10. DL Applications I: Computer Vision

CS 5242 Neural Networks and Deep Learning

Ai Xin

Agenda

- Object detection
  - Region Based Detector
    - R-CNN
    - Fast R-CNN
    - Faster R-CNN
  - Single Stage
    - YOLO
    - SSD
- Image segmentation
  - U-Net
  - Mask R-CNN

Introduction

What is Object Detection?

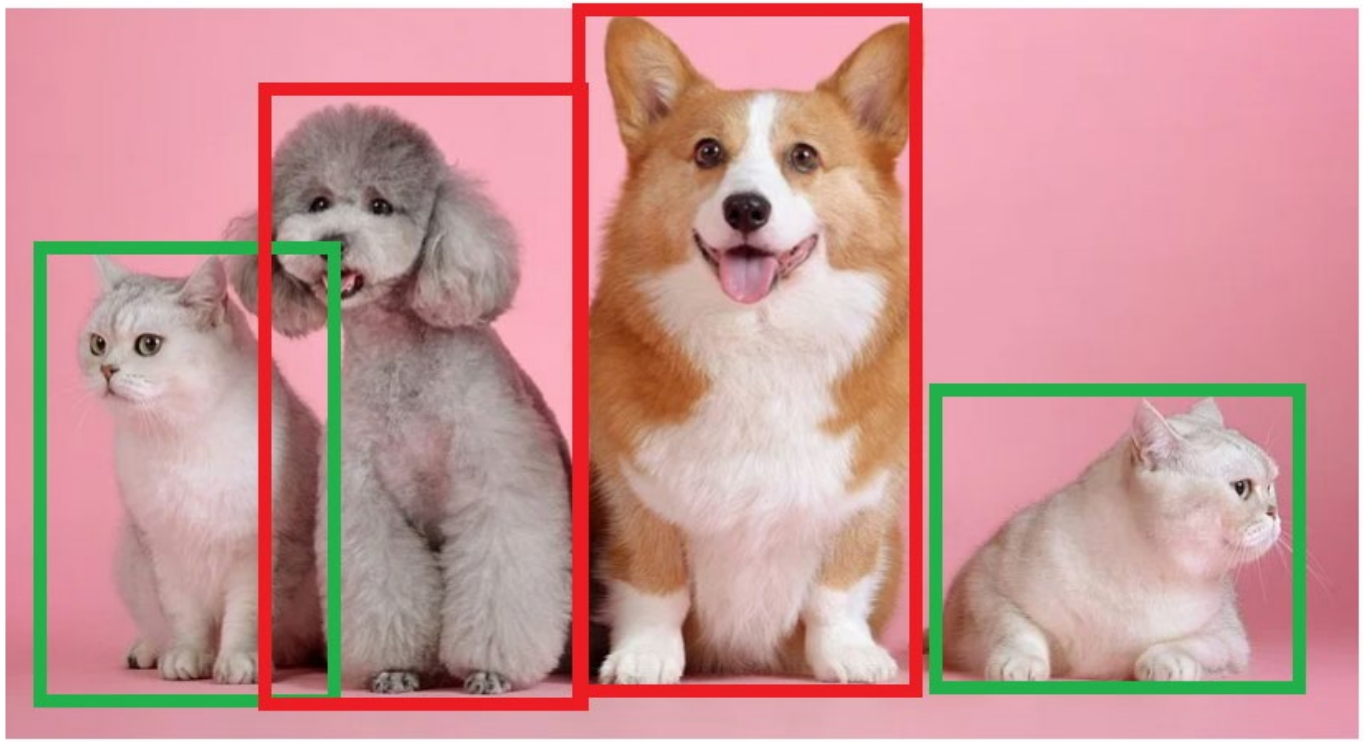
Object Classification

CAT

Object Localization

CAT

Object Detection



CAT, DOG, DOG, CAT

Object Detection Applications

- Face detection
  - Point-and-shoot camera
- Surveillance
  - Count cars, peoples, animals
- Indexing
  - Get objects from images for search

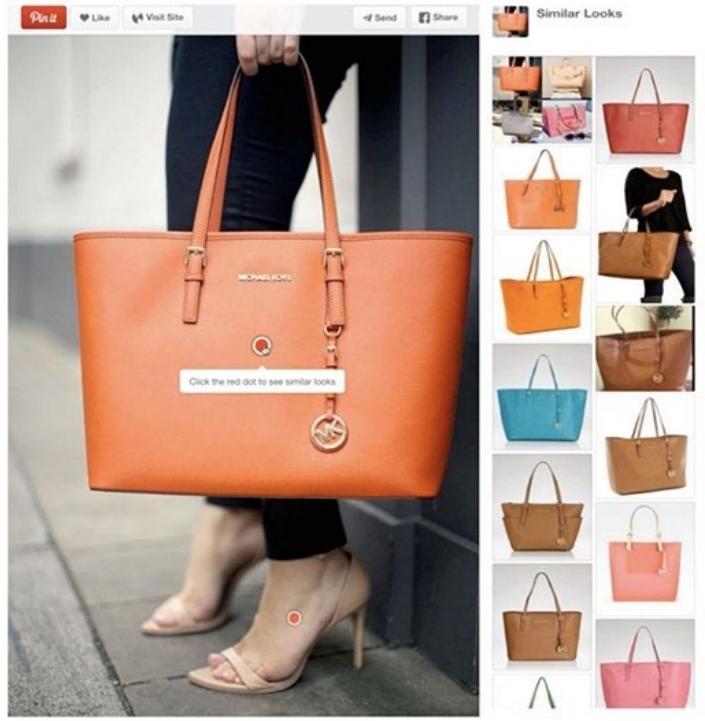
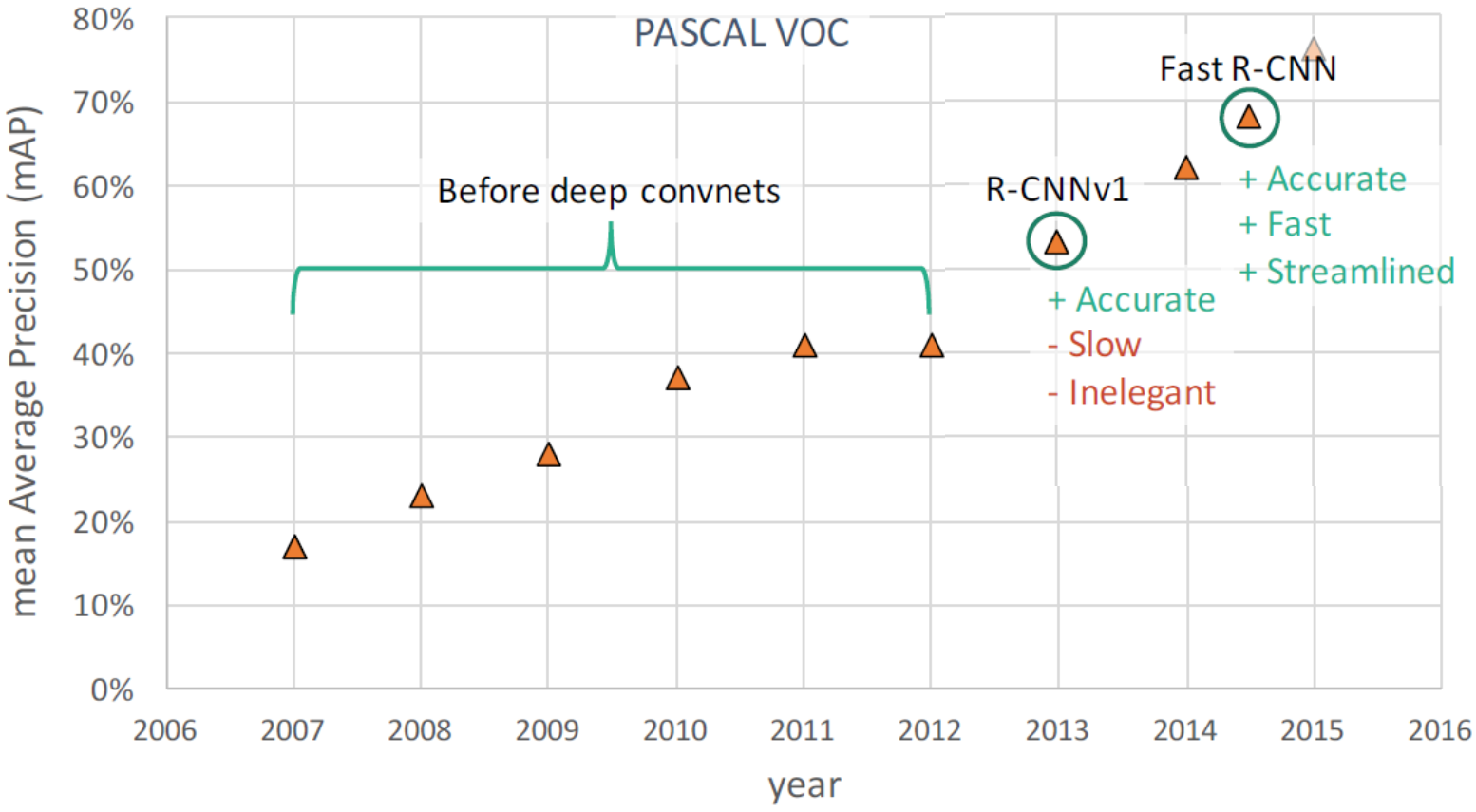


Figure 1: Similar Looks: We apply object detection to localize products such as bags and shoes. In this prototype, users click on automatically tagged objects to view similar-looking products.

Object Recognition Renaissance



Recap: Image Classification

This image is CC0 public domain

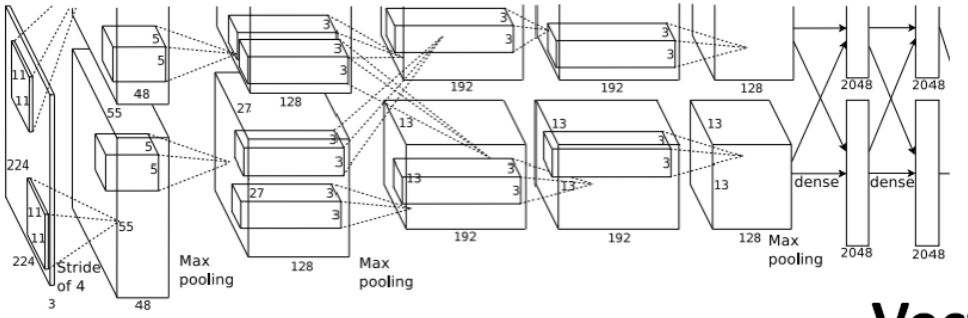


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

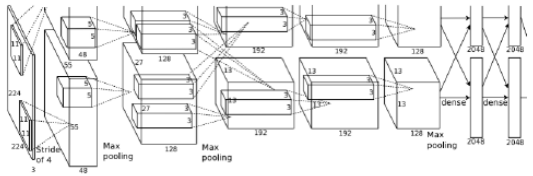
Vector:
4096

Fully-Connected:
4096 to 1000

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

Detecting a single object

This image is CC0 public domain



Treat localization as a regression problem!

Vector: 4096

Fully Connected: 4096 to 1000

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

Softmax Loss

Correct label: Cat

Fully Connected: 4096 to 4

Box Coordinates (x, y, w, h)

L2 Loss

Correct box: (x', y', w', h')

Weighted Sum

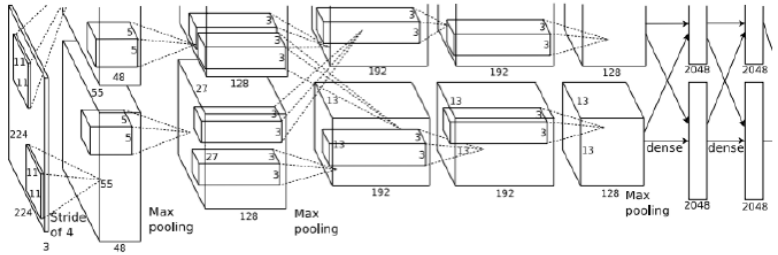
Loss

"What"

"Where"

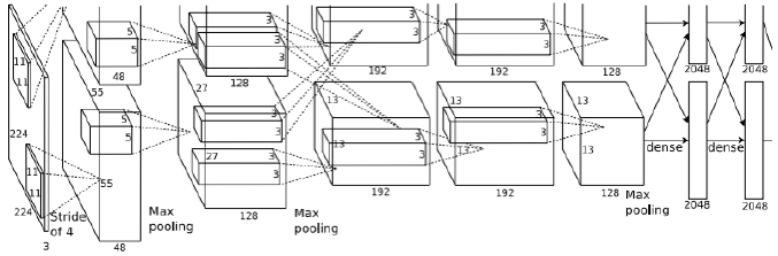
Detecting Multiple Objects

Need different numbers of outputs per image



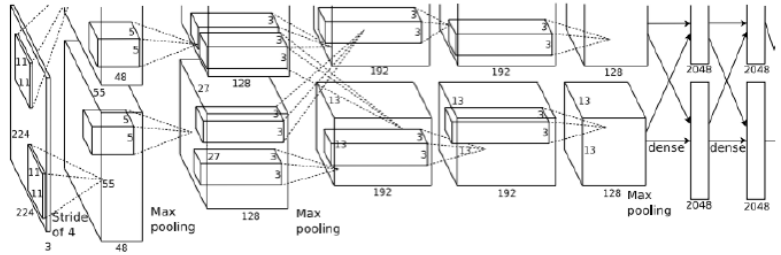
CAT: (x, y, w, h)

4 numbers



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

16 numbers



DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
....

Many numbers!

Object Detection Task

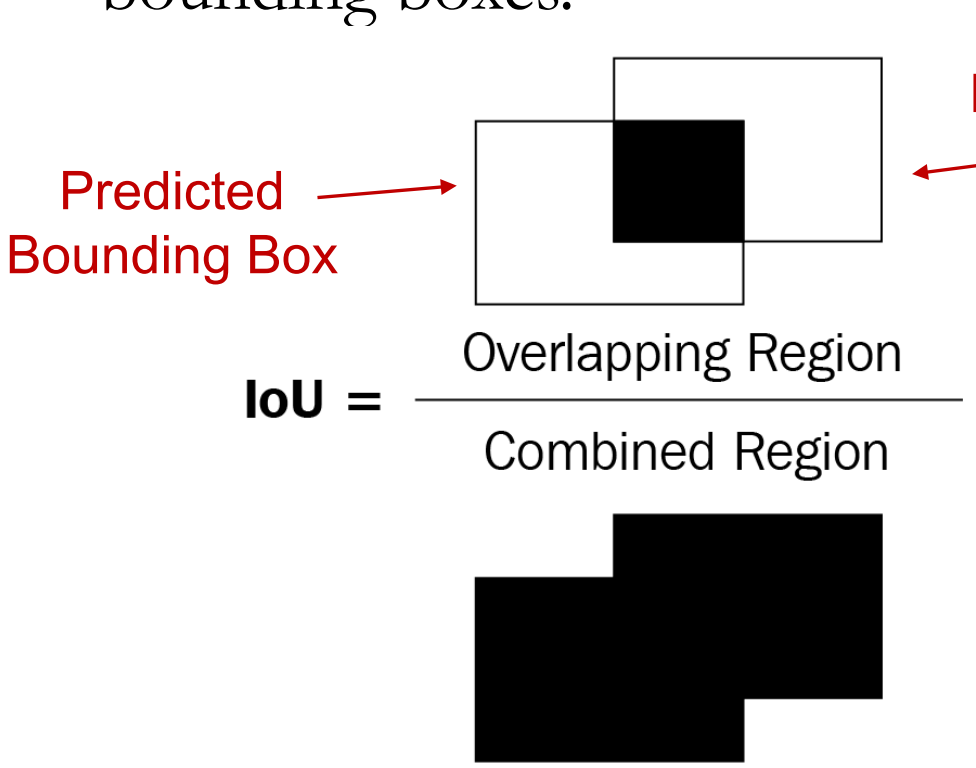
- Detect the location of all object instances of all classes "What & where?"
- Training label
  - a list of <class, bounding box of each object instance>
- Prediction output
  - a list of <class, probability, bounding box of each object instance>

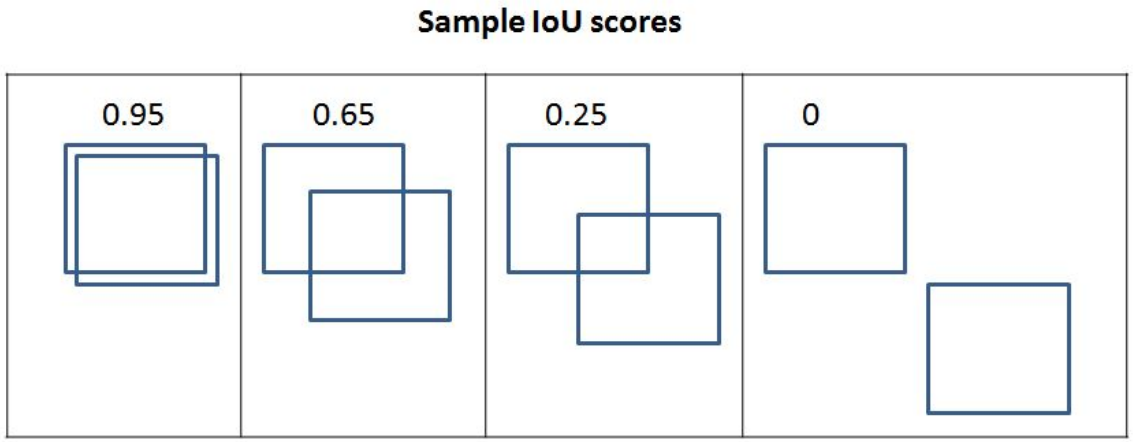
Training label:
bicycle (10, 100, 110, 110) (200, 200, 180, 80) …
People (200, 80, 71, 71) (300, 50, 20, 80) …

Prediction output:
bicycle 0.9 (9, 93, 100, 111), 0.8 (200, 200, 180, 80), …
people 0.8 (200, 80, 71, 71), …
….

How to Evaluate Performance?

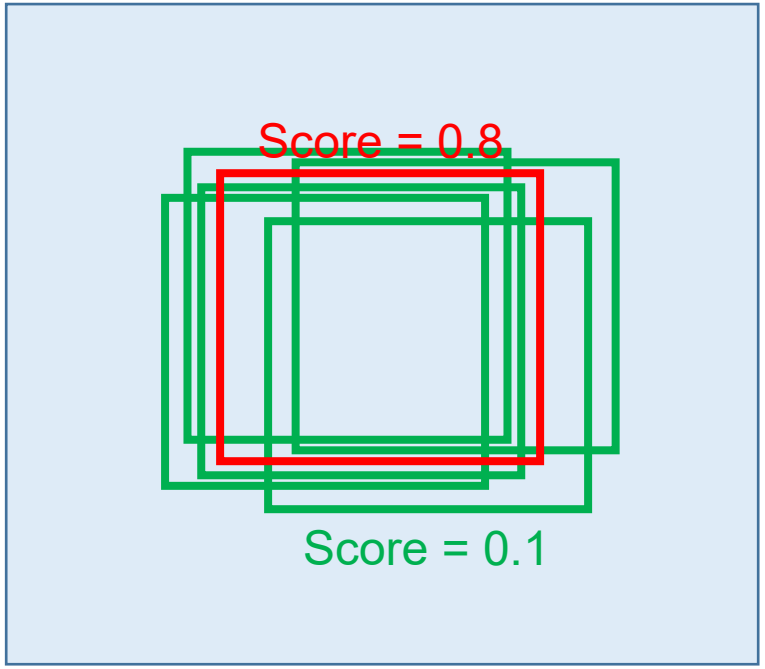
- Intersection Over Union (IoU): the ratio of the overlapping region between the two bounding boxes over the combined region of both the bounding boxes.





How to Evaluate Performance?

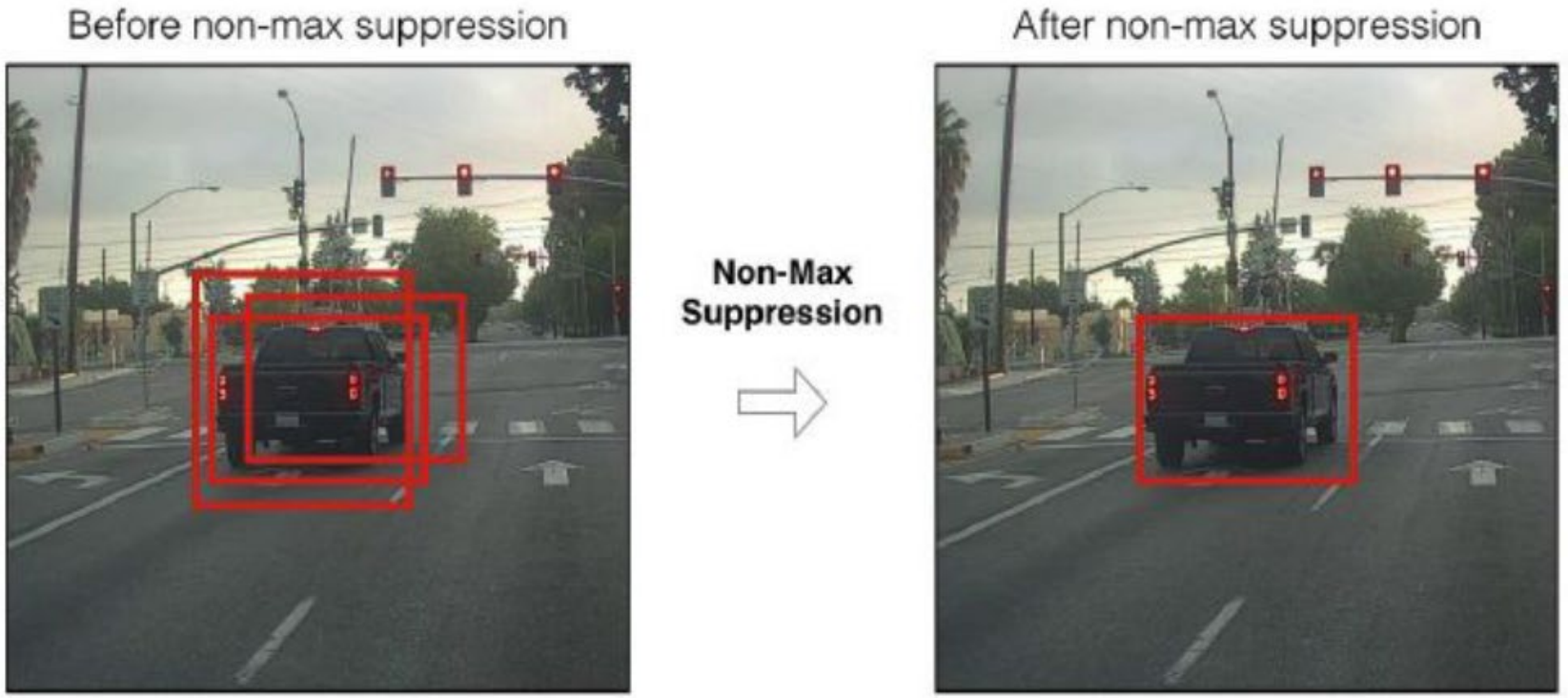
- Non-max suppression:
  - Multiple predicted bounding boxes are generated and significantly overlapped
  - Identify the bounding box that has the highest confidence score (e.g. 0.8)
  - Discard all the other bounding boxes that have an IoU greater than a certain threshold (e.g. 0.5)



Adapted from J. Hays

How to Evaluate Performance?

