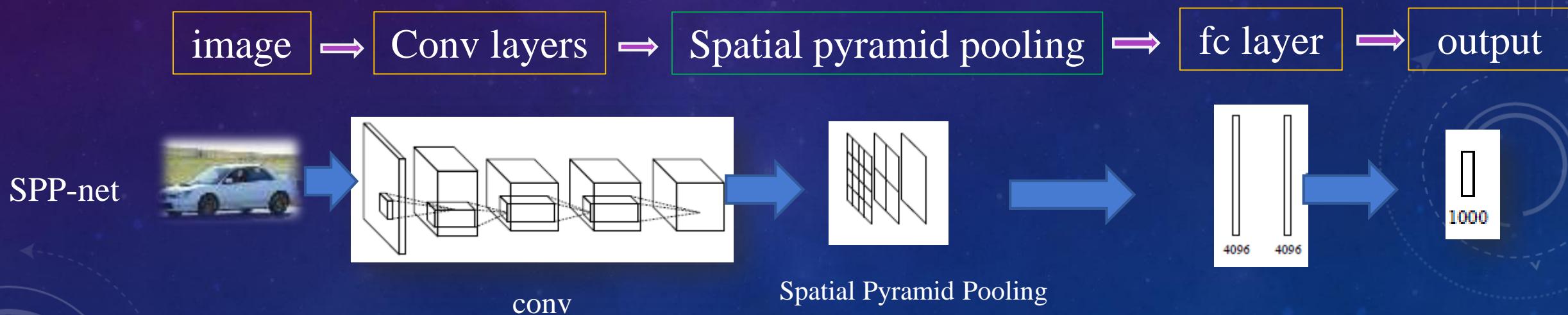
# R-CNN

- In a typical CNN structure, a full connection is usually connected after the convolutional layer. The number of features of the fully connected layer is fixed, so the input size is fixed when the network is in use. But in reality, the image size of our input is always unable to meet the size required for input. The usual way to handle this issue about input is by cropping and warping.



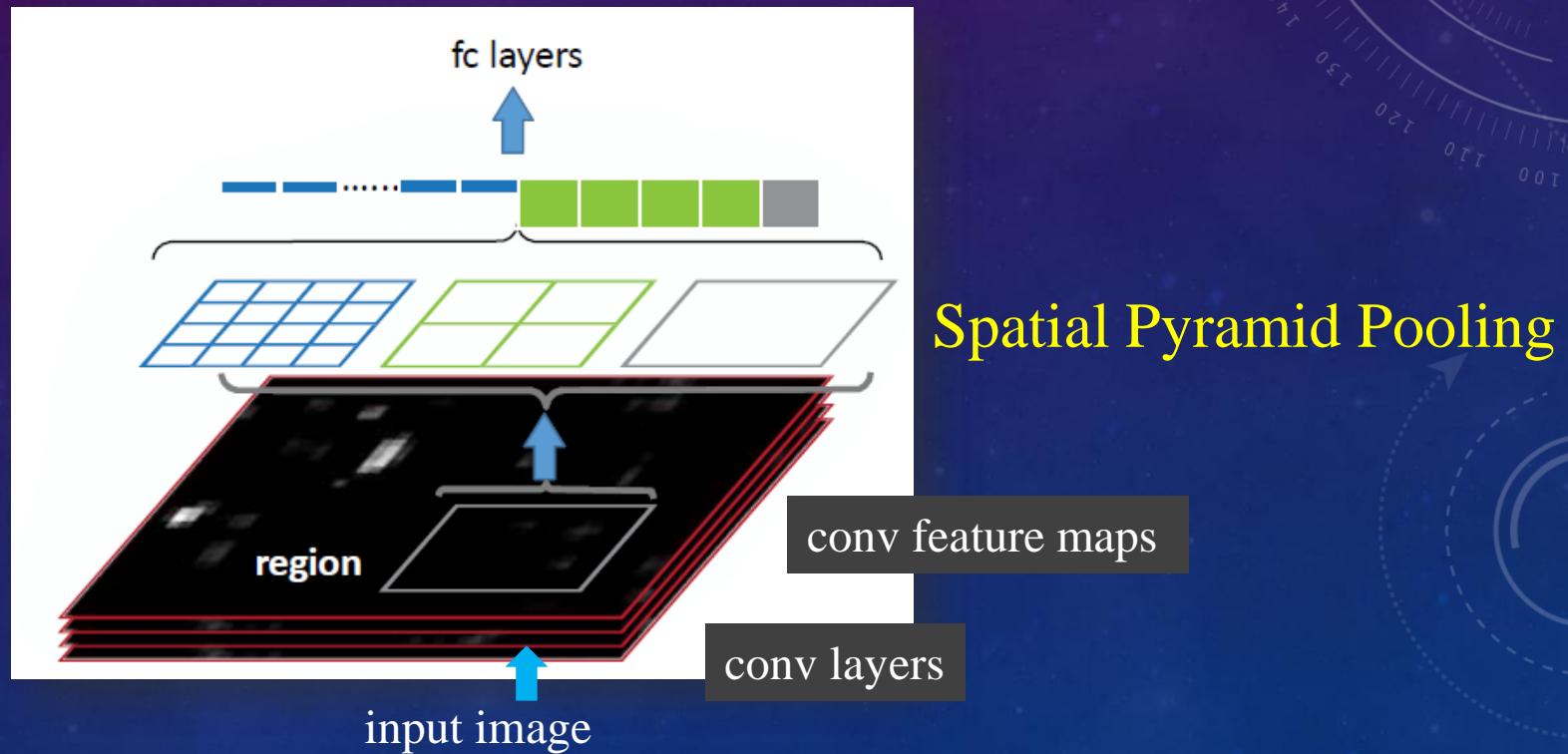
# Spatial Pyramid Pooling

- SPP can produce a fixed size output regardless of the input size
- Use multiple windows (pooling window)
- SPP can use the same image with different scales as input to get the same length of pooling features.

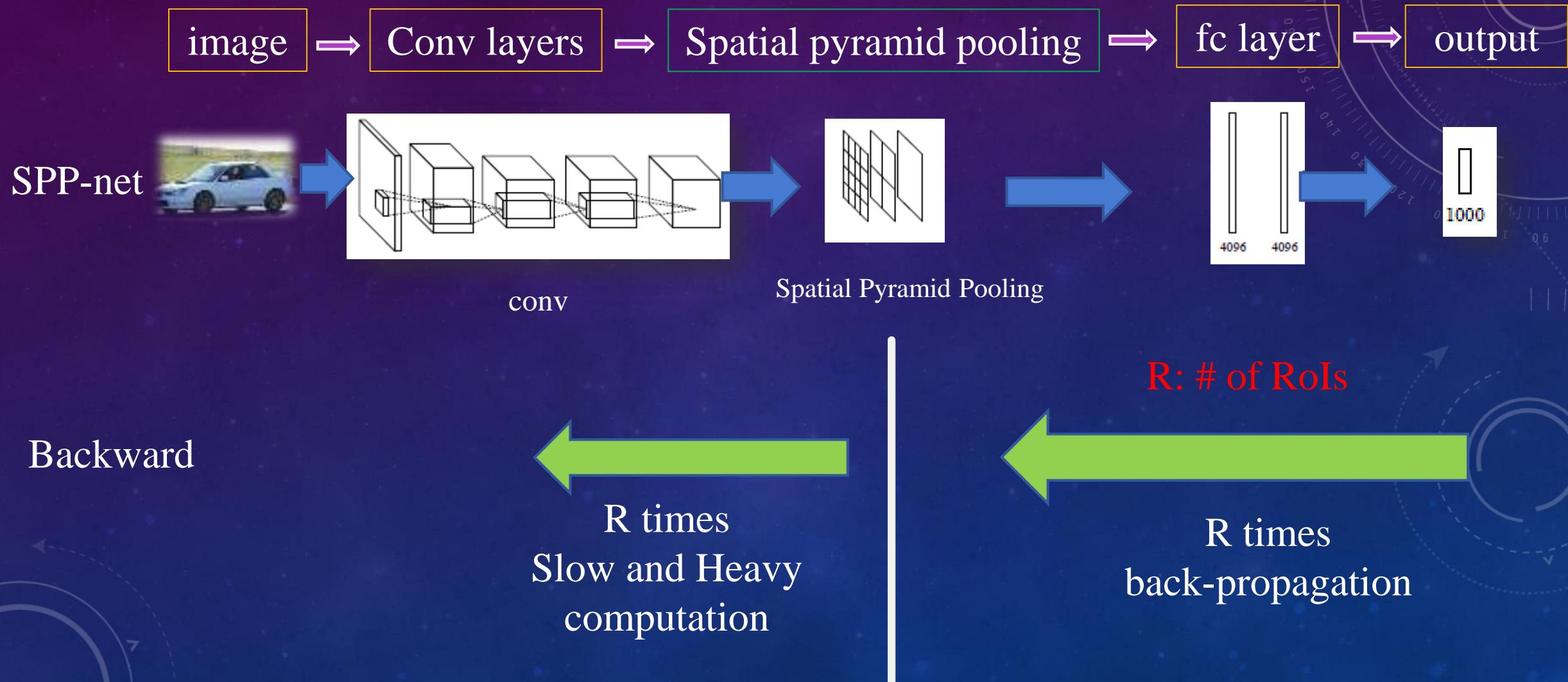


# Spatial Pyramid Pooling

SPP NET uses multiple windows to detect multiple scales.

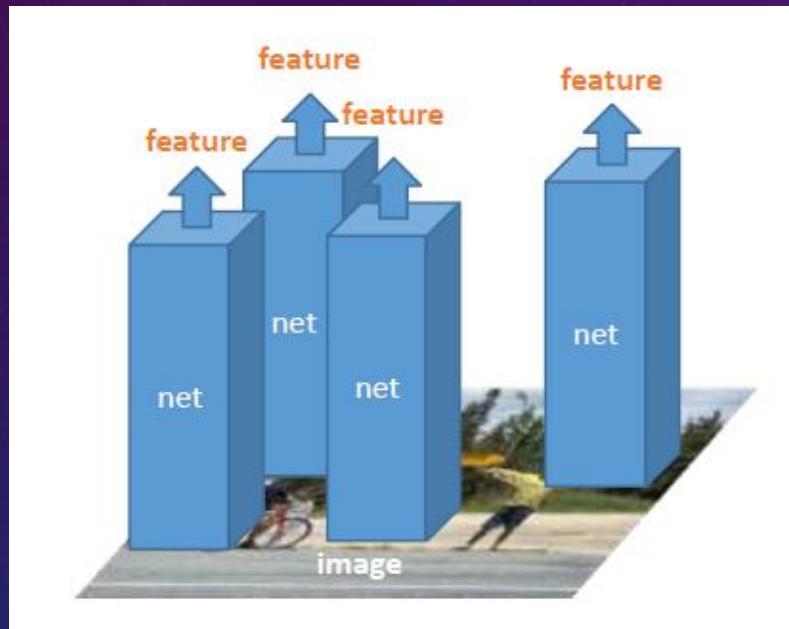


# Spatial Pyramid Pooling (SPP)



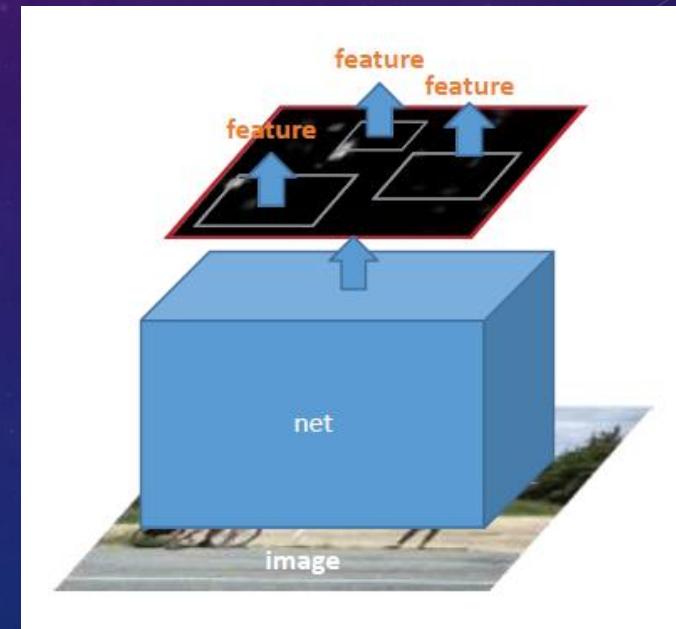
# RCNN vs. SPP

Image Regions vs. Feature Map Regions



R-CNN

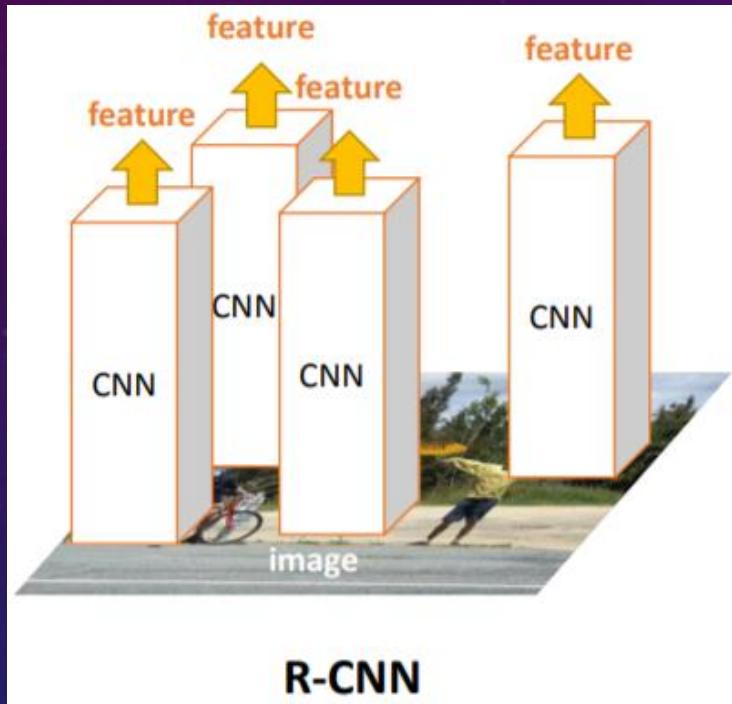
2000 nets on image regions



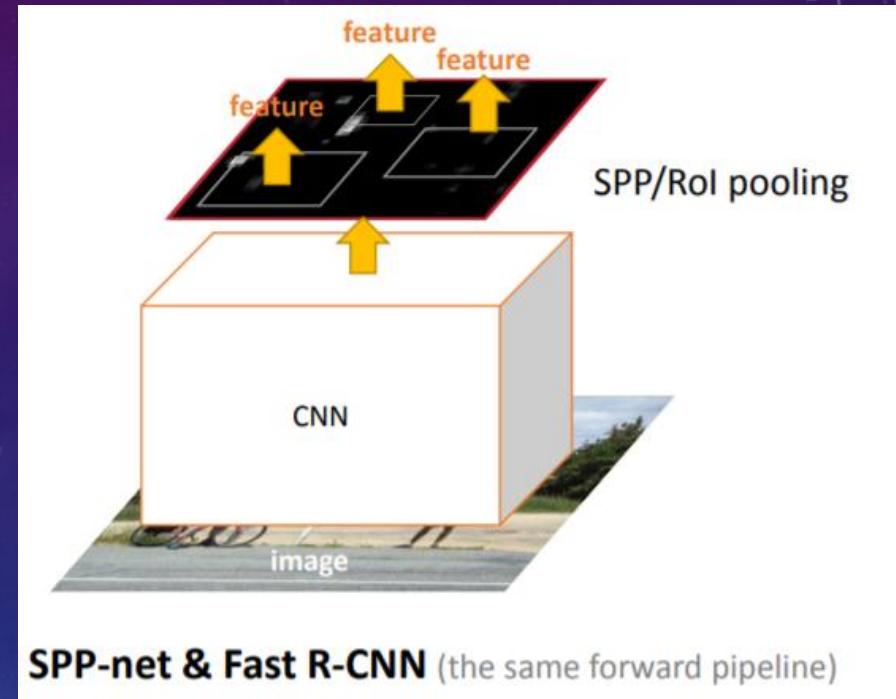
SPP-net

1 net on full image

# R-CNN vs. Fast R-CNN (Forward Pipeline)

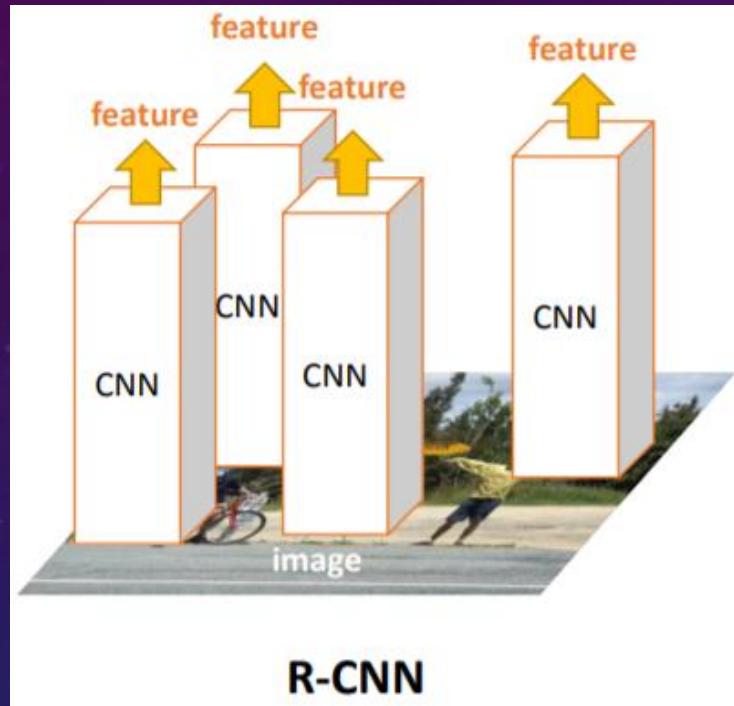


- Extract image regions
- 1 CNN per region (2000 CNNs)
- Classify region-based features

