

Deep Learning for Computer Vision

# CNNs for Object Detection

Vineeth N Balasubramanian

Department of Computer Science and Engineering  
Indian Institute of Technology, Hyderabad



# Classification vs Localization vs Detection

## Classification



CAT

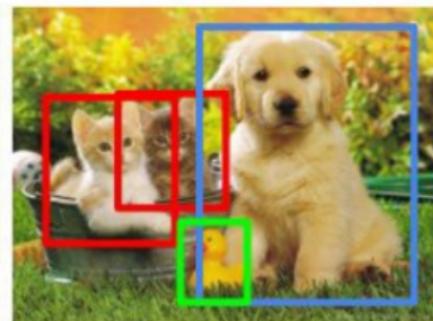
## Single object

## Classification + Localization



CAT

## Object Detection



## CAT, DOG, DUCK

## Instance Segmentation

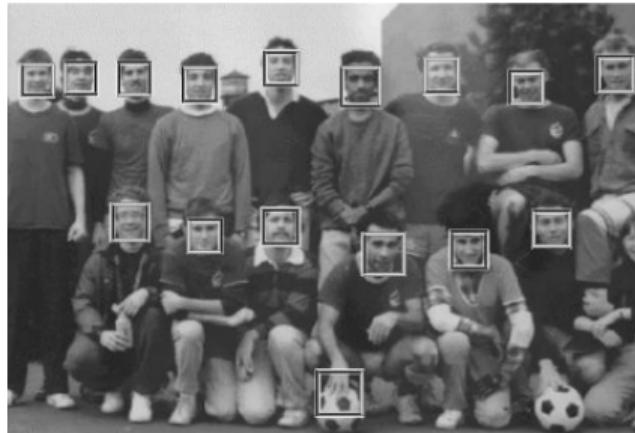


CAT, DOG, DUCK

*Credit: Mike Tamir, SIG & UC Berkeley*

# Detection in the Pre-Deep Learning Era: Viola-Jones Algorithm<sup>1</sup>

- A framework to perform object detection in real-time
- Used predominantly for detecting frontal upright faces.
- Consists of three main components:
  - Weak classification using **Haar-like features** and **Integral Images**
  - Adaboost to build strong classifier
  - Classifier cascades



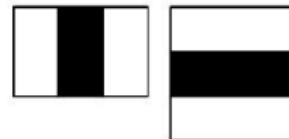
<sup>1</sup>Viola, Jones, Rapid Object Detection using a Boosted Cascade of Simple Features, CVPR 2001

## Haar-like Features

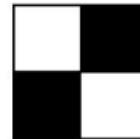
- Rectangular features based on Haar wavelets
  - Feature value given by sum of pixels in white rectangles subtracted from sum in black rectangles, i.e.
$$f = \sum(\text{pixels inside black rectangle}) - \sum(\text{pixels inside white rectangle})$$
  - To normalize for images of all scales, features are scaled through height and width



(a) Edge Features

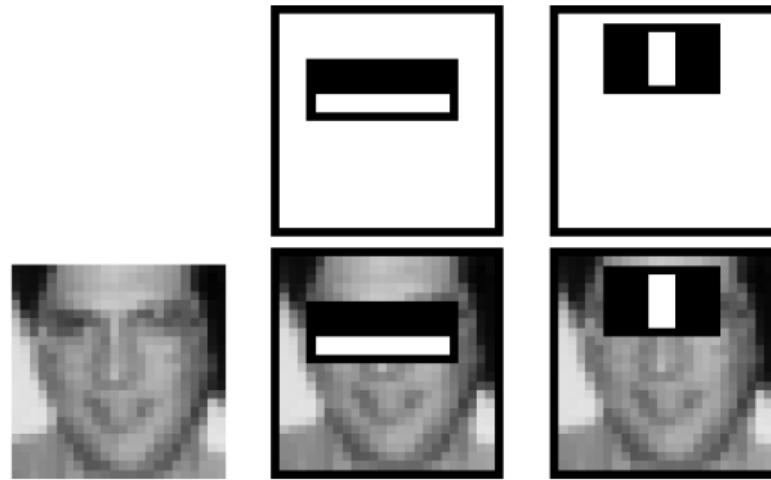


### (b) Line Features



### (c) Four-rectangle features

## Haar Features: Intuition



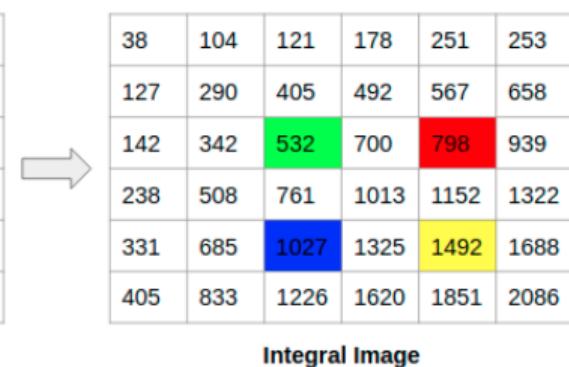
# Integral Image

- Reduces computational complexity caused while adding pixel intensities for any image operation
- For image  $i$ , Integral image  $ii$  is given by:

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y')$$

38	66	17	57	73	2
89	97	98	30	2	89
15	37	75	81	23	50
96	70	63	84	41	29
93	84	89	46	28	26
74	74	51	96	64	39

Image



38	104	121	178	251	253
127	290	405	492	567	658
142	342	532	700	798	939
238	508	761	1013	1152	1322
331	685	1027	1325	1492	1688
405	833	1226	1620	1851	2086

Integral Image

# Integral Image

- Reduces computational complexity caused while adding pixel intensities for any image operation
- For image  $i$ , Integral image  $ii$  is given by:

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y')$$

38	66	17	57	73	2
89	97	98	30	2	89
15	37	75	81	23	50
96	70	63	84	41	29
93	84	89	46	28	26
74	74	51	96	64	39

Image

38	104	121	178	251	253
127	290	405	492	567	658
142	342	532	700	798	939
238	508	761	1013	1152	1322
331	685	1027	1325	1492	1688
405	833	1226	1620	1851	2086

Integral Image

- Recurrent definition of  $ii(x, y)$ :

$$s(x, y) = s(x, y - 1) + i(x, y)$$

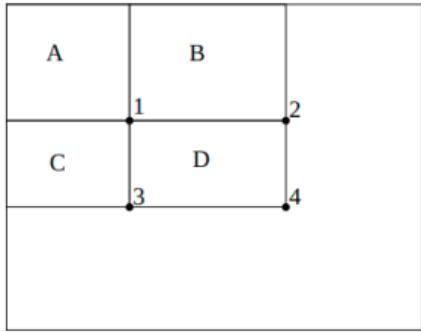
$$ii(x, y) = ii(x - 1, y) + s(x, y)$$

where  $s(x, y)$  is cumulative row sum

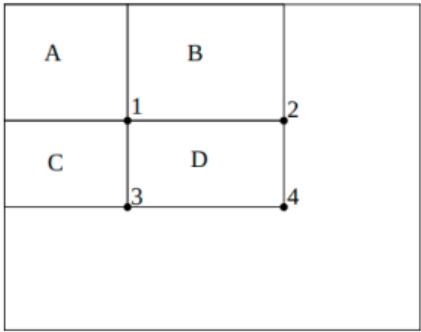
$$s(x, -1) = 0, ii(-1, y) = 0$$

# Integral Image

- Sum of pixels within rectangle 1234 = D+A-(B+C)  
(Why?)



# Integral Image



- Sum of pixels within rectangle 1234 = D+A-(B+C) (Why?)
- More formally, sum of pixels in rectangle with upper-left pixel as  $(x_1, y_1)$  and bottom-right pixel as  $(x_2, y_2)$  is given by:

$$\text{Sum} = ii(x_2, y_2) + ii(x_1 - 1, y_1 - 1) - (ii(x_1 - 1, y_2) + ii(x_2, y_1 - 1))$$

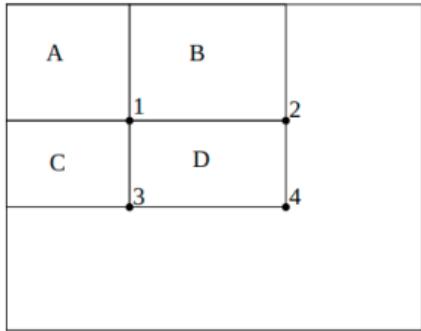
38	66	17	57	73	2
89	97	98	30	2	89
15	37	75	81	23	50
96	70	63	84	41	29
93	84	89	46	28	26
74	74	51	96	64	39

Sum of pixels = 199

38	104	121	178	251	253
127	290	405	492	567	658
142	342	532	700	798	939
238	508	761	1013	1152	1322
331	685	1027	1325	1492	1688
405	833	1226	1620	1851	2086

Sum of pixels = 1492 + 532 - 1027 - 798 = 199

# Integral Image



- Sum of pixels within rectangle 1234 = D+A-(B+C) (Why?)
- More formally, sum of pixels in rectangle with upper-left pixel as  $(x_1, y_1)$  and bottom-right pixel as  $(x_2, y_2)$  is given by:

$$\text{Sum} = ii(x_2, y_2) + ii(x_1 - 1, y_1 - 1) - (ii(x_1 - 1, y_2) + ii(x_2, y_1 - 1))$$

38	66	17	57	73	2
89	97	98	30	2	89
15	37	75	81	23	50
96	70	63	84	41	29
93	84	89	46	28	26
74	74	51	96	64	39

Sum of pixels = 199

38	104	121	178	251	253
127	290	405	492	567	658
142	342	532	700	798	939
238	508	761	1013	1152	1322
331	685	1027	1325	1492	1688
405	833	1226	1620	1851	2086

Sum of pixels = 1492 + 532 - 1027 - 798 = 199

2-rectangle features now require 6 look-ups. How many lookups do 3-rectangle and 4-rectangle features need? **Homework!**

## Weak Classifiers

- Each feature is considered as a weak classifier
- Given feature  $f_j$ , a threshold  $\theta_j$  and a parity  $p_j$  indicating the direction of the inequality sign, classifier can be defined as:

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases}$$

- Each classifier weak on its own, but combination gives strong classifiers

**Do you see any problem here?**

## Weak Classifiers

- Each feature is considered as a weak classifier
- Given feature  $f_j$ , a threshold  $\theta_j$  and a parity  $p_j$  indicating the direction of the inequality sign, classifier can be defined as:

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases}$$

- Each classifier weak on its own, but combination gives strong classifiers

**Do you see any problem here?**

For a sliding window base detector of size  $24 \times 24$ , there exist over 160k features. How to identify the best ones?

# Classifier Training using AdaBoost

Recall: AdaBoost learns a strong classifier as a linear combination of weak classifiers

- Given example images  $(x_1, y_1), \dots, (x_n, y_n)$  where  $y_i = 0, 1$  for negative and positive examples respectively.
- Initialize weights  $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$  for  $y_i = 0, 1$  respectively, where  $m$  and  $l$  are the number of negatives and positives respectively.
- For  $t = 1, \dots, T$ :
  - Normalize the weights,

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^n w_{t,j}}$$

so that  $w_t$  is a probability distribution.

- For each feature,  $j$ , train a classifier  $h_j$  which is restricted to using a single feature. The error is evaluated with respect to  $w_t$ ,  $\epsilon_j = \sum_i w_i |h_j(x_i) - y_i|$ .

- Choose the classifier,  $h_t$ , with the lowest error  $\epsilon_t$ .
- Update the weights:

$$w_{t+1,i} = w_{t,i} \beta_t^{1-\epsilon_t}$$

where  $e_i = 0$  if example  $x_i$  is classified correctly,  $e_i = 1$  otherwise, and  $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$ .