- Non-max suppression:



How to Evaluate Performance?

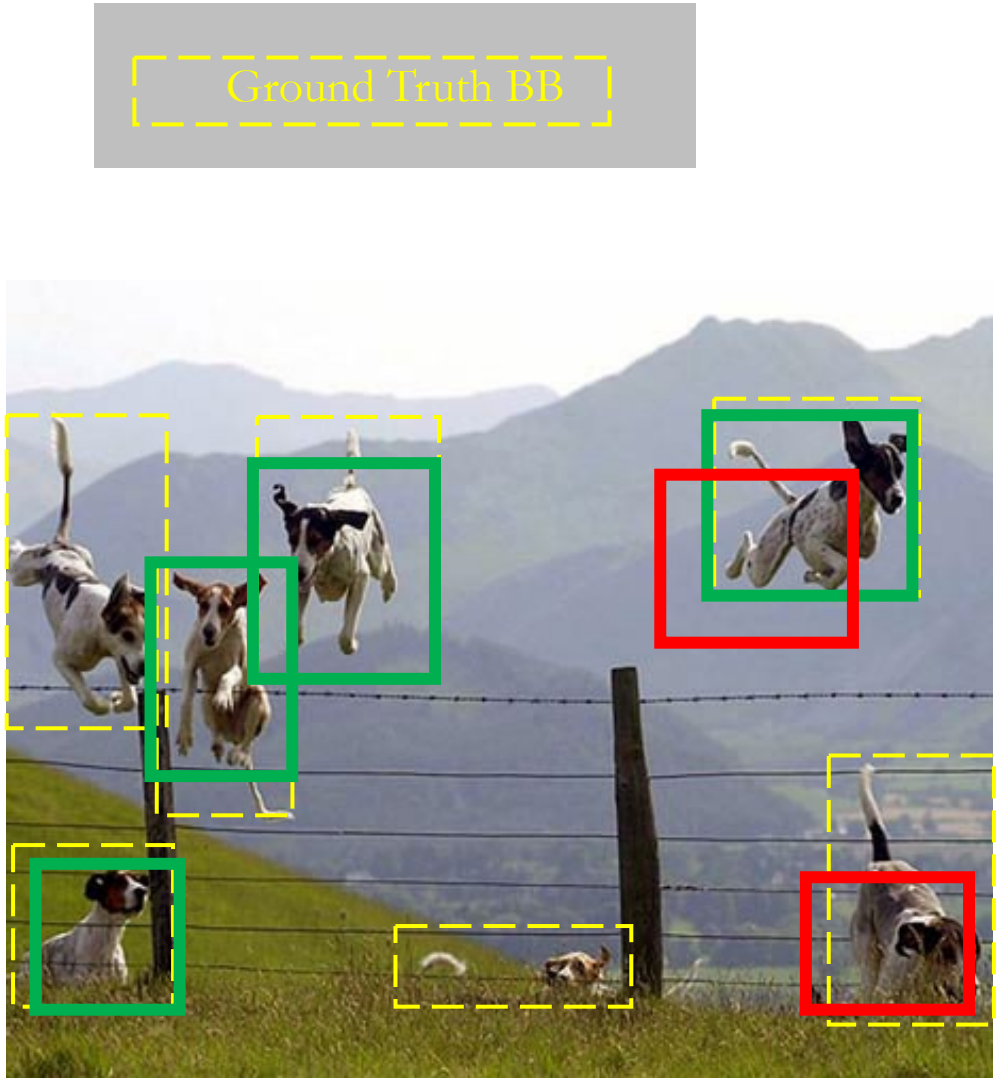
- Mean Average Precision (mAP)
  - **Precision:** TP / (TP + FP)
  - **Recall:** TP / (TP + FN)
  - **Average Precision:** Area under Precision vs Recall Curve
  - **mAP:** the average of AP for each class

Precision & Recall

After detection score thresholding & non-maximum suppression:

- True Positive (TP) – detected objects which align with the ground truth by IoU threshold *t* (usu. *t=0.5*)
- False Positive (FP) – detected objects w/out corresponding ground truth.
  - detections which do not meet IoU threshold are also counted as false positives!
- False Negative (FN) – objects marked in ground truth which were not detected

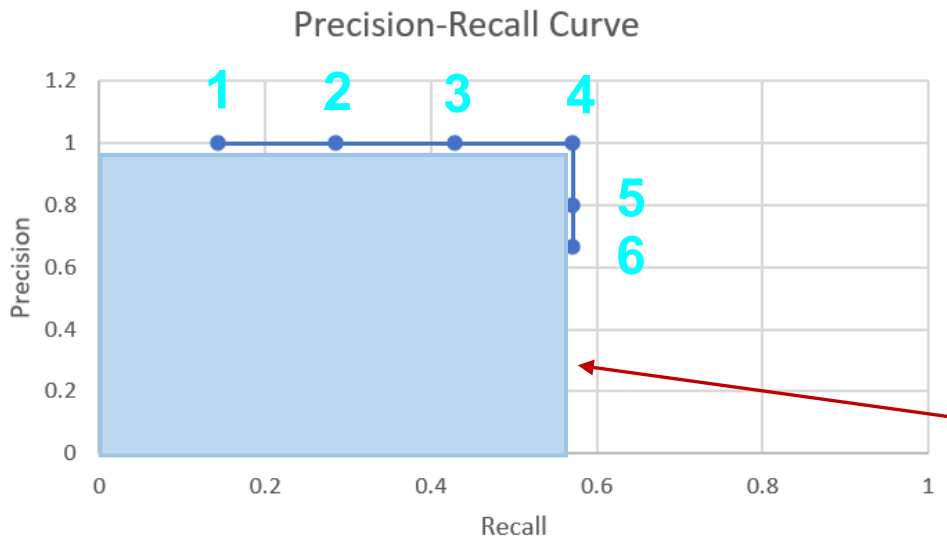
Precision = TP / (TP + FP) = 4 / (4 + 2) = 0.67
Recall = TP / (TP + FN) = 4 / (4 + 3) = 0.57



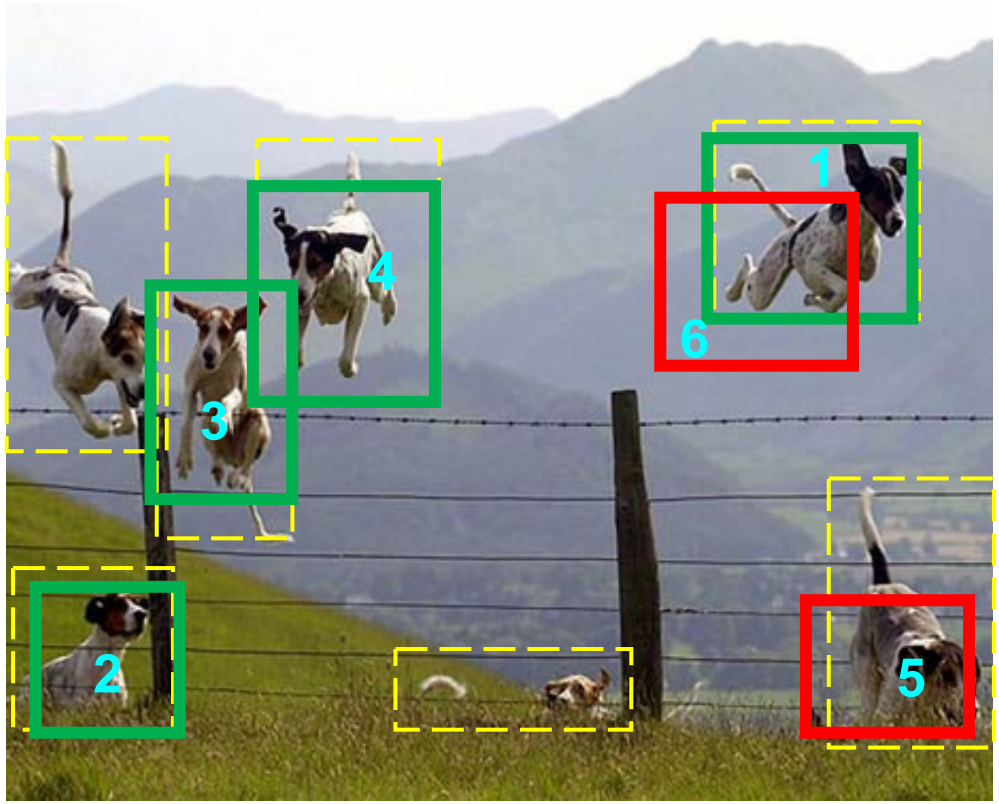
Average Precision

Go through the Predicted Bounding Box from 1 to 6 to generate six data points

Data Point #	TP	FP	FN	Precision	Recall
1	1		6	1/1 = 1	1/7 = 0.14
2	2		5	2/2 = 1	2/7 = 0.29
3	3		4	3/3 = 1	3/7 = 0.43
4	4		3	4/4 = 1	4/7 = 0.57
5	4	1	3	4/5 = 0.8	4/7 = 0.57
6	4	2	3	4/6= 0.67	4/7 = 0.57



Predicted Bounding Box is sorted based on the confidence score from 1 to 6



AP = 1 * 4/7 = 0.57
(Area under Precision-Recall Curve)

Region-based Convolutional Neural Network (R-CNN)

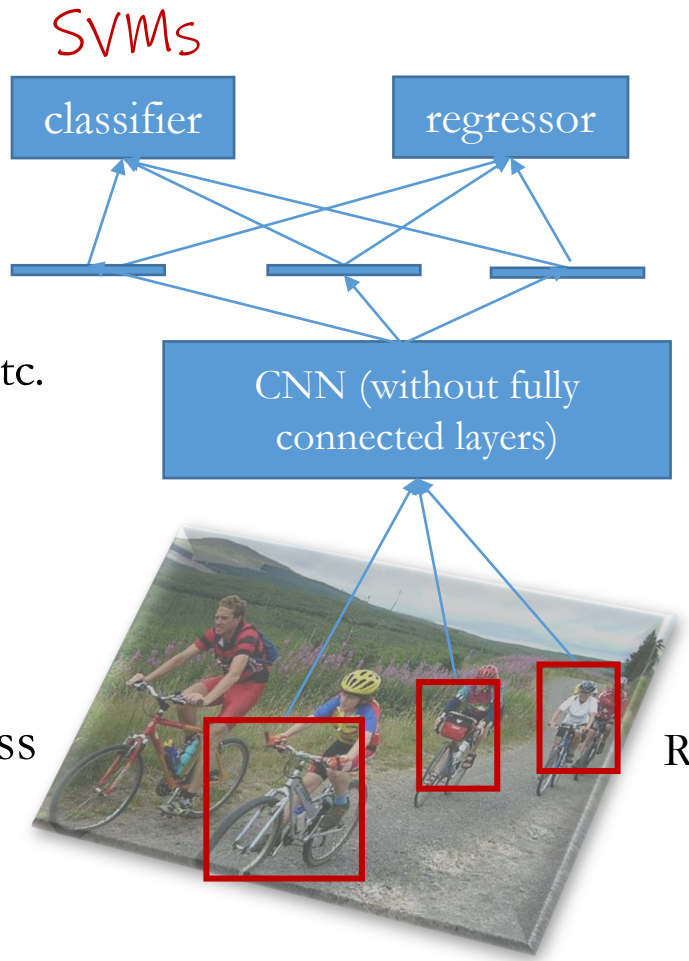
R-CNN: Region-Based ConvNets [pdf]

1. Generate region proposal
   - Candidate object regions
   - Resize region into the same size

2. Extract CNN features
   - Forward-propagate each region via CNN
   - Using popular CNN architectures: VGG, ResNet, etc.

3. Create Data for Model Training
   - Input: CNN feature maps
   - Target: Class and Region Proposal **Offset**

4. Train the Model to minimize the combined loss
   - Classify the image: Softmax
   - Predict the region proposal **offset**: Linear Layer
     - L2 loss for each value
     - 4 values: (dx, dy, dw, dh)



Slide credit: Wang Wei

Region Proposal

- **Region proposal** is a technique that helps in identifying islands of regions where the pixels are similar to one another.

Original Image

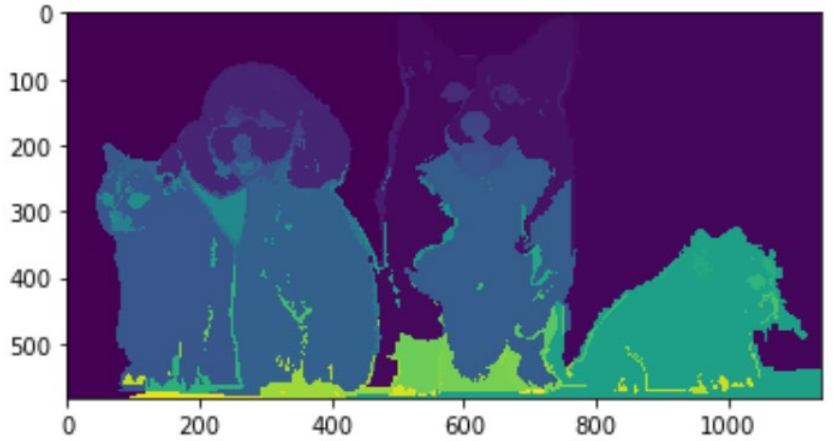
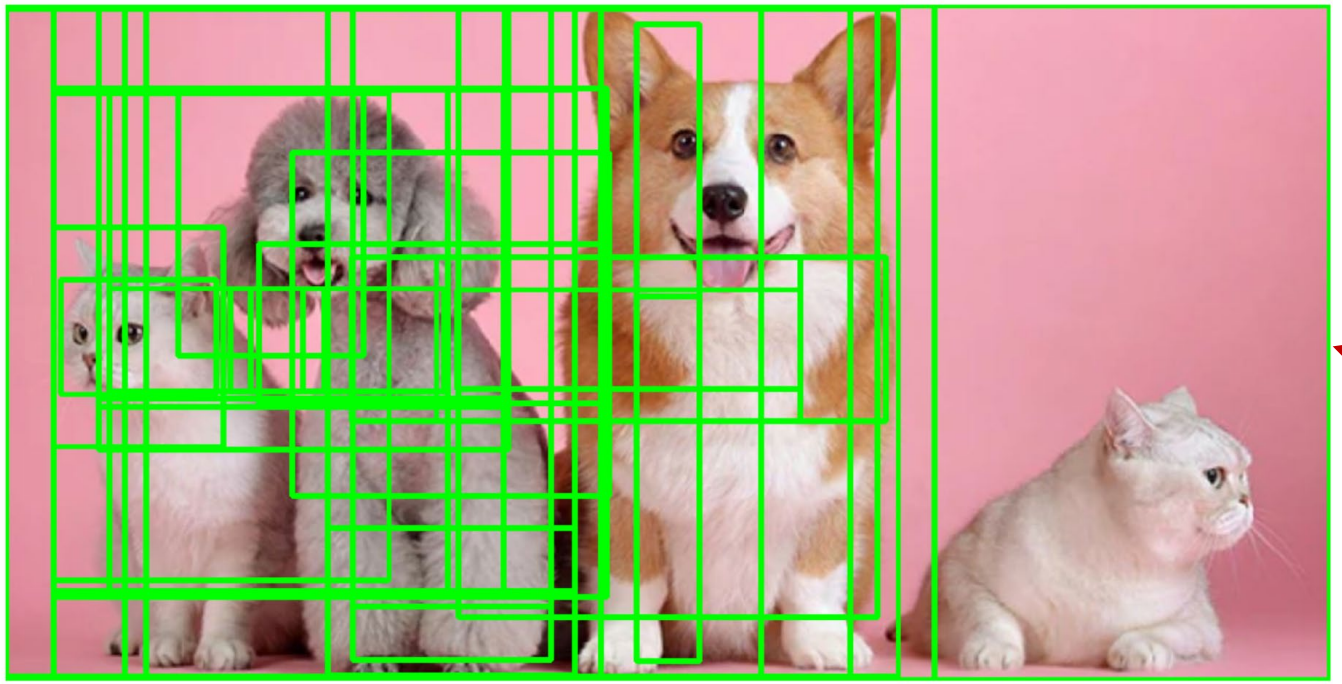
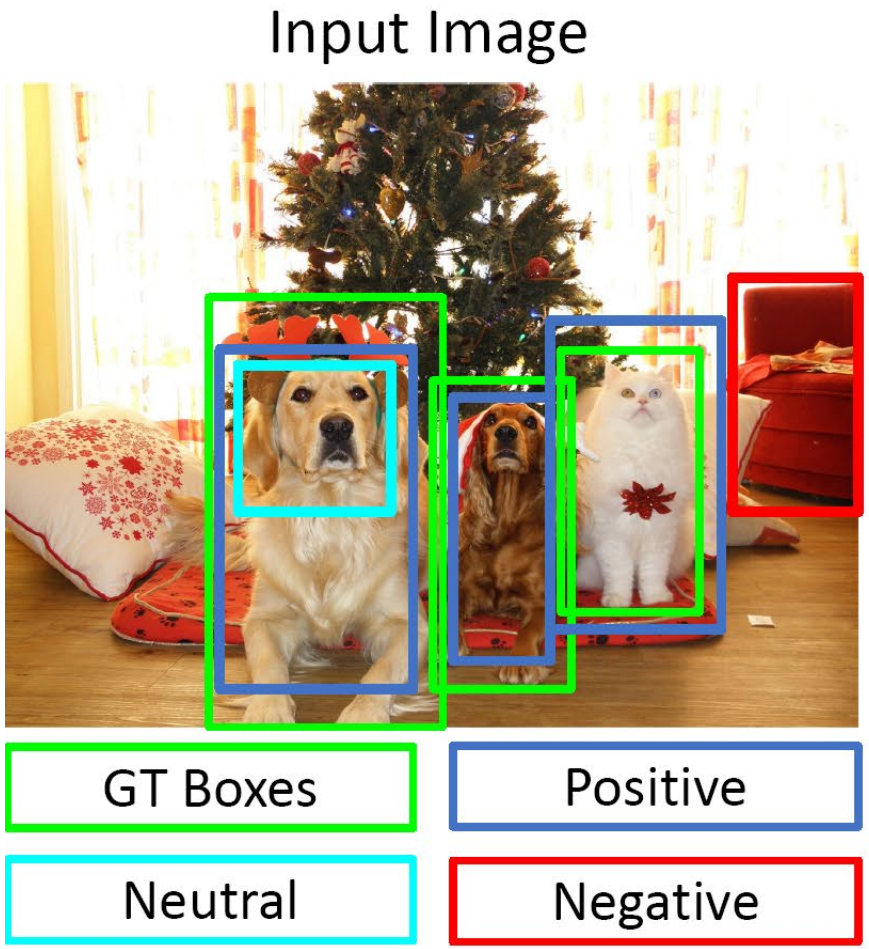