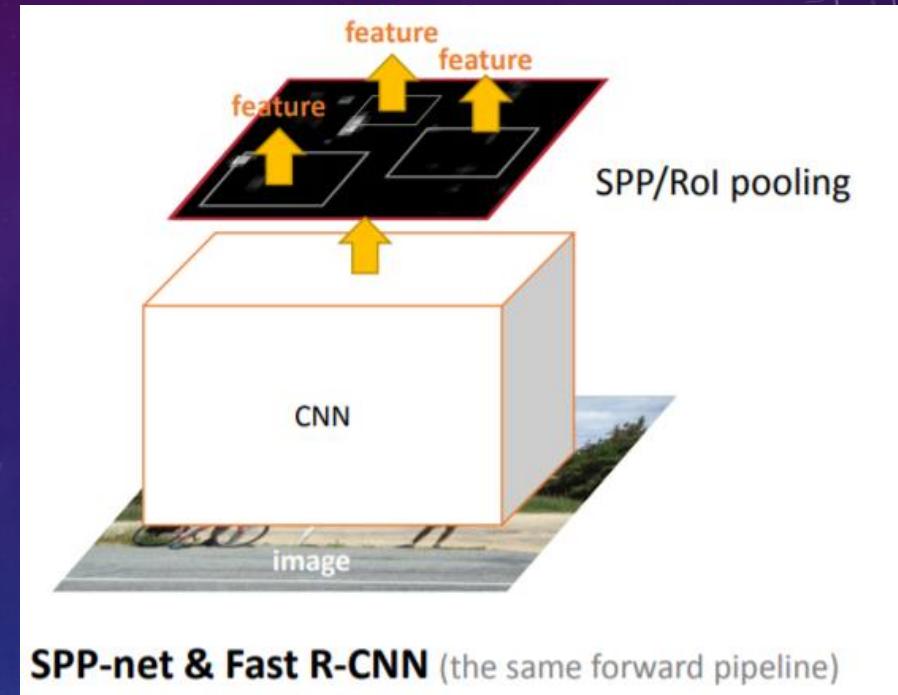


- 1 CNN on the entire image
- Extract features from **feature map regions**
- Classify region-based features

# R-CNN vs. Fast R-CNN (Forward Pipeline)



- Complexity:  $\sim 224 \times 224 \times 2000$



- Complexity:  $\sim 600 \times 1000 \times 1$
- $\sim 160x$  faster than R-CNN

# R-CNN vs. Fast R-CNN

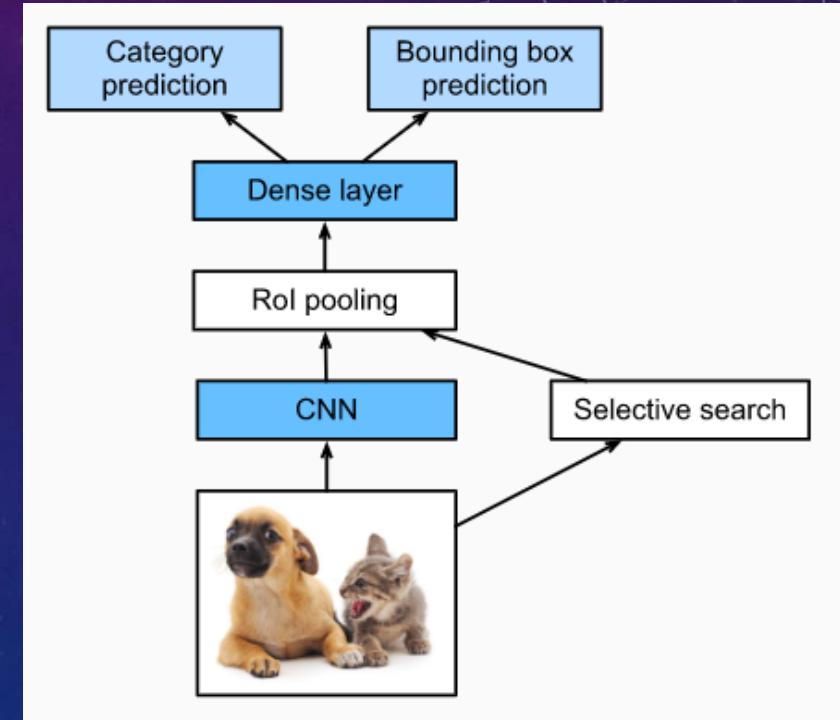
- Fix most of the slow processing issues with R-CNN and SPP-net
- Train the detector in a single stage, end-to-end
  - No caching features to disk
  - No post hoc training steps
- Train all layers of the network

# Problems of R-CNN

- **Slow at test-time:** need to run full forward path of CNN for each region proposal
  - 13s/image on a GPU(K40)
  - 53s/image on a CPU
- **SVM and regressors are post-hoc:** CNN features not updated in response to SVMs and regressors
- **Complex multistage training pipeline** (84 hours using K40 GPU)
  - Fine-tune network with softmax classifier (log loss)
  - Train post-hoc linear SVMs (hinge loss)
  - Train post-hoc bounding-box regressions (squared loss)

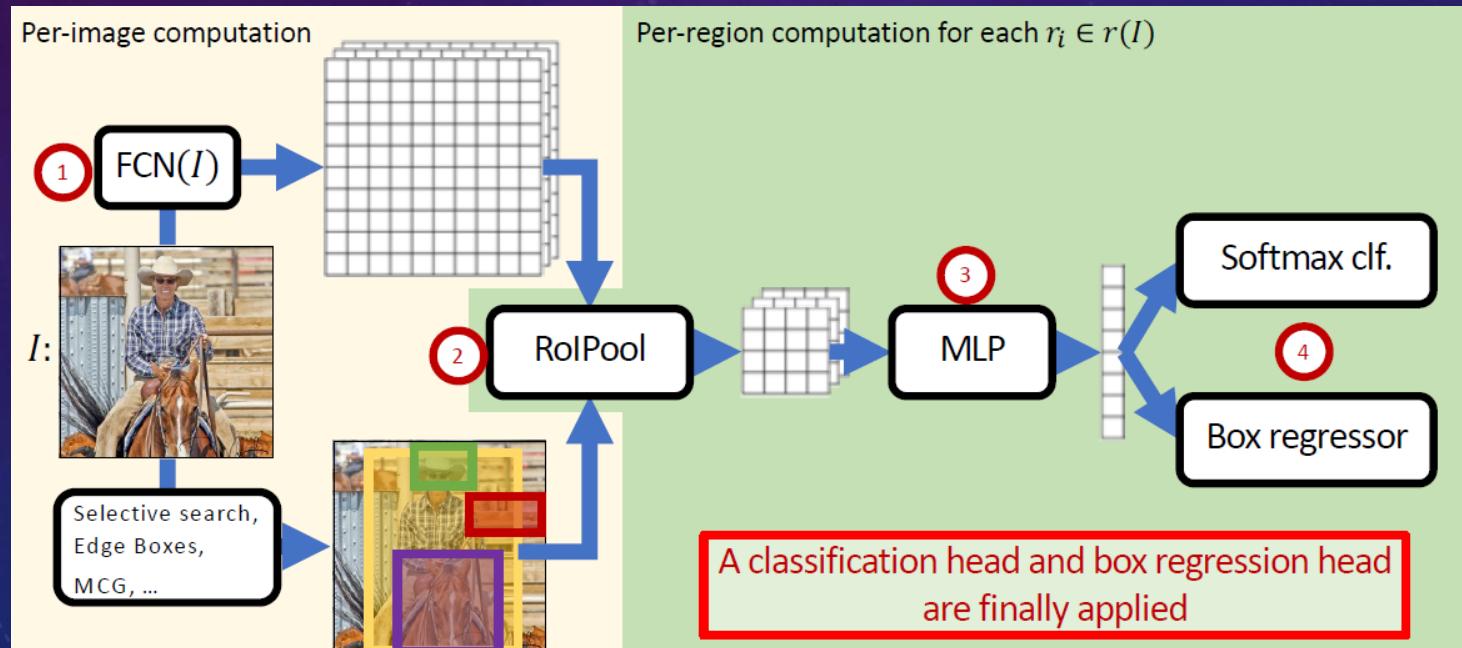
# FAST R-CNN

- The main performance bottleneck of an R-CNN model is the need to independently extract features for each proposed region.
- As these regions have a high degree of overlap, independent feature extraction results in a high volume of repetitive computations. Fast R-CNN improves on the R-CNN by only performing CNN forward computation on the image as a whole.



# FAST R-CNN

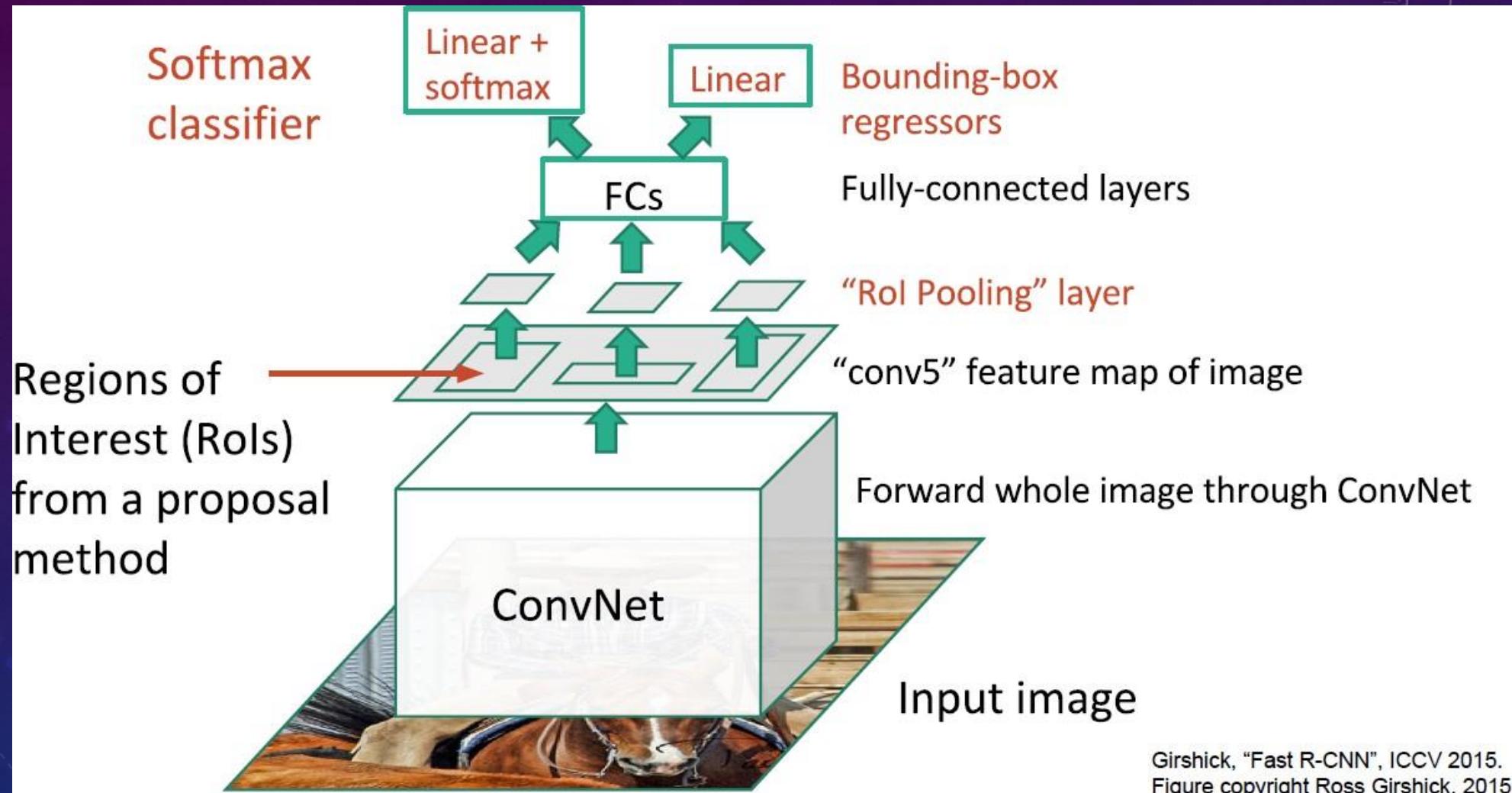
- Use fully convolutional network to extract features
- Extract region of interest from feature map and use region of interest ( $RoI$ ) pooling layer extracts a fixed-length feature vector from the feature map.
- A sequence of fully connected ( $fc$ ) layers that finally branch into two sibling output layers: one that produces softmax probability estimates over  $K$  object classes plus a “background” class and another layer that outputs four real-valued numbers for each of the  $K$  object classes.



