

CSE428: Image Processing

Lecture 16

Object Detection

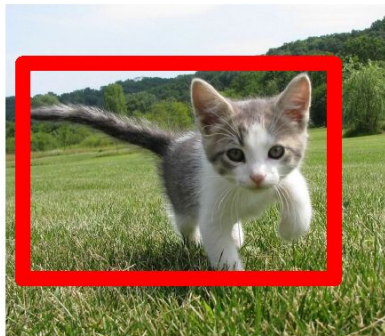
Image Recognition Problems

Classification



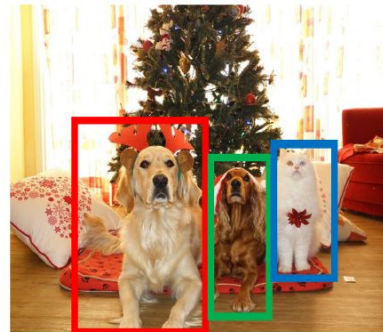
$[\{\text{CAT}\}]$

Classification +
Localization



$[\{\text{CAT}, (x, y, h, w)\}]$

Object Detection



$[\{\text{DOG}, (x, y, h, w)\},$
 $\{\text{DOG}, (x, y, h, w)\},$
 $\{\text{CAT}, (x, y, h, w)\}]$

Object Detection Methods

Methods for object detection generally fall into either neural network-based or non-neural approaches.

1. Non-neural approaches:

- Viola–Jones object detection framework based on Haar features
- Scale-invariant feature transform (SIFT)
- Histogram of oriented gradients (HOG) features

2. Neural network approaches:

- Region Proposals (R-CNN, Fast R-CNN, Faster R-CNN)
- Single Shot MultiBox Detector (SSD)
- You Only Look Once (YOLO)

Non-neural Object Detection Methods

Non-neural approaches:

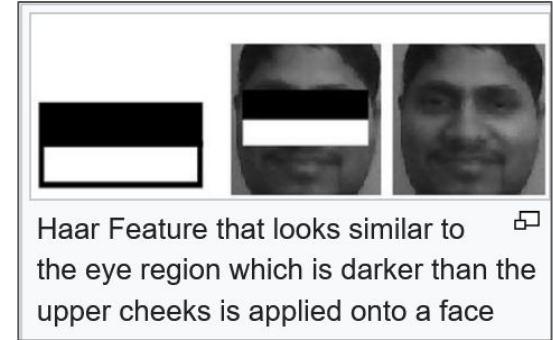
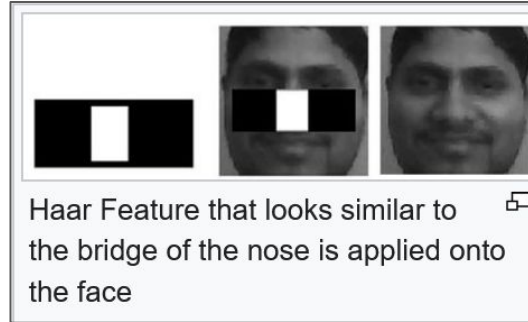
- Viola–Jones object detection framework based on Haar features
- Scale-invariant feature transform (SIFT)
- Histogram of oriented gradients (HOG) features

For non-neural approaches, it becomes necessary to first define **features** (which are preferably: **scale, position, rotation & lighting invariant**) using one of the methods above, then using a technique such as support vector machine (SVM) to do the classification.

Viola-Jones Algorithm (Face Detection)

The algorithm has four stages:

1. Haar Feature Selection
2. Creating an Integral Image
3. Adaboost Training
4. Cascading Classifiers

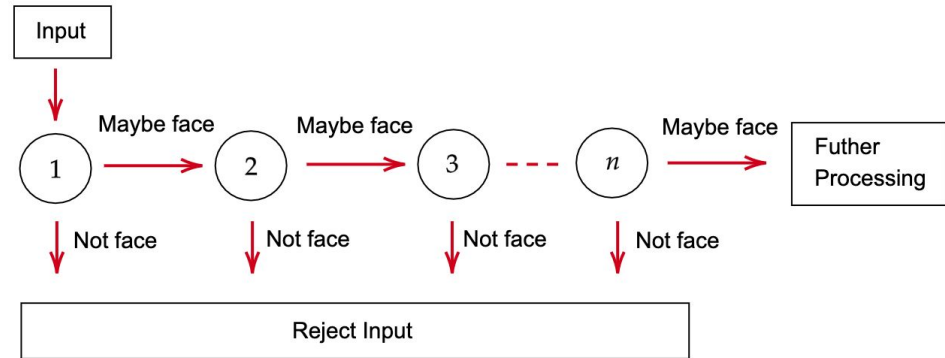


Scale invariant: zoom in/out in your selfie camera?