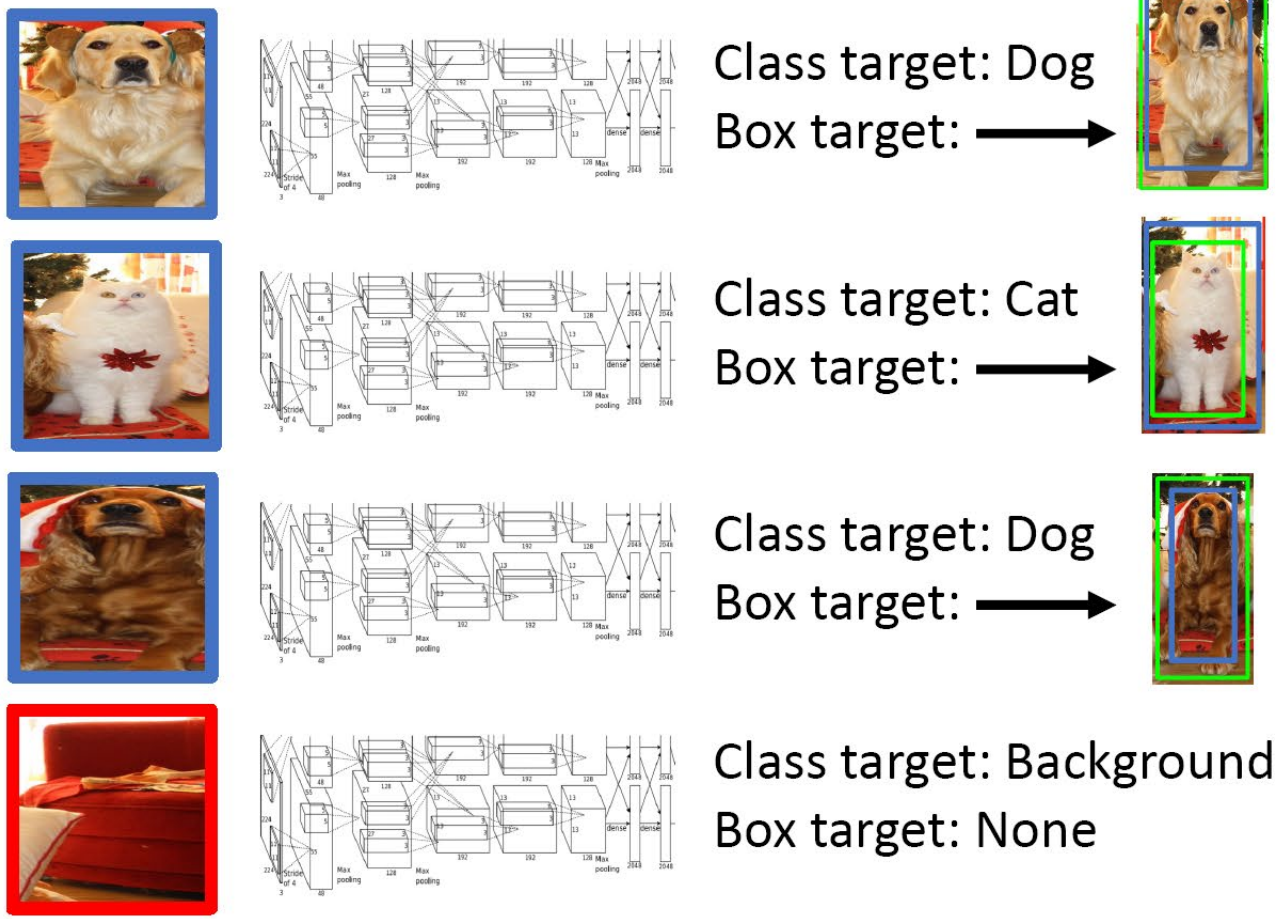
Image post felzenszwalb segmentation



Candidate Regions generated using selective search method

Create Data for Model Training

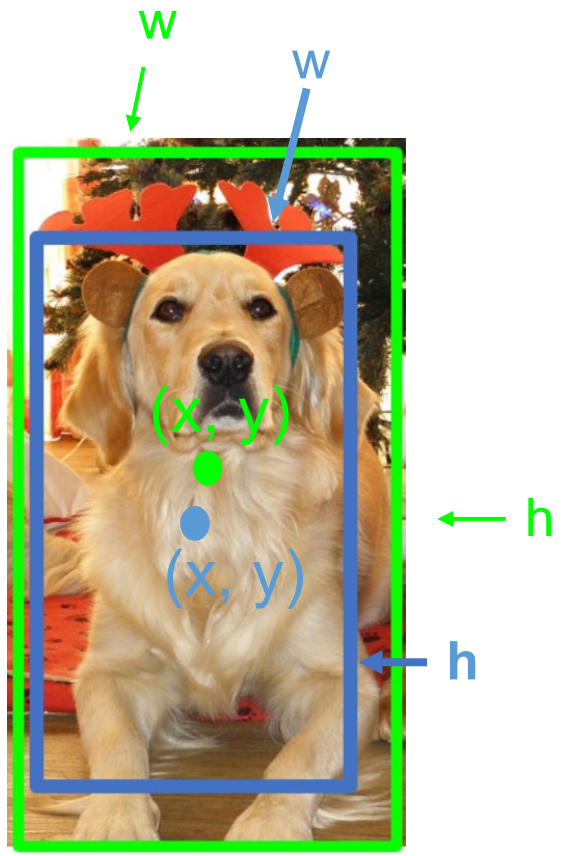




Create Data for Model Training

- An Example
  - A region proposal (blue box) has an IoU greater than 0.5 with the ground truth (green box) BB of dog ➔ Label this candidate region as "dog"
  - Region proposal Offset

(dx, dy, dw, dh) = (x, y, w, h) - (x, y, w, h)



The actual offset transformation fuction

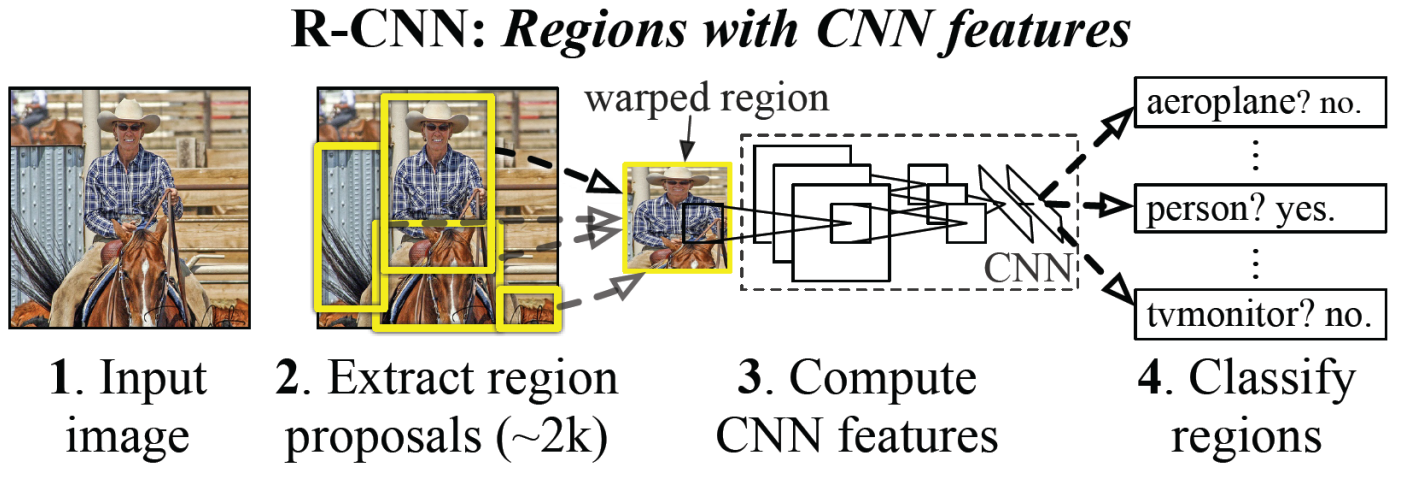
- A region proposal box A is assigned a ground-truth bounding box B
- Apply below transformations may lead to more uniformly distributed offsets that are easier to fit.

$$\left(\frac{\frac{x_b - x_a}{w_a} - \mu_x}{\sigma_x}, \frac{\frac{y_b - y_a}{h_a} - \mu_y}{\sigma_y}, \frac{\log \frac{w_b}{w_a} - \mu_w}{\sigma_w}, \frac{\log \frac{h_b}{h_a} - \mu_h}{\sigma_h} \right)$$

- By default: $\mu_x = \mu_y = \mu_w = \mu_h = 0$, $\sigma_x = \sigma_y = 0.1$, $\sigma_w = \sigma_h = 0.2$

Problems with R-CNN

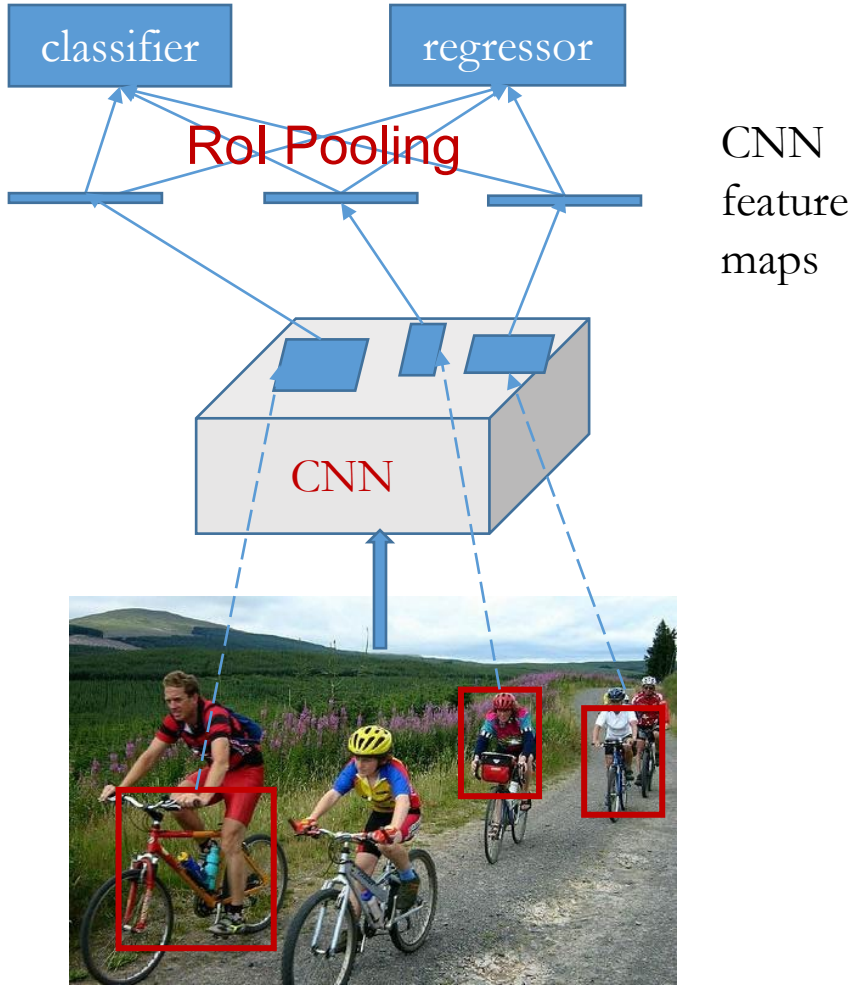
- Slow
  - Too many region proposals~2000 candidate regions per image
  - Each has to go through the CNN
- Training is ad-hoc
  - Train the CNN for feature maps extraction
  - Train SVM classifier for Classification
  - Train Regressor for Region Proposal Offset



Fast R-CNN

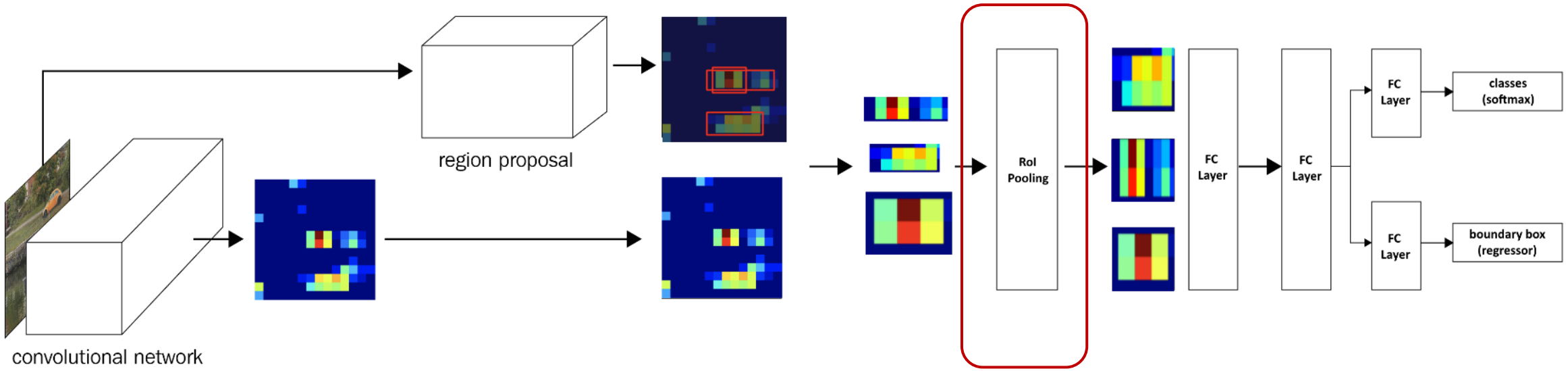
Fast R-CNN [pdf]

- Generate region proposal
  - Candidate object regions
  - Using existing methods
- Extract CNN feature
  - For the **whole input image**
  - Forward-propagate the whole image via CNN
- Get CNN feature for each region
  - Using the region coordinates to locate areas in CNN feature maps
  - Extract the RoI (Region of Interest) from feature maps corresponding to the region proposals
- RoI Pooling Layer
  - Resize all the feature maps of region proposals to a similar shape
- Fully Connected Layers
  - Classify the image
  - Predict the region proposal offset



Slide credit: Wang Wei

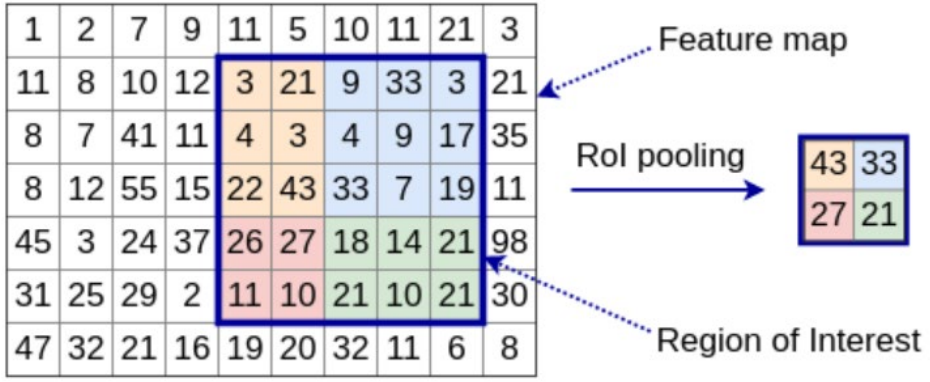
Fast R-CNN


region proposal
convolutional network
RoI Pooling
FC Layer
FC Layer
FC Layer
FC Layer
classes (softmax)
boundary box (regressor)

ROI Pooling

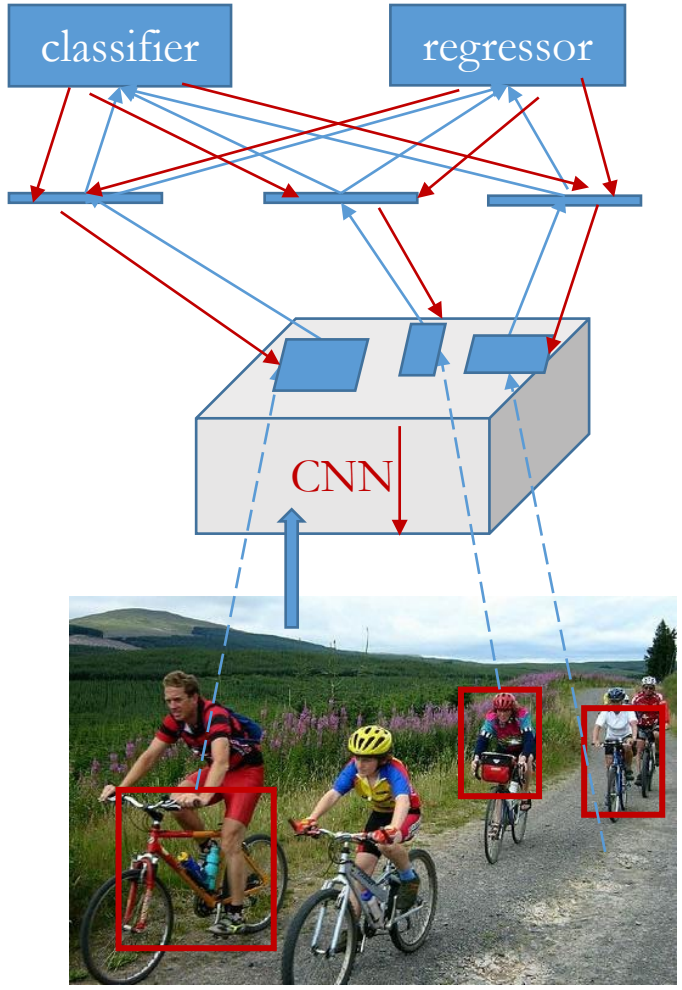
- Why?
  - ROI in feature maps corresponding to different Region proposals have different shapes
  - How can we convert different shapes of ROI into a fixed sized input for the classifier / regressor?
  - Use pooling to aggregate them into a fixed size

- An Example:
  - Feature Map: 10 x 7
  - Region of Interest: 5 x 5
  - RoI Max Pooling: 2 x 2



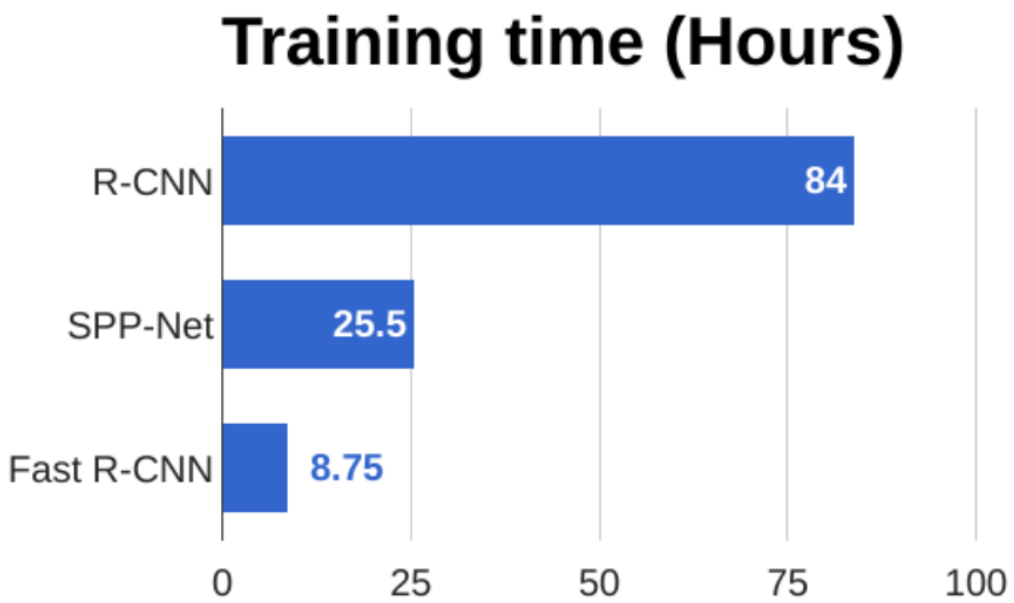
Fast R-CNN Training

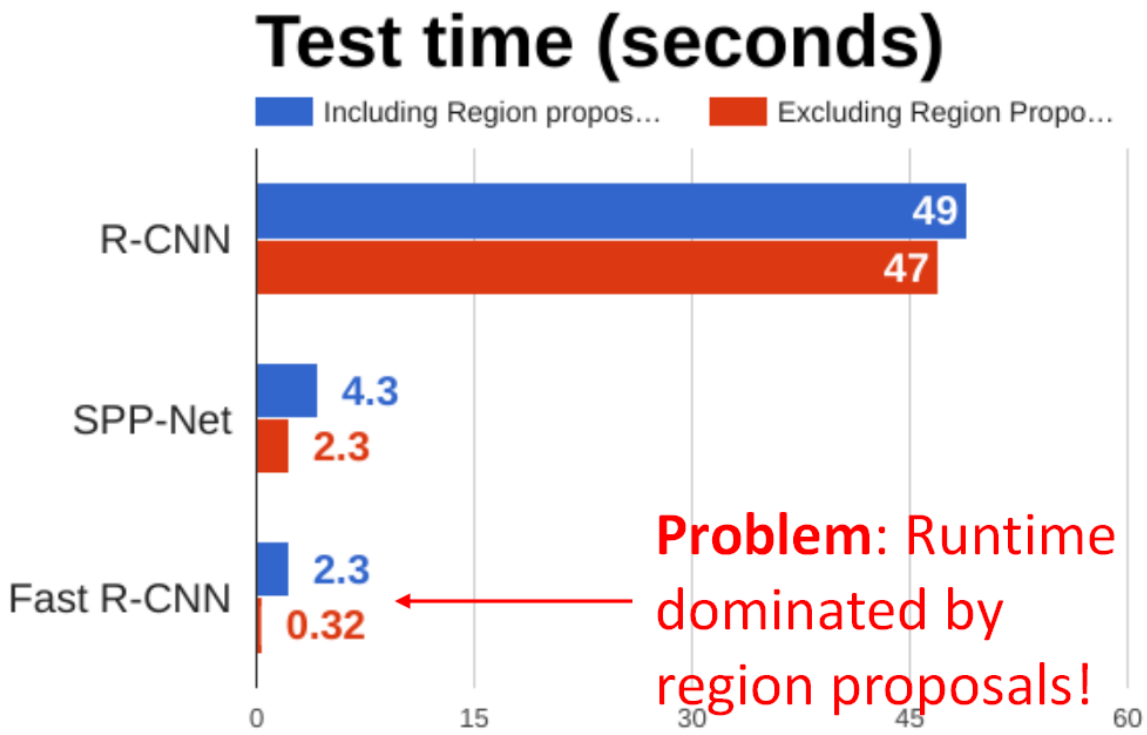
- Run CNN forwarding once
- End-to-end training
  - Softmax classifier
  - Linear regressor for bounding box