**Problem:**

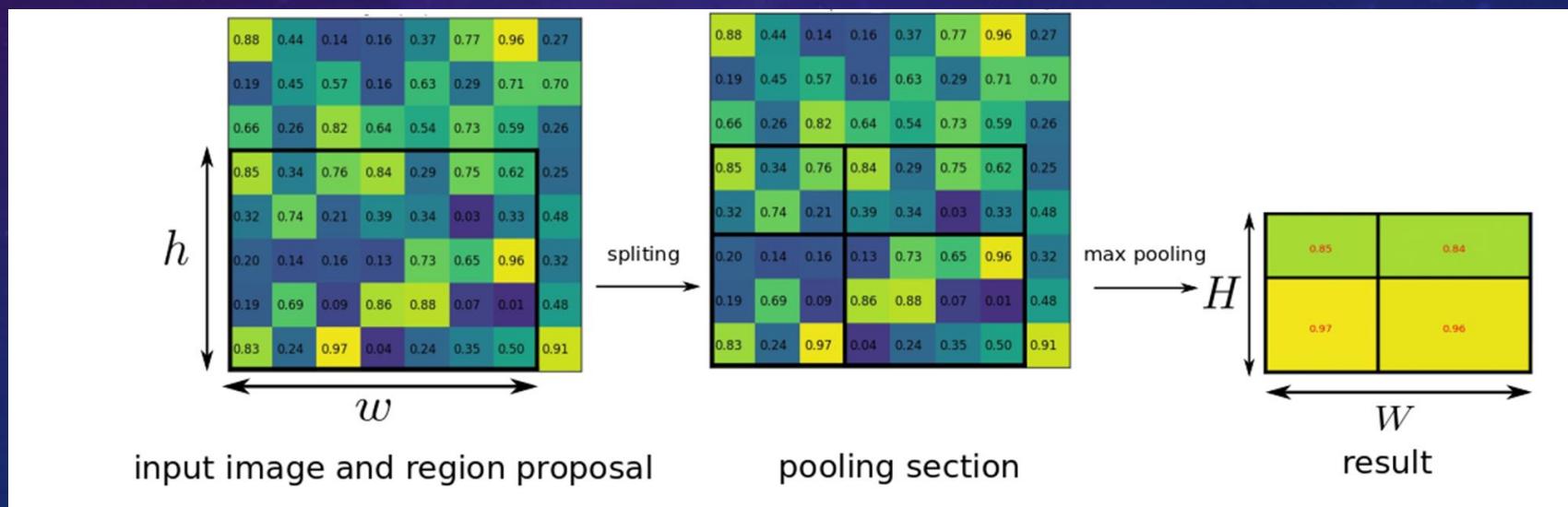
- ✓ Use selective search still slow
- ✓ Not end-to-end training

# FAST R-CNN



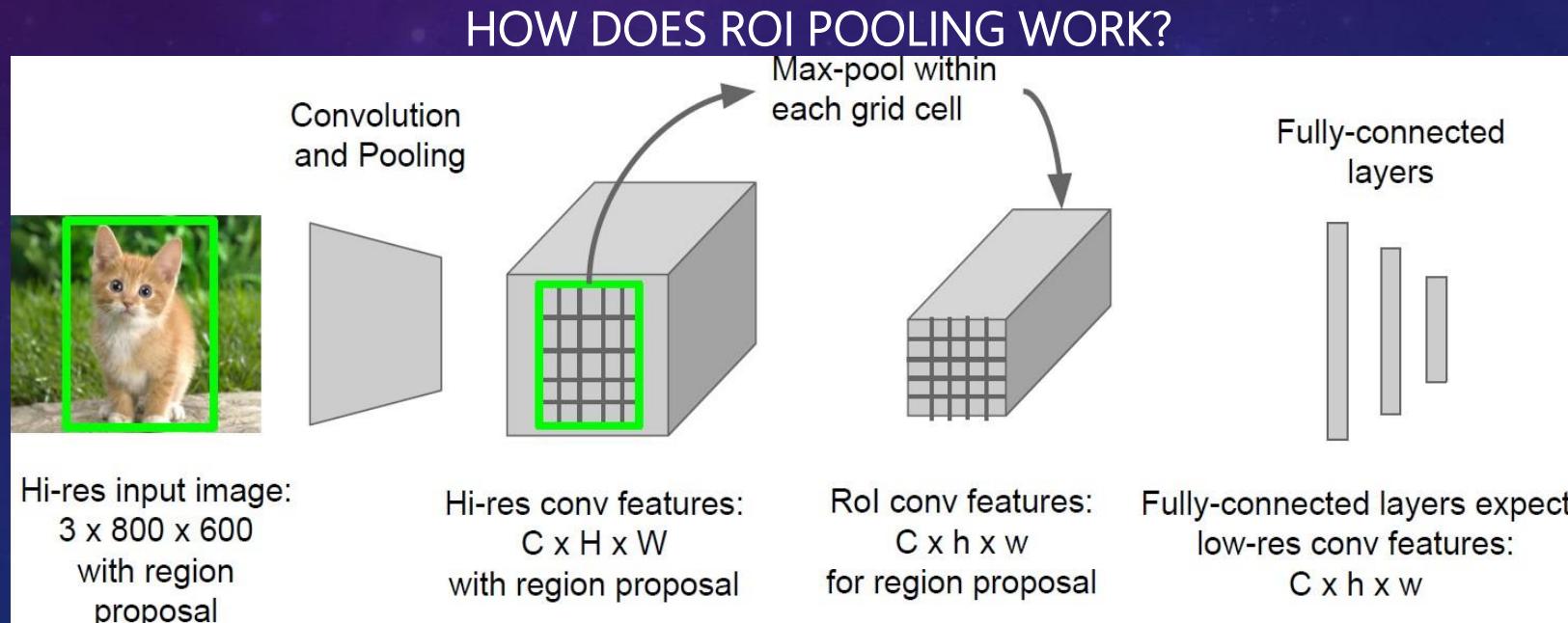
# What is the ROI pooling layer?

- Suppose we obtain the region proposal (left) with  $h \times w$ , and we would like to have an output (right) of  $H \times W$  sizes of output layer after pooling. Then, the area for each pooling area (middle) =  $h/H \times w/W$ .
- In the example below, with input ROI of  $5 \times 7$ , and output of  $2 \times 2$ , the area for each pooling area is  $2 \times 3$  or  $3 \times 3$  after rounding.
- The maximum value within the pooling window is taken as output value for each grid which is the same idea of conventional max pooling layer.



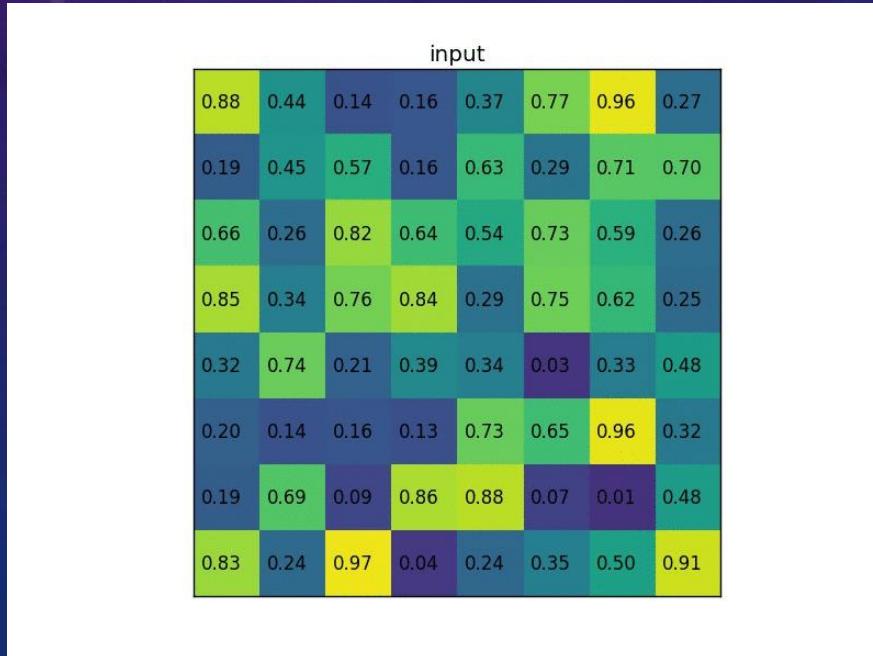
# How does ROI pooling work?

- ROI max pooling works by dividing the  $\mathbf{h} \times \mathbf{w}$  ROI window into an  $\mathbf{H} \times \mathbf{W}$  grid of sub-windows of approximate size  $\mathbf{h}/\mathbf{H} \times \mathbf{w}/\mathbf{W}$  and then max-pooling the values in each sub-window into the corresponding output grid cell, i.e. the “max-pooling” filter size becomes  $\mathbf{h}/\mathbf{H} \times \mathbf{w}/\mathbf{W}$ . Pooling is applied how it is normally applied, independently to each feature map channel.



# ROI pooling Summary

- ROI pooling summary:
- It's used for object detection tasks
- It allows us to reuse the feature map from the convolutional network
- It can significantly speed up both train and test time
- It allows to train object detection systems in an end-to-end manner



# Problems of Fast R-CNN

- Out-of-network region proposals are the test-time computational bottleneck
- Is it fast enough??

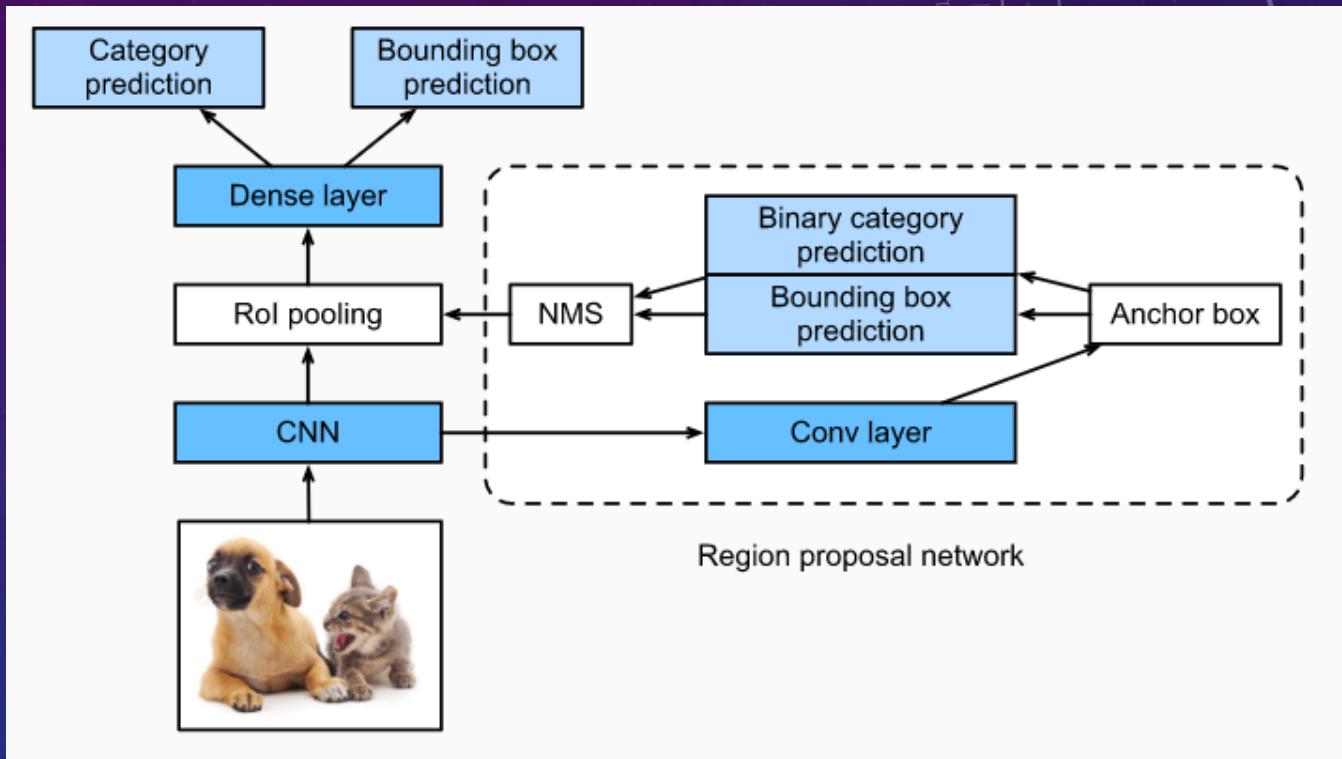


# Faster R-CNN

- Faster R-CNN architecture
- Faster R-CNN(RPN + fast R-CNN)
- How to train faster R-CNN
- Training goal : share features
- How RPN (region proposal networks) works
- Region Proposal Network
- RPN(fully convolutional network)

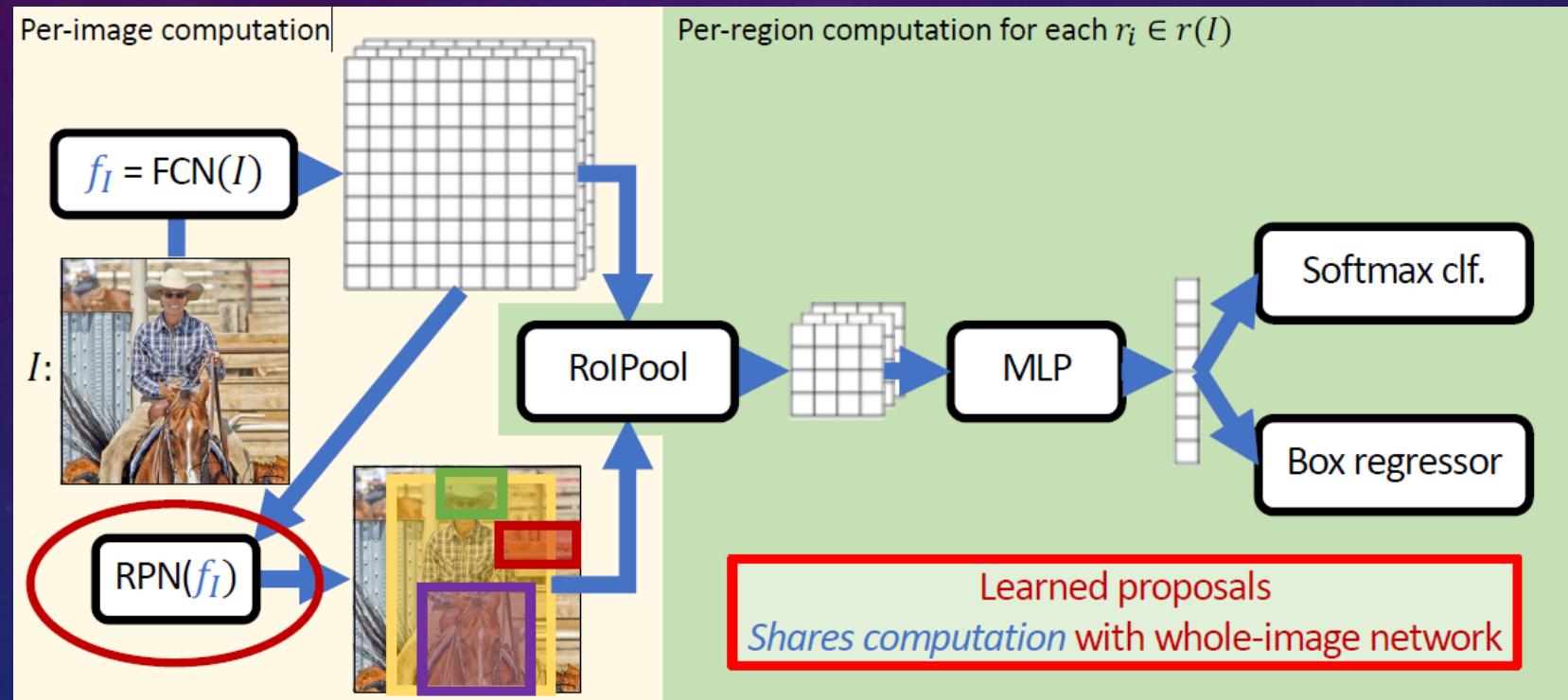
# Faster R-CNN

- In order to obtain precise object detection results, Fast R-CNN generally requires that many proposed regions be generated in selective search. Faster R-CNN replaces selective search with a region proposal network. This reduces the number of proposed regions generated, while ensuring precise object detection.