- The final strong classifier is:

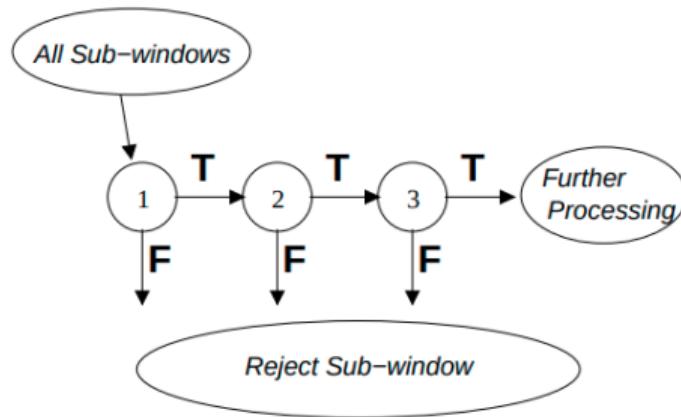
$$h(x) = \begin{cases} 1 & \sum_{t=1}^T \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^T \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

where  $\alpha_t = \log \frac{1}{\beta_t}$

# Classifier Cascade

## Intuition:

- Very less number of sub-windows actually have objects (False positive rate very high)
- But computation is equal across all sub-windows
- Eliminate negative windows earlier thereby, reducing computation time spent on them

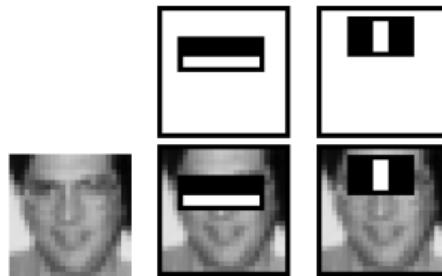


## Classifier Cascade: Why does it matter?

- A 1 MP has  $\sim 10^6$  pixels and a comparable number of candidate face locations  $\implies$  our ideal false positive (FP) rate has to be less than  $10^{-6}$

## Classifier Cascade: Why does it matter?

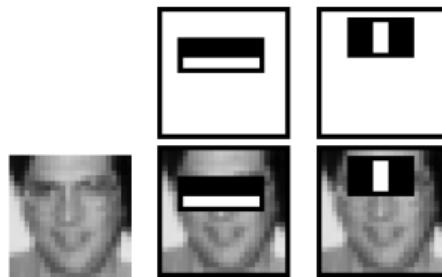
- A 1 MP has  $\sim 10^6$  pixels and a comparable number of candidate face locations  $\implies$  our ideal false positive (FP) rate has to be less than  $10^{-6}$



- This feature combination can yield 100% detection rate and 50% FP rate

## Classifier Cascade: Why does it matter?

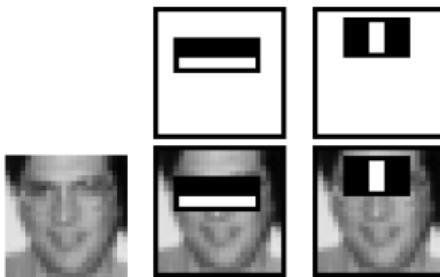
- A 1 MP has  $\sim 10^6$  pixels and a comparable number of candidate face locations  $\implies$  our ideal false positive (FP) rate has to be less than  $10^{-6}$



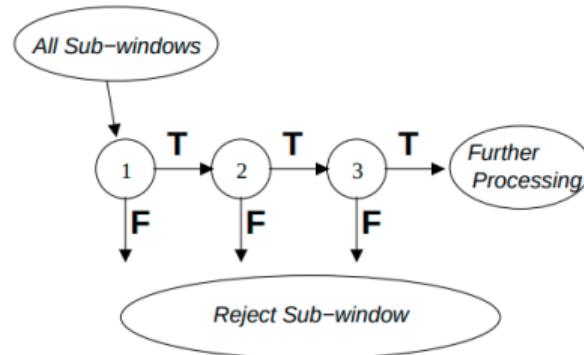
- This feature combination can yield 100% detection rate and 50% FP rate
- A 200-feature classifier can yield 95% detection rate and FP rate of 1 in 14084.  
Not enough!

## Classifier Cascade: Why does it matter?

- A 1 MP has  $\sim 10^6$  pixels and a comparable number of candidate face locations  $\implies$  our ideal false positive (FP) rate has to be less than  $10^{-6}$



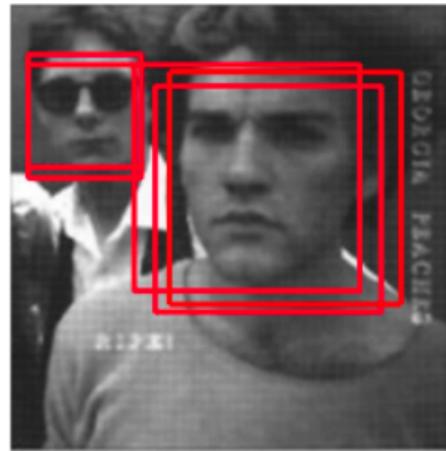
- This feature combination can yield 100% detection rate and 50% FP rate
- A 200-feature classifier can yield 95% detection rate and FP rate of 1 in 14084. Not enough!



- Detection rate and FP rate of cascade are found by multiplying respective rates of individual stages
- Detection rate of 0.9 and FP rate on the order of  $10^{-6}$  can be achieved by a 10-stage cascade if each stage has detection rate of 0.99 ( $0.99^{10} \approx 0.9$ ) and FP rate of about 0.30 ( $0.3^{10} \approx 6 \times 10^{-6}$ )

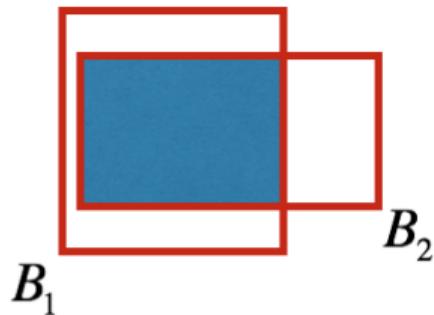
# Non-Maximum Suppression (NMS)

- Usually, more than one bounding box is detected for same object
- Use bounding box similarity measures to identify duplicate bounding boxes which can be removed

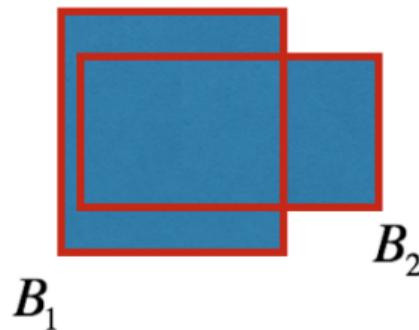


## Intersection over Union (IoU)

Intersection



Union



Intersection over Union

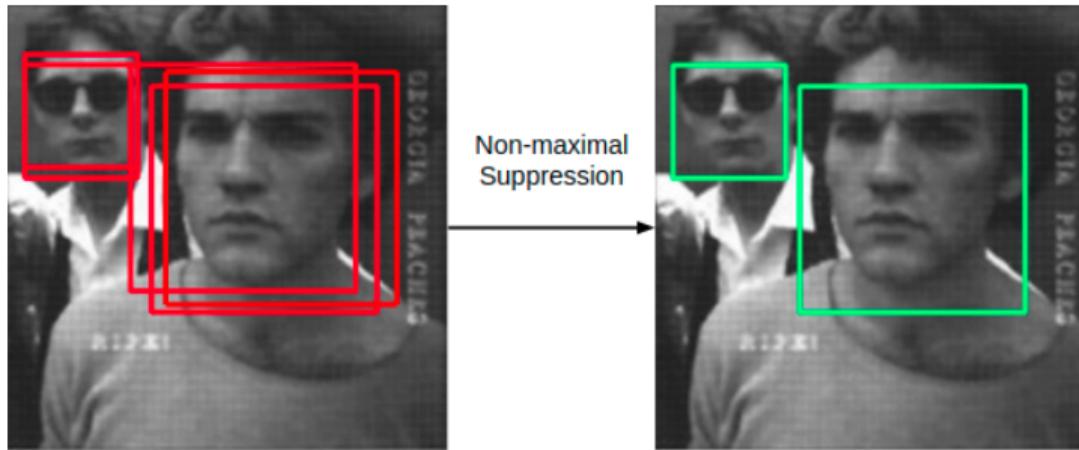
$$IoU = \frac{B_1 \cap B_2}{B_1 \cup B_2} = \frac{\text{Area of intersection}}{\text{Area of union}}$$

A diagram illustrating the calculation of IoU. It shows two overlapping red-outlined rectangles labeled  $B_1$  and  $B_2$ . The intersection of the two boxes is filled with blue. To the right, the total union of the two boxes is shown as a larger red-outlined rectangle, which includes the overlapping area and the non-overlapping parts of both  $B_1$  and  $B_2$ .

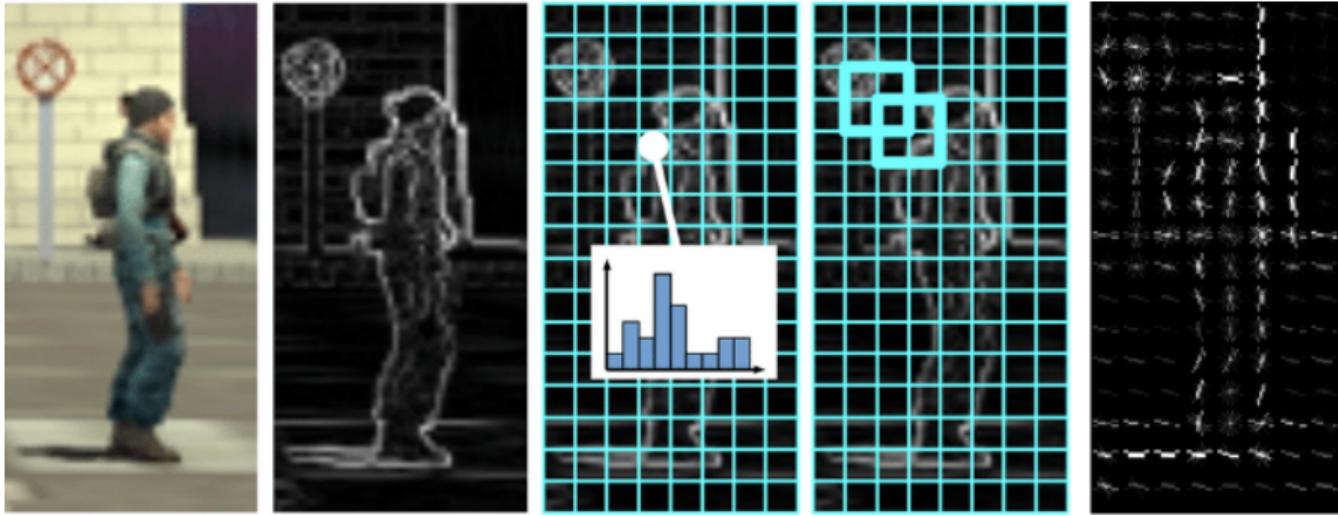
Credit: Hands-On Convolutional Neural Networks with TensorFlow, Iffat Zafar

# Non-Maximum Suppression (NMS)

- Select a random box from a bounding box proposal list
- Compare it with rest of the boxes; if  $\text{IoU} > 0.5$ , remove box from list
- Repeat until boxes left are exclusive



# Histograms of Oriented Gradients<sup>2</sup>



detection window  
slides over an  
image

at each location where  
the window is applied,  
gradients are  
computed

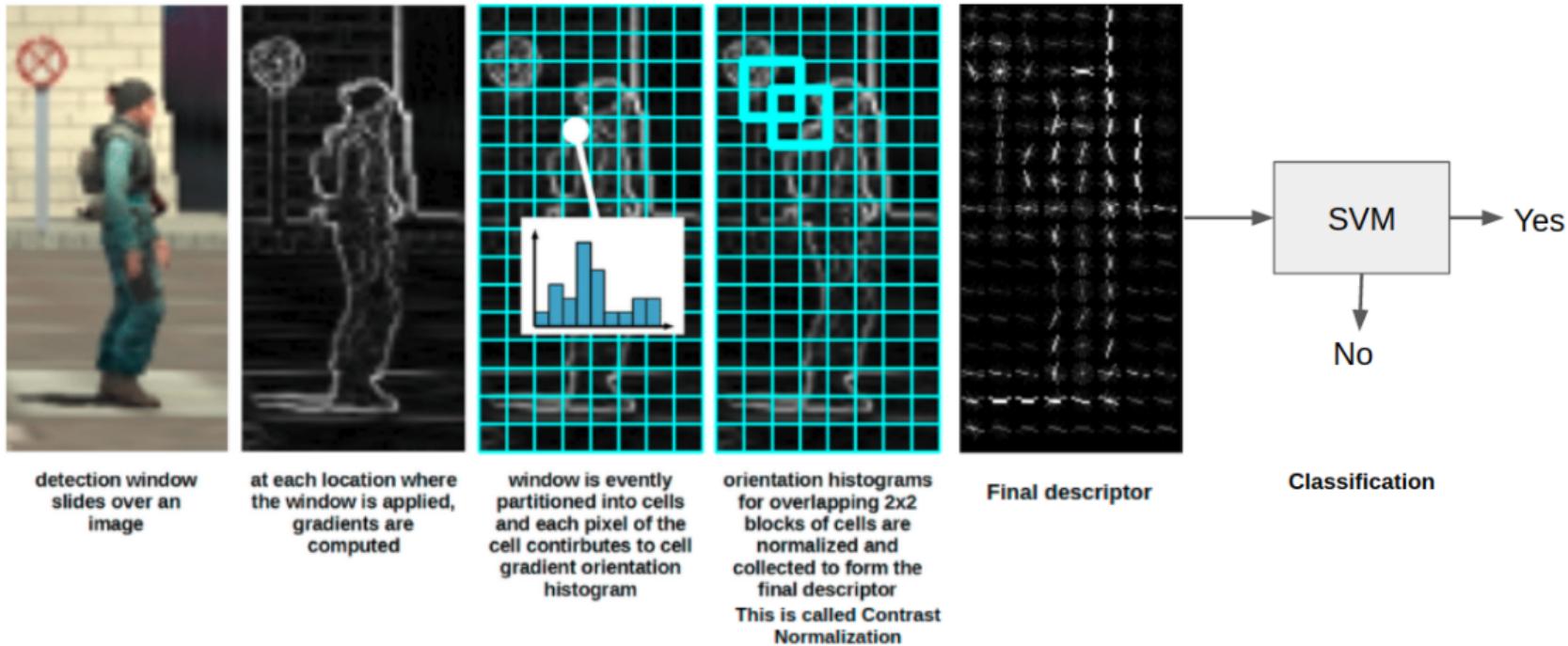
window is evenly  
partitioned into cells  
and each pixel of the  
cell contributes to cell  
gradient orientation  
histogram

orientation histograms  
for overlapping 2x2  
blocks of cells are  
normalized and  
collected to form the  
final descriptor  
**This is called Contrast  
Normalization**

Final descriptor

<sup>2</sup>Dalal and Triggs, Histograms of Oriented Gradients for Human Detection, CVPR 2005