Slide credit: Wang Wei

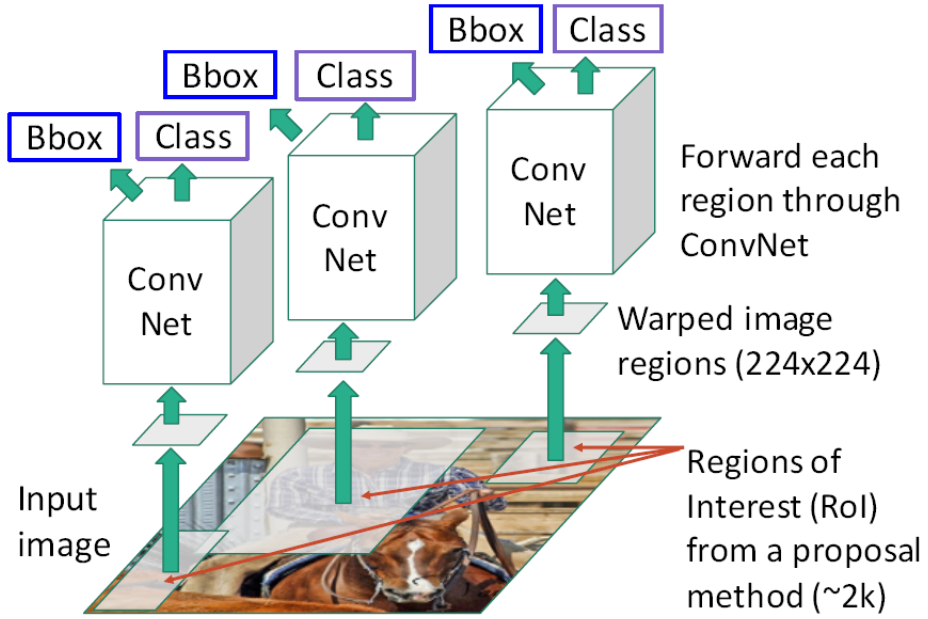
Fast R-CNN vs "Slow" R-CNN





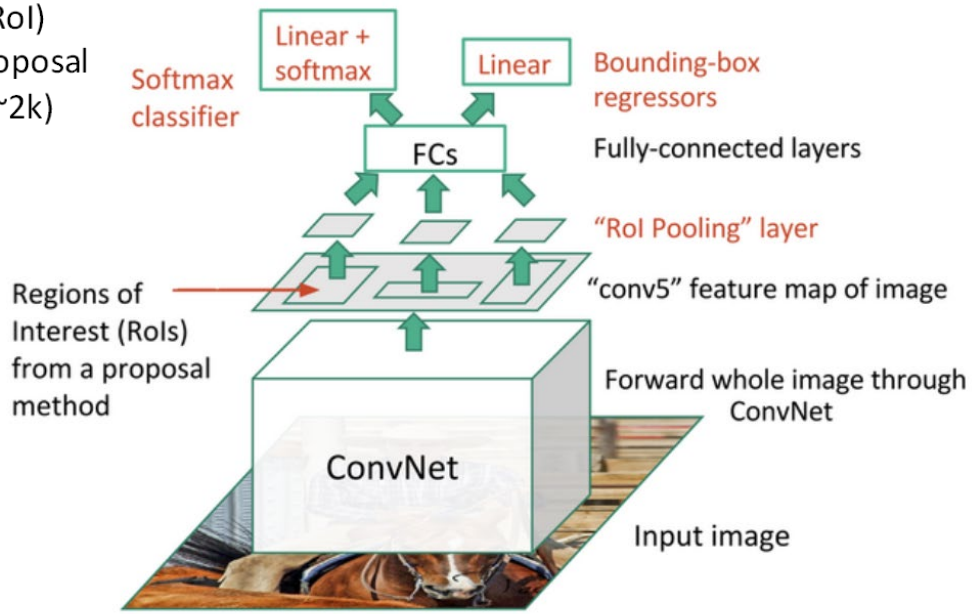
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
He et al, "Spatial pyramid pooling in deep convolutional networks for visual recognition", ECCV 2014
Girshick, "Fast R-CNN", ICCV 2015

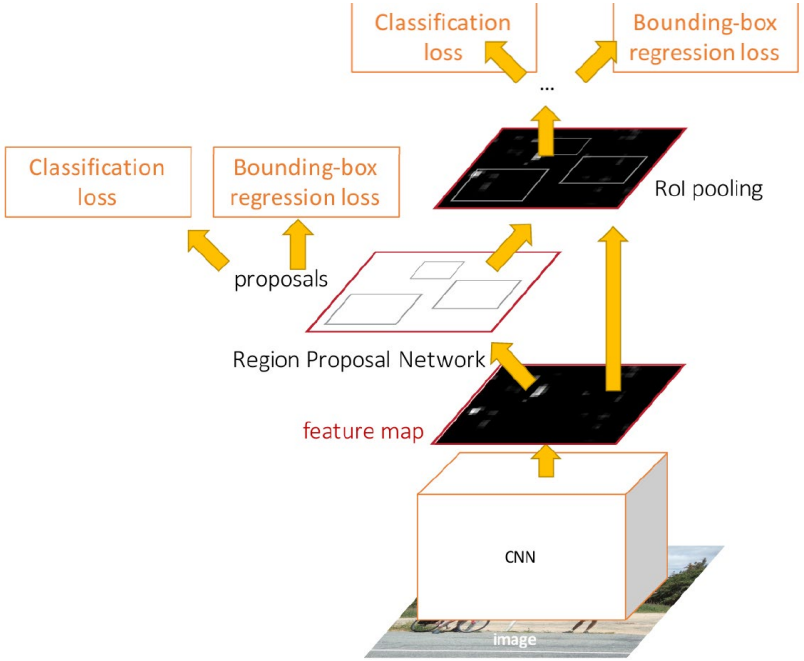
Faster R-CNN



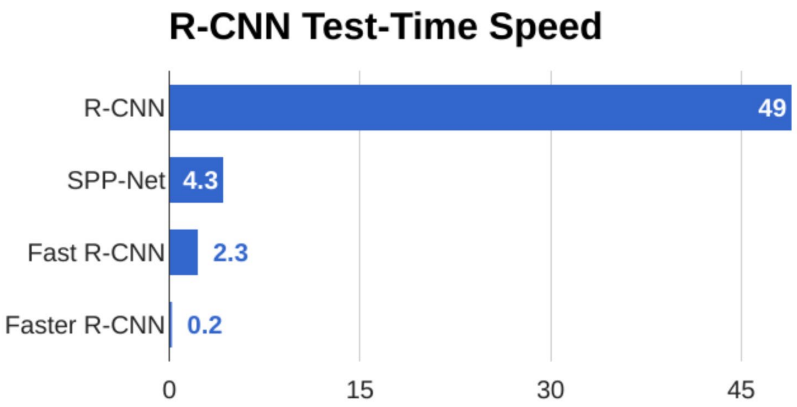
R-CNN

Fast R-CNN



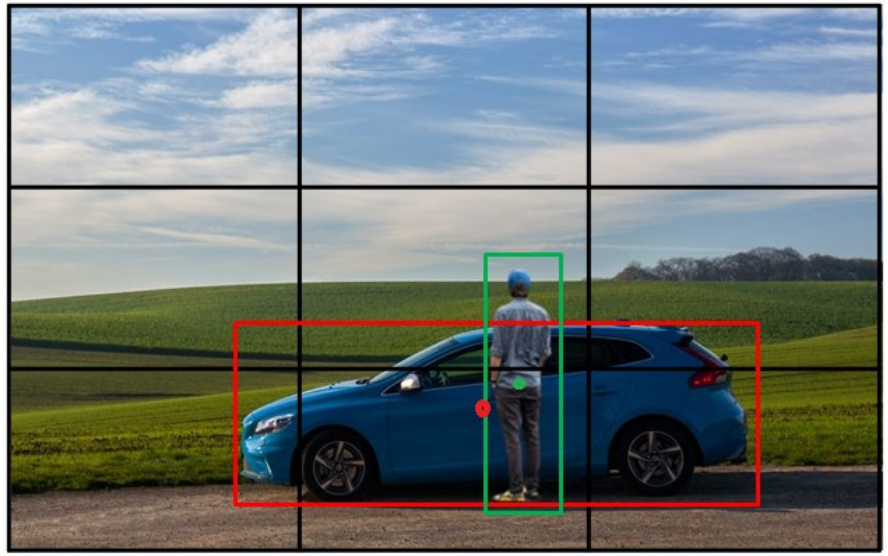


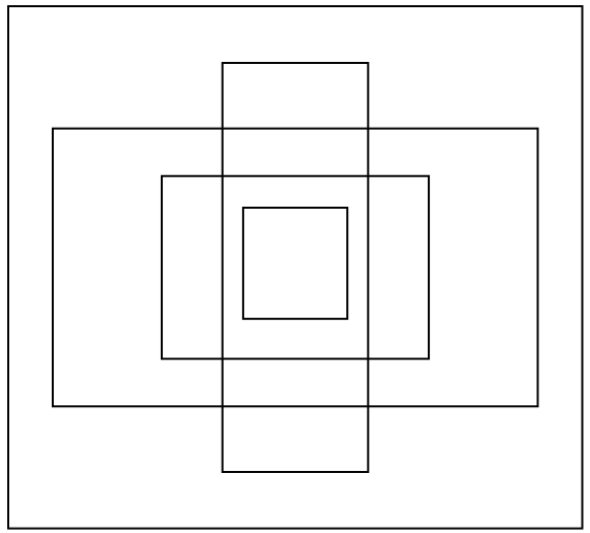
Faster R-CNN



Anchor Box

- A handy replacement for selective search
- Majority of objects in the same category have a similar shape
- In same image, the object might be scaled, but still maintain the aspect ratio (i.e. height / width)
- Employ K-means clustering on all the ground truth BB to obtain the Anchor Box s.t. different aspect ratios scales





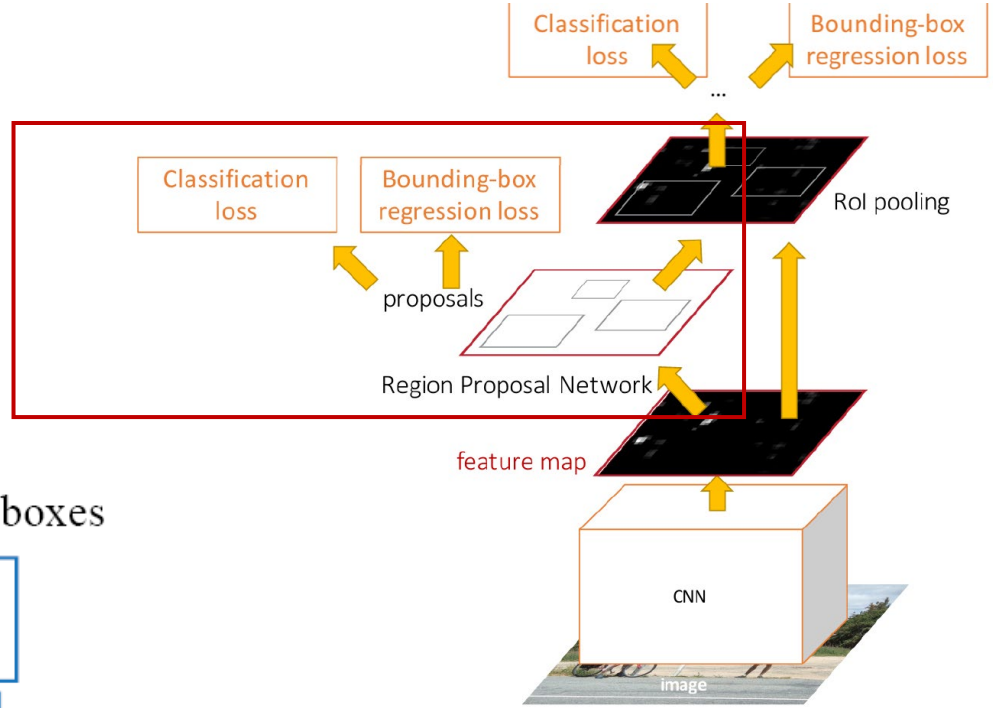
Slide the Anchor Box

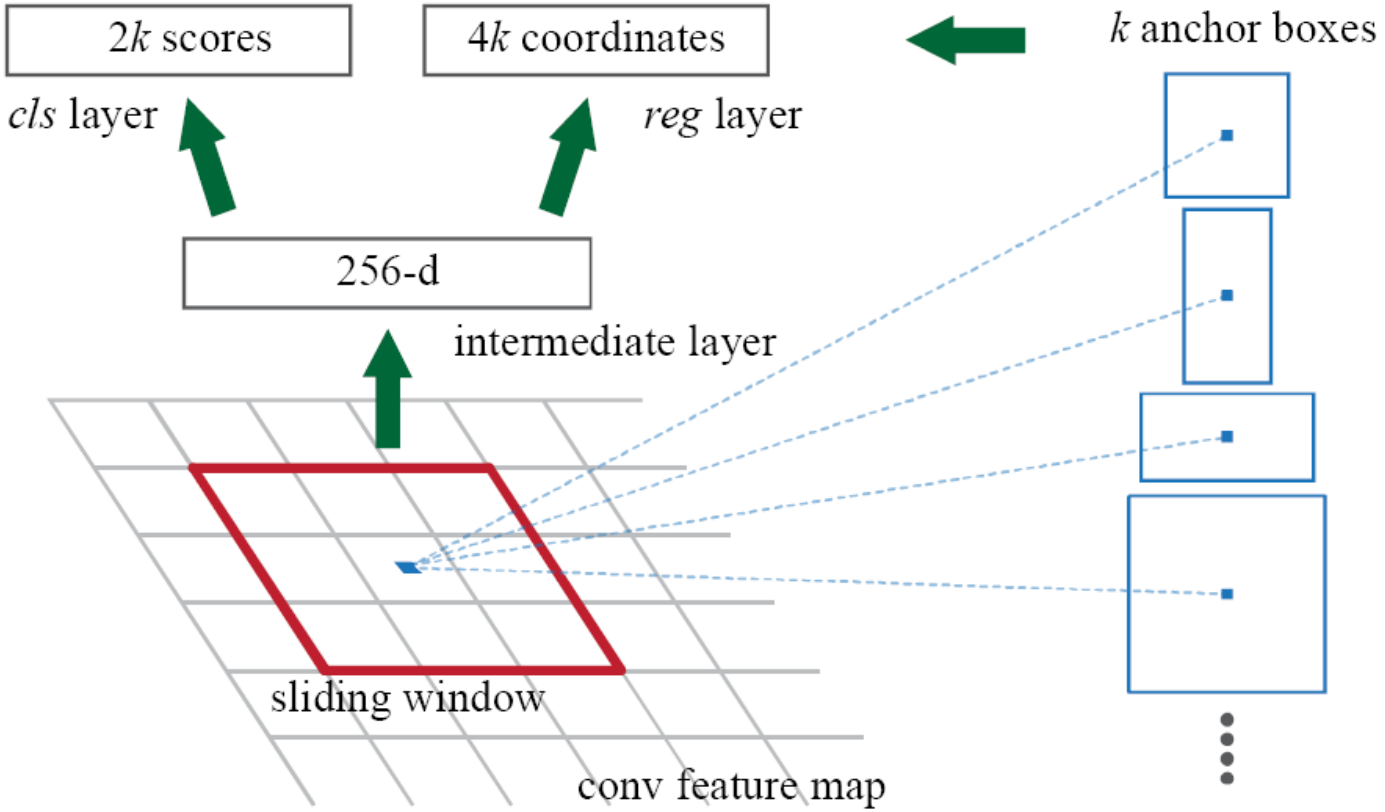
- Slide each anchor box over an image from top left to bottom right

- For anchor box, check its IoU with the ground truth BB:
  - If IoU > threshold, Labeled as 1 ("Object)
  - Otherwise, Labeled as 0 ("Background")