Rotation invariant: ?

Lighting: ?

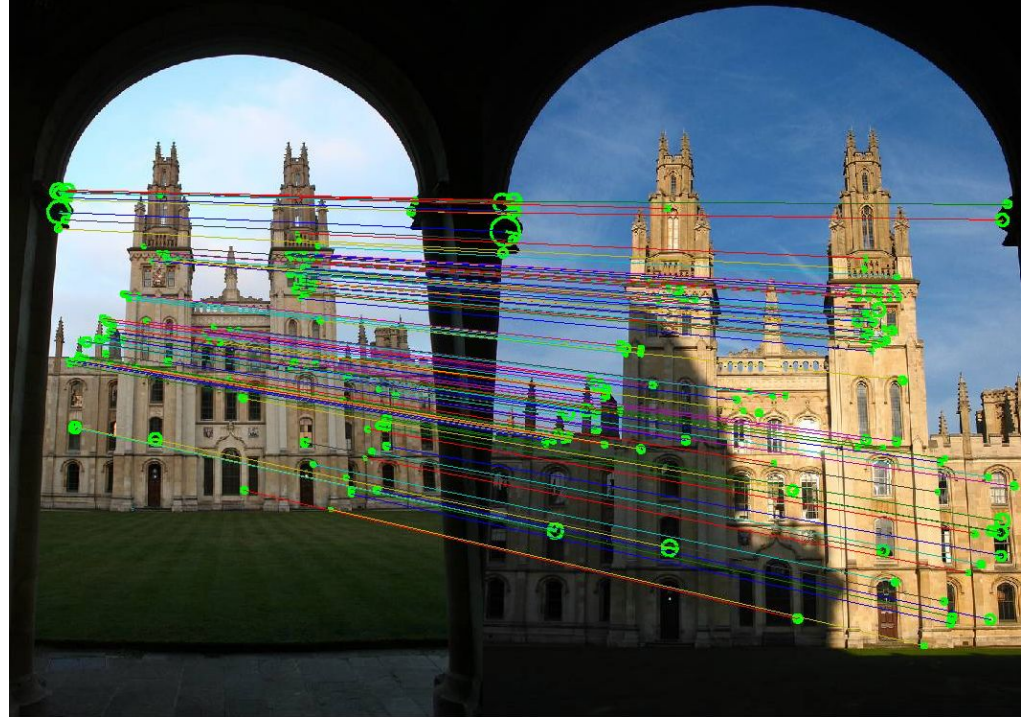
Position: ?



Scale-invariant feature transform (SIFT)

SIFT **keypoints** of objects are first extracted from a set of reference images and stored in a database.

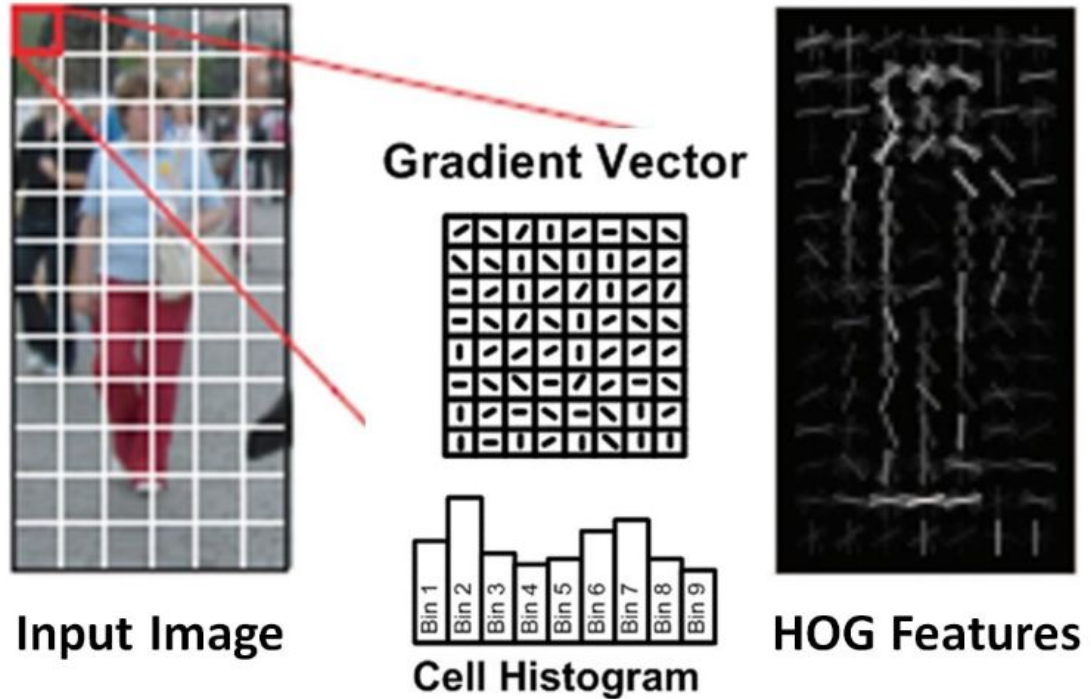
An object is recognized in a new image by individually comparing each feature from the new image to this database and finding candidate matching features based on Euclidean distance of their **feature vectors**.