# Faster R-CNN

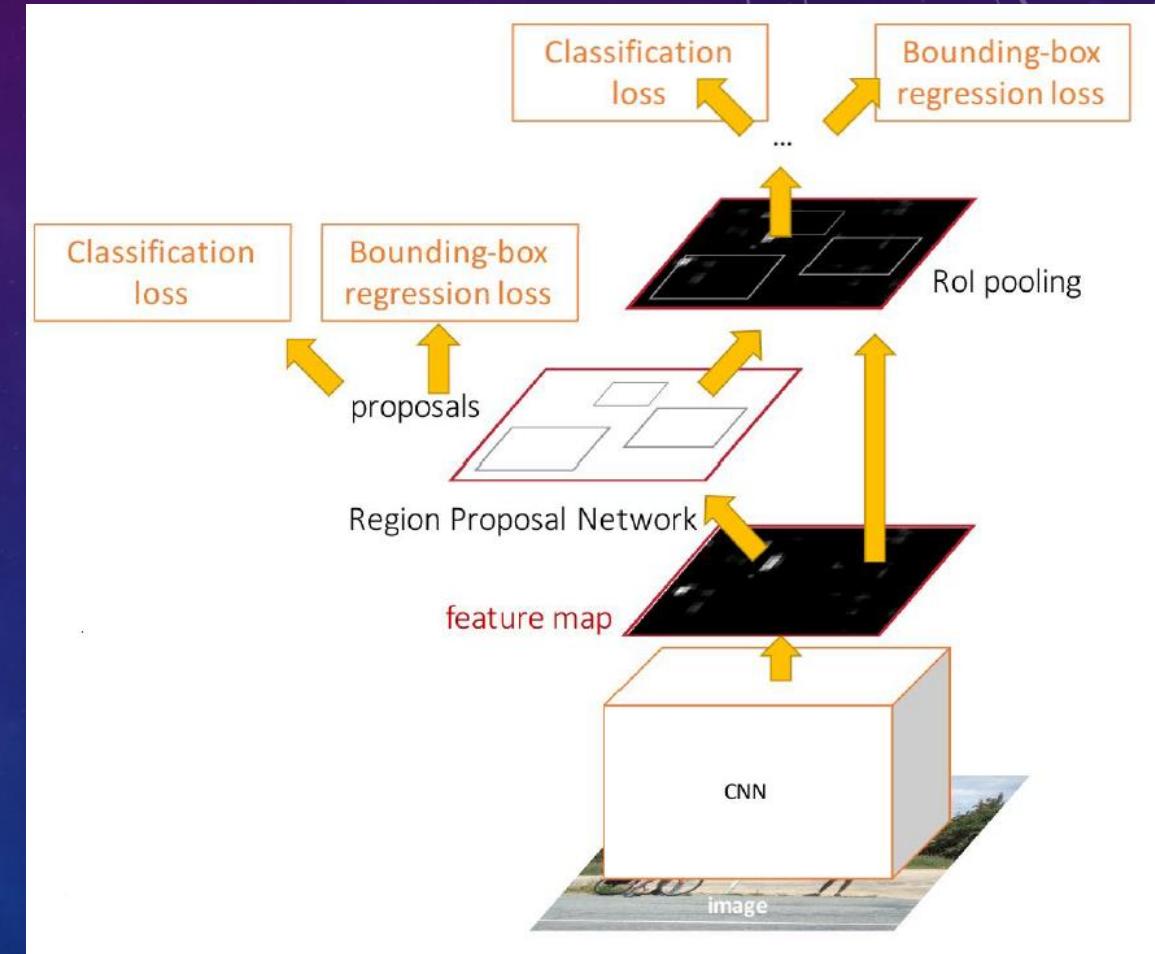
- Replace selective search with Region Proposal Network
- End-to-end training



# Faster R-CNN

- Jointly train the whole network with 2 losses
  - RPN classifies object/non-object
  - RPN regresses bbox coordinates
- Results show the final classification score and bbox coordinates.

$$L = L_{cls} + L_{reg}$$



# Loss Function

$$L_{cls} = -\frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*)$$

- Here i is the index of the anchor in the mini-batch.
- $L_{cls}(p_i, p_i^*)$ : log loss over two classes (object vs non-object)
- $p_i$ : output score from the classification branch for anchor i
- $p_i^*$ : groundtruth label (1 or 0).

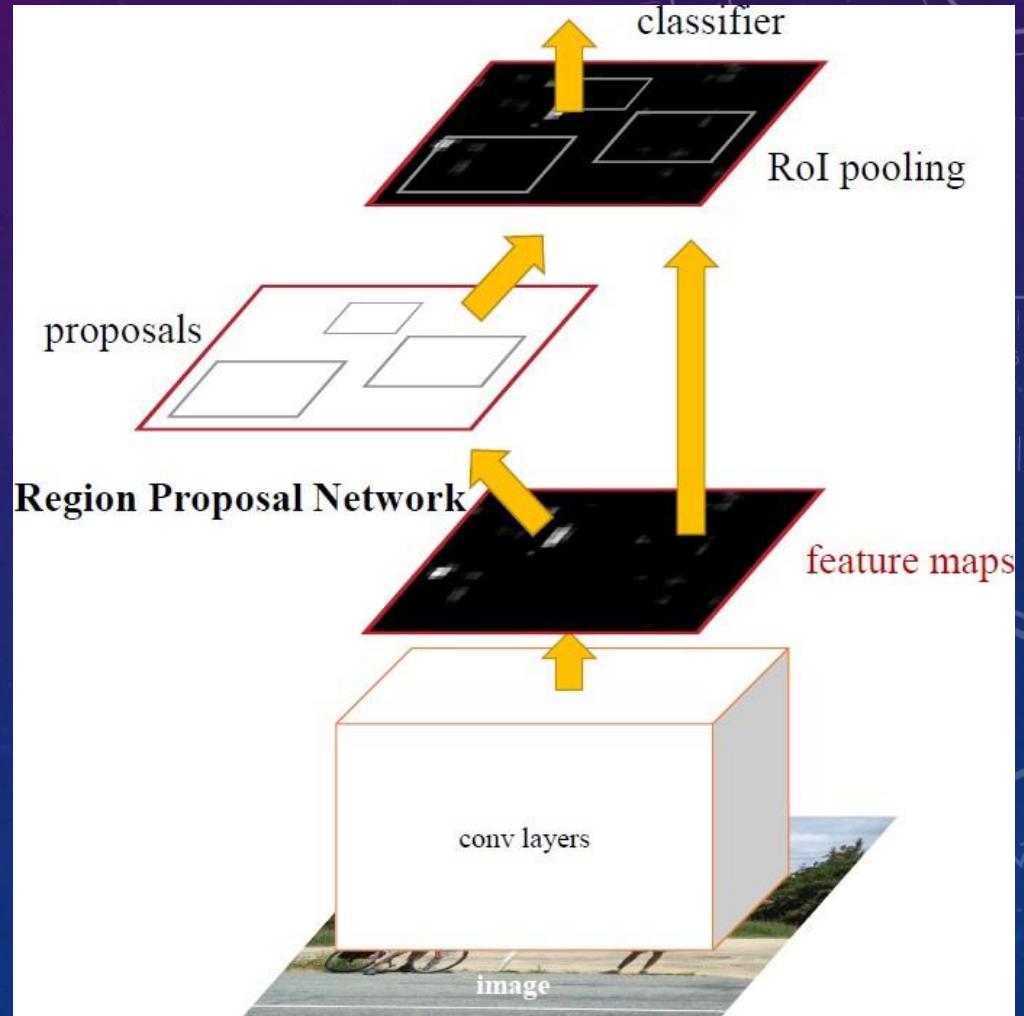
# Loss Function

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

- The regression loss  $L_{reg}(t_i, t_i^*)$  is activated only if the anchor actually contains an object.
- $t_i$ : The output prediction of the regression layer and consists of 4 variables  $[t_x, t_y, t_w, t_h]$ .
- $t_i^*$ :  $t_x^* = (x^* - x_a)/w_a$ ,  $t_y^* = (y^* - y_a)/h_a$ ,  $t_w^* = \log(w^*/w_a)$ ,  $t_h^* = \log(h^*/h_a)$

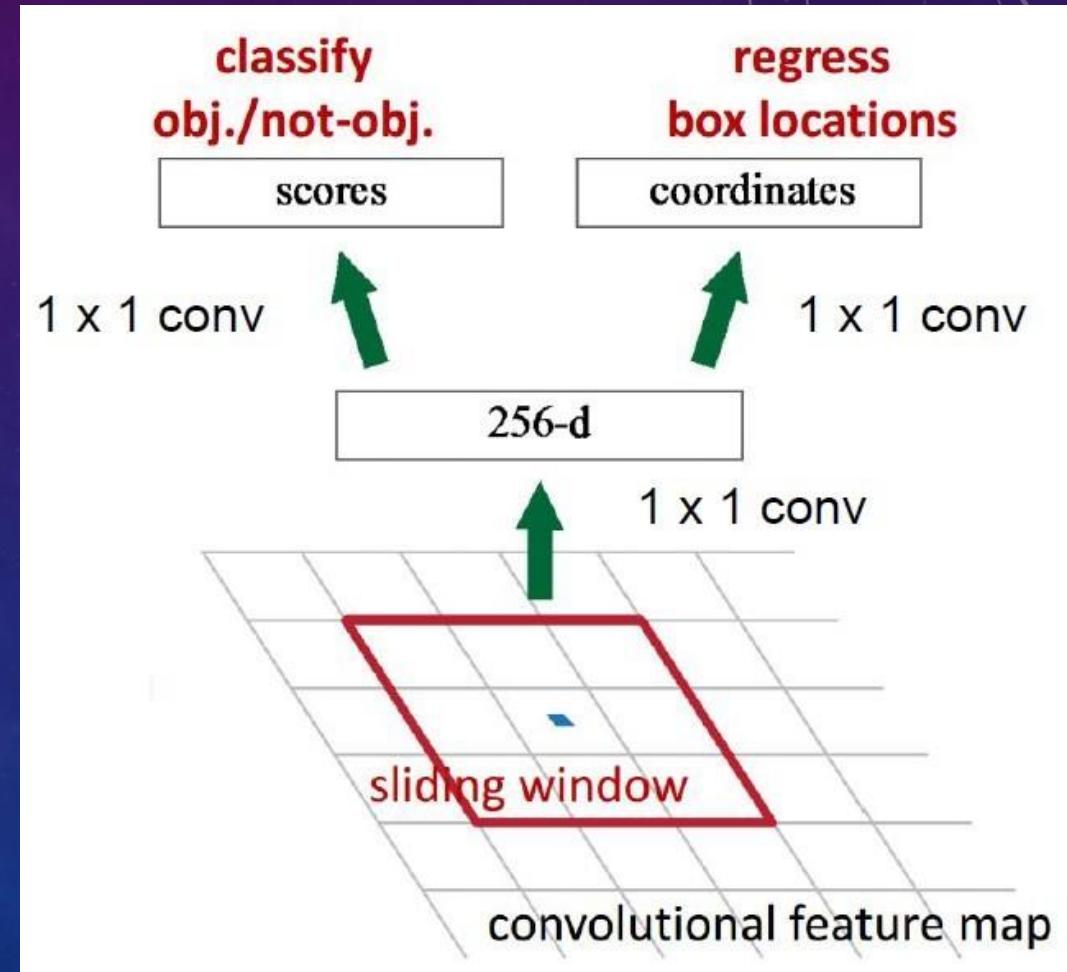
# Faster R-CNN (RPN + Fast R-CNN)

- Insert a Region Proposal Network (RPN) after the last convolutional layer
- RPN trained to produce region proposals directly; no need for external region proposals
- After RPN, use ROI Pooling and an upstream classifier and bbox regressor just like Fast R-CNN



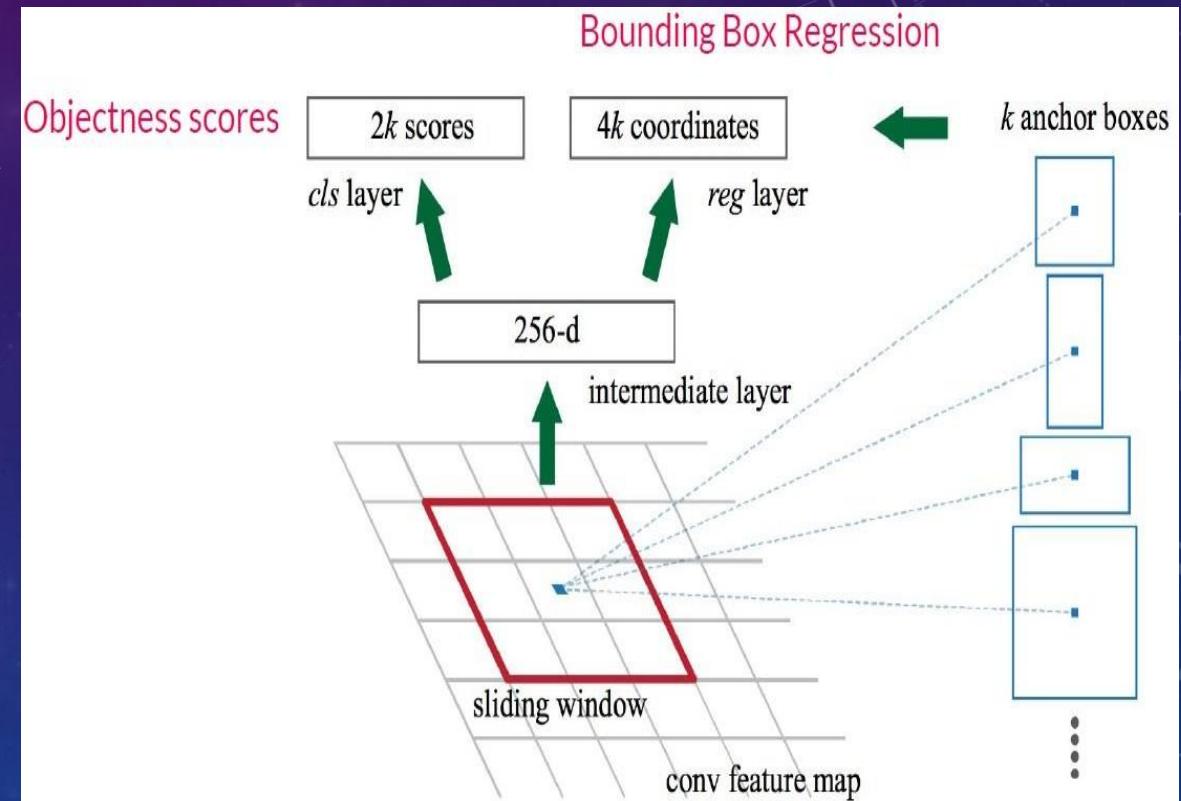
# Region Proposal Network

- Slide a small window on the feature map
- Build a small network for
  - Classifying object or not-object
  - Regressing bbox locations
- Position of the sliding window provides localization information with reference to the image
- Box regression provides finer localization information with reference to this sliding window