Adapted from J. Hays

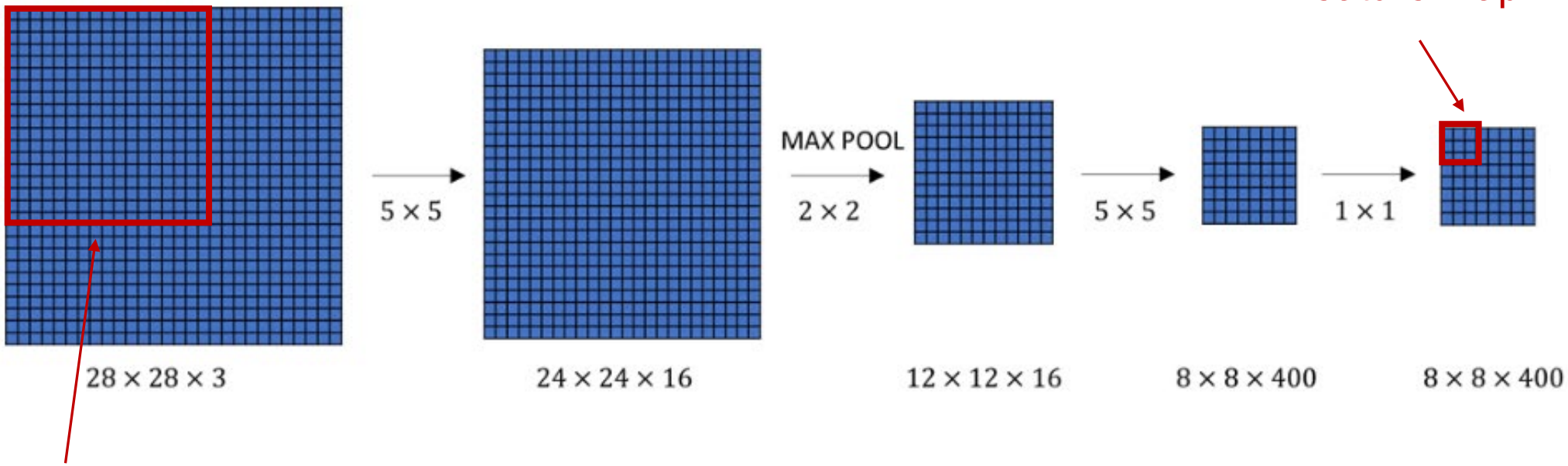
Region Proposal Network





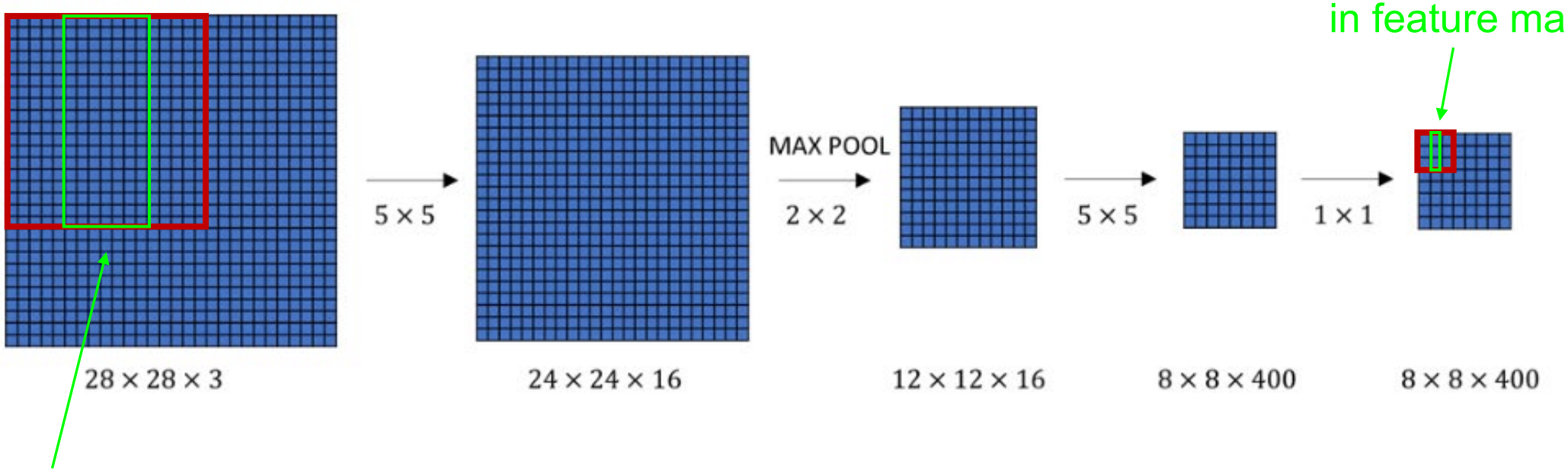
Region Proposal Network

- Convolutional implementation of sliding window

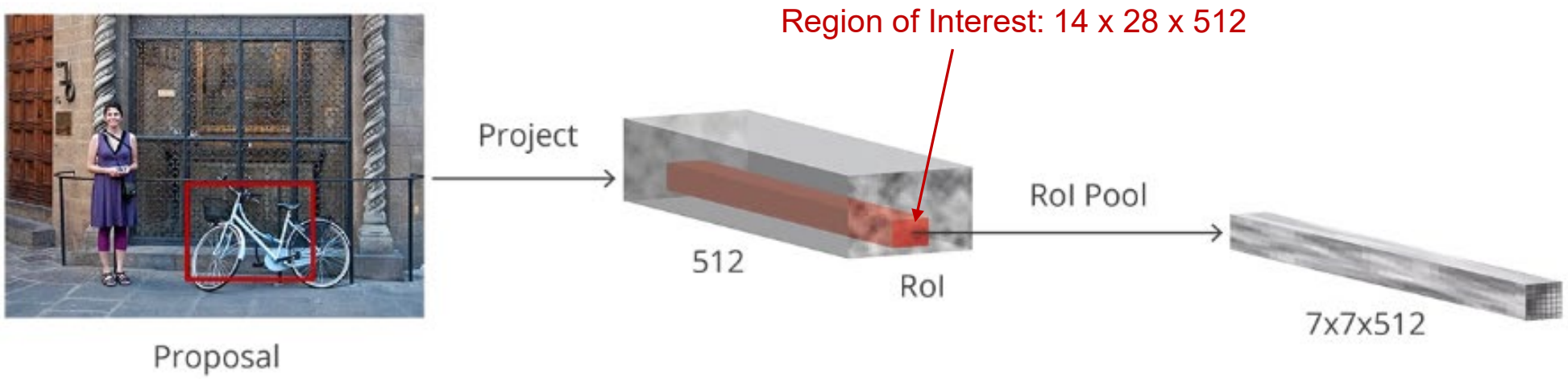


Region Proposal Network

- Convolutional implementation of sliding window

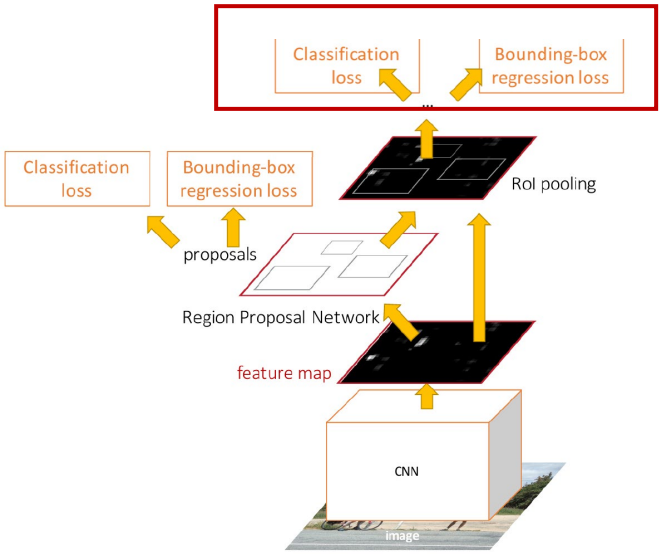


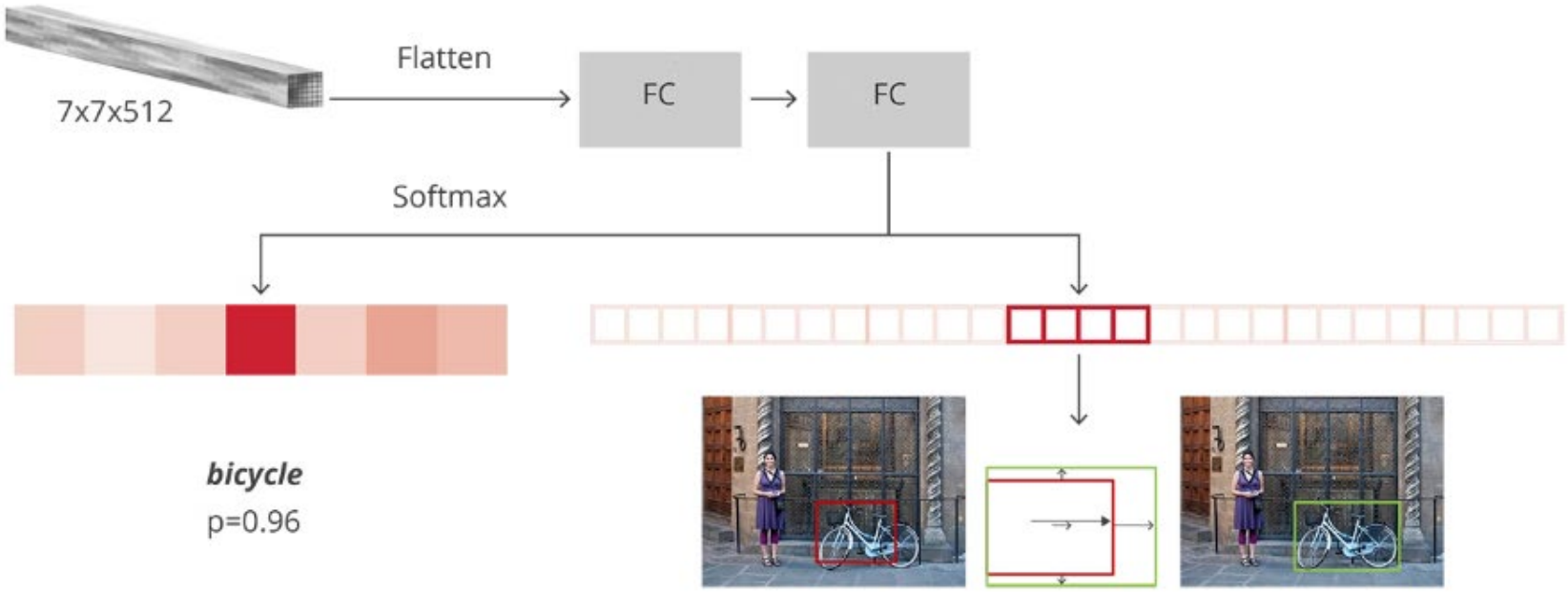
Region of Interest Pooling



Region of Interest Pooling

Detection Network

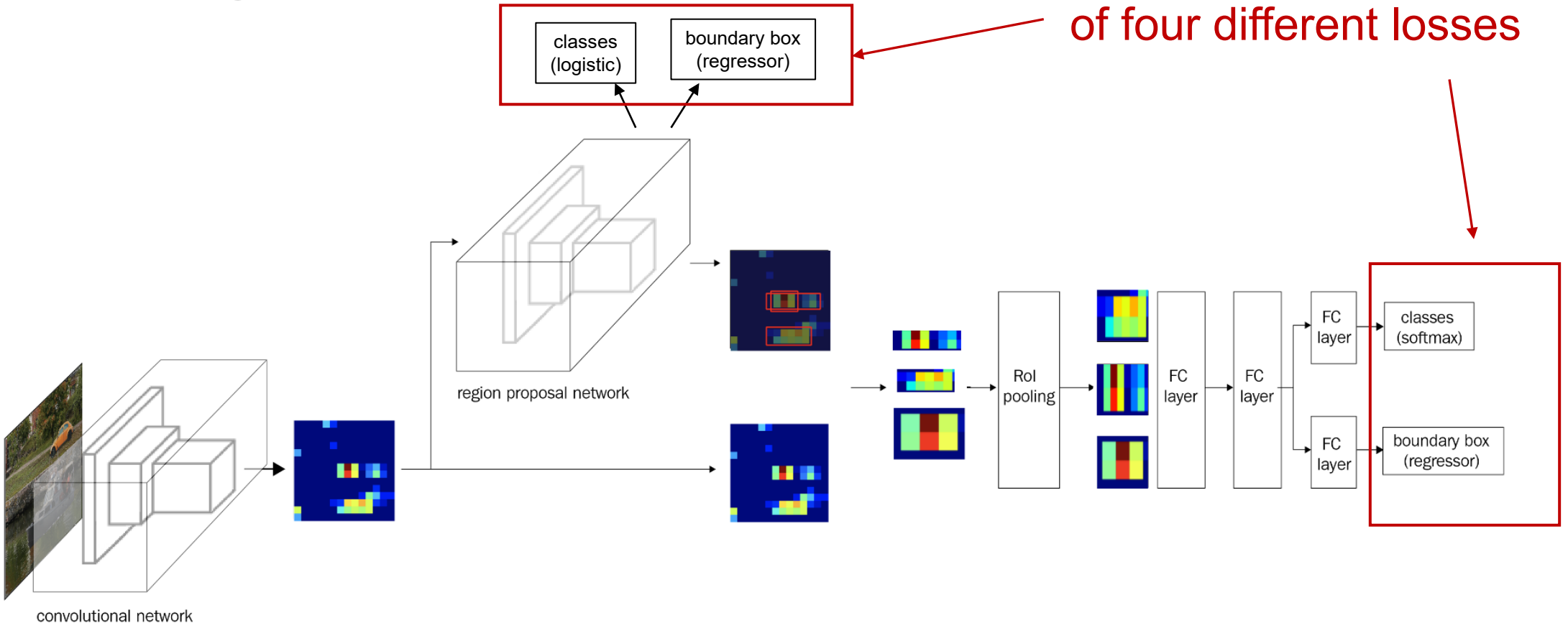




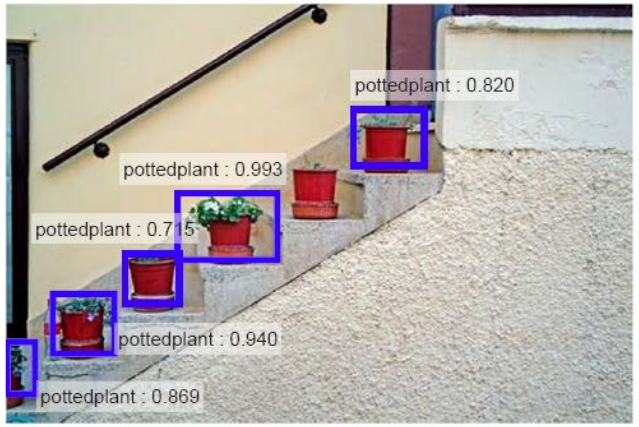
R-CNN architecture

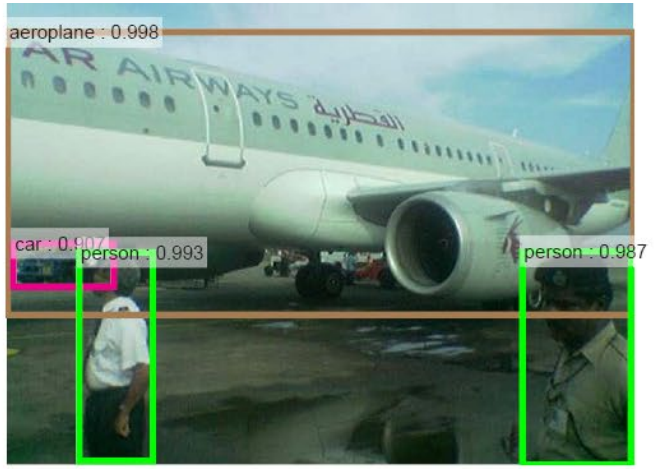
Training Faster R-CNN:

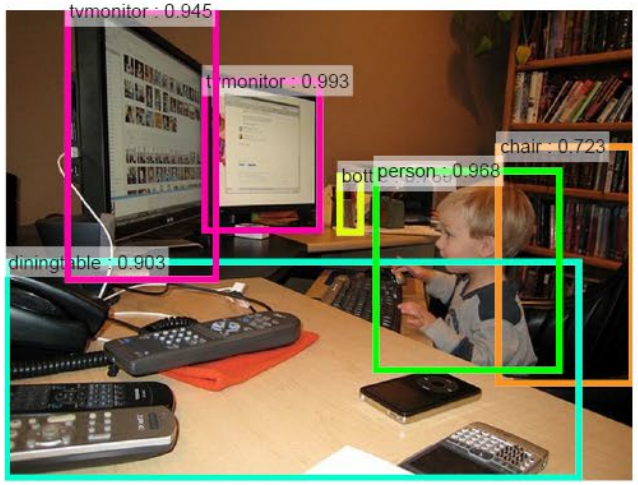
Minimize a weighted sum of four different losses

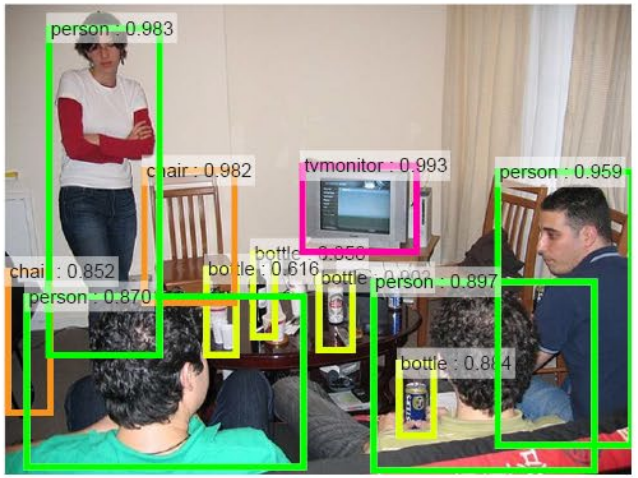


Performance of Faster R-CNN





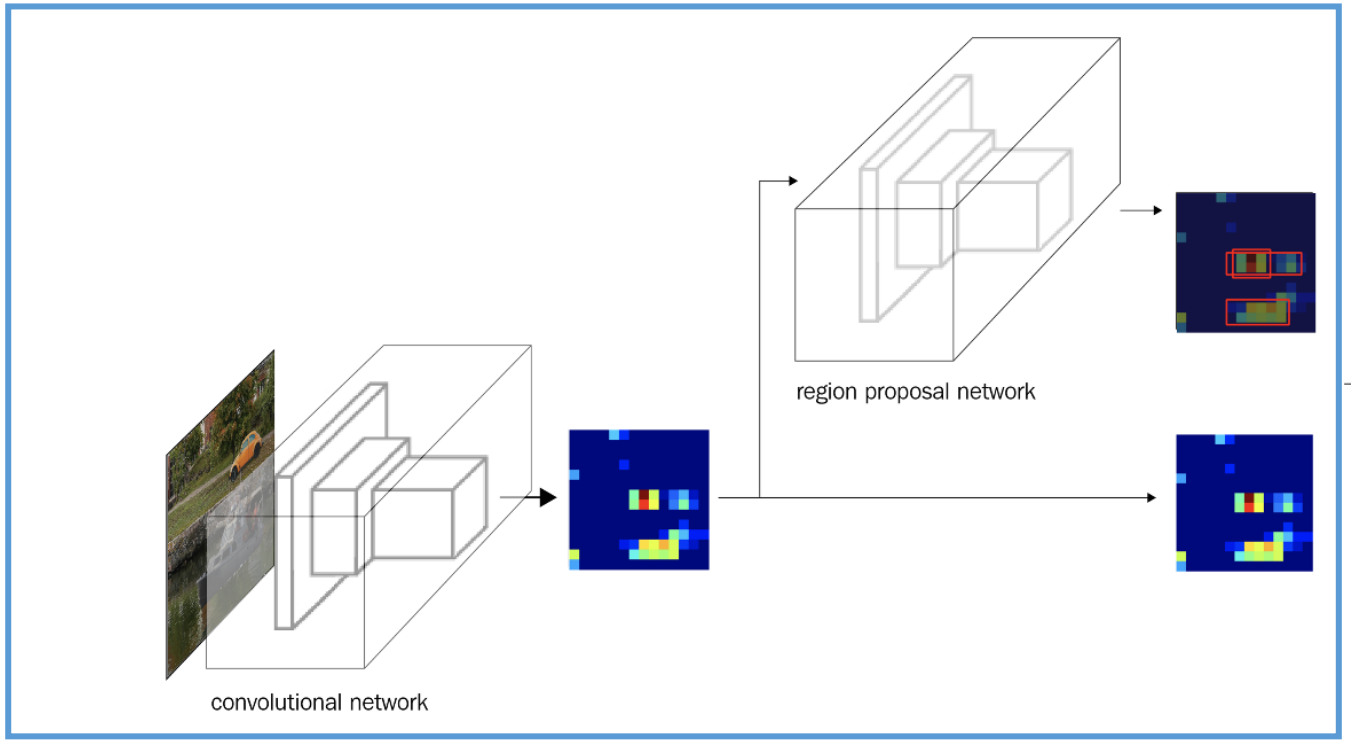


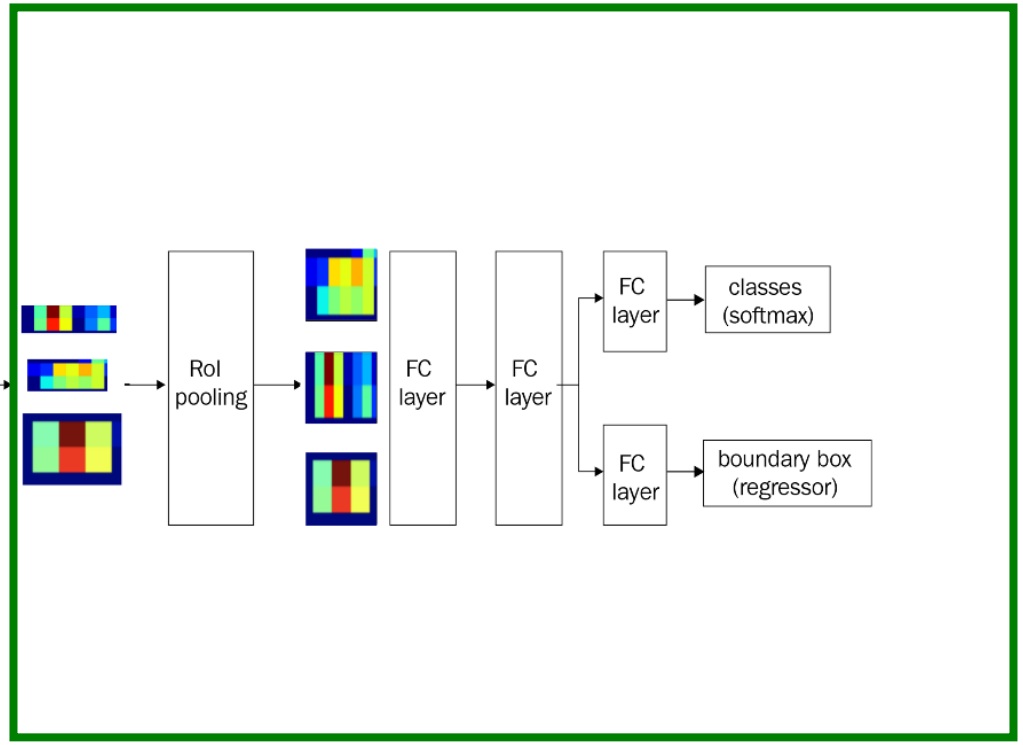


Can we complete what & where in one stage?

Two Stage Object Detector

- Stage One: Run Once per Image
  - Detect Whether there is an Object?
  - Where is the Object?
- Stage Two: Run Once per region
  - Detect What is the Object?
  - Where is the object





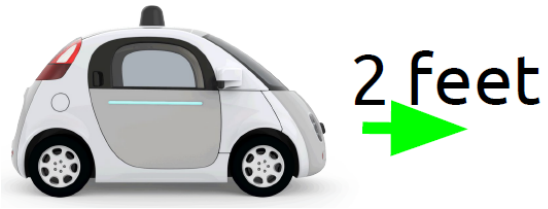
One-Shot Object Detection

One-Shot Object Detection

- Region-based classification is slow; can we do everything in one-pass?
- Two key frameworks:
  - YOLO (You only look once)
  - SSD (Single-shot detection)

Accurate object detection is slow!

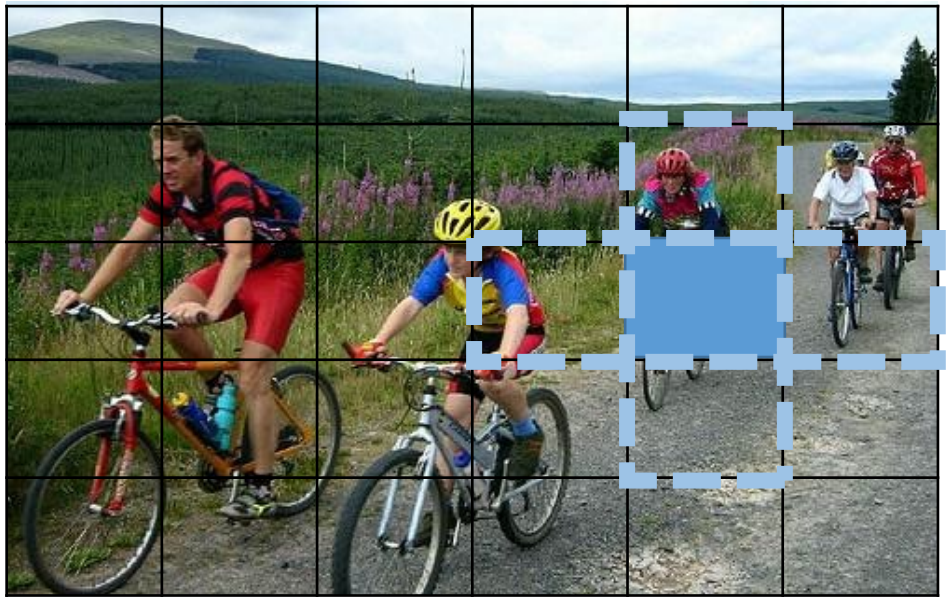
	Pascal 2007 mAP	Speed	
DPM v5	33.7	.07 FPS	14 s/img
R-CNN	66.0	.05 FPS	20 s/img
Fast R-CNN	70.0	.5 FPS	2 s/img
Faster R-CNN	73.2	7 FPS	140 ms/img
YOLO	63.4	45 FPS	22 ms/img



J. Redmon

Real-time Object Detection: YOLO [pdf]

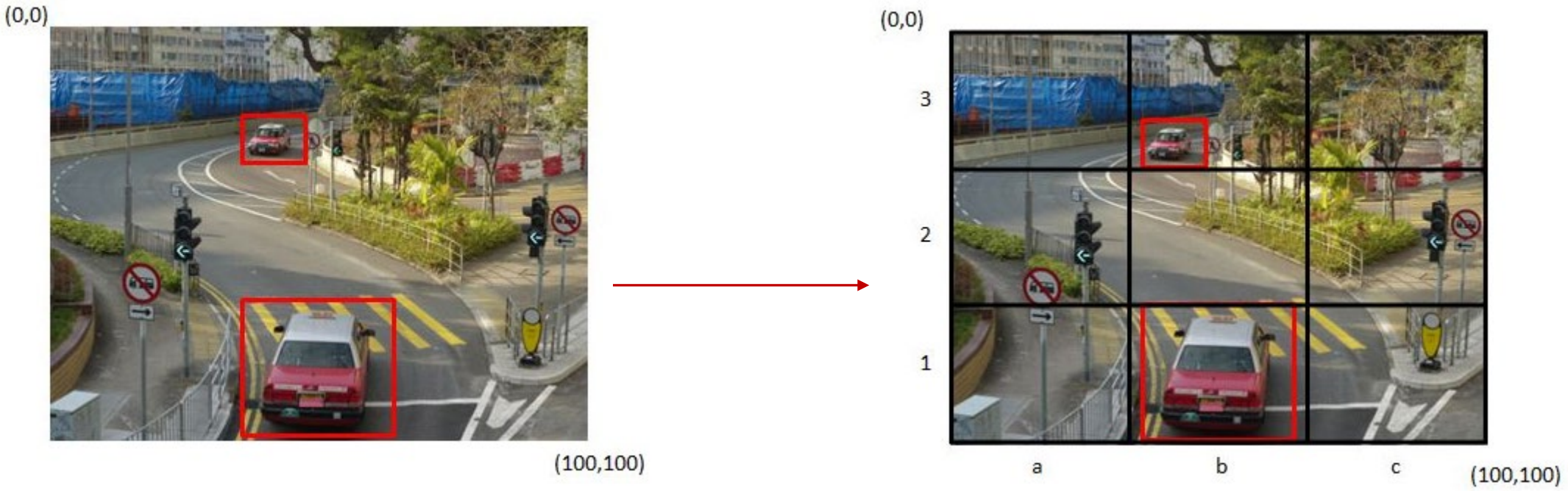
- Partition the image into grids
- Each grid is associated with multiple Anchor Boxes with different aspect ratios:
  - 1:1, 1:3, 3:1, etc.
- Make predictions from each grid location and each aspect ratio