Histogram of oriented gradients (HOG)

Local object appearance and shape within an image can be described by the distribution of intensity gradients or edge directions.

The image is divided into small connected regions called cells, and for the pixels within each cell, a histogram of gradient directions is compiled. The descriptor is the concatenation of these histograms.



Deep Learning Based Object Detection Methods

Neural network approaches:

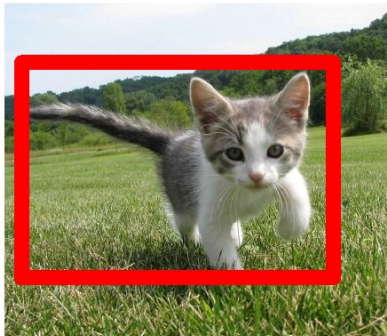
- Region Proposals (R-CNN, Fast R-CNN, Faster R-CNN)
- Single Shot MultiBox Detector (SSD)
- You Only Look Once (YOLO)

neural techniques are able to do end-to-end object detection without specifically defining features, and are typically based on convolutional neural networks (CNN).

Classification + Localization

Only applicable when there is **one object/image!**

Classification +
Localization

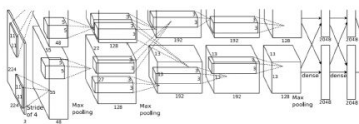


[{CAT, (x, y, h, w)}]

Classification + Localization



[This image is CC0 public domain](#)



Feature
vector: 4096

FC layer 4096 \rightarrow #classes

Class scores
Dog: 0.1
Cat: 0.8
Goat: ...

FC layer 4096 \rightarrow 4

Bounding
box
(x, y, h, w)

Correct label: Cat

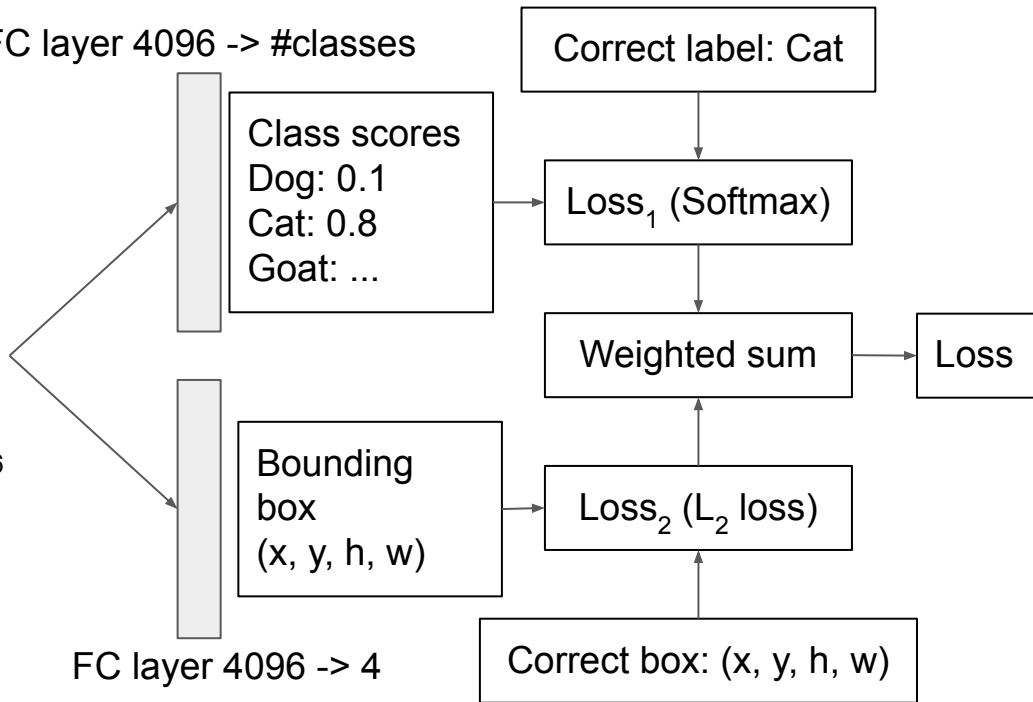
Loss₁ (Softmax)