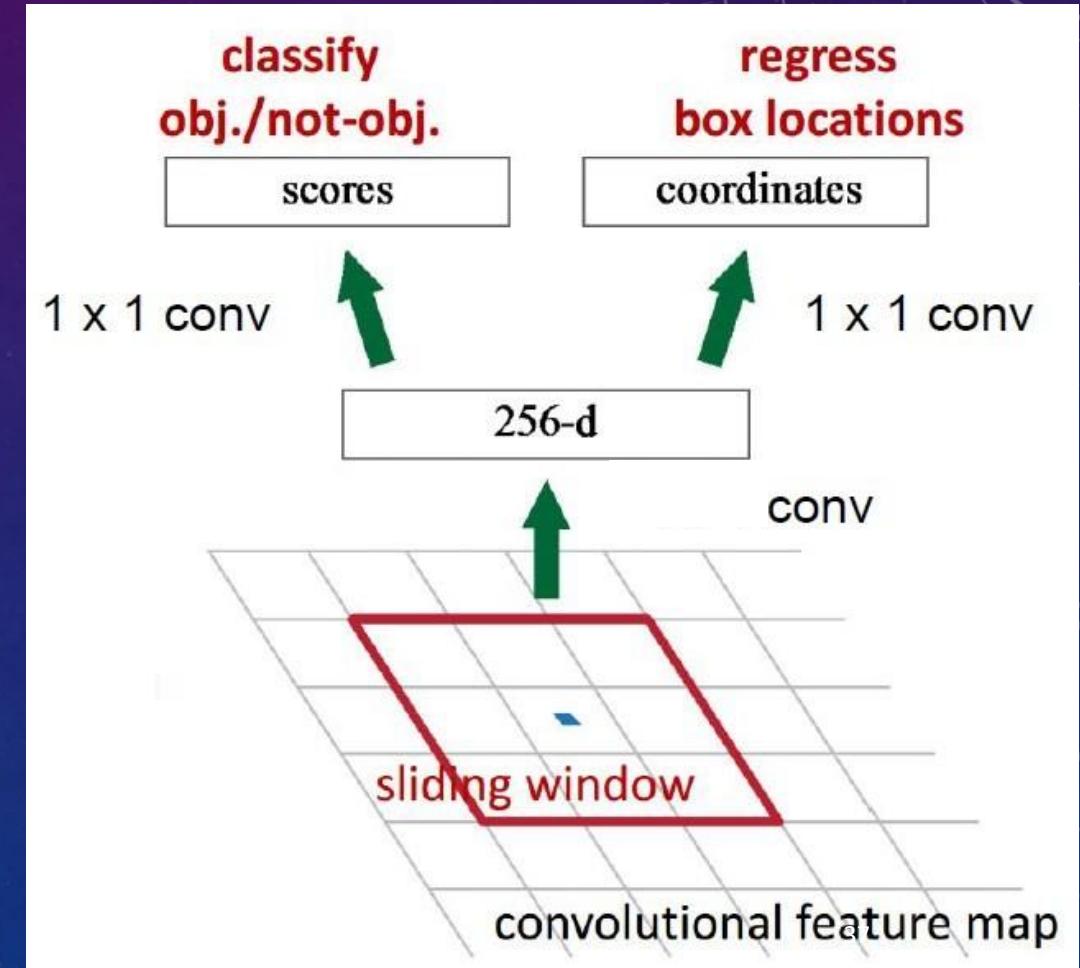
# Region Proposal Network

- Use K anchor boxes at each location
- Anchors are translation invariant: use the same ones at every location
- Regression gives offsets from anchor boxes
- Classification gives the probability that each (regressed) anchor shows an object



# RPN (Fully Convolutional Network)

- Intermediate Layer – 256(or 512) 3x3 filter, stride 1, padding 1
- Cls layer – 18(9x2) 1x1 filter, stride 1, padding 0
- Reg layer – 36(9x4) 1x1 filter, stride 1, padding 0



# Anchors As References

- Anchors: pre-defined reference boxes
- Multi-scale/size anchors:
  - Multiple anchors are used at each position:  
3 scale(128x128, 256x256, 512x512) and 3 aspect ratios(2:1, 1:1, 1:2)  
yield 9 anchors
  - Each anchor has its own prediction function
  - Single-scale features, multi-scale predictions

# Is Faster RCNN enough?

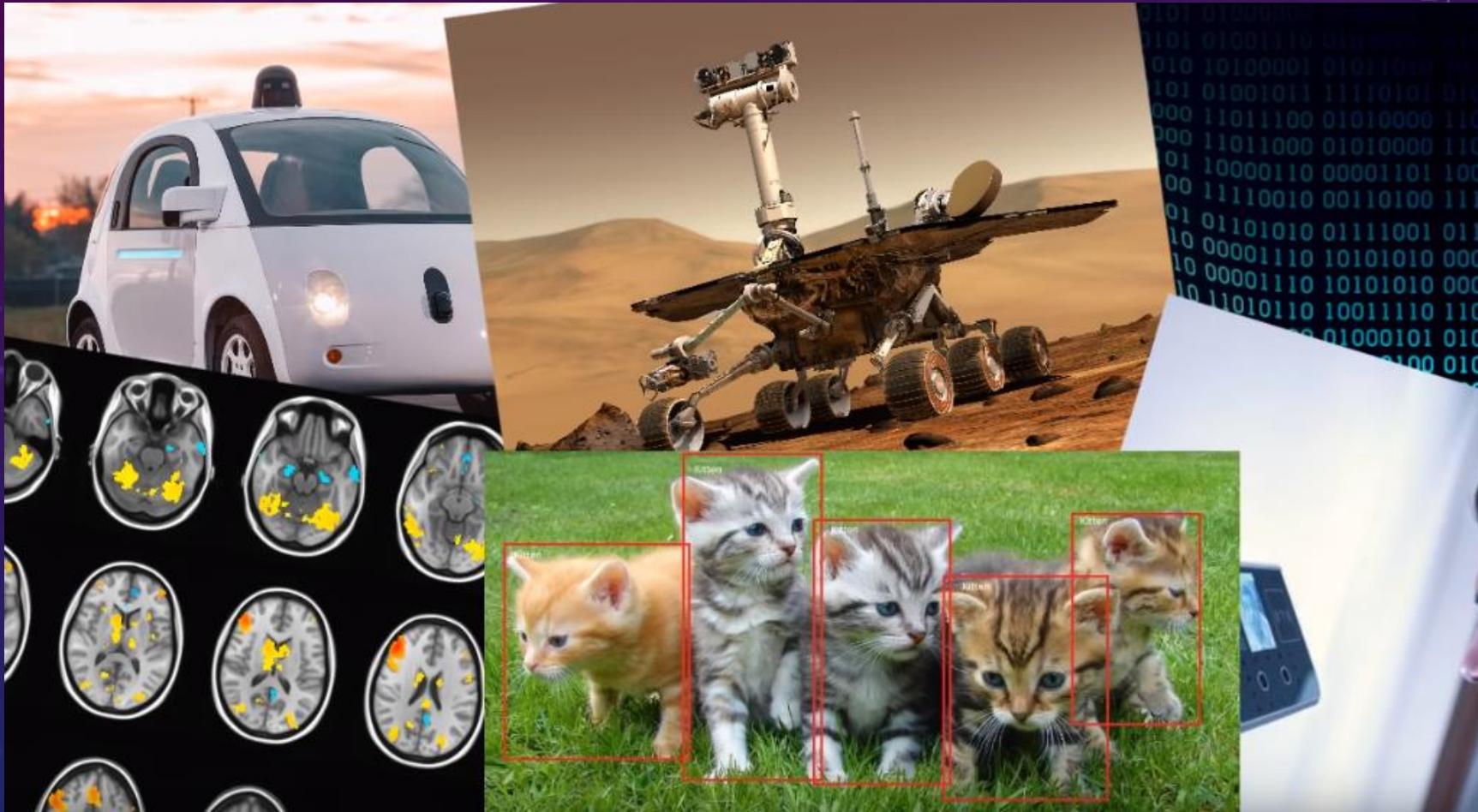
- RoI Pooling has some quantization operations
- These quantizations introduce misalignments between the RoI and the extracted features
- While this may not impact classification, it can make a negative effect on predicting bbox

See Coding Manual, Chapter 4, Page 2.

# Mask R-CNN

- Abstract
- Semantic Segmentation
- Instance Segmentation
- General Architecture
- Mask R-CNN RoI Alignment
- Mask R-CNN Network architecture

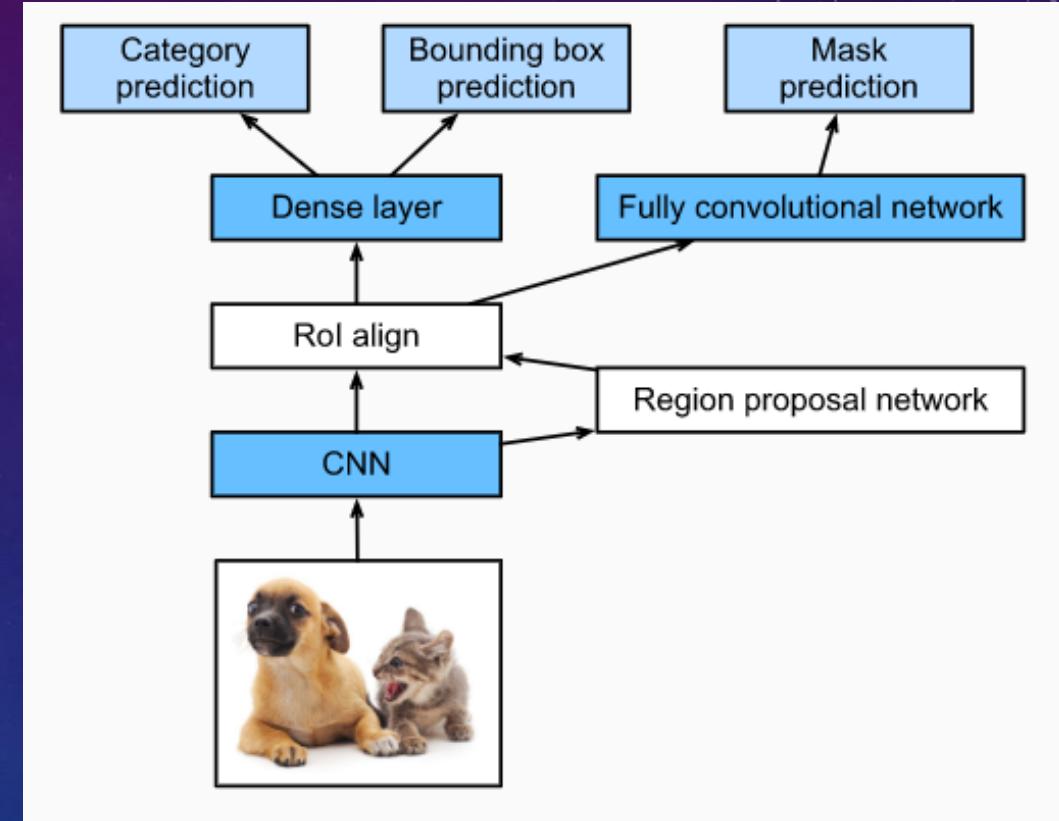
# Mask R-CNN



<https://www.youtube.com/watch?v=4tkgOzQ9yyo> [9:34]

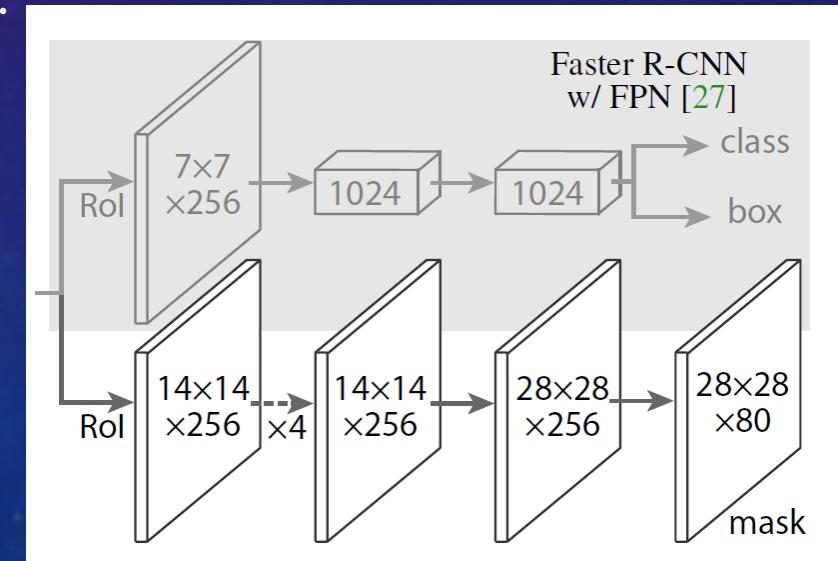
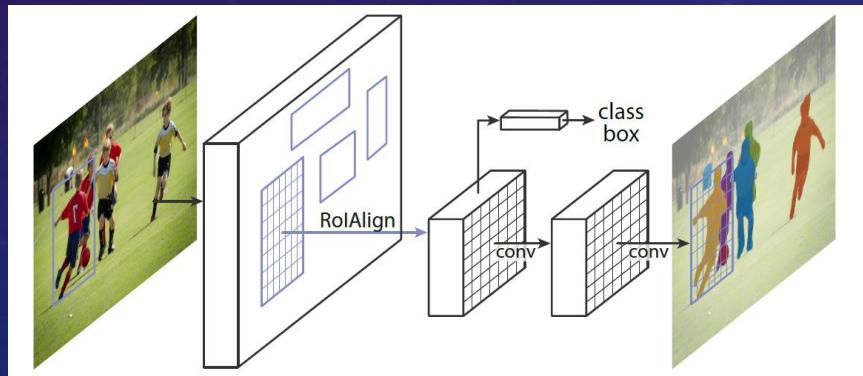
# Mask R-CNN

If training data is labeled with the pixel-level positions of each object in an image, a Mask R-CNN model can effectively use these detailed labels to further improve the precision of object detection.



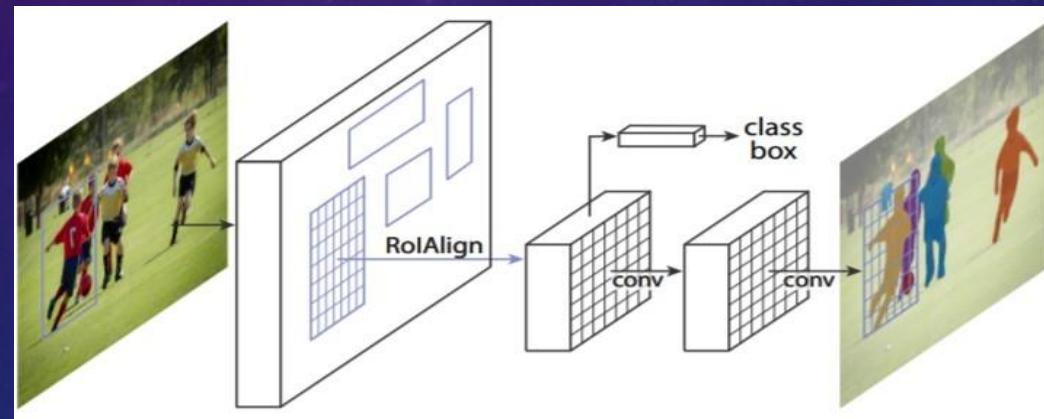
# Mask R-CNN

- Mask R-CNN, extends Faster R-CNN by adding a branch for predicting segmentation masks on each Region of Interest (RoI), in parallel with the existing branch for classification and bounding box regression
- The mask branch is a small FCN applied to each RoI, predicting a segmentation mask in a pixel-to-pixel manner.
- ROI Alignment: use bilinear interpolation to compute the exact values of the input features at four regularly sampled locations in each ROI bin.



# Abstract

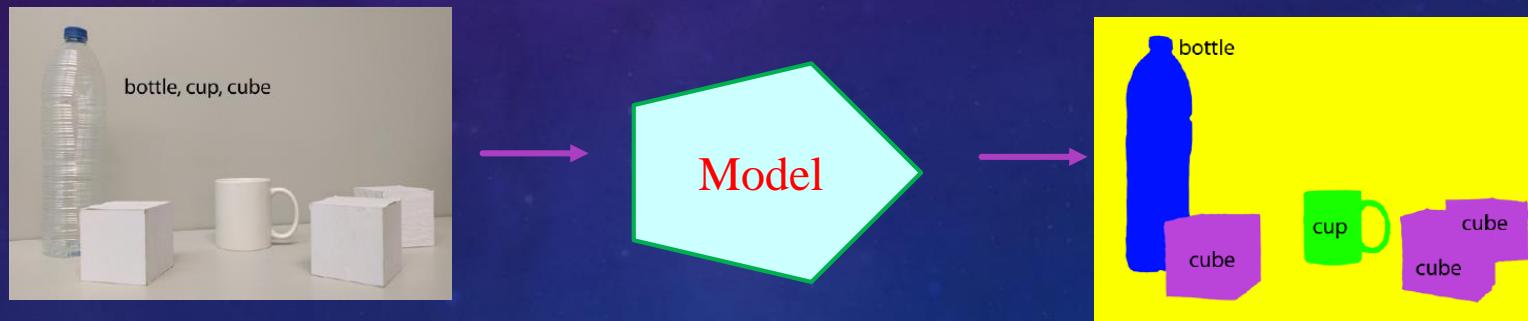
- Our approach efficiently detects objects in an image while simultaneously generating a high-quality segmentation mask for each instance.
- Mask R-CNN extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition.