# Histograms of Oriented Gradients<sup>2</sup>



<sup>2</sup>Dalal and Triggs, Histograms of Oriented Gradients for Human Detection, CVPR 2005

## HOG: SVM Weights



Average gradient image over training examples

Maximum positive and negative SVM weights respectively centred on the block

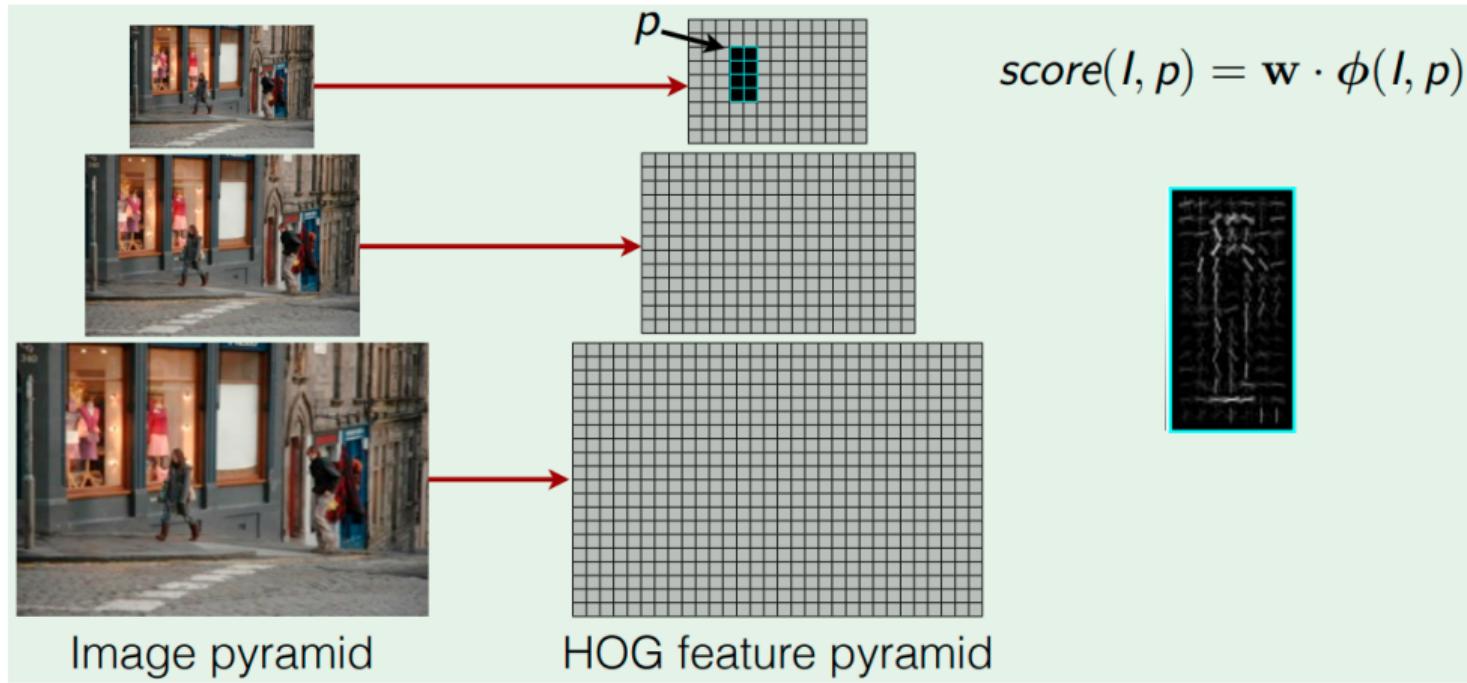
Test Image

HOG descriptor for test image

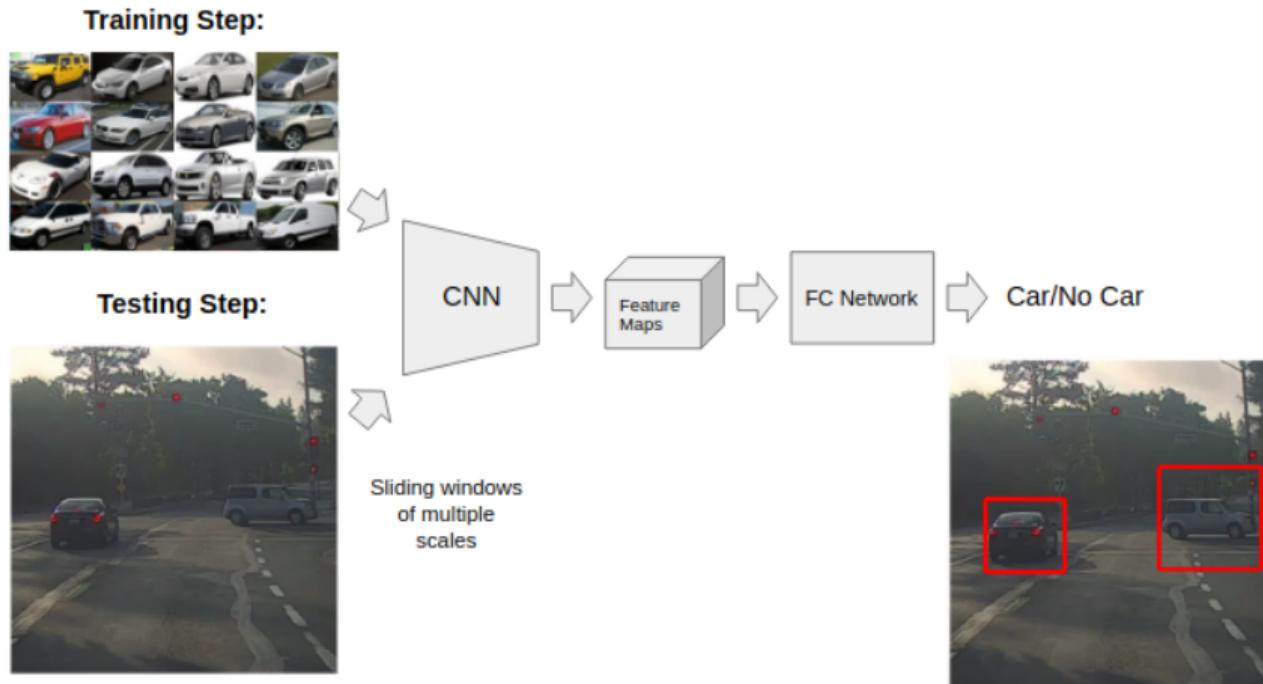
HOG descriptor weighted by positive and negative SVM weights respectively

# Image Pyramid

To identify images of all sizes, repeat the process across multiple image scales

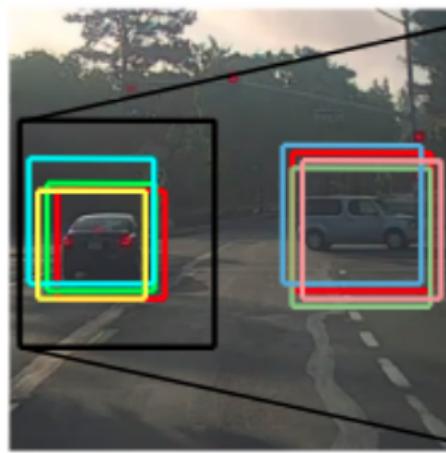


# Object Detection using CNNs



Credit: Stanford Cars dataset, Jonathan Krause

# Non-Maximum Suppression using Class Scores



List of all bounding boxes



Identify box with highest class signal(0.93 red)



Eliminate the boxes with  $\text{IOU} > 0.5$  w.r.t red box

# Object Detection using CNNs and Sliding Windows: Disadvantages

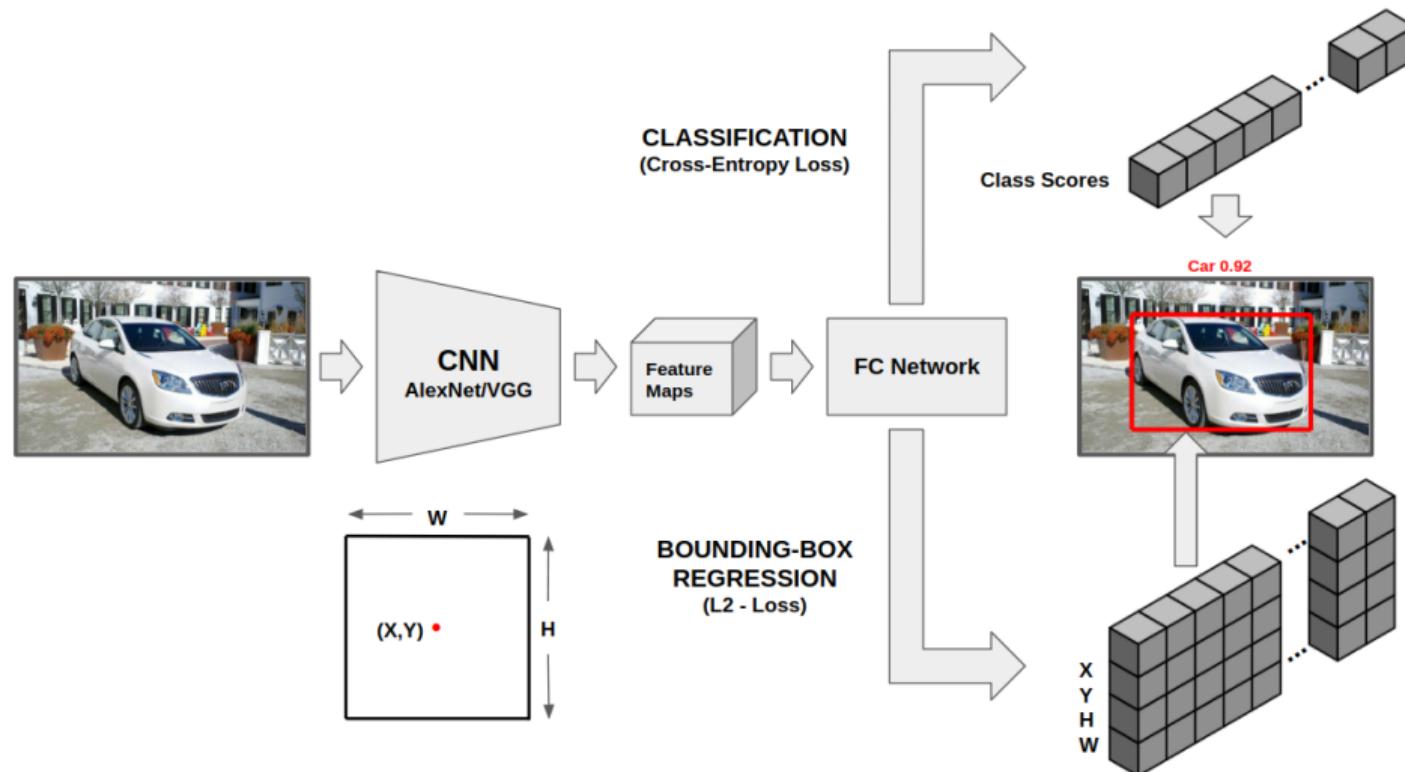
# Object Detection using CNNs and Sliding Windows: Disadvantages

- Bounding boxes may not be tightly around the object



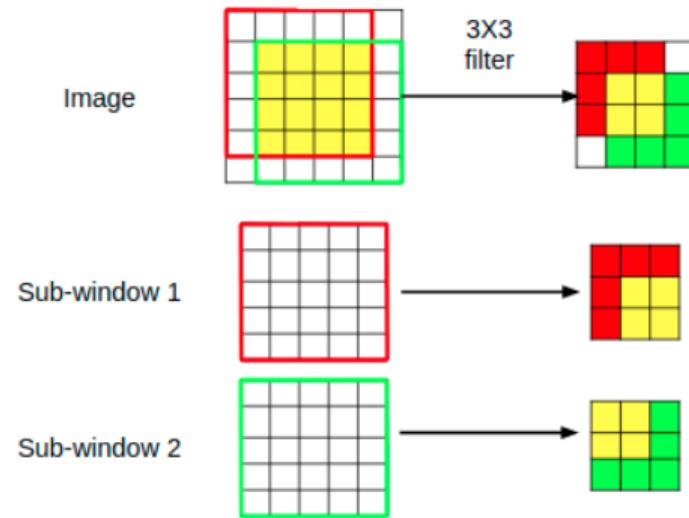
- Evaluating a CNN network on all sliding windows is computationally intensive and time consuming