Slide credit: Wang Wei

Partition the Image into grids

- Divide the image into 3 x 3 grid cells



Create ground truths output for each grid cell

Contain the centre pixel of a ground truths BB

b1

y =	
	1
	0.5
	0.5
	0.8
	0.95
	0
	1
	0

y=	
	pc
	bx
	by
	bw
	bh
	c1
	c2
	c3

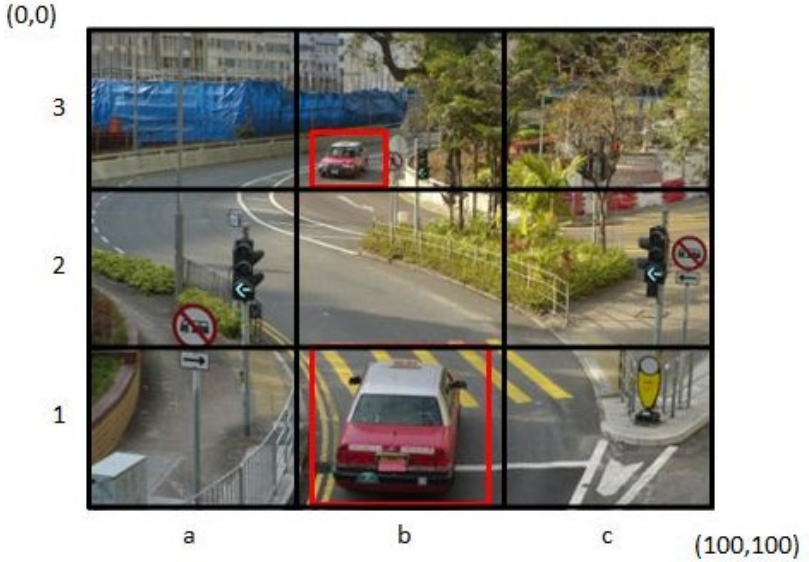
has an object?

truck
car
bus

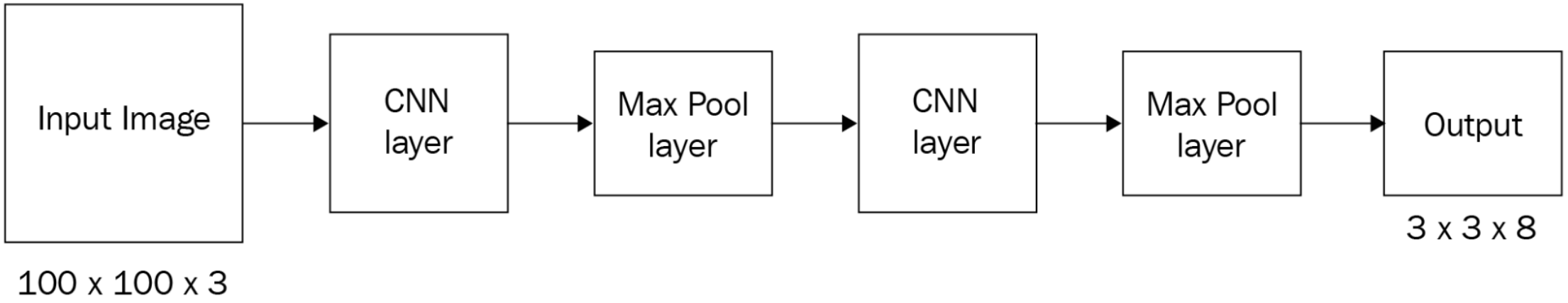
a3

y=	
	0
	?
	?
	?
	?
	?
	?
	?

NOT contain the centre pixel of any ground truths BB



Define a Model

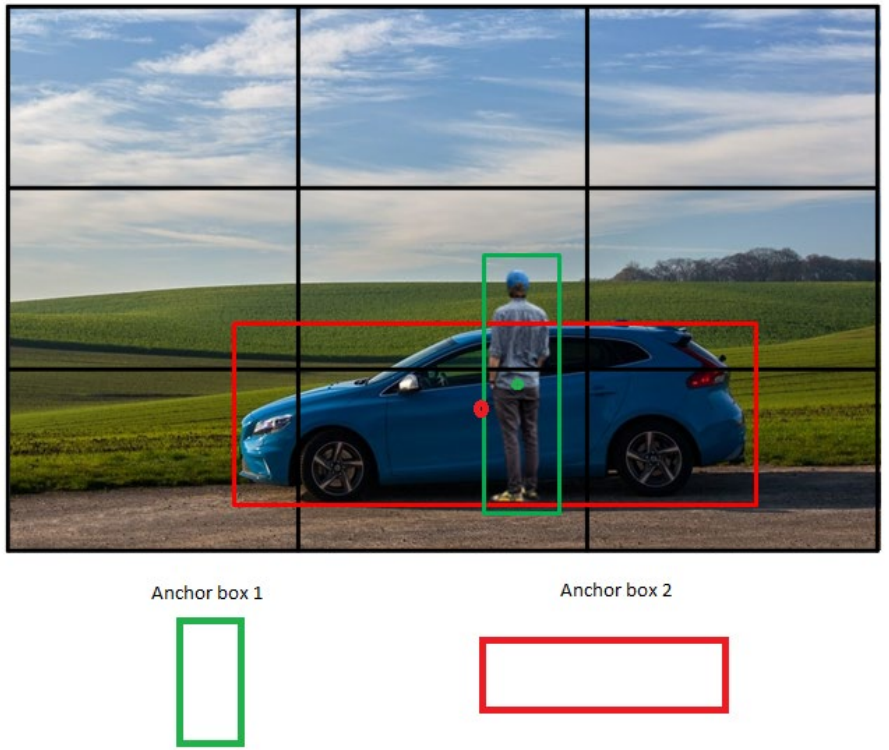


Considering the Anchor Boxes

y =	pc
	bx
	by
	bh
	bw
	c1
	c2
	c3
	pc
	bx
	by
	bh
	bw
	c1
	c2
	c3

Anchor Box 1

Anchor Box 2



Loss Function to Train the Model

grid cells

Ground Truth BB

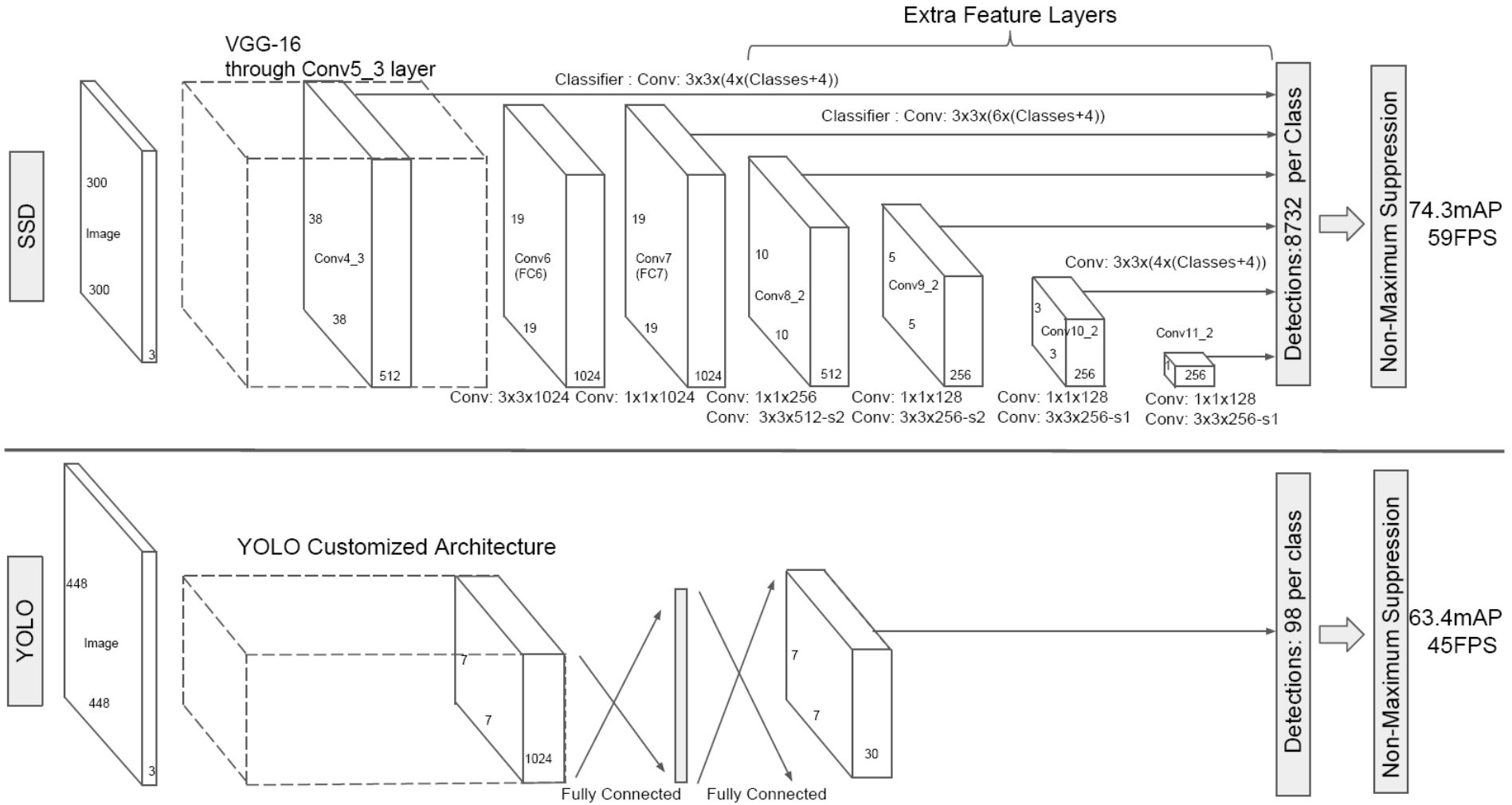
Whether the cell contains an object

$$L_{loc} = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2]$$

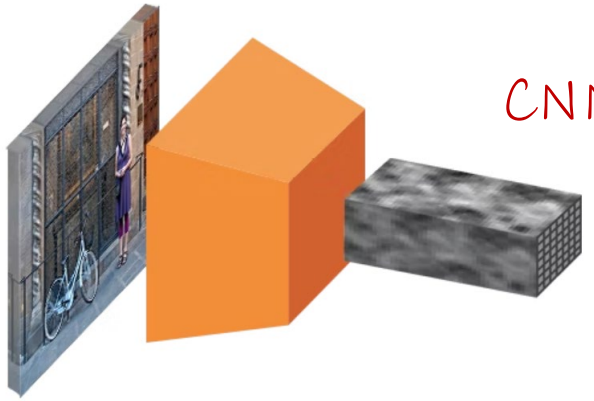
$$L_{cls} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} (1_{ij}^{obj} + \lambda_{noobj}(1 - 1_{ij}^{obj}))(C_{ij} - \hat{C}_{ij})^2 + \sum_{i=0}^{S^2} \sum_{c \epsilon C} 1_i^{obj} (p_i(c) - \hat{p}_i(c))^2$$

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

SSD: Single Shot MultiBox Detector



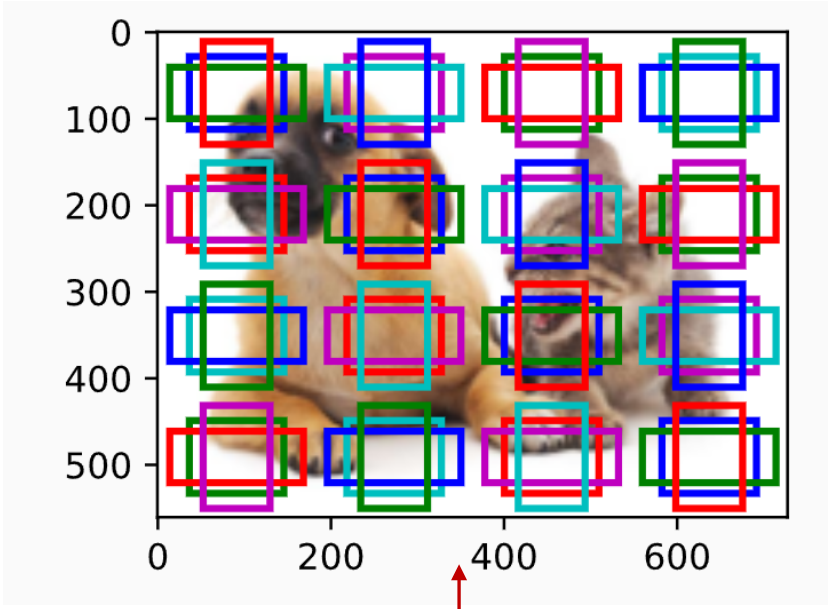
Original Image



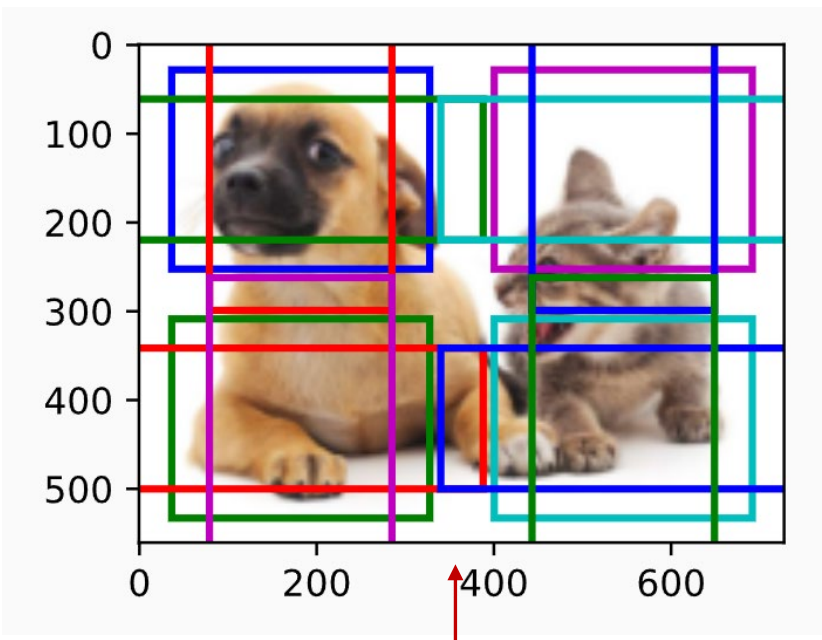
CNN Feature Map

Image to convolutional feature map

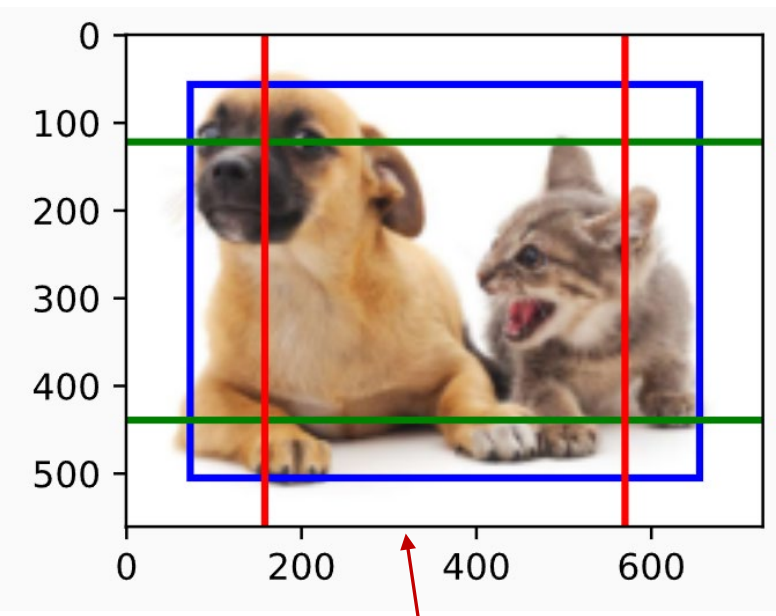
Receptive Fields in the Original Image with different CNN Feature Maps



4 x 4 Feature Map



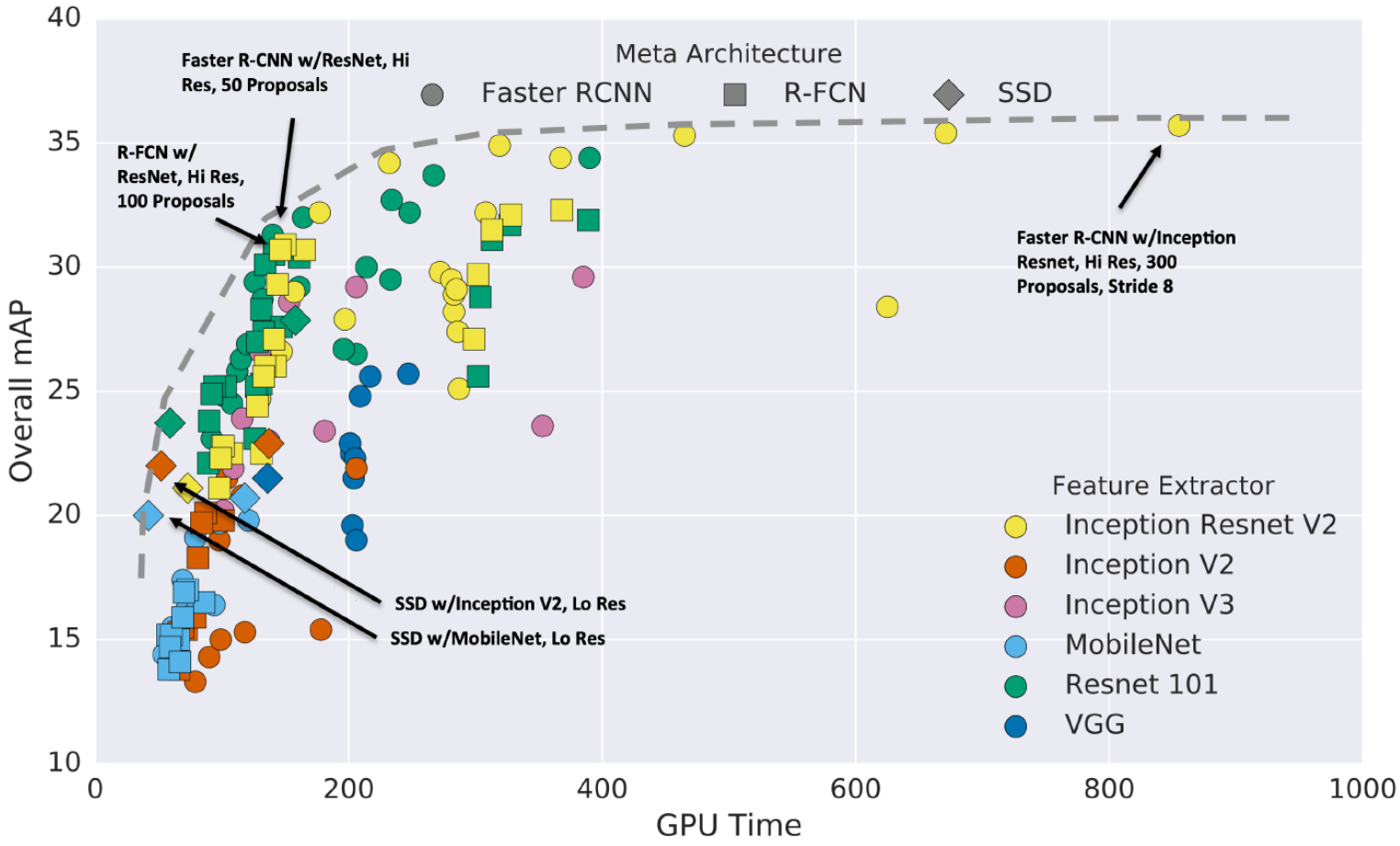
2 x 2 Feature Map



1 x 1 Feature Map

Performance Comparison

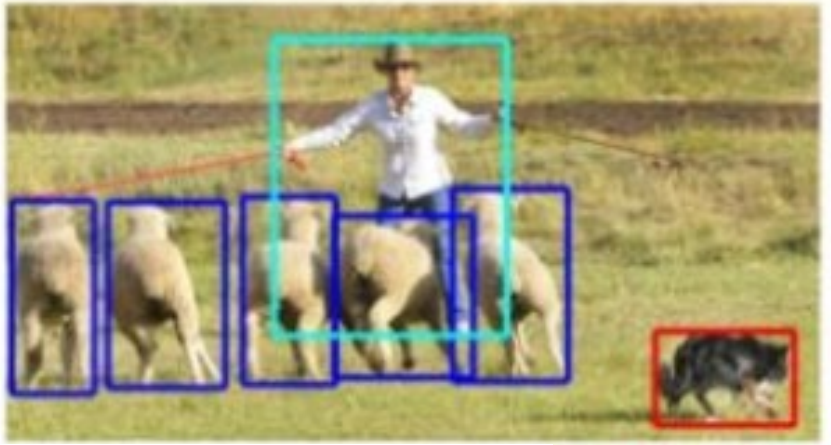
- Two Stage Methods
  - Better Accuracy
  - Slower
- One Stage Methods
  - Faster
  - Less Accurate
- Bigger backbones improve accuracy but slower



Huang et al, "Speed/accuracy trade-offs for modern convolutional object detectors", CVPR 2017

Image Segmentation

(a) Image classification



(b) Object detection

(c) Semantic segmentation

(d) Instance segmentation

Image segmentation

- Label each pixel with a class
- Training label
  - A class (index) per pixel
- Prediction output
  - For each pixel, a probability vector (one per class)

Training label:
(0, 0) background …
(200, 80) people (200, 81) people

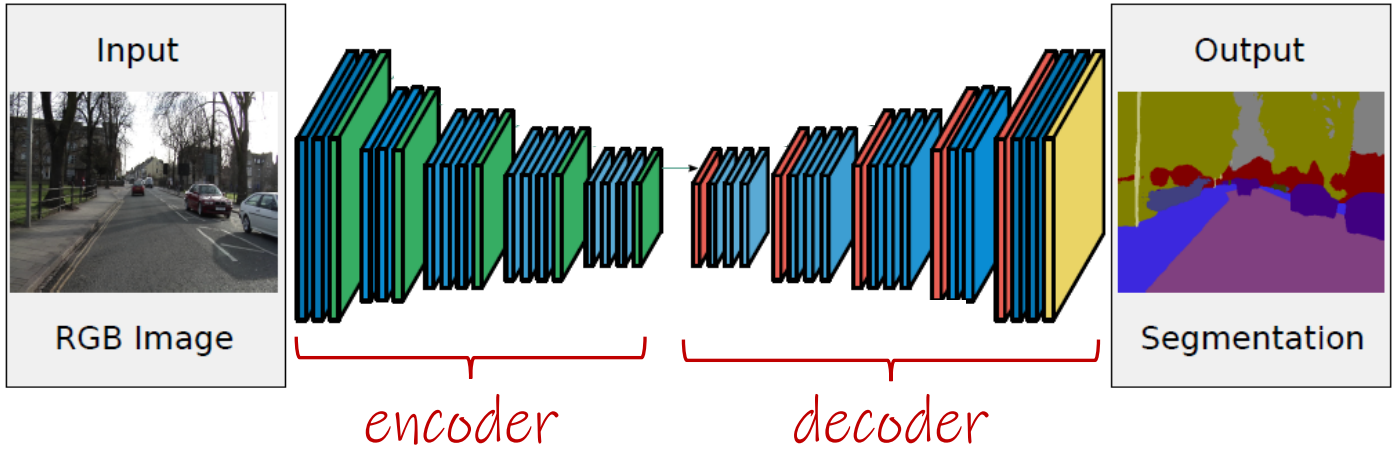
Prediction output:
(0, 0) background 0.9; mountain: 0.1 …
(200, 80) people 0.8; bicycle 0.1; tree: 0.1

Slide credit: Wang Wei

Encoder-Decoder Paradigm

- Encoders
  - "read" inputs into (hidden) feature representations *h*
  - inputs can be of variable size / length, while hidden feature representations are usually of fixed dimensions
- Decoders
  - "write" outputs conditioned on *h*
  - transforms fixed length representations into (variable) sized outputs
- Examples:
  - RNNs: sequence-to-sequence tasks such as translation
  - CNNs: image-to-image mappings e.g. segmentation!