Mask R-CNN is simple to train and adds only a small overhead to Faster R-CNN, running at 5fps.

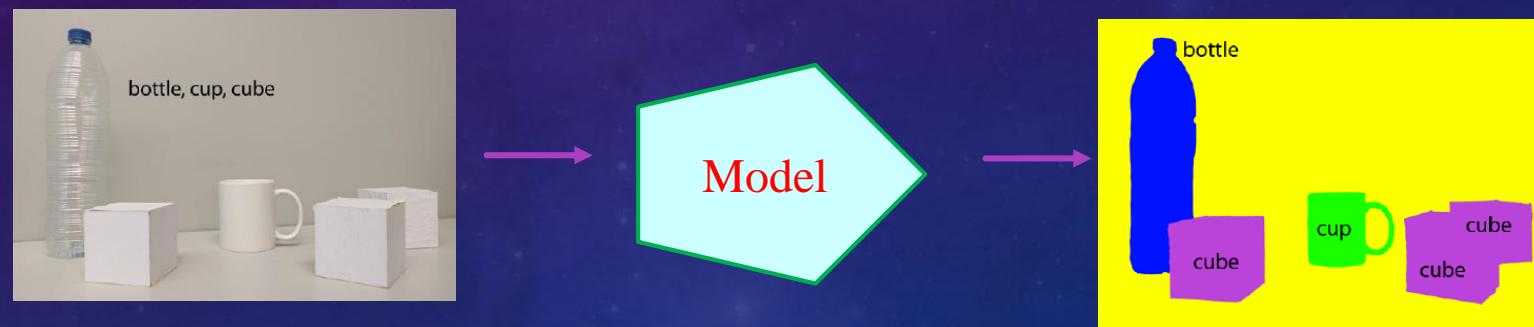
# Semantic Segmentation

- Semantic segmentation refers to the process of linking each pixel in an image to a class label.
- These labels could include a person, car, flower, piece of furniture, etc., just to mention a few. We can think of semantic segmentation as image classification at a pixel level.



# Instance Segmentation

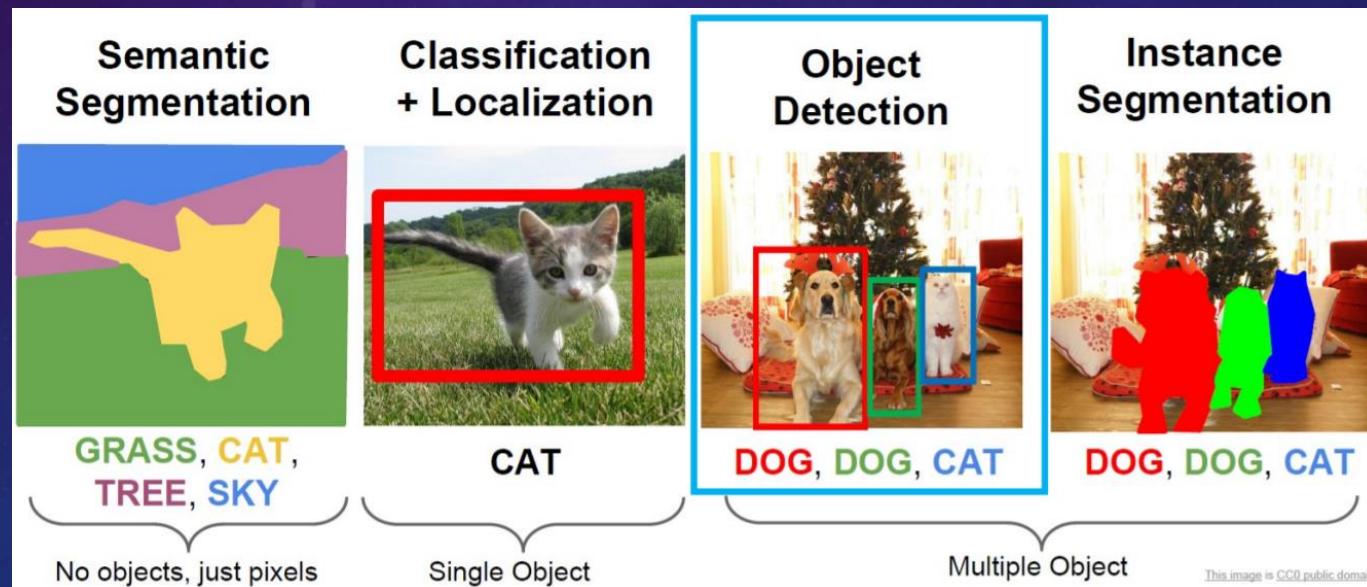
- For each pixel  $I$ , predict semantic label  $l$  and instance id  $z$
- Instance segmentation = semantic segmentation + distinguished instances per class
- Closer to real scene understanding



# Instance Segmentation

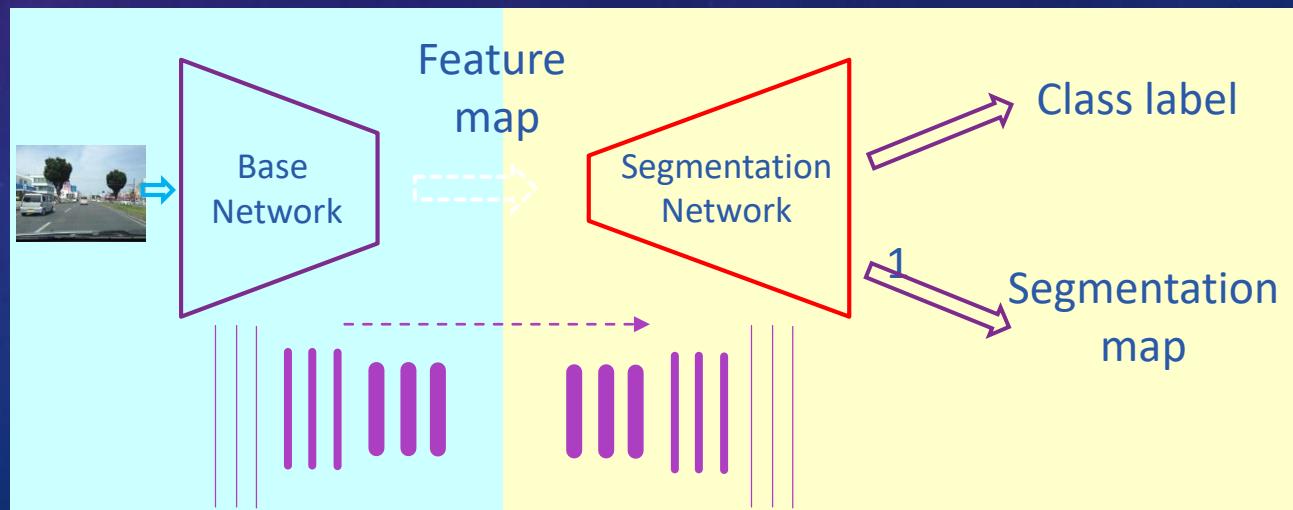
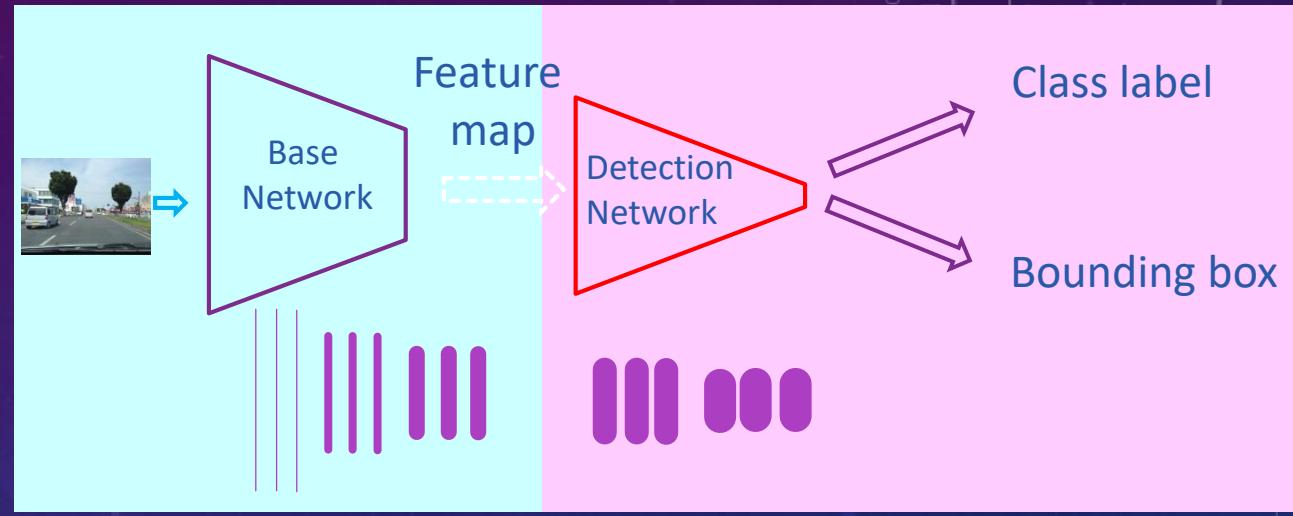
Instance segmentation combines elements from the classical computer vision tasks of object detection and segmentation.

- Object detection is to classify individual objects and localize each with a bounding box
- Semantic segmentation, where the goal is to classify each pixel into a fixed set of categories without differentiating object instances.

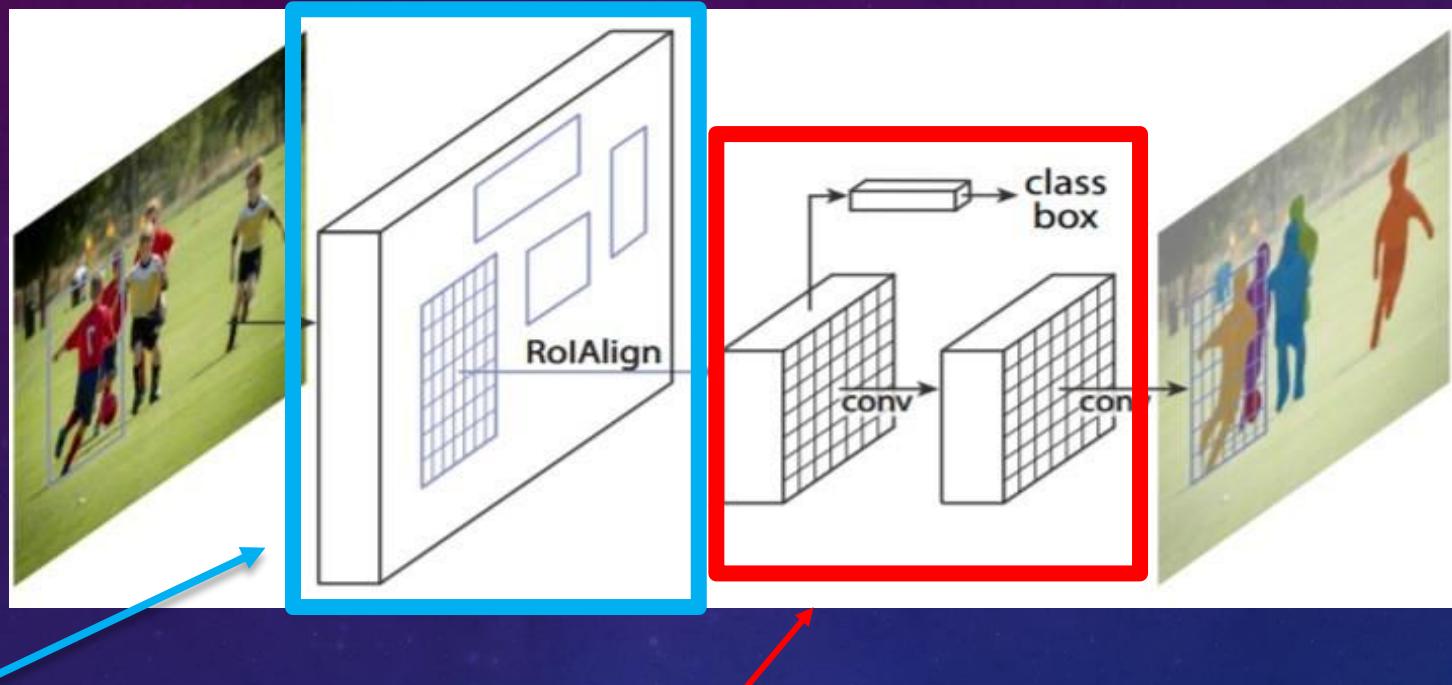


# General Architecture

- Object Detection
  - Height and width decrease
  - Channel increase
  - Base network is pre-trained model
  - Output is a list of scalars or vectors
- Segmentation
  - Height and width decrease, then increase again.
  - Channel increases and then decreases
  - Base network is pre-trained model
  - Outputs are map and a lists of scalar.



# Mask R-CNN Contribution

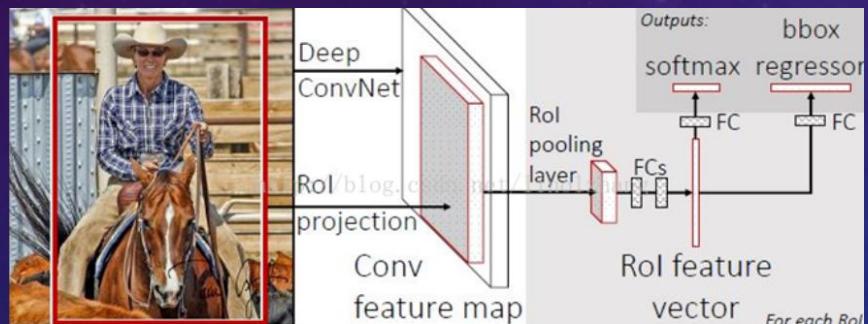


1. To fix the mis-alignment, we propose a simple, quantization-free layer, called **RoI Alignment**, that faithfully preserves exact spatial locations.
2. Adding a branch for predicting segmentation masks on each Region of Interest (RoI), in parallel with the existing branch for classification and bounding box regression.