# Object Localization using Bounding Box Regression



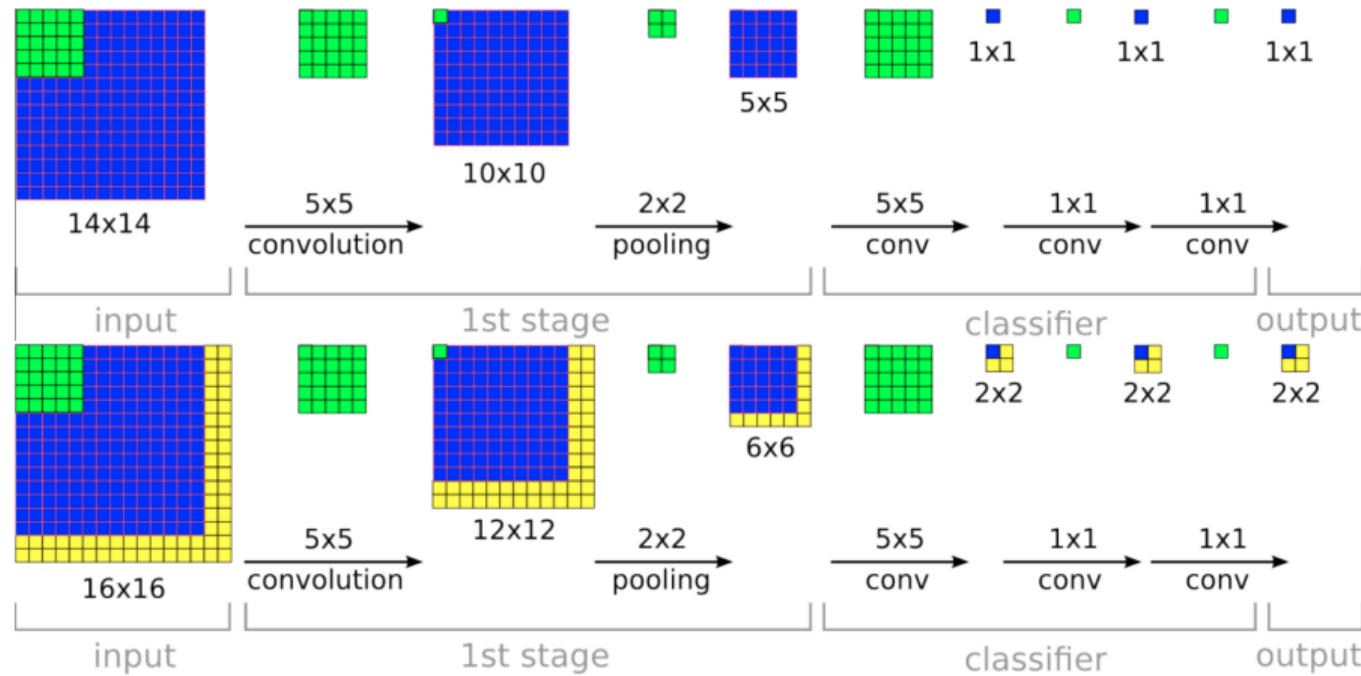
# OverFeat: Integrated Recognition, Localization and Detection<sup>3</sup>

- Winner of ILSVRC 2013 Localization Challenge
- **Intuition:** Avoid computation time over sub-windows by applying filter directly to image



<sup>3</sup>Sermanet et al, OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks, ICLR 2014

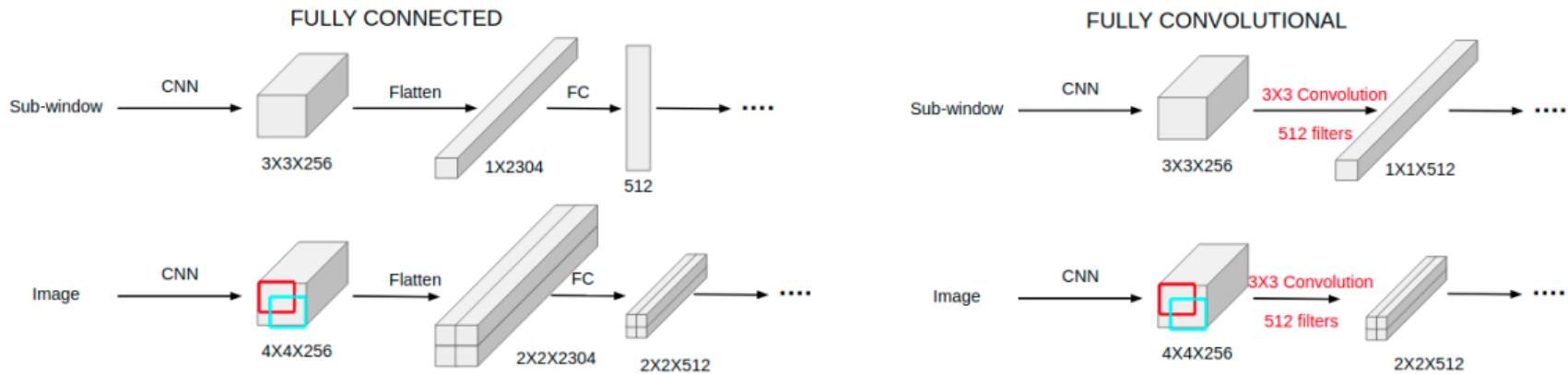
# OverFeat: Integrated Recognition, Localization and Detection<sup>4</sup>



<sup>4</sup>Sermanet et al, OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks, ICLR 2014

# OverFeat: Integrated Recognition, Localization and Detection<sup>5</sup>

The network used is fully convolutional, replacing fully connected layers with convolutional layers. Why?



<sup>5</sup>Sermanet et al, OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks, ICLR 2014

# Contemporary Object Detection Methods

## Region Proposal-based:

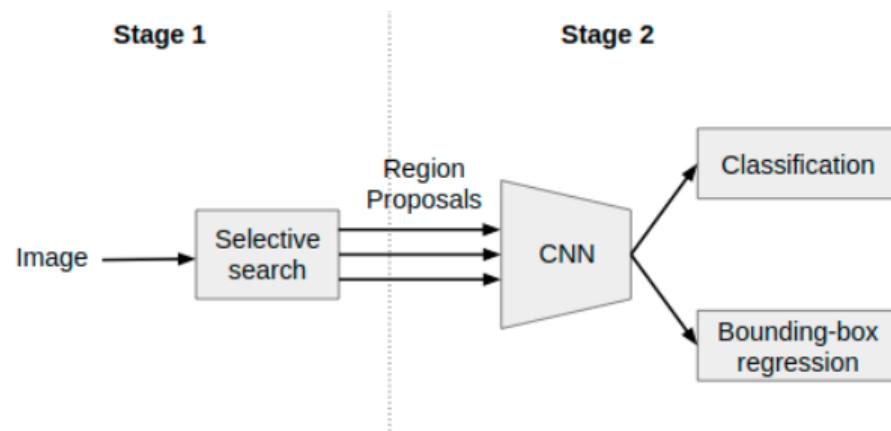
- Two-stage detection framework
- In the first stage, potential object regions are proposed (through methods such as Selective Search or Region Proposal Network, which we will see soon)
- In the second stage, a classifier processes the candidate regions
- More robust in performance but slower

## Dense Sampling-based:

- One-stage detection framework
- Integrates region proposals and detection by acting on a dense sampling of possible locations
- Simple and fast but performance not as good as Region Proposal-based methods

# R-CNN (Region-based Convolutional Neural Networks)<sup>6</sup>

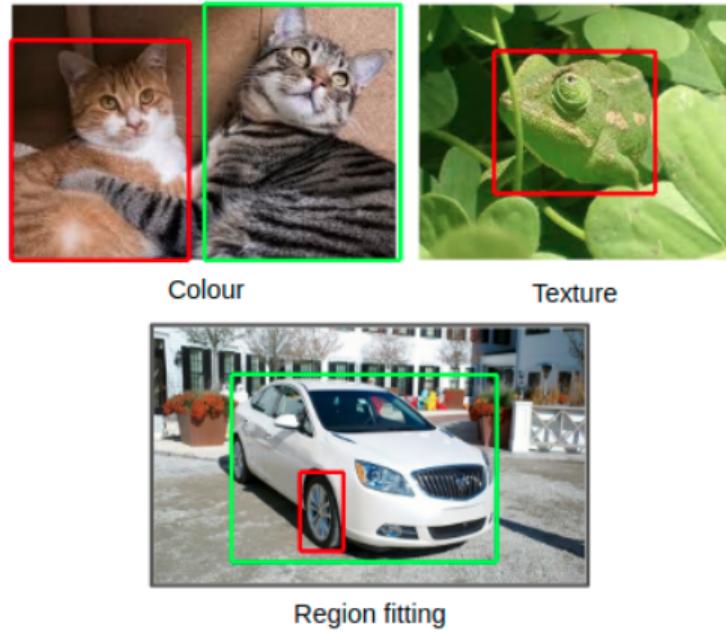
- Region proposal-based object detection network
- Uses **Selective Search** as region proposal algorithm to identify potential objects
- Each region is then evaluated by a CNN that performs classification and bounding box regression



<sup>6</sup>Girshick et al, Rich feature hierarchies for accurate object detection and semantic segmentation, CVPR 2014

## R-CNN: Selective Search

- Uses graph-based image segmentation or mean shift method for initial set of region hypotheses
- Hypotheses are hierarchically grouped/combined based on region similarity measures like color, texture, size and region-filling