Weighted sum

Loss

Loss₂ (L₂ loss)

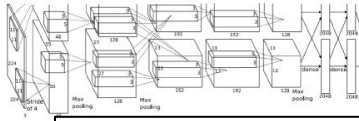
Correct box: (x, y, h, w)



Classification + Localization



[This image is CC0 public domain](#)

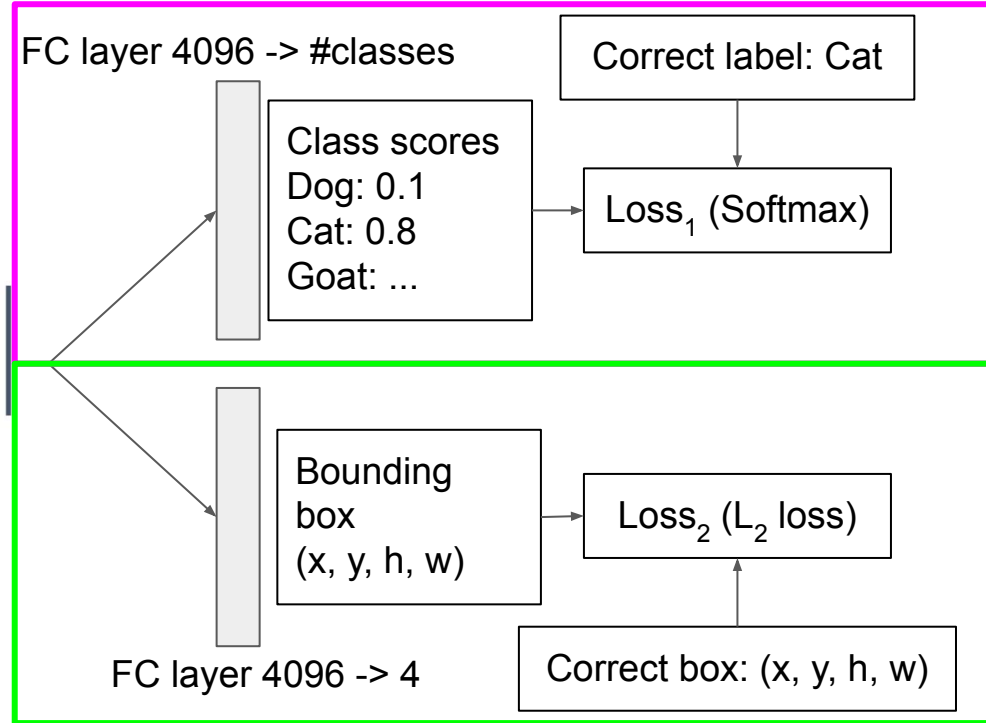


Classification

Network: Pretrained

Localization

Question: What about multiple objects/image?