Image Encoding / Decoding



- Encoder
  - extracts a semantic-rich representation which can be used directly for label prediction
  - Subsampling by (pooling or convolution with stride > 1)
- Decoder
  - incorporates location information for a learned form of up-sampling
  - generates a final feature map as the same size as the input
- Loss: pixel-wise loss (softmax for multiple classification)

Adapted from Wang Wei

U-Net

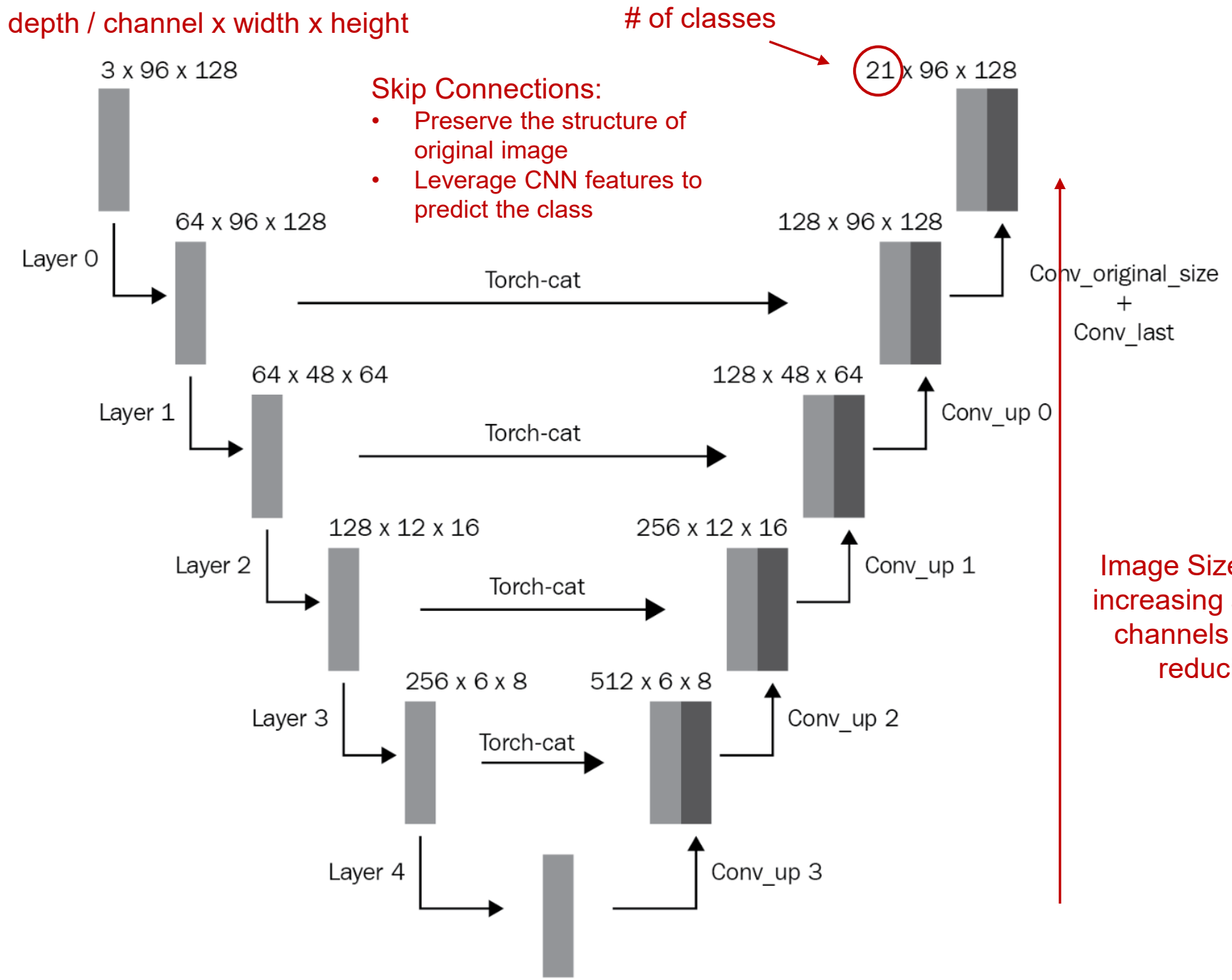
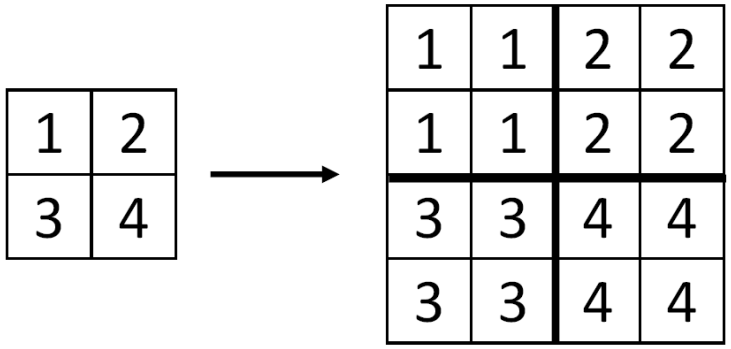


Image Size keeps reducing and # of channels keeps increasing

Image Size keeps increasing and # of channels keeps reducing

Upsampling

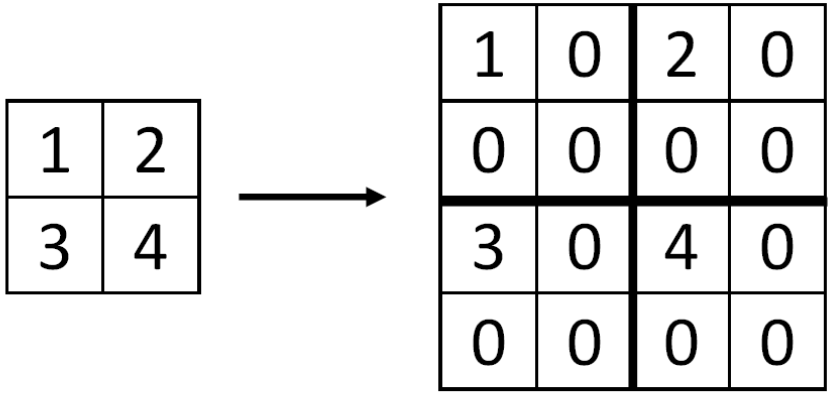
Nearest Neighbor



Input
C x 2 x 2

Output
C x 4 x 4

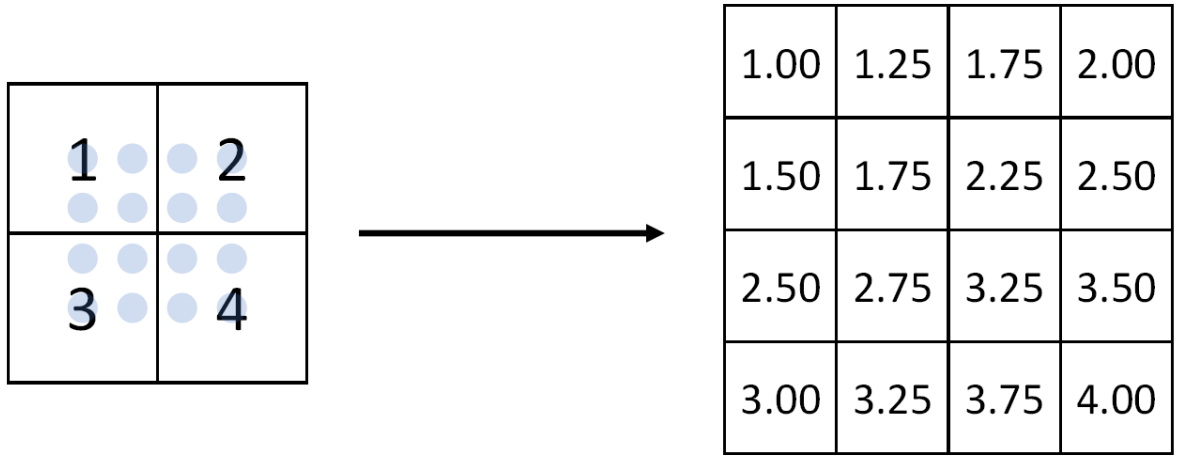
Bed of Nails



Input
C x 2 x 2

Output
C x 4 x 4

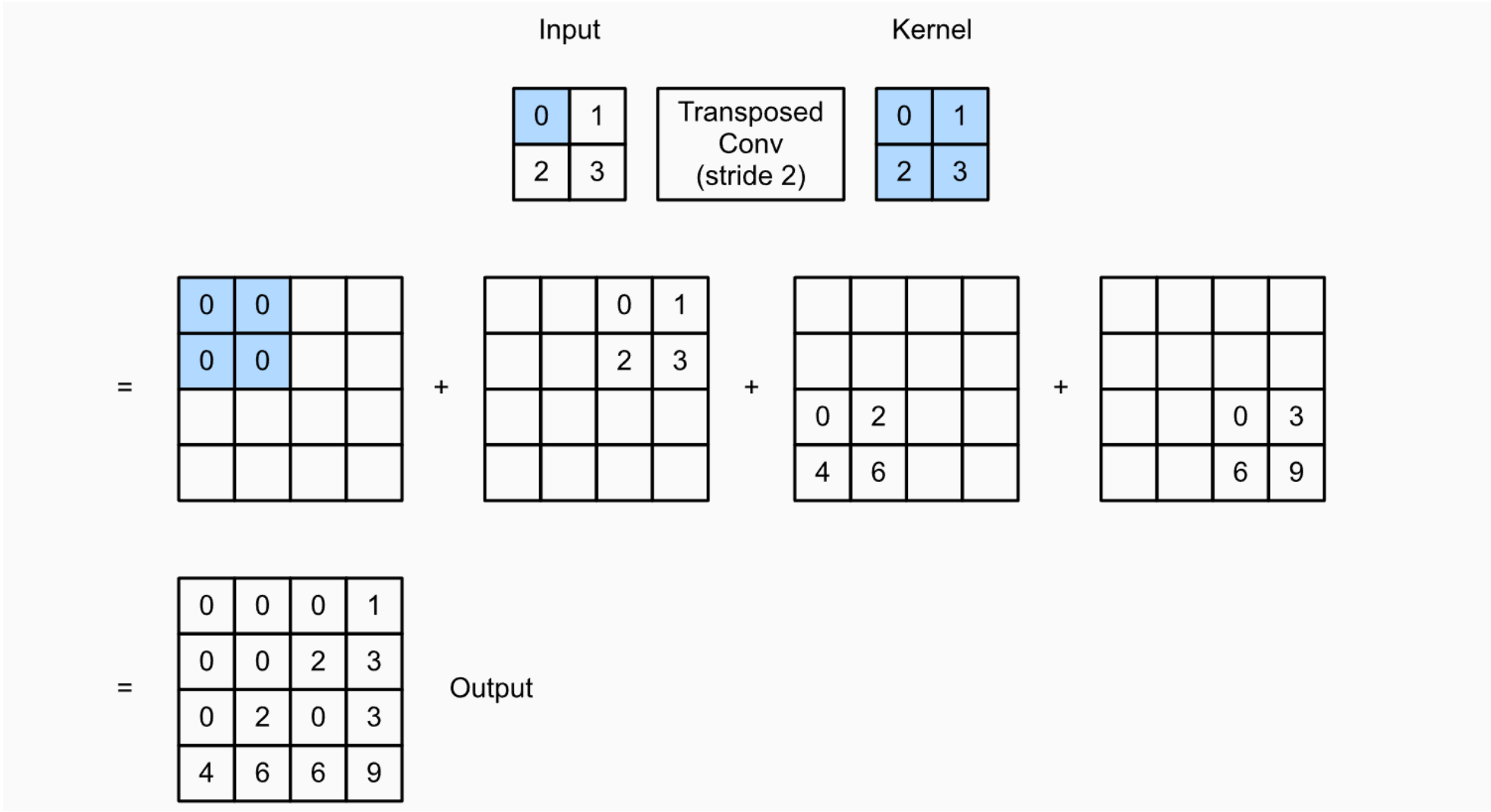
Bilinear Interpolation



Input: C x 2 x 2

Output: C x 4 x 4

Learnable Upsampling: Transposed Convolution



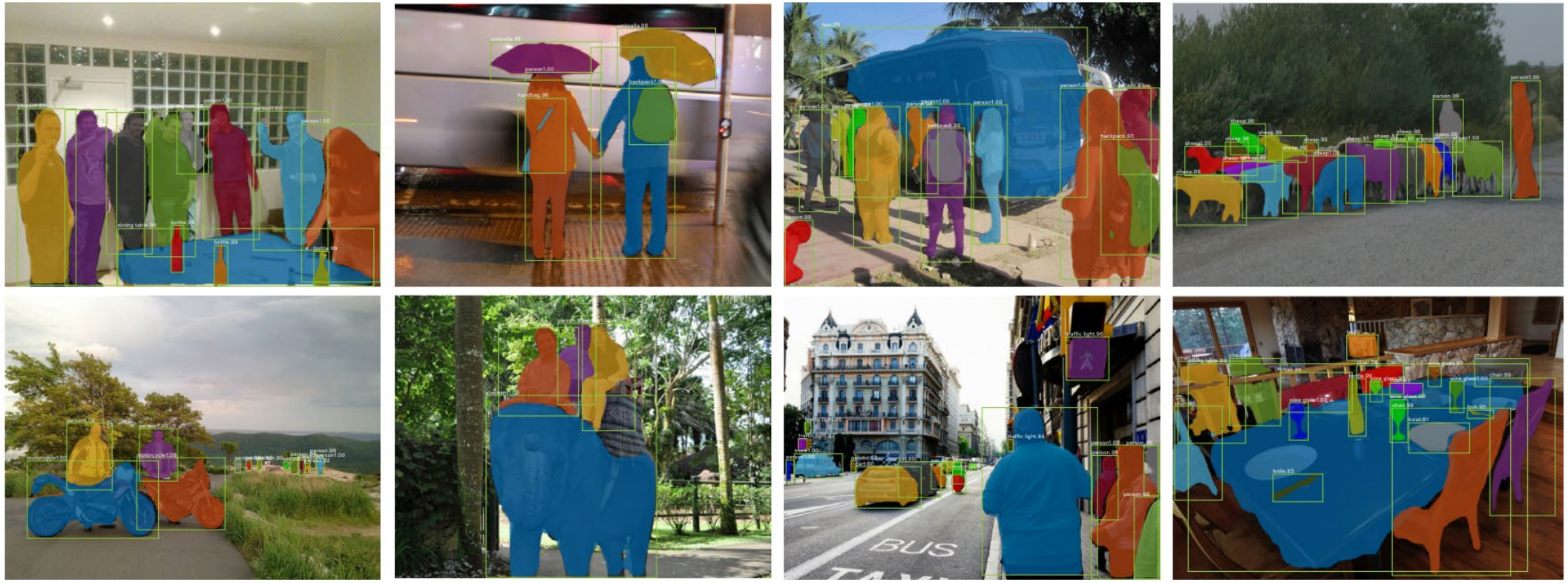
U-Net Performance

How to differentiate the different persons in the image?

Mask R-CNN

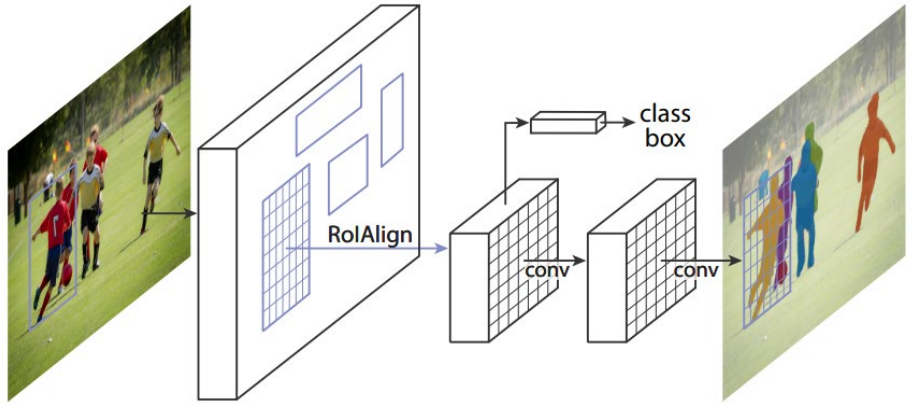
Mask R-CNN

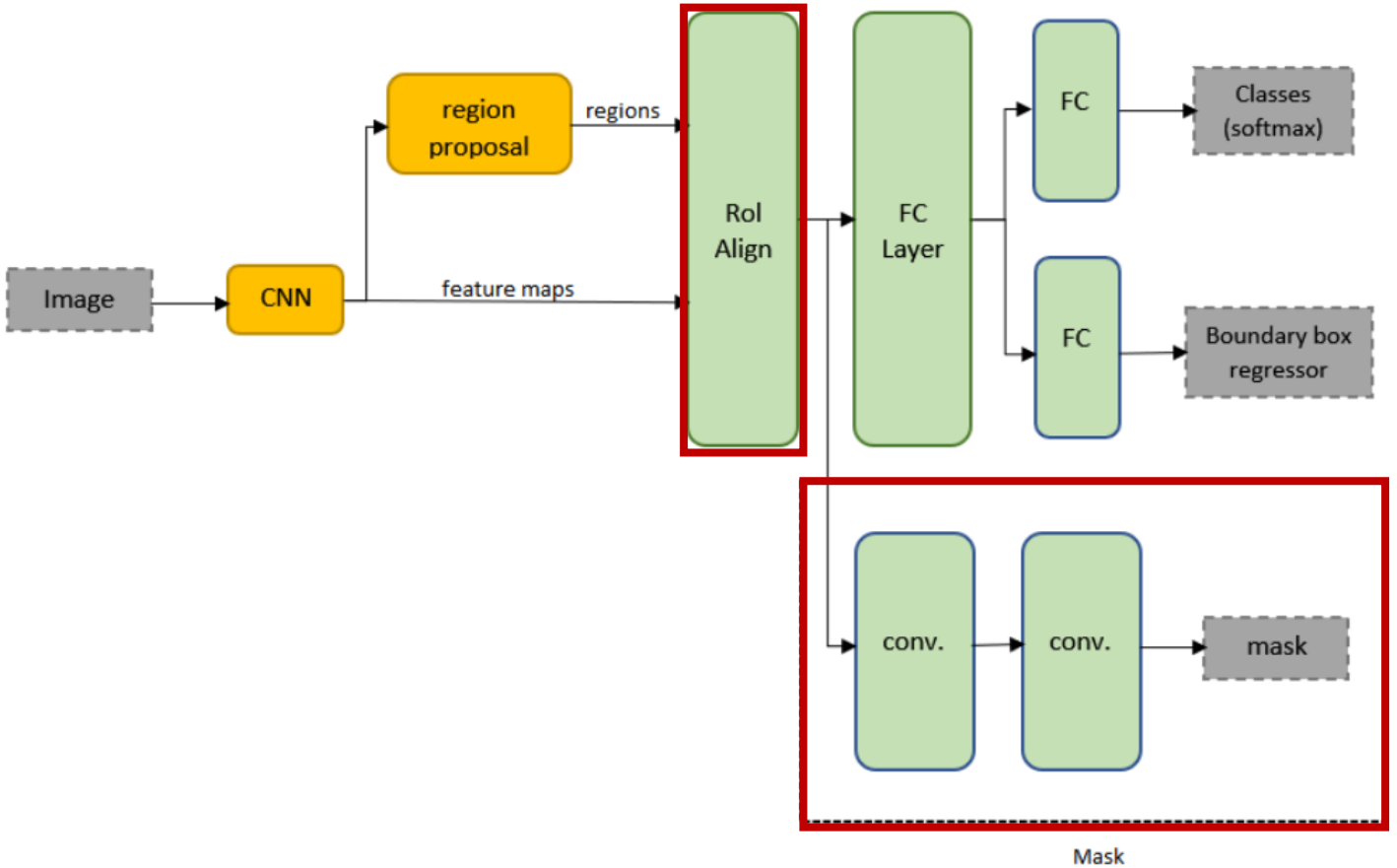
- For any class, identify all the instances of objects in the image
- Mask: the segmentation is done at the pixel level for each instance