## R-CNN: Selective Search

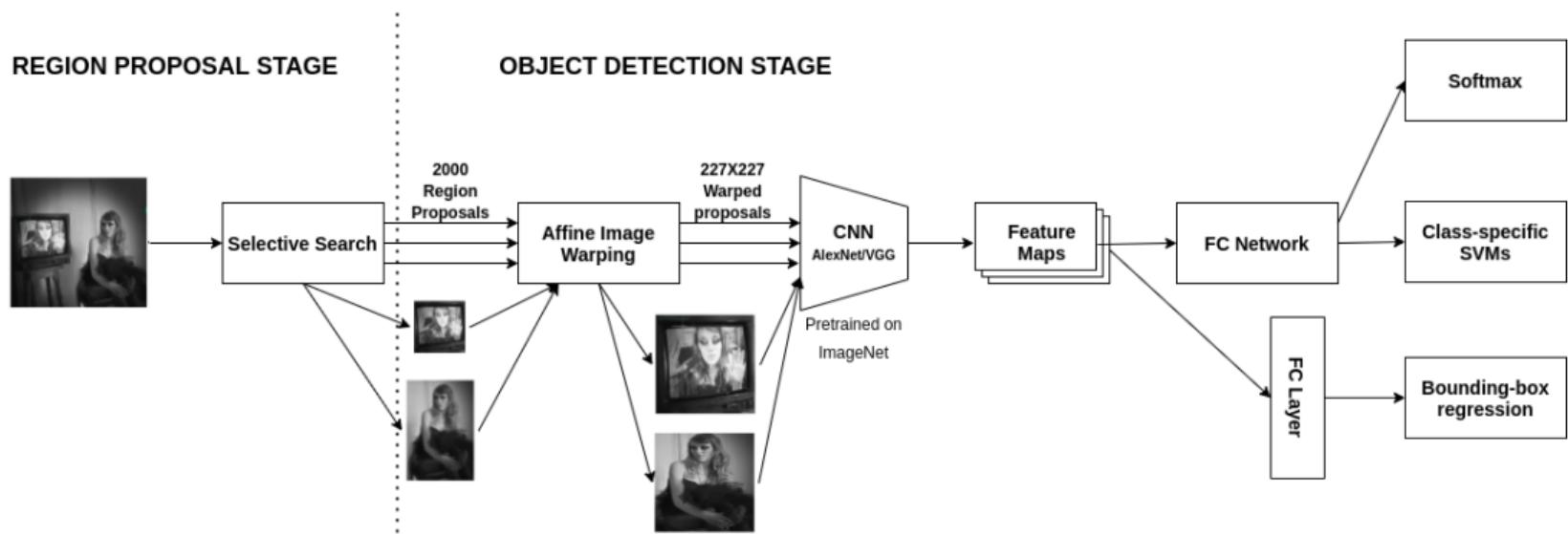


Initial proposals

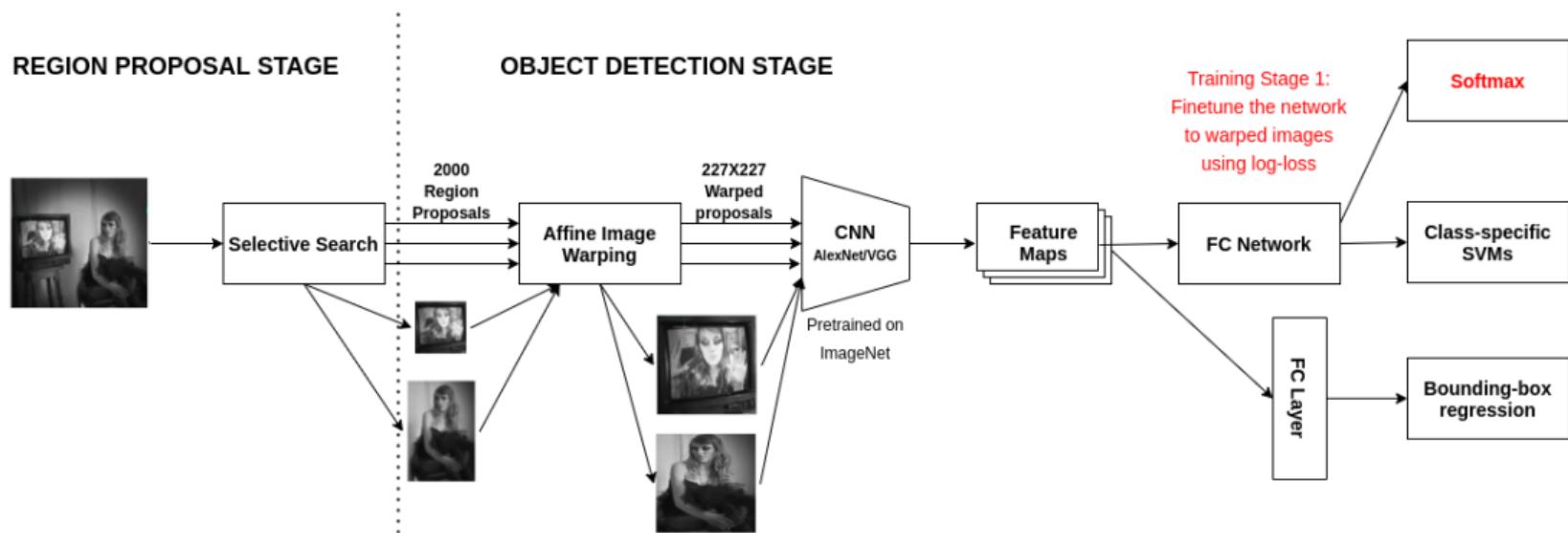


Regions coalesce as we travel up the hierarchy

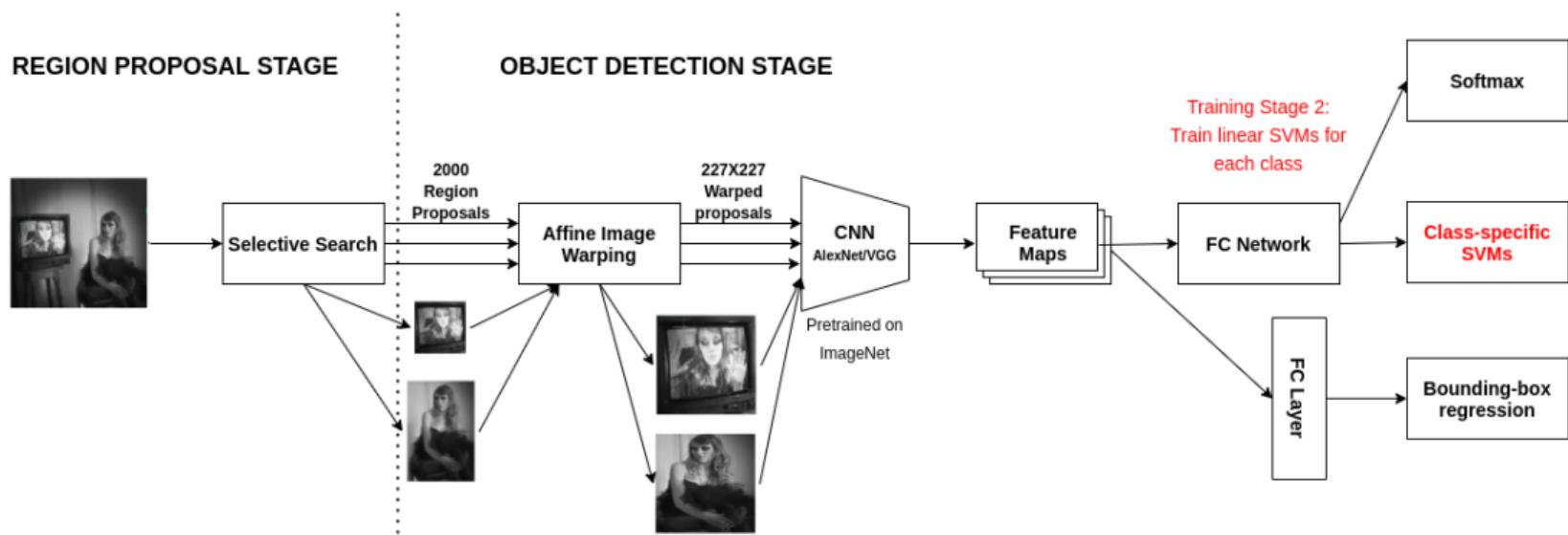
# R-CNN



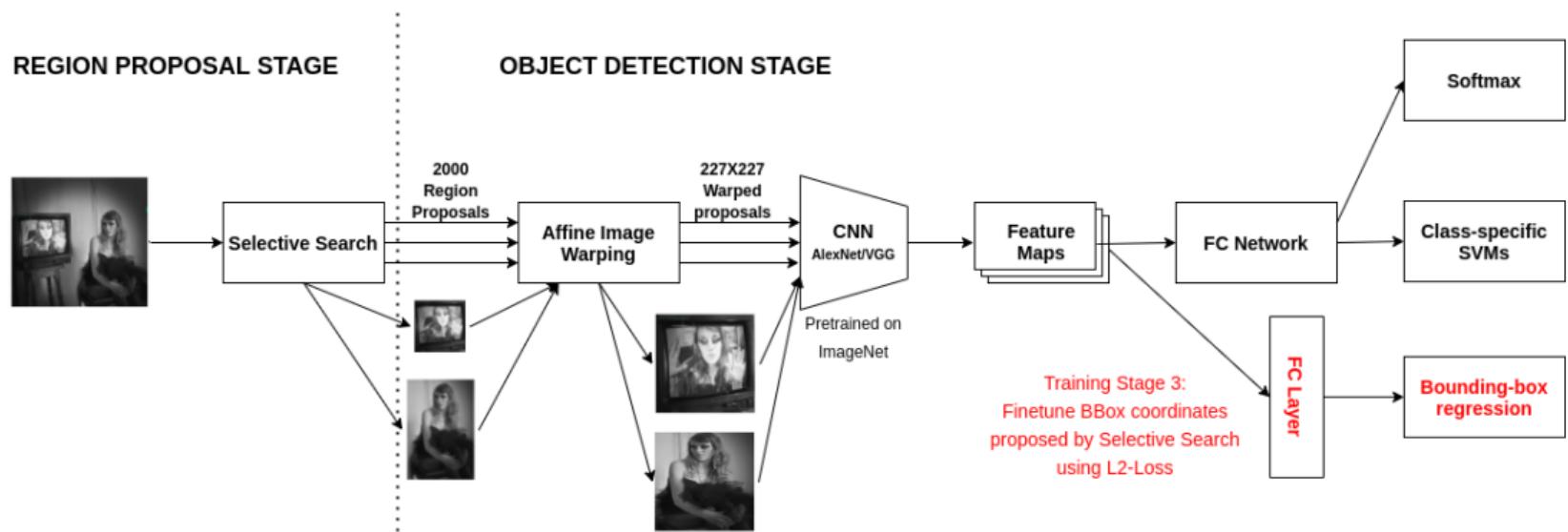
# R-CNN



# R-CNN



# R-CNN



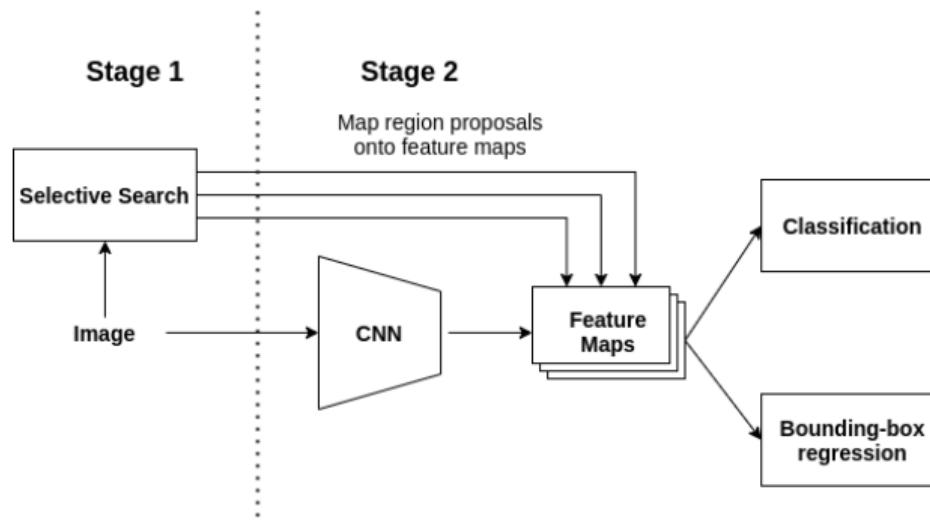
# R-CNN: Limitations

## R-CNN: Limitations

- Multi-stage pipeline of training
- Feature extraction of region proposals requires a lot of time and space; e.g. PASCAL VOC07 dataset with 5k images requires 2.5 GPU-days and several hundred GBs
- Object detection takes 47s/image at test time

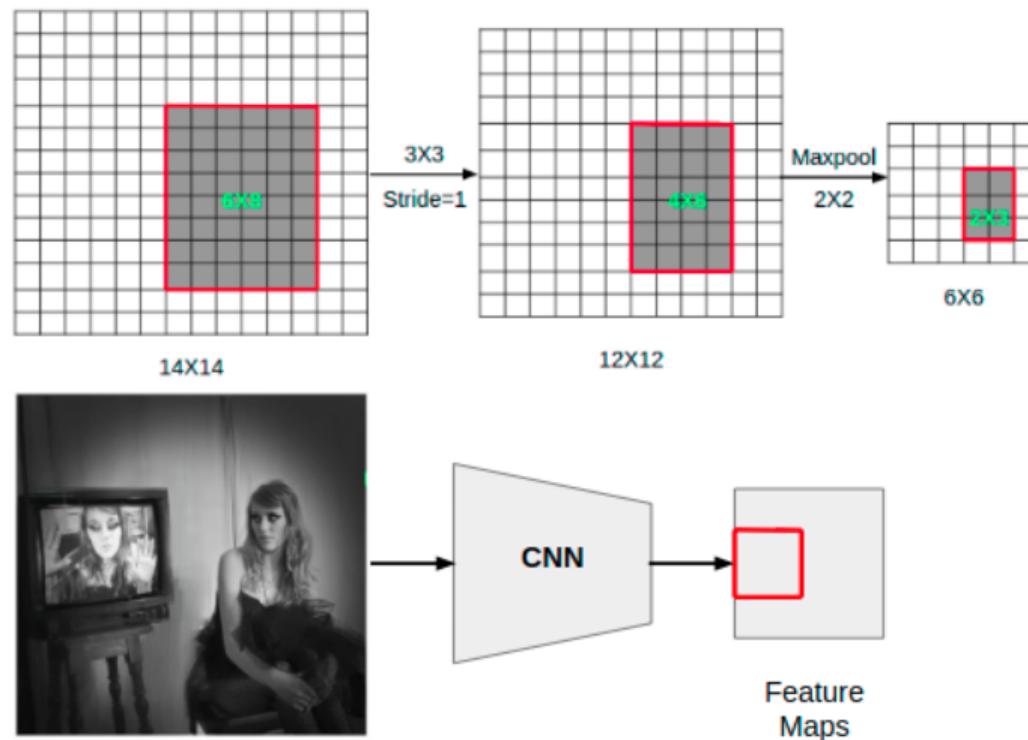
# Fast R-CNN<sup>7</sup>

- Instead of passing each region proposal through CNN, extract their features directly from feature maps
- Use multi-task loss to integrate classification and bounding box regression training stages (no SVMs)



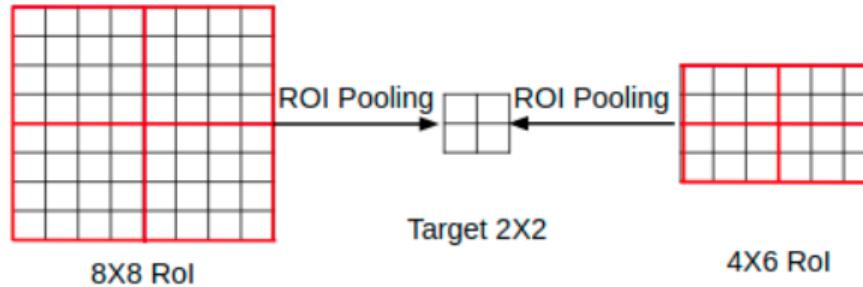
<sup>7</sup>Girshick, Fast R-CNN, ICCV 2015

# Fast R-CNN: RoI Projection



## Fast R-CNN: ROI Pooling

- ROIs can be of different scales
- We need a scale-invariant way of pooling
- Given an ROI of size  $h \times w$ , **ROI Pooling** converts this region into  $H \times W$  grid of subwindows with each subwindow of size approximately equal to  $h/H, w/W$



## Fast R-CNN: Multi-Task Loss

- Let  $u$  be true class, and  $\mathbf{v}$  be ground truth bounding box regression targets
- Let  $p$  be predicted class probability, and  $\mathbf{t}^u = (t_x^u, t_y^u, t_w^u, t_h^u)$  be bounding box regression offsets for true class
- Loss for each ROI is then given by:

$$L(p, u, \mathbf{t}^u, \mathbf{v}) = L_{cls}(p, u) + \lambda[u \geq 1]L_{loc}(\mathbf{t}^u, \mathbf{v})$$

where:

$$L_{cls}(p, u) = -\log p_u$$

and Iverson bracket indicator function  $[u \geq 1] = \begin{cases} 1 & \text{if } u \geq 1 \\ 0 & \text{otherwise} \end{cases}$

## Fast R-CNN: Multi-task Loss

- Localization loss  $L_{loc}$  given by:

$$L_{loc}(\mathbf{t}^u, \mathbf{v}) = \sum_{i \in \{x, y, w, h\}} smooth_{L1}(\mathbf{t}_i^u - \mathbf{v}_i)$$

where  $smooth_{L1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$

- Smooth L1 loss is less sensitive to outliers than L2 loss. Why?