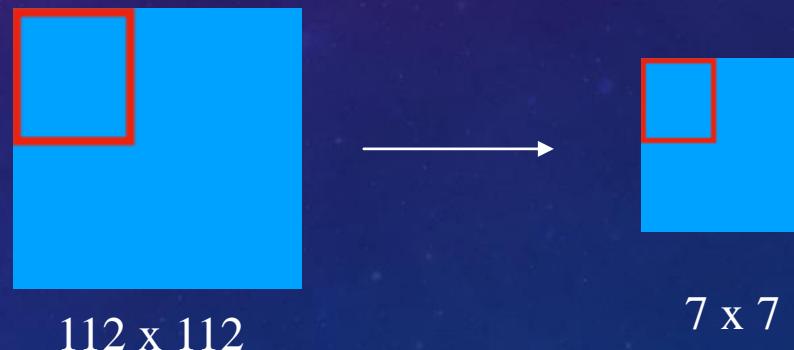
# Mask R-CNN

We propose an ROI Alignment layer that removes the harsh quantization of ROI Pooling, properly aligning the extracted features with the input. ROI Alignment improves mask accuracy by relative 10% to 50%, showing bigger gains under stricter localization metrics

- Previous works - RoIPool



Max-pooling (Input is rounded off)



# Faster R-CNN ROI Pooling

0.88	0.44	0.14	0.16	0.37	0.77	0.96	0.27
0.19	0.45	0.57	0.16	0.63	0.29	0.71	0.70
0.66	0.26	0.82	0.64	0.54	0.73	0.59	0.26
0.85	0.34	0.76	0.84	0.29	0.75	0.62	0.25
0.32	0.74	0.21	0.39	0.34	0.03	0.33	0.48
0.20	0.14	0.16	0.13	0.73	0.65	0.96	0.32
0.19	0.69	0.09	0.86	0.88	0.07	0.01	0.48
0.83	0.24	0.97	0.04	0.24	0.35	0.50	0.91

Input activation

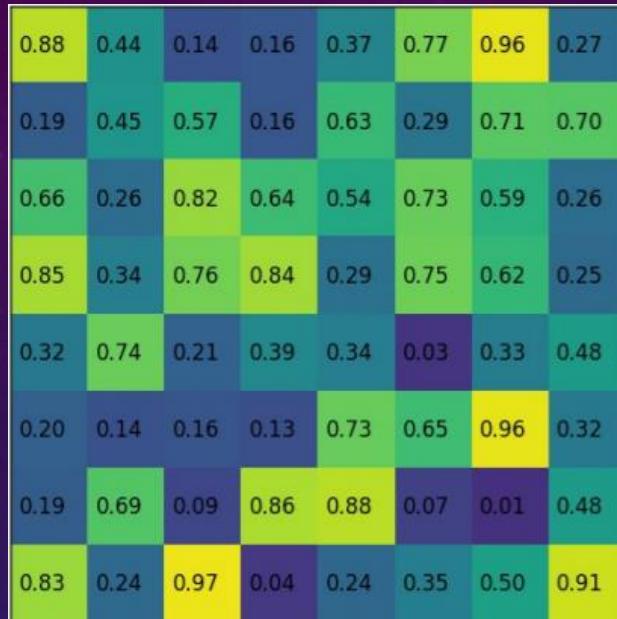
0.88	0.44	0.14	0.16	0.37	0.77	0.96	0.27
0.19	0.45	0.57	0.16	0.63	0.29	0.71	0.70
0.66	0.26	0.82	0.64	0.54	0.73	0.59	0.26
0.85	0.34	0.76	0.84	0.29	0.75	0.62	0.25
0.32	0.74	0.21	0.39	0.34	0.03	0.33	0.48
0.20	0.14	0.16	0.13	0.73	0.65	0.96	0.32
0.19	0.69	0.09	0.86	0.88	0.07	0.01	0.48
0.83	0.24	0.97	0.04	0.24	0.35	0.50	0.91

Region projection and pooling sections

0.85	0.84
0.97	0.96

Max pooling output

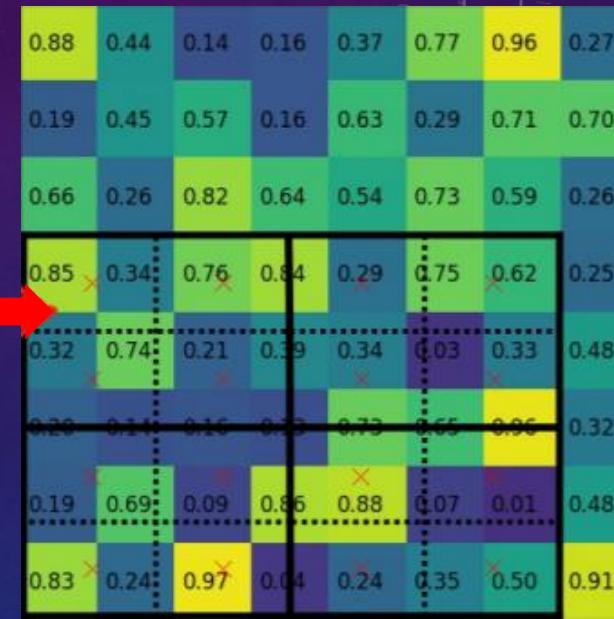
# Mask R-CNN RoI Alignment



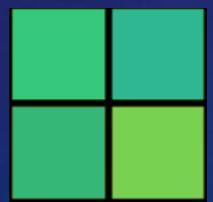
Input activation



Region projection and pooling sections



Sampling locations

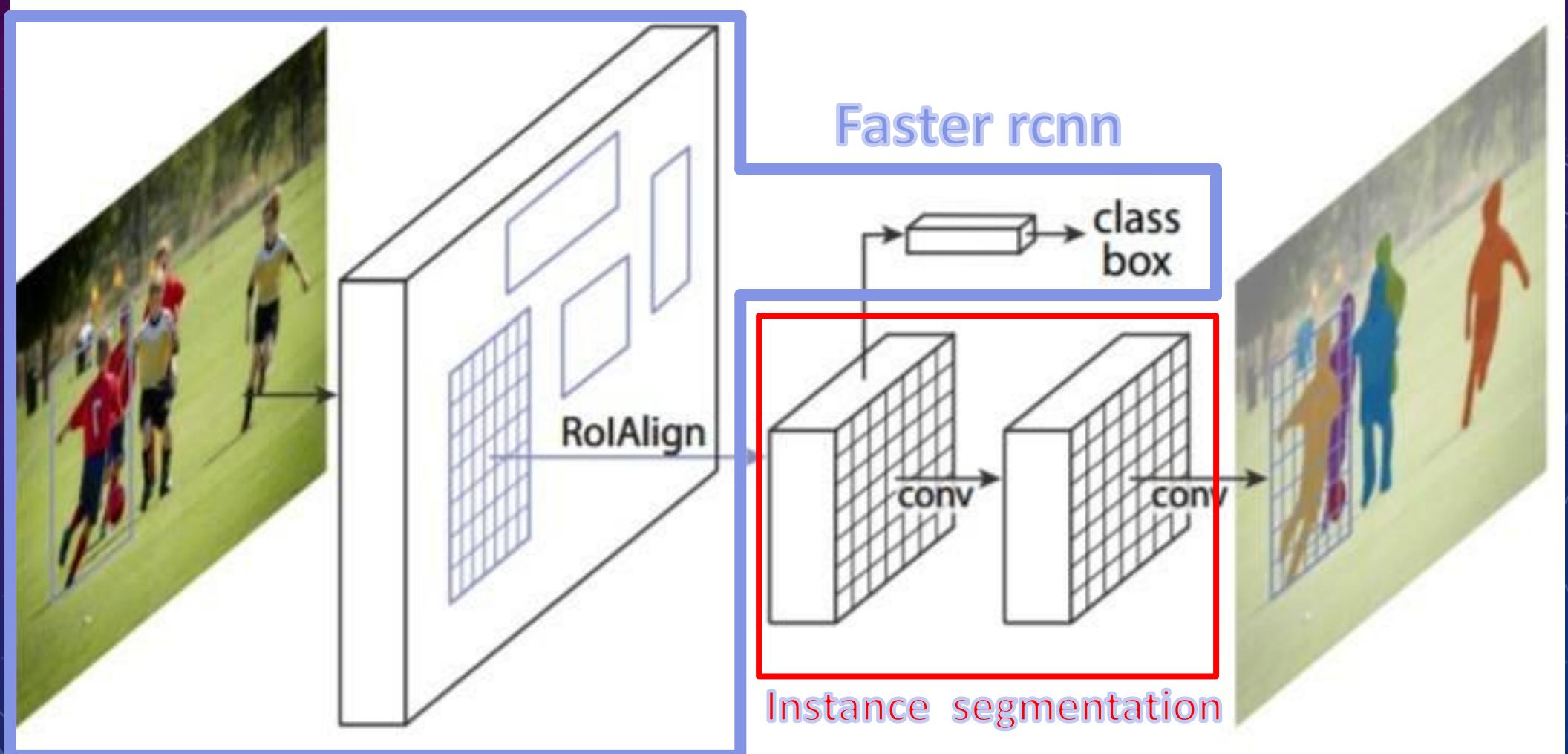


Max pooling output



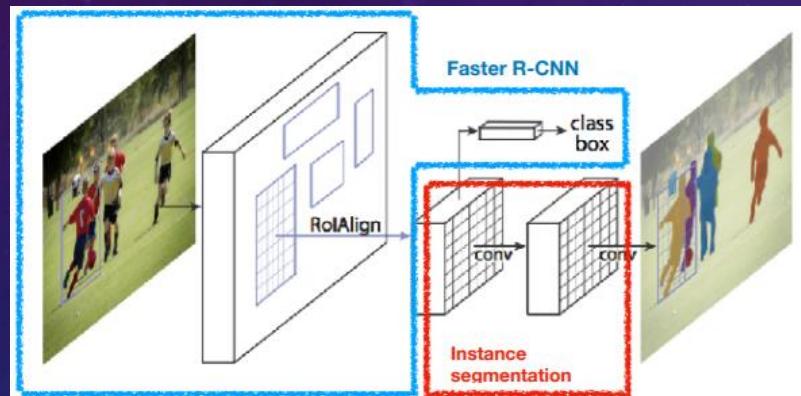
Bilinear interpolated values

# Mask R-CNN Architecture

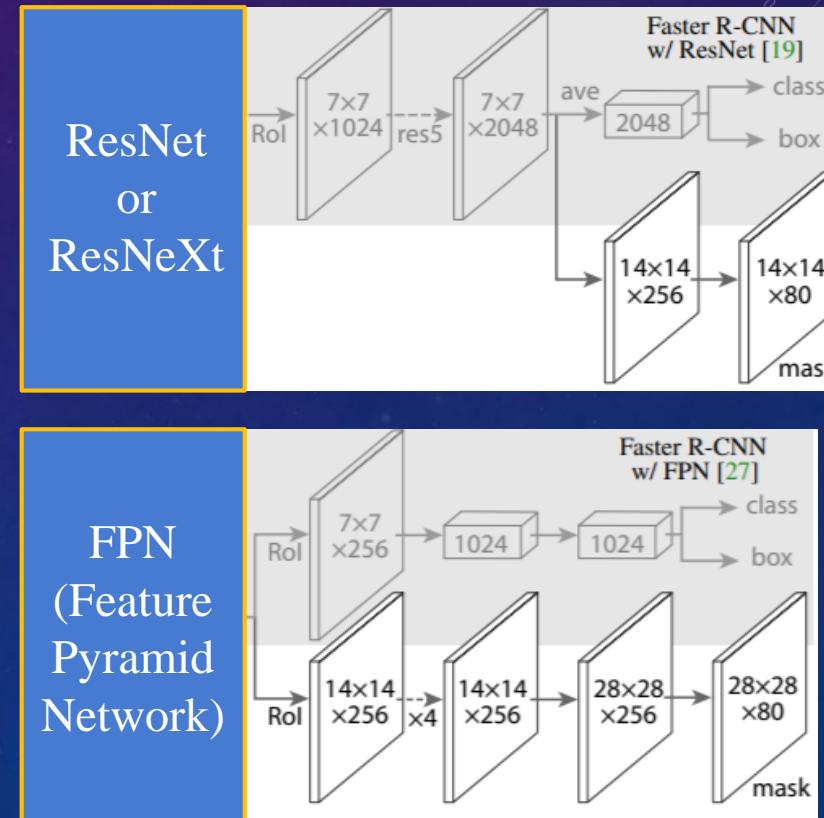


# Mask R-CNN Network Architecture

- Mask R-CNN extends Faster R-CNN by adding a branch for predicting segmentation masks on each Region of Interest (RoI), in parallel with the existing branch for classification and bounding box regression



Faster R-CNN + Instance segmentation



FPN  
(Feature  
Pyramid  
Network)

# Loss Function

$$L = L_{cls} + L_{reg} + L_{mask}$$

- $L_{cls}, L_{reg}$  is the same in the faster R-CNN

# Loss Function

$$L_{mask} = -\frac{1}{m^2} \sum_{1 \leq i,j \leq m} [y_{i,j} \log \hat{y}_{i,j}^k + (1 - y_{i,j}) \log(1 - \hat{y}_{i,j}^k)]$$

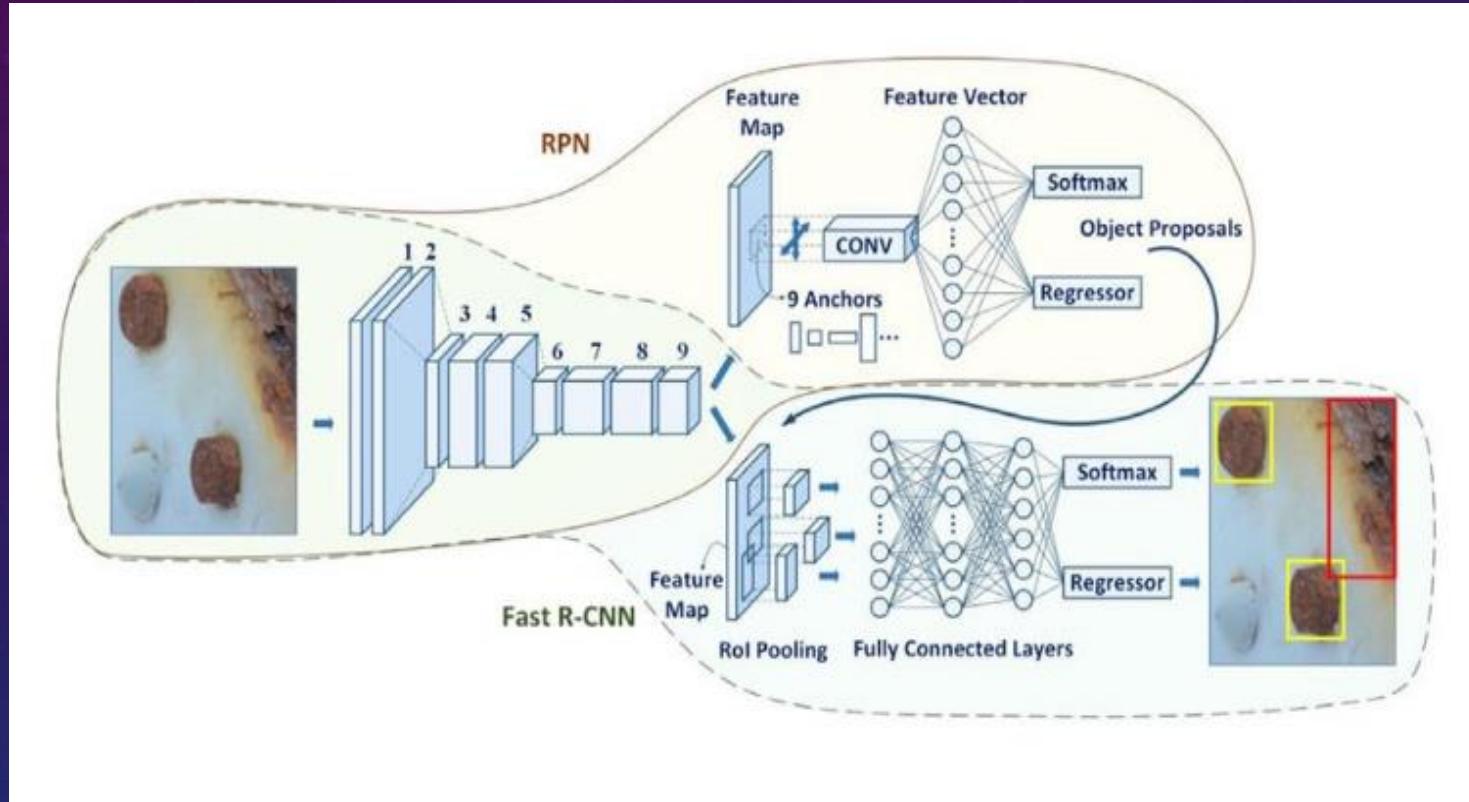
- The mask branch has a  $K \times m \times m$  - dimensional output for each ROI and each class;  $K$  classes in total.
- $L_{mask}$  is defined as the average binary cross-entropy loss, only including  $k$ -th mask if the region is associated with the ground truth class  $k$

# Summary

- **R-CNN** model selects several proposed regions and uses a CNN to perform forward computation and extract the features from each proposed region. It then uses these features to predict the categories and bounding boxes of proposed regions.
- **Fast R-CNN** improves on the R-CNN by only performing CNN forward computation on the image as a whole. It introduces an ROI pooling layer to extract features of the same shape from ROIs of different shapes.
- **Faster R-CNN** replaces the selective search used in Fast R-CNN with a region proposal network. This reduces the number of proposed regions generated, while ensuring precise object detection.
- **Mask R-CNN** uses the same basic structure as Faster R-CNN, but adds a fully convolution layer to help locate objects at the pixel level and further improve the precision of object detection.