Mask R-CNN architecture

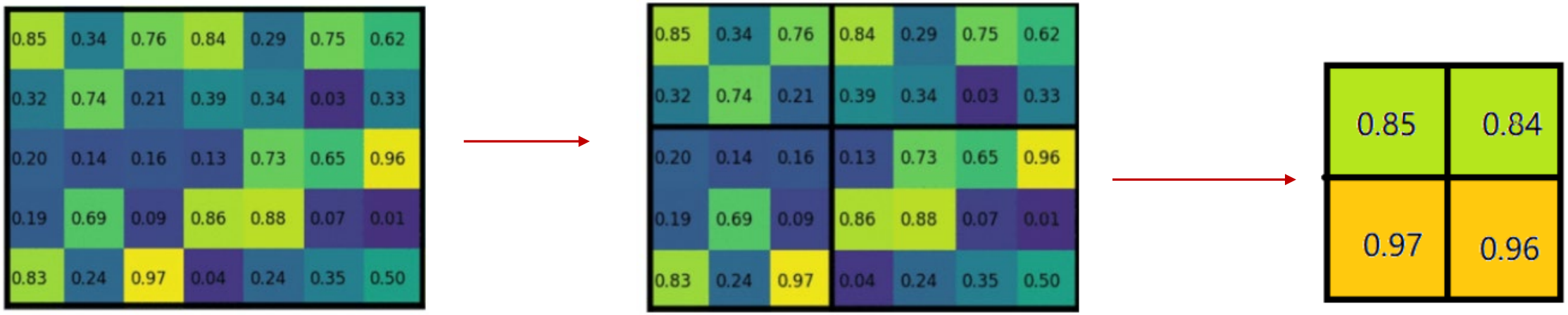
- RoI Pooling layer is replaced by RoI Aligh layer
- A mask head has been added to predict a mask of objects
- A fully convolutional network (FCN) is leveraged for mask prediction





RoI Pooling

- RoI Pooling tends to loose information
- Certain parts of the region have more weight than others



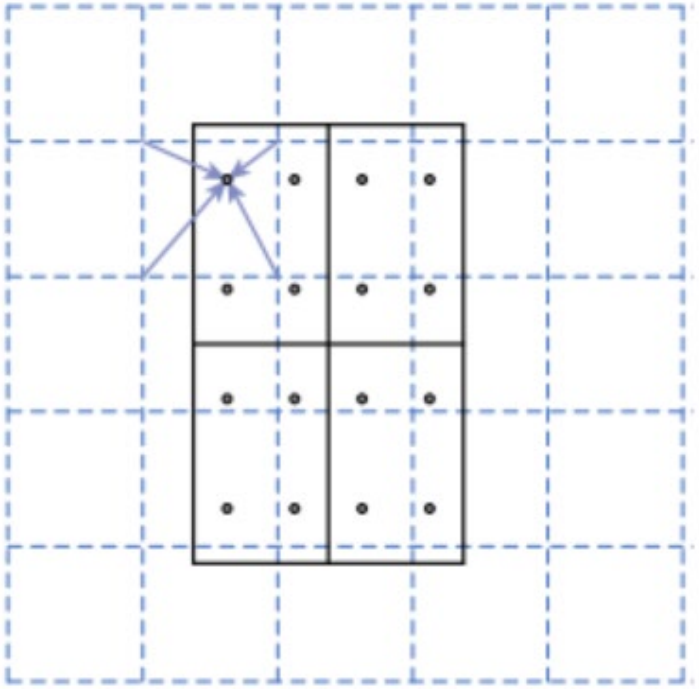
CNN Feature

Divide the Feature into Sections

Max Pooling

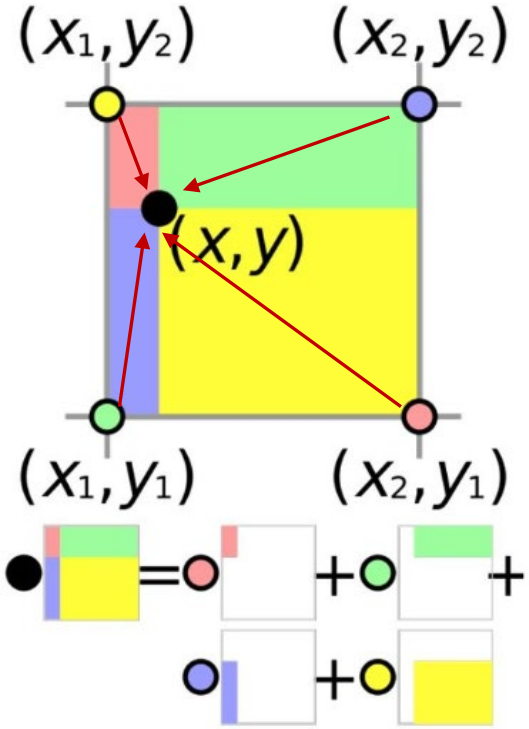
RoI Align

- Get a more accurate representation of region proposal from RPN



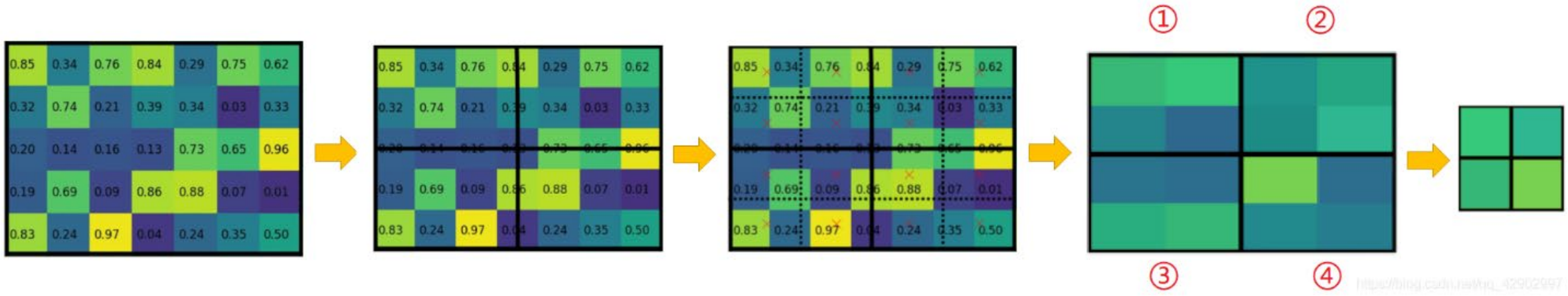
1. Divide the region into equal sections

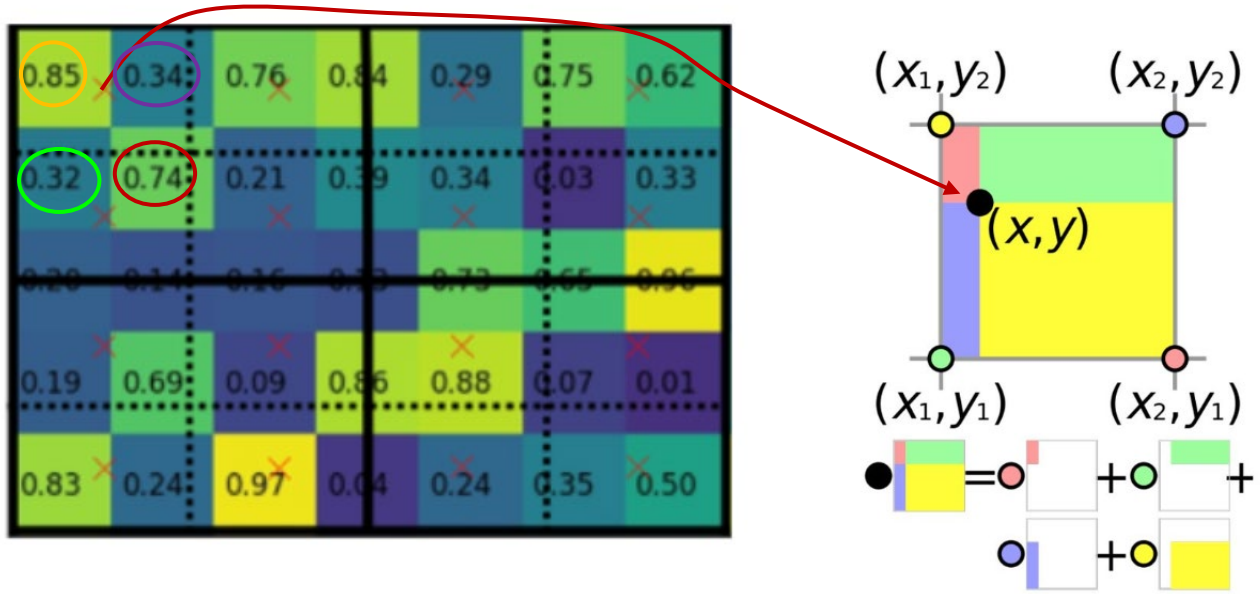
2.Define four points that are equally spaced within each cell



3. Calculate the value of each point using bilinear interpolation

RoI Align: an example

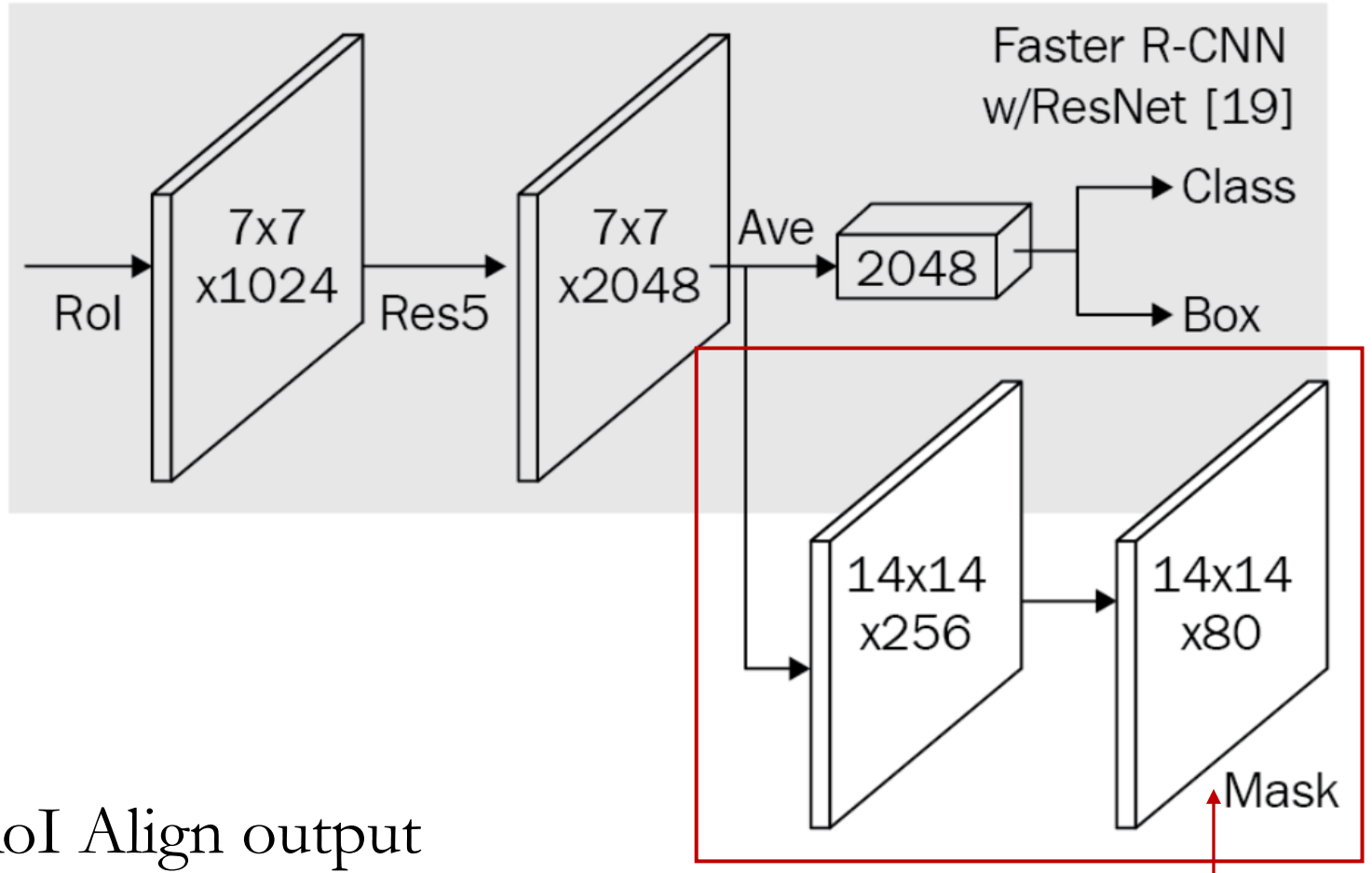




Assume Coordinates:
(x1, y1) = (5, 5)
(x2, y2) = (6, 6)
(x, y) = (5.3, 5.7)

Pixel Value at (x, y) =
0.74 * 0.3 * 0.5 +
0.32 * 0.3 * 0.5 +
0.34 * 0.5 * 0.7 +
0.85 * 0.5 * 0.7

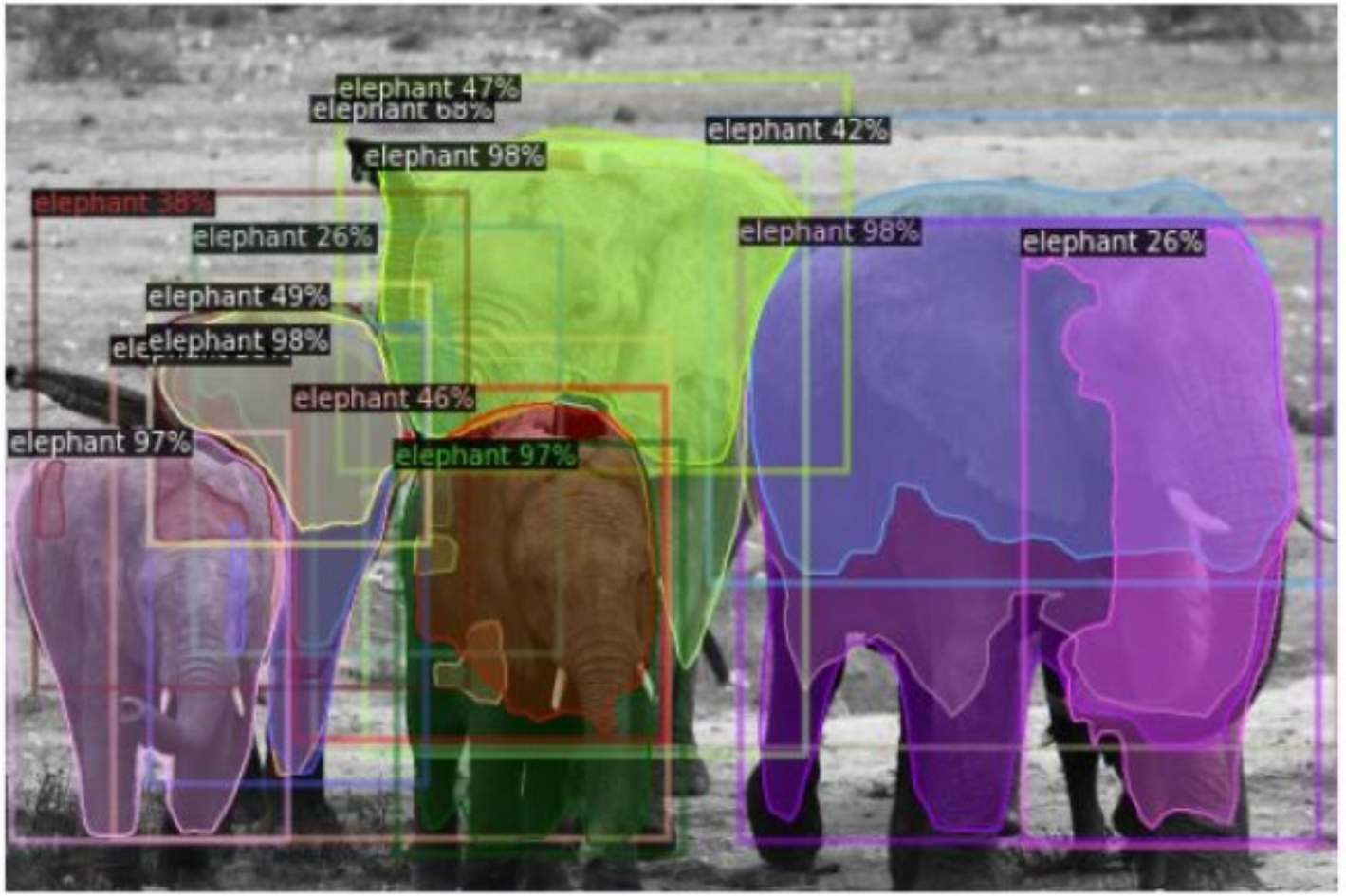
Mask Head



- For every region proposal
- Given a standard shaped RoI Align output
- Obtain the segmentation (mask) output

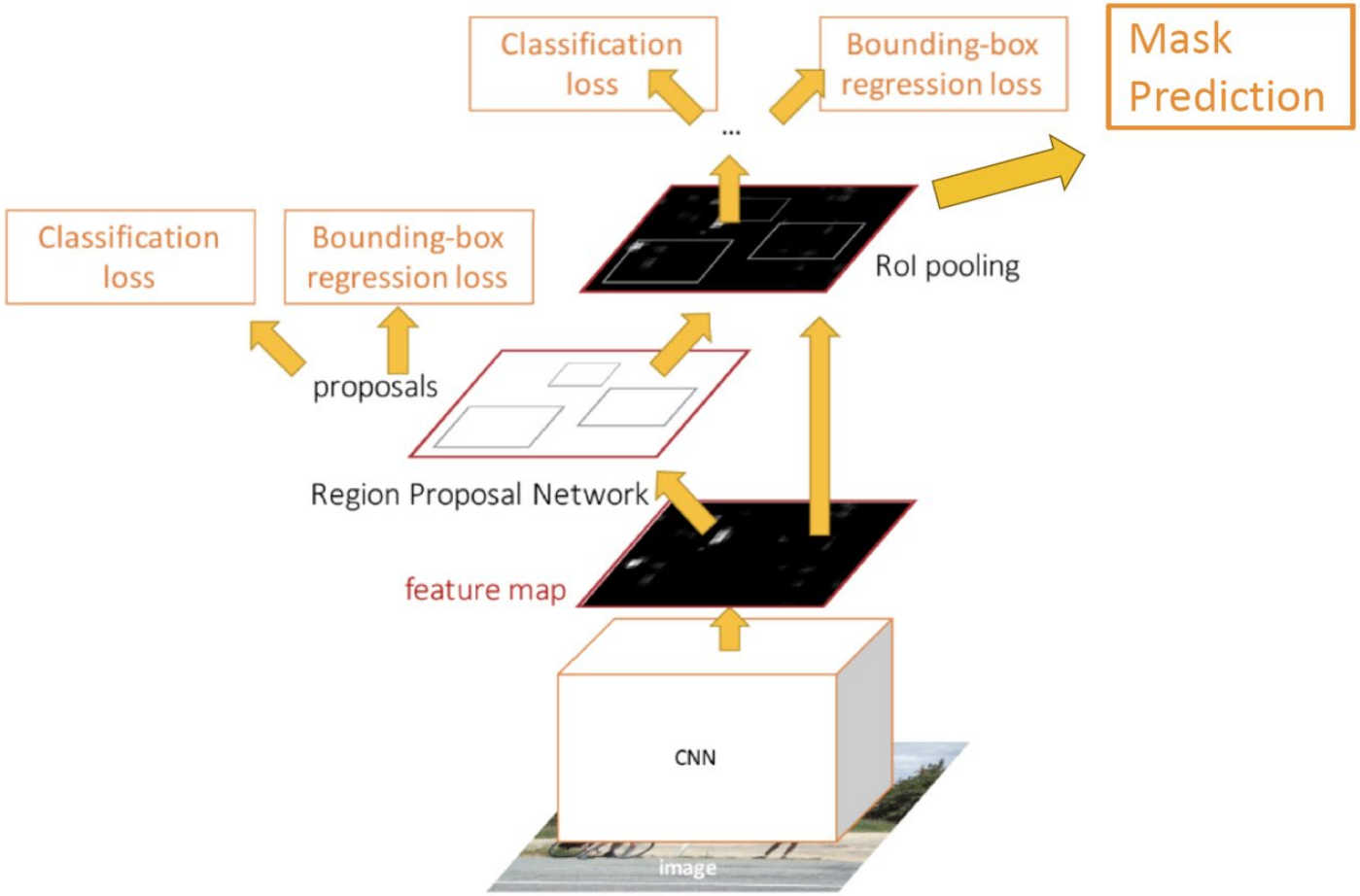
Predict 80 unique classes at pixel level

Mask R-CNN Performance



Summary

- Object detection
  - Region Based Detector
    - R-CNN
    - Fast R-CNN
    - Faster R-CNN
  - Single Stage
    - YOLO
    - SSD
- Image segmentation
  - U-Net
  - Mask R-CNN



Reference

- R-CNN Paper https://arxiv.org/abs/1311.2524
- Fast R-CNN Paper https://arxiv.org/abs/1504.08083
- Faster R-CNN Paper https://arxiv.org/abs/1506.01497
- Yolo Paper https://arxiv.org/abs/1506.02640
- SSD Paper https://arxiv.org/abs/1512.02325
- U-Net Paper https://arxiv.org/pdf/1505.04597.pdf
- Mask R-CNN Paper https://arxiv.org/abs/1703.06870
- Some slides are taken from Stanford Course CS231n