## Fast R-CNN: Multi-task Loss

- Localization loss  $L_{loc}$  given by:

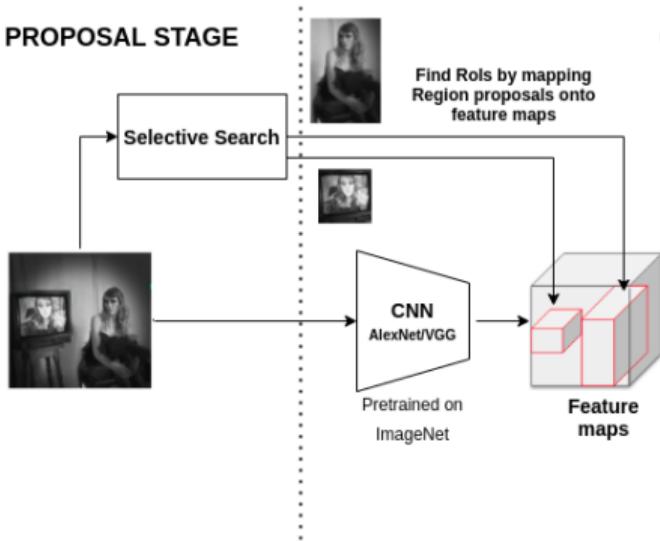
$$L_{loc}(\mathbf{t}^u, \mathbf{v}) = \sum_{i \in \{x, y, w, h\}} smooth_{L1}(\mathbf{t}_i^u - \mathbf{v}_i)$$

where  $smooth_{L1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$

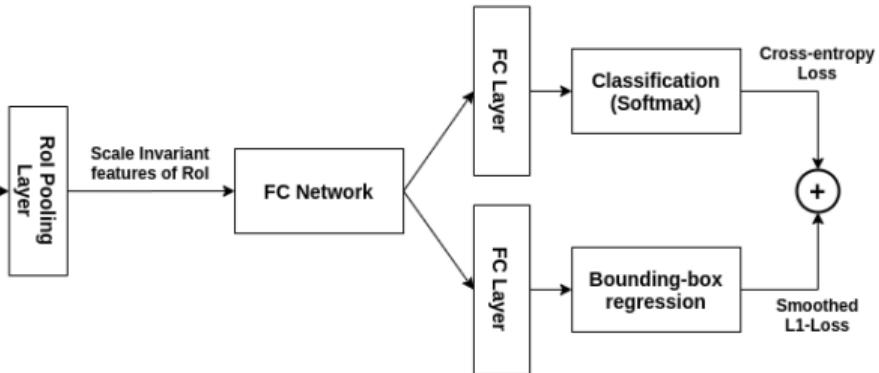
- Smooth L1 loss is less sensitive to outliers than L2 loss. Why? **Homework!**

# Fast R-CNN: Summary

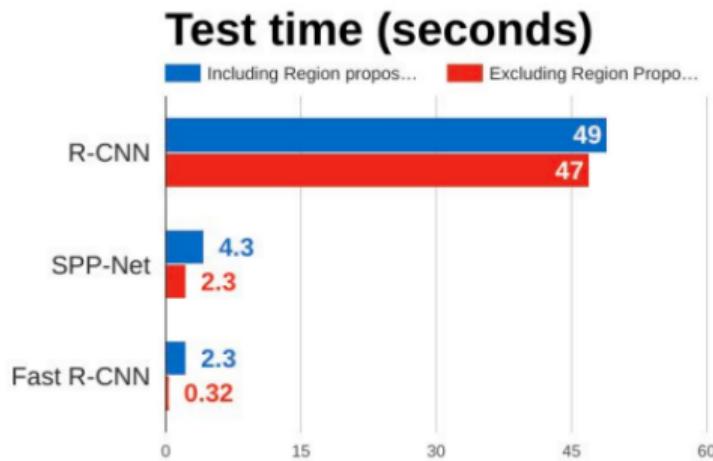
## REGION PROPOSAL STAGE



## OBJECT DETECTION STAGE



# R-CNN vs Fast R-CNN: Time Comparison



Region proposals are slowing down Fast R-CNN

Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, cs231n, Stanford Univ

## Faster R-CNN<sup>8</sup>

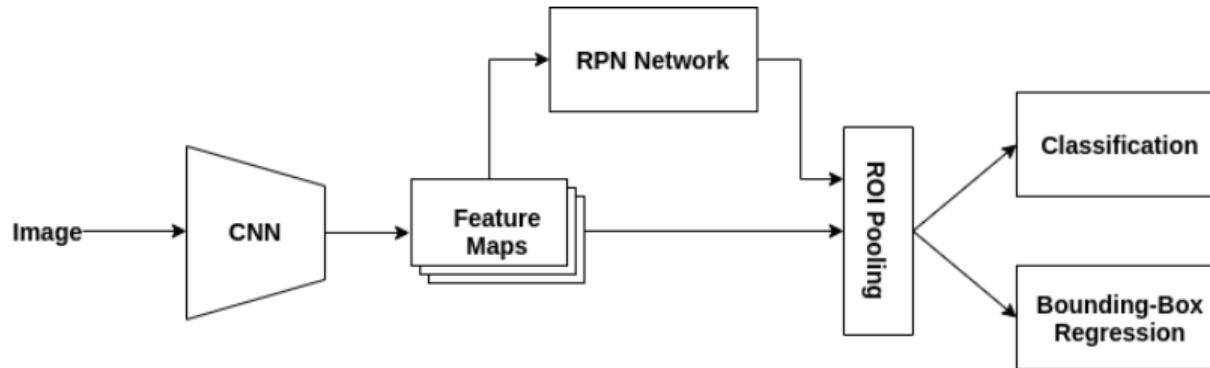
- Replaces Selective Search with a **Region Proposal Network** (RPN) to reduce time to compute regions

---

<sup>8</sup>Ren et al, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, NeurIPS 2015

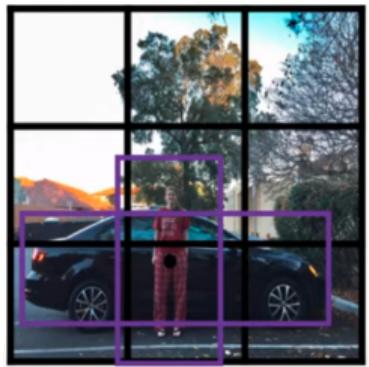
# Faster R-CNN<sup>8</sup>

- Replaces Selective Search with a **Region Proposal Network** (RPN) to reduce time to compute regions
- Uses **anchor boxes** of different scales and aspect ratios to identify multiple objects present in the same window

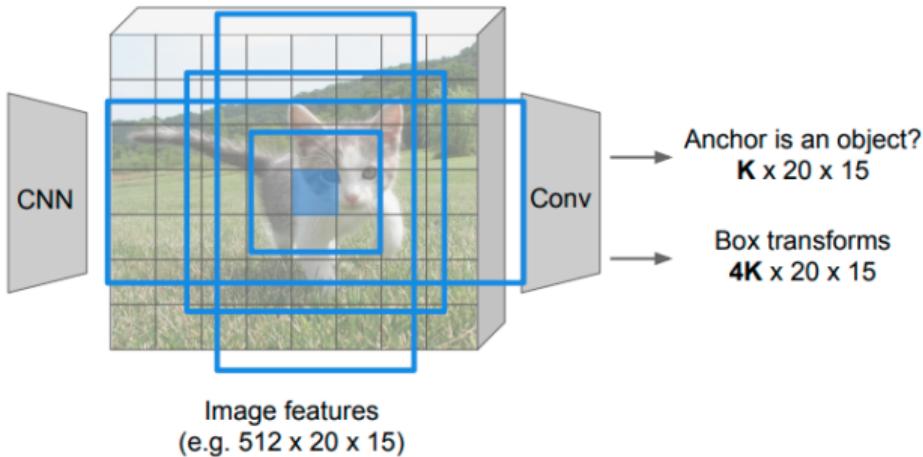


<sup>8</sup>Ren et al, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, NeurIPS 2015

# Faster R-CNN: Anchor Boxes

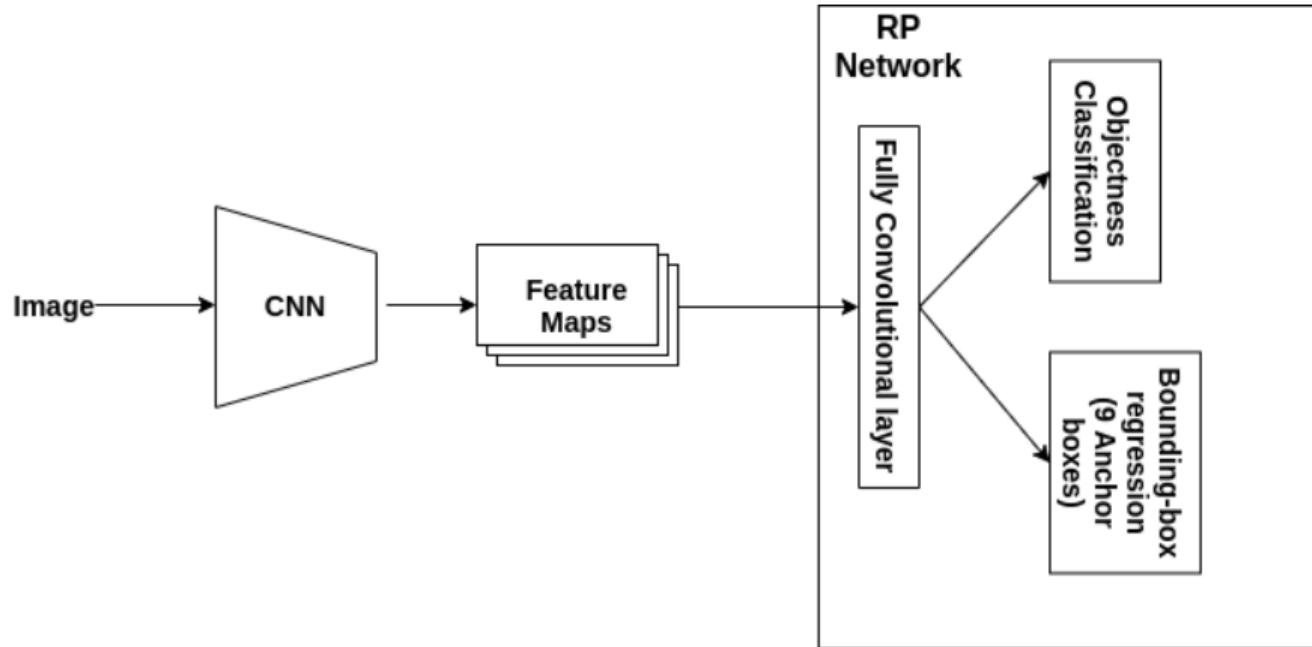


Input Image  
(e.g.  $3 \times 640 \times 480$ )