# Appendix

# Introduction to How Faster R-CNN, Fast R-CNN and R-CNN Works



[https://www.youtube.com/watch?v=v5bFVbQvFRk&list=PLkRkKTC6HZMzp28TxR\\_fJYZ-K8Yu3EQw0&index=1](https://www.youtube.com/watch?v=v5bFVbQvFRk&list=PLkRkKTC6HZMzp28TxR_fJYZ-K8Yu3EQw0&index=1) [8:40]

# Faster R-CNN architecture

1. Introduction

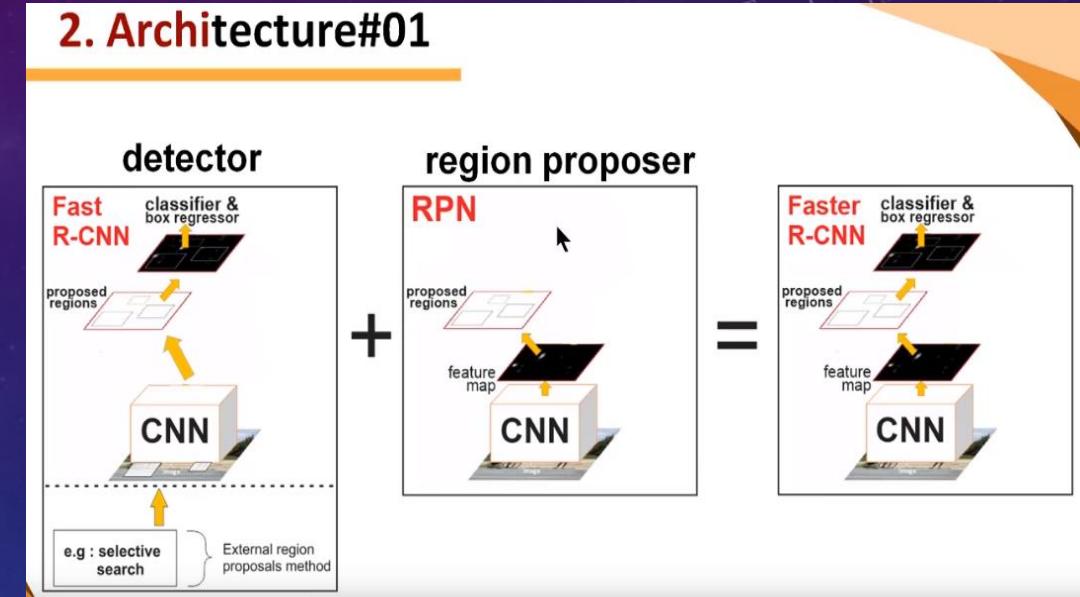
## 2. Architecture

- 2.1 CNN (Convolutional Neural Networks)
- 2.2 RPN (Regional Proposal Networks)
- 2.3 Fast R-CNN (detector)
- 2.4 Putting all together → Faster R-CNN

3. Training Phase (sharing CNN)

4. Experiments and Results

By Ardian Umam



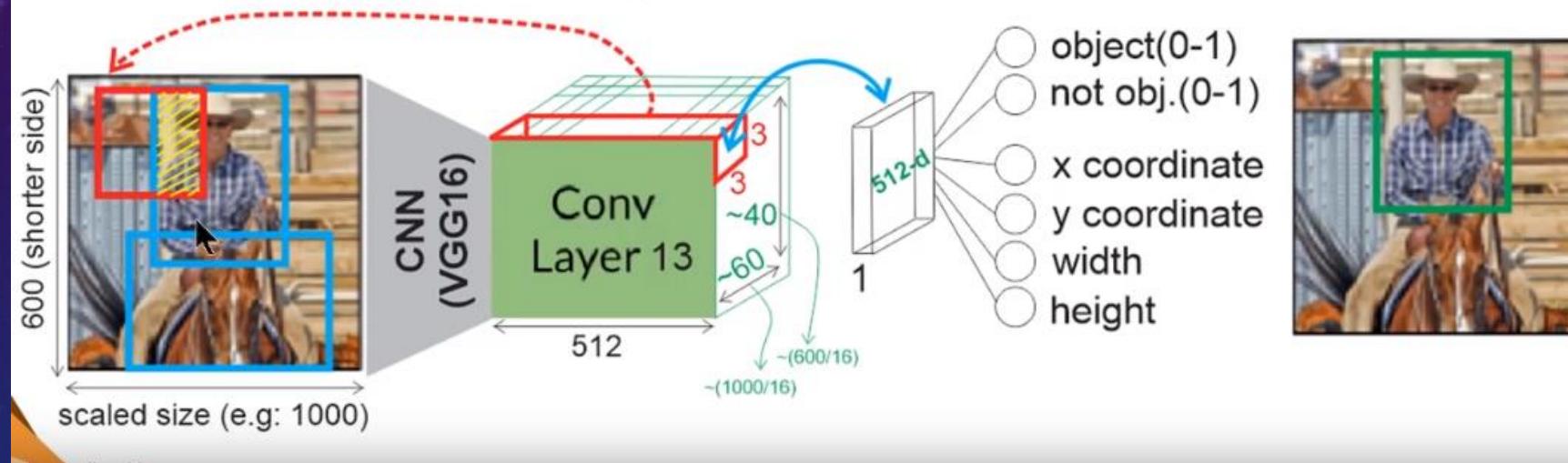
[https://www.youtube.com/watch?v=c1\\_g6tw69bU&list=PLkRkKTC6HZMzp28TxR\\_fJYZ-K8Yu3EQw0&index=2](https://www.youtube.com/watch?v=c1_g6tw69bU&list=PLkRkKTC6HZMzp28TxR_fJYZ-K8Yu3EQw0&index=2) [5:07]

# How RPN (Region Proposal Network) works

## 2. Architecture#06

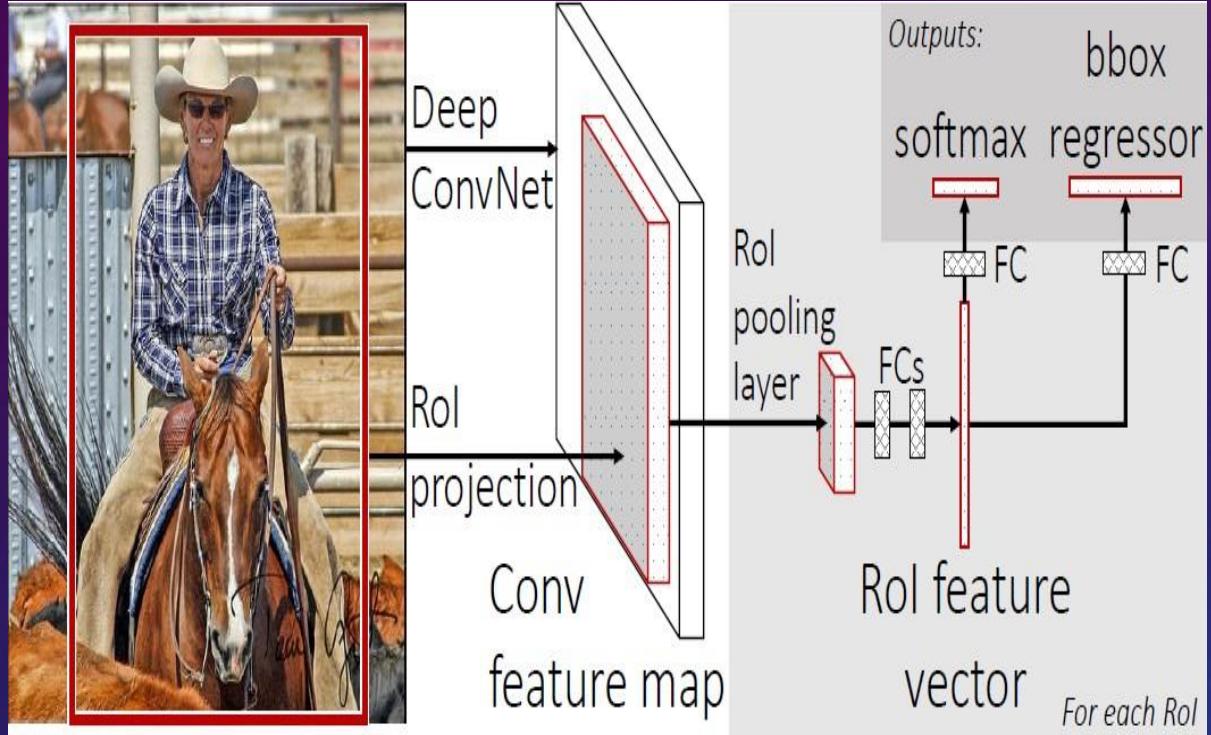
### RPN as reg. proposer

$$\text{IoU} = \frac{A \cap Gt}{A \cup Gt} \begin{cases} > 0.7 = \text{object} \\ < 0.3 = \text{not object} \end{cases}$$



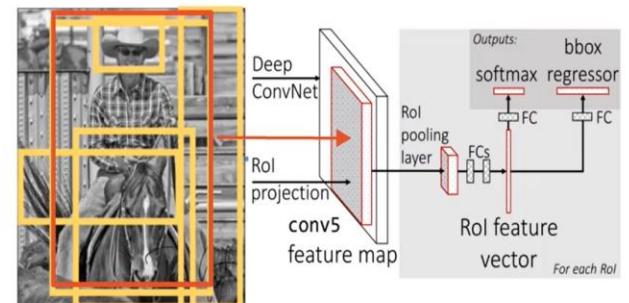
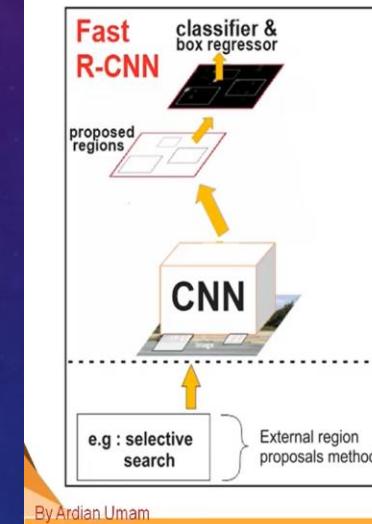
[https://www.youtube.com/watch?v=X3IlbjQs190&list=PLkRkKTC6HZMzp28TxR\\_fJYZ-K8Yu3EQw0&index=3](https://www.youtube.com/watch?v=X3IlbjQs190&list=PLkRkKTC6HZMzp28TxR_fJYZ-K8Yu3EQw0&index=3) [18:10]

# How Fast R-CNN Works



## 2. Architecture#08

### Fast R-CNN as detector



17

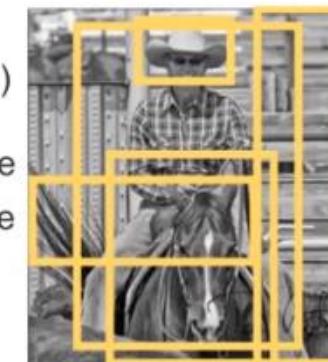
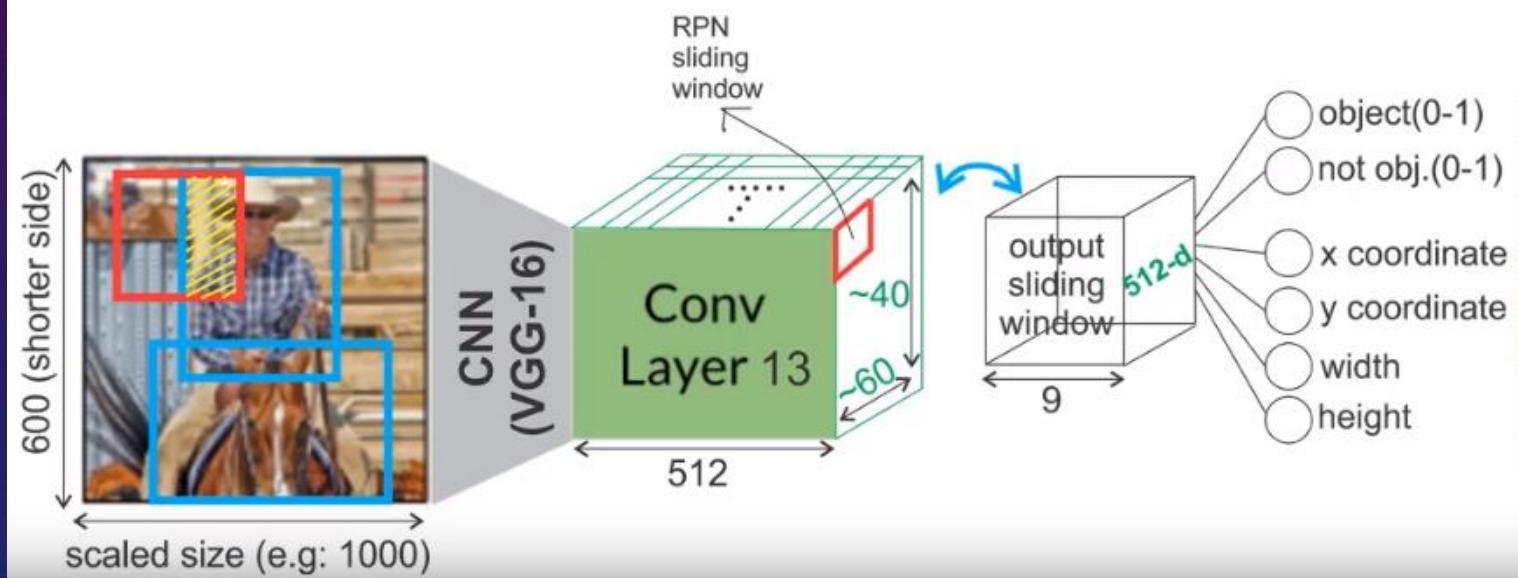
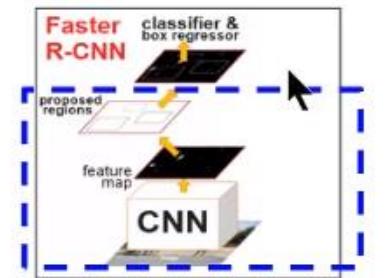
[https://www.youtube.com/watch?v=xzw3lcldlOU&list=PLkRkKTC6HZMzp28TxR\\_fJYZ-K8Yu3EQw0&index=4](https://www.youtube.com/watch?v=xzw3lcldlOU&list=PLkRkKTC6HZMzp28TxR_fJYZ-K8Yu3EQw0&index=4) [2:51]

# How To Train Faster R-CNN

## 3. Training phase#03

### Step 1 (Proposer)

1. Train RPN, initialized with ImageNet pre-trained model



[https://www.youtube.com/watch?v=cSO1nUj495Y&list=PLkRkKTC6HZMzp28TxR\\_fJYZ-K8Yu3EQw0&index=5](https://www.youtube.com/watch?v=cSO1nUj495Y&list=PLkRkKTC6HZMzp28TxR_fJYZ-K8Yu3EQw0&index=5) [13:55]

# Experiments and Results of Faster R-CNN

## Experiments & Results#1

### Experiments: Datasets



Visual Object Classes Challenge 2012 (VOC2012)



VOC 2007

5k trainval images  
5k test images  
20 object categories

By Ardian Umam

80k training images  
40k validation set images  
20k test images  
80 object categories



29

[https://www.youtube.com/watch?v=306ieamtvGM&list=PLkRkKTC6HZMzp28TxR\\_fJYZ-K8Yu3EQw0&index=6](https://www.youtube.com/watch?v=306ieamtvGM&list=PLkRkKTC6HZMzp28TxR_fJYZ-K8Yu3EQw0&index=6) [20:51]