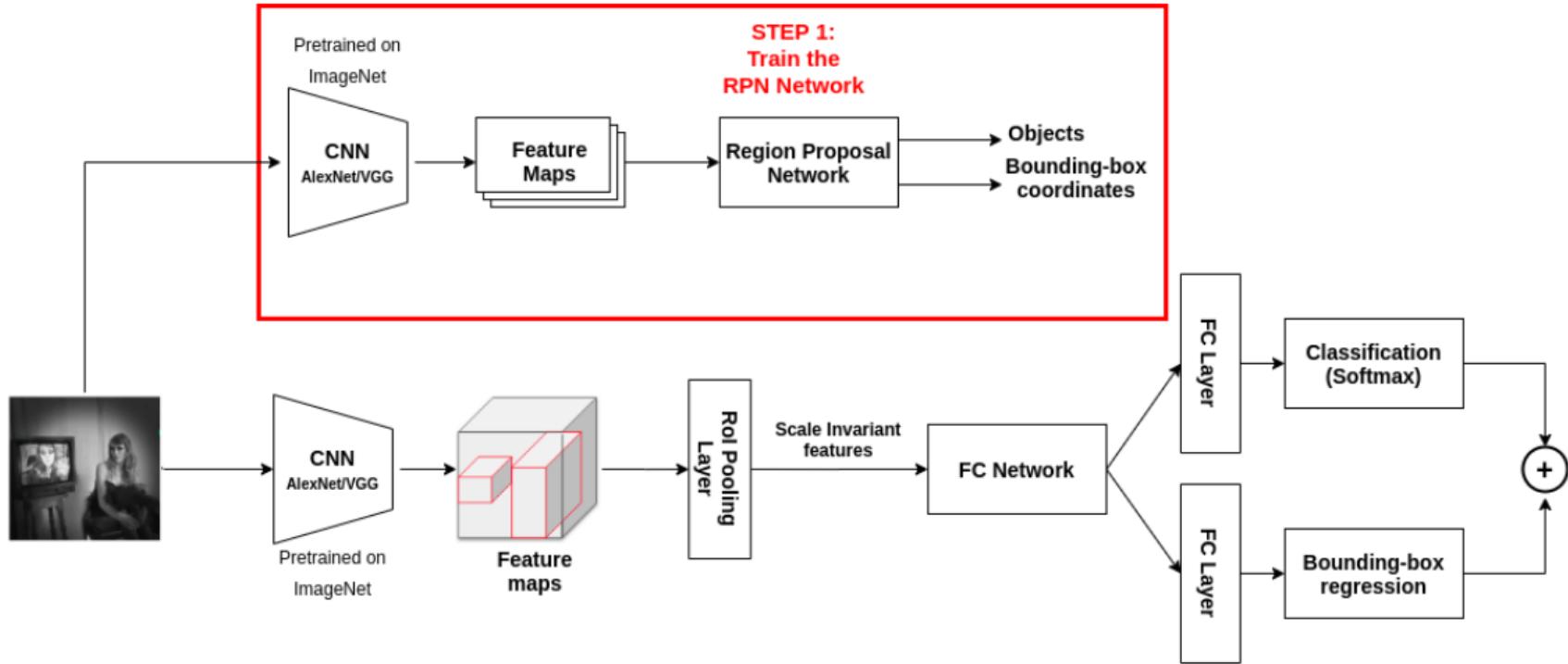


Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, cs231n, Stanford Univ

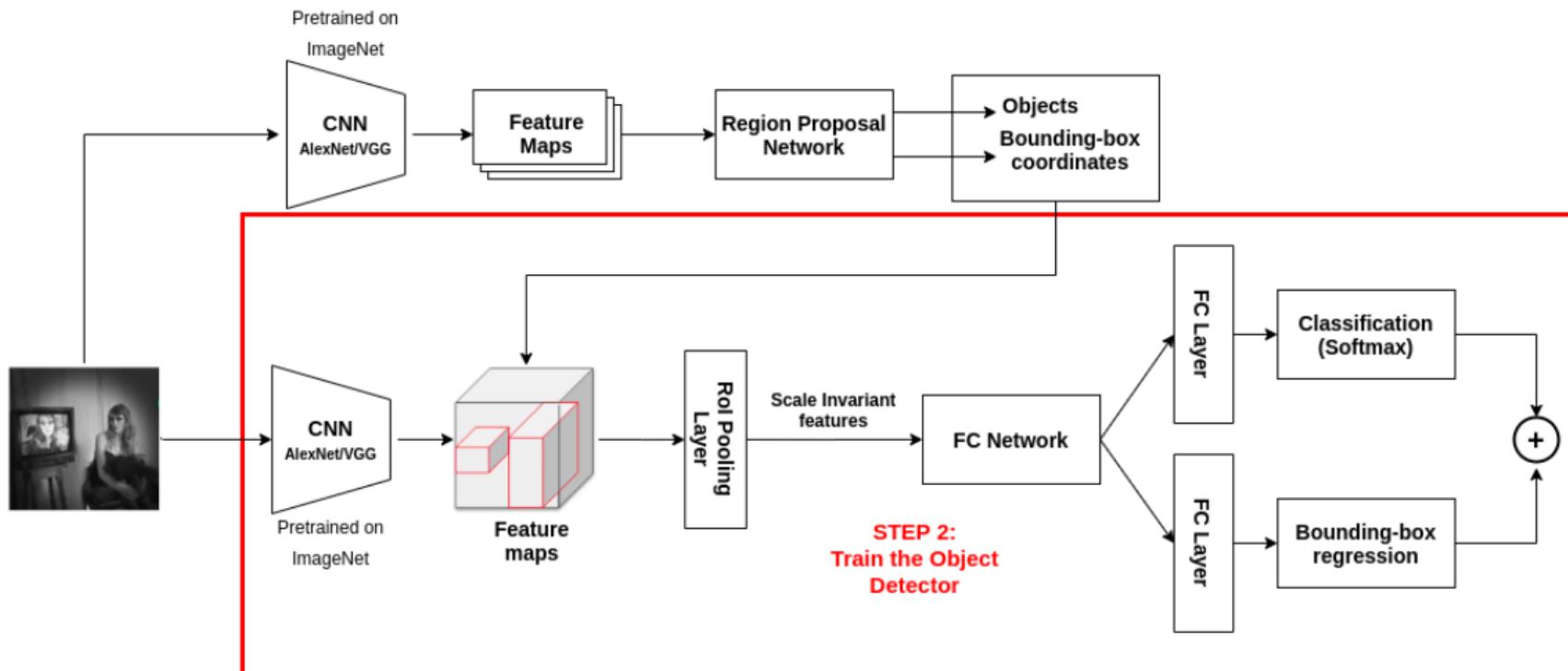
# Faster R-CNN: Region Proposal Network



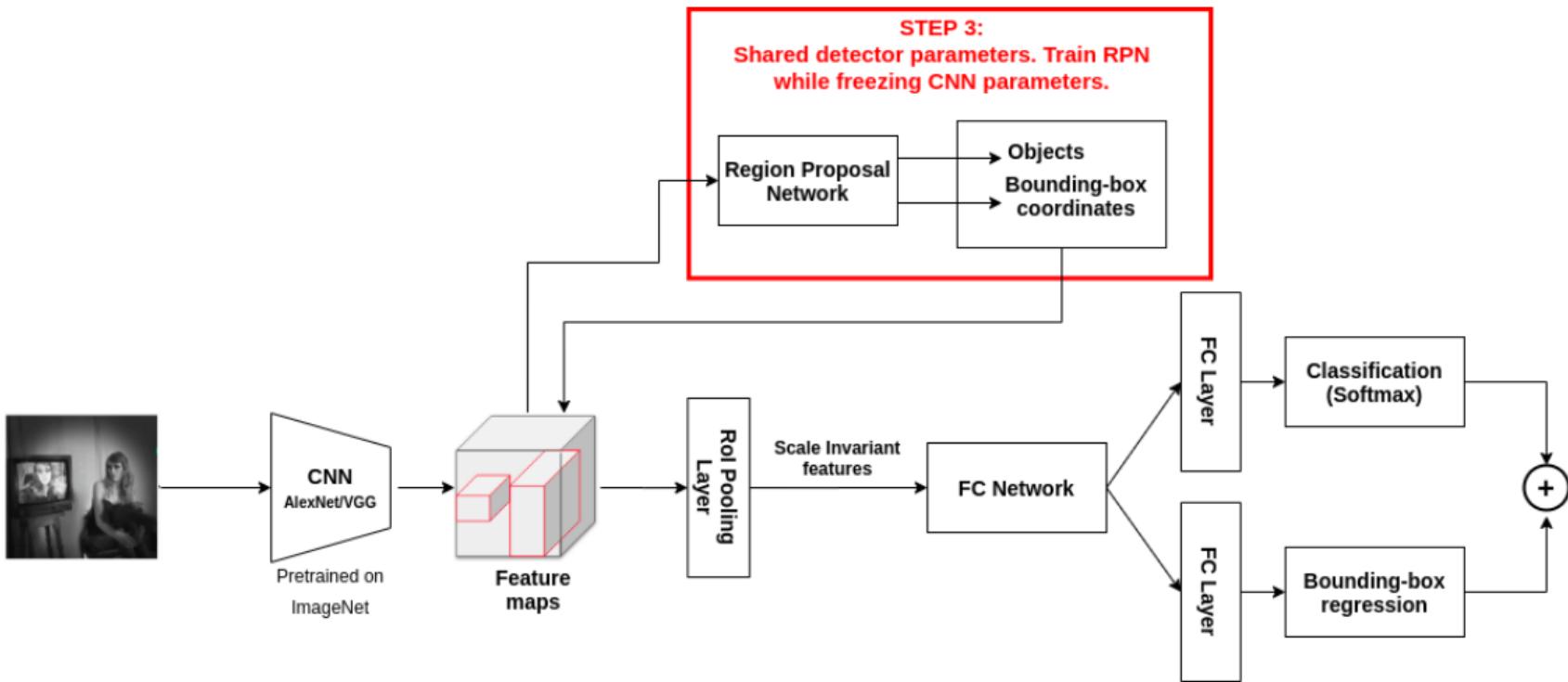
# Faster R-CNN: 4-step Training



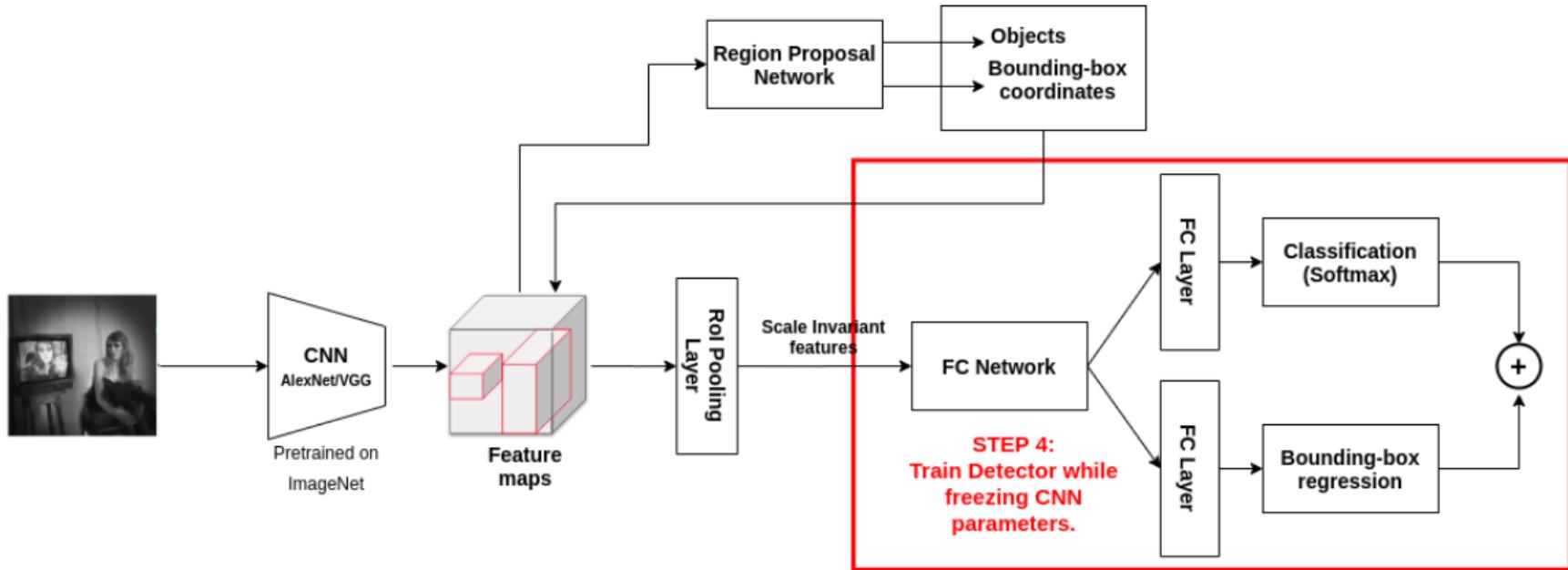
# Faster R-CNN: 4-step Training



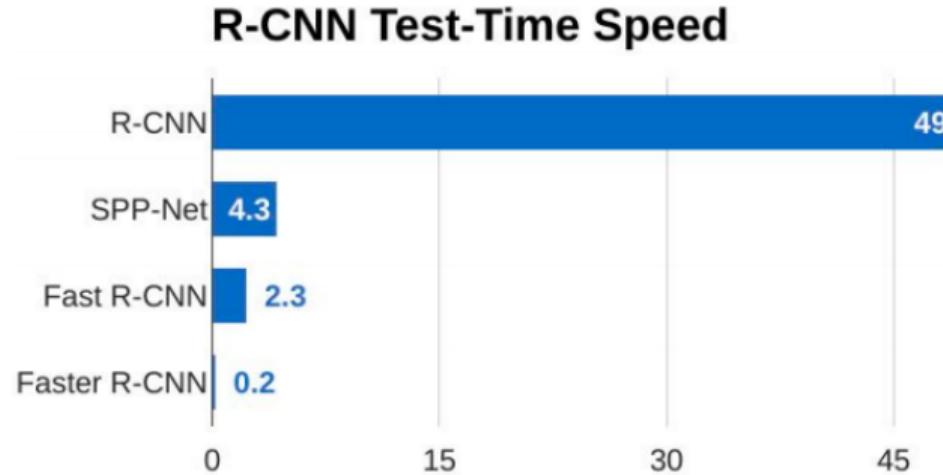
# Faster R-CNN: 4-step Training



# Faster R-CNN: 4-step Training



# Faster R-CNN Performance



Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, cs231n, Stanford Univ

# Evaluating Object Detectors: How?<sup>9</sup>

For classification, we measure accuracy; for detection?

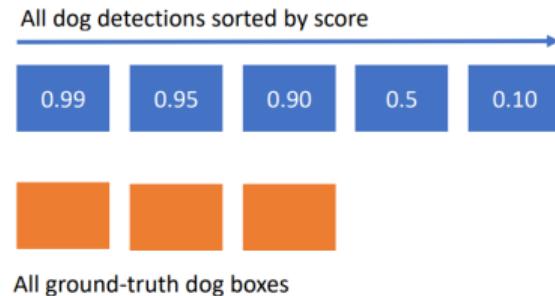
---

<sup>9</sup>Credit: Justin Johnson, Univ of Michigan

# Evaluating Object Detectors: How?<sup>9</sup>

For classification, we measure accuracy; for detection?

- ① Run object detector on all test images (with NMS)
- ② For each category, compute Average Precision (AP)  
= area under Precision vs Recall Curve
  - ① For each detection (highest score to lowest score)

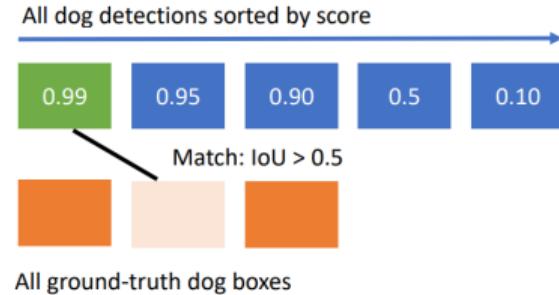


<sup>9</sup>Credit: Justin Johnson, Univ of Michigan

# Evaluating Object Detectors: How?<sup>9</sup>

For classification, we measure accuracy; for detection?

- ① Run object detector on all test images (with NMS)
- ② For each category, compute Average Precision (AP)  
= area under Precision vs Recall Curve
  - ① For each detection (highest score to lowest score)
    - ① If it matches some GT box with  $\text{IoU} > 0.5$ , mark it as positive and eliminate the GT
    - ② Otherwise mark it as negative

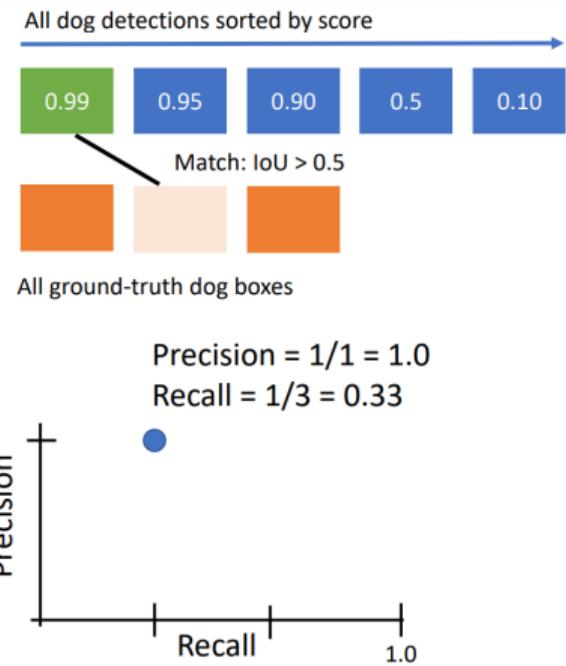


<sup>9</sup>Credit: Justin Johnson, Univ of Michigan

# Evaluating Object Detectors: How?<sup>9</sup>

For classification, we measure accuracy; for detection?

- ① Run object detector on all test images (with NMS)
- ② For each category, compute Average Precision (AP)  
= area under Precision vs Recall Curve
  - ① For each detection (highest score to lowest score)
    - ① If it matches some GT box with  $\text{IoU} > 0.5$ , mark it as positive and eliminate the GT
    - ② Otherwise mark it as negative
    - ③ Plot a point on PR Curve

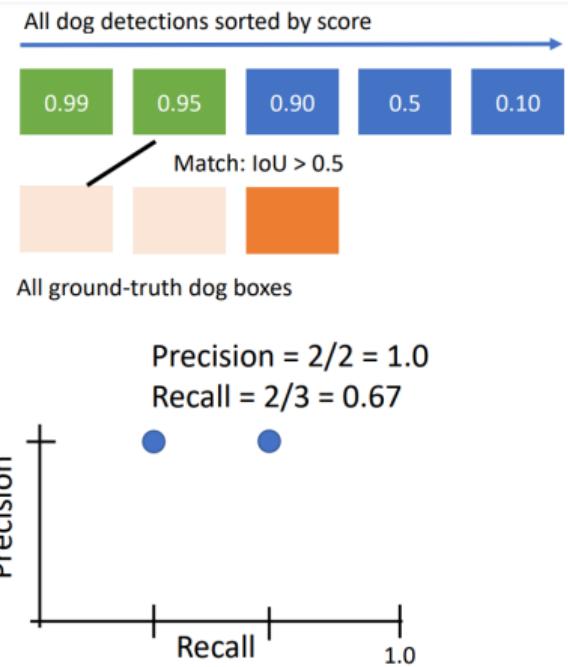


<sup>9</sup>Credit: Justin Johnson, Univ of Michigan

# Evaluating Object Detectors: How?<sup>9</sup>

For classification, we measure accuracy; for detection?

- ① Run object detector on all test images (with NMS)
- ② For each category, compute Average Precision (AP)  
= area under Precision vs Recall Curve
  - ① For each detection (highest score to lowest score)
    - ① If it matches some GT box with  $\text{IoU} > 0.5$ , mark it as positive and eliminate the GT
    - ② Otherwise mark it as negative
    - ③ Plot a point on PR Curve

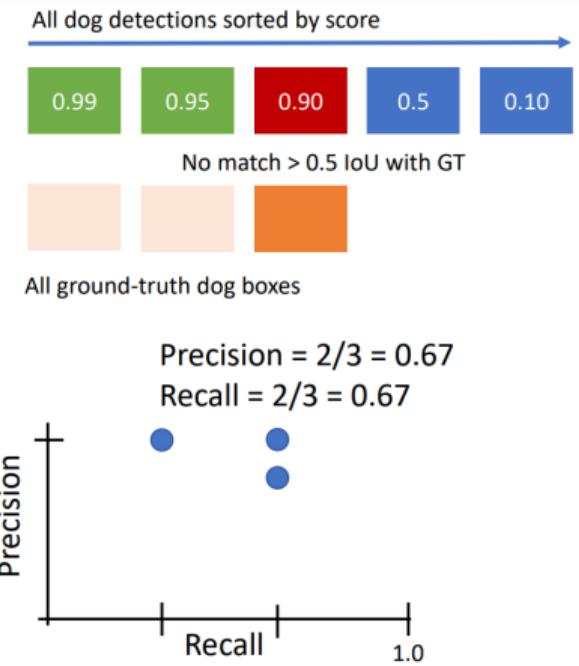


<sup>9</sup>Credit: Justin Johnson, Univ of Michigan

# Evaluating Object Detectors: How?<sup>9</sup>

For classification, we measure accuracy; for detection?

- ① Run object detector on all test images (with NMS)
- ② For each category, compute Average Precision (AP)  
= area under Precision vs Recall Curve
  - ① For each detection (highest score to lowest score)
    - ① If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
    - ② Otherwise mark it as negative
    - ③ Plot a point on PR Curve

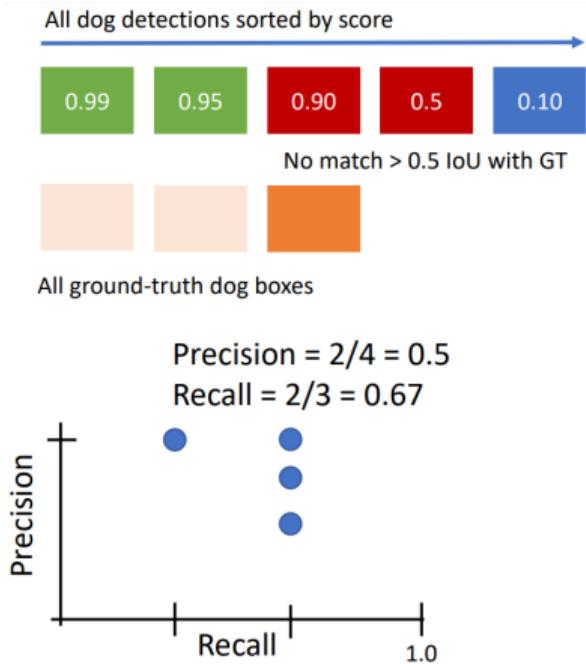


<sup>9</sup>Credit: Justin Johnson, Univ of Michigan

# Evaluating Object Detectors: How?<sup>9</sup>

For classification, we measure accuracy; for detection?

- ① Run object detector on all test images (with NMS)
- ② For each category, compute Average Precision (AP)  
= area under Precision vs Recall Curve
  - ① For each detection (highest score to lowest score)
    - ① If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
    - ② Otherwise mark it as negative
    - ③ Plot a point on PR Curve

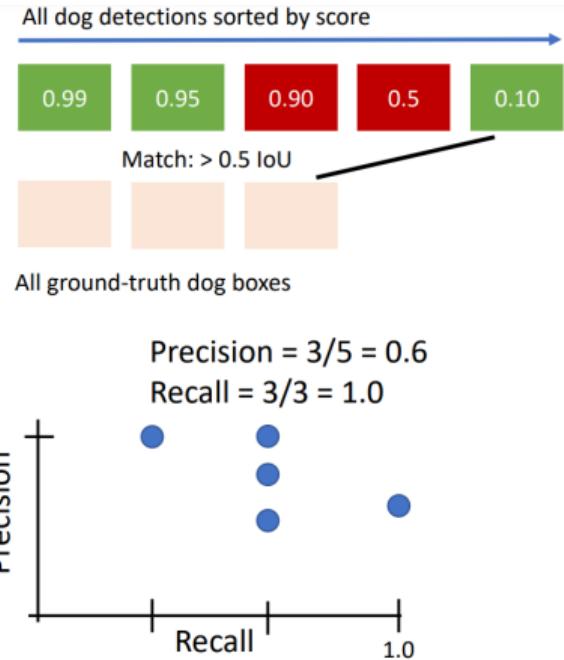


<sup>9</sup>Credit: Justin Johnson, Univ of Michigan

# Evaluating Object Detectors: How?<sup>9</sup>

For classification, we measure accuracy; for detection?

- ① Run object detector on all test images (with NMS)
- ② For each category, compute Average Precision (AP)  
= area under Precision vs Recall Curve
  - ① For each detection (highest score to lowest score)
    - ① If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
    - ② Otherwise mark it as negative
    - ③ Plot a point on PR Curve

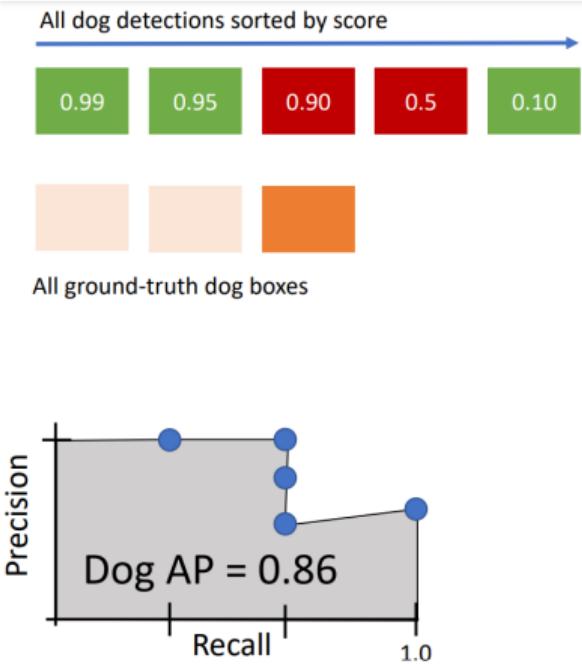


<sup>9</sup>Credit: Justin Johnson, Univ of Michigan

# Evaluating Object Detectors: How?<sup>9</sup>

For classification, we measure accuracy; for detection?

- ① Run object detector on all test images (with NMS)
- ② For each category, compute Average Precision (AP)  
= area under Precision vs Recall Curve
  - ① For each detection (highest score to lowest score)
    - ① If it matches some GT box with  $\text{IoU} > 0.5$ , mark it as positive and eliminate the GT
    - ② Otherwise mark it as negative
    - ③ Plot a point on PR Curve
  - ② Average Precision (AP) = area under PR curve



<sup>9</sup>Credit: Justin Johnson, Univ of Michigan

## Evaluating Object Detectors: How?<sup>9</sup>

For classification, we measure accuracy; for detection?

- ① Run object detector on all test images (with NMS)
- ② For each category, compute Average Precision (AP)  
= area under Precision vs Recall Curve
  - ① For each detection (highest score to lowest score)
    - ① If it matches some GT box with  $\text{IoU} > 0.5$ , mark it as positive and eliminate the GT
    - ② Otherwise mark it as negative
    - ③ Plot a point on PR Curve
  - ② Average Precision (AP) = area under PR curve
- ③ Mean Average Precision (mAP) = average of AP for each category

$$\text{Car AP} = 0.65$$

$$\text{Cat AP} = 0.80$$

$$\text{Dog AP} = 0.86$$

$$\text{mAP}@0.5 = 0.77$$

---

<sup>9</sup>Credit: Justin Johnson, Univ of Michigan

# Evaluating Object Detectors: How?<sup>9</sup>

For classification, we measure accuracy; for detection?

- ① Run object detector on all test images (with NMS)
- ② For each category, compute Average Precision (AP)  
= area under Precision vs Recall Curve
  - ① For each detection (highest score to lowest score)
    - ① If it matches some GT box with  $\text{IoU} > 0.5$ , mark it as positive and eliminate the GT
    - ② Otherwise mark it as negative
    - ③ Plot a point on PR Curve
  - ② Average Precision (AP) = area under PR curve
- ③ Mean Average Precision (mAP) = average of AP for each category
- ④ For COCO dataset: Compute mAP for IoU thresholds (0.5, 0.55, 0.6, ..., 0.95) and average

$$\text{mAP}@0.5 = 0.77$$

$$\text{mAP}@0.55 = 0.71$$

$$\text{mAP}@0.60 = 0.65$$

...

$$\text{mAP}@0.95 = 0.2$$

$$\text{COCO mAP} = 0.4$$

<sup>9</sup>Credit: Justin Johnson, Univ of Michigan

# Homework

## Readings

- Viola-Jones Object Detection Framework
- Object Detection for Dummies (Parts 1-3)
- Understanding Overfeat

## Video Series

- Evolution of Object Detection Models (Chapter 5-8)

## Exercise

Smooth L1 loss is less sensitive to outliers than L2 loss. Why?

# References

- [1] Paul Viola and Michael Jones. "Rapid object detection using a boosted cascade of simple features". In: *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*. Vol. 1. IEEE. 2001, pp. I–I.
- [2] Navneet Dalal and Bill Triggs. "Histograms of oriented gradients for human detection". In: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. Vol. 1. IEEE. 2005, pp. 886–893.
- [3] Ross Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.
- [4] Ross Girshick. "Fast r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.
- [5] Shaoqing Ren et al. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*. 2015, pp. 91–99.