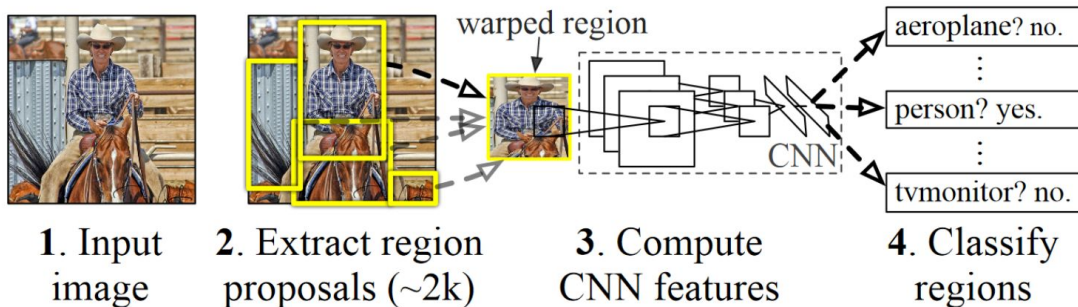
For multiple objects/image this algorithm
would be highly inefficient!

Region-based CNN (R-CNN)

Region-based CNN

1. takes an input image
2. extracts around 2000 bottom-up region proposals
3. computes features for each proposal using a large CNN
4. classifies each region using class-specific linear SVMs

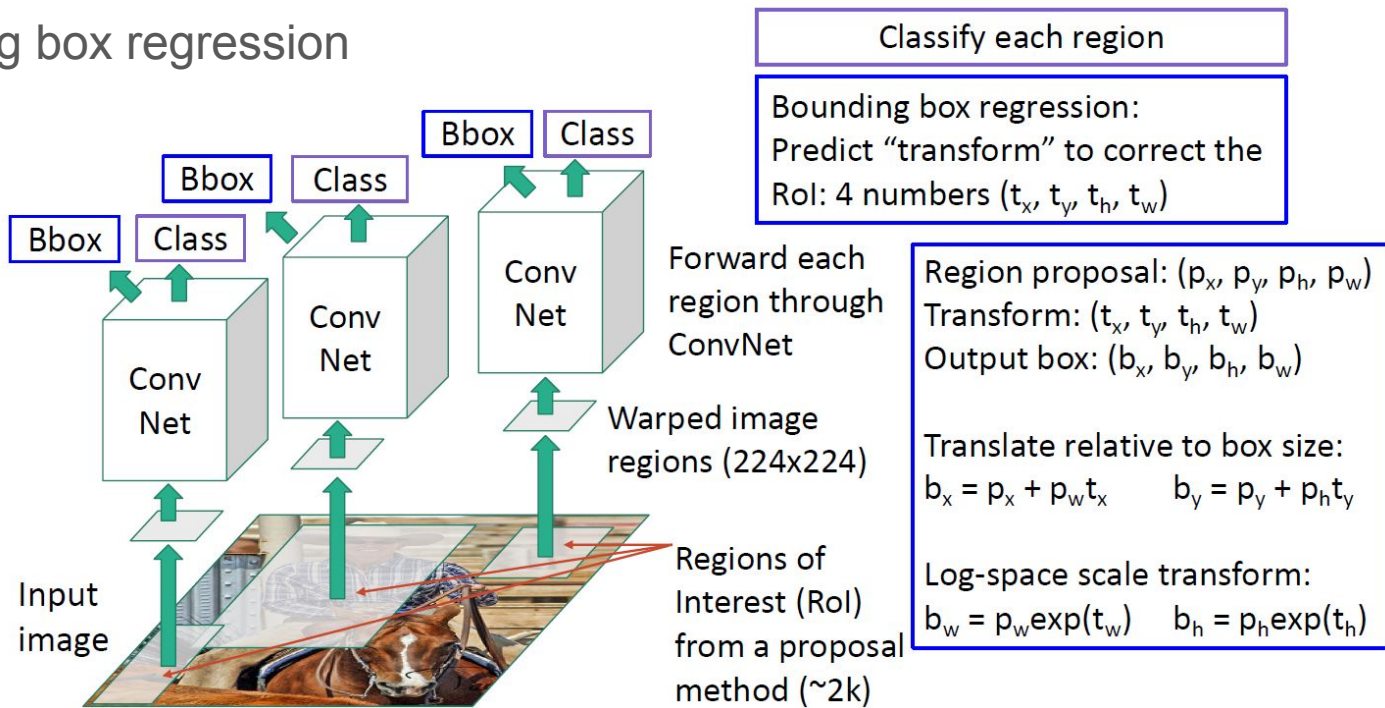
R-CNN: *Regions with CNN features*



- R-CNN achieves a mean average precision (mAP) of 53.7% on PASCAL VOC 2010.
- On the 200-class ILSVRC 2013 detection dataset, R-CNN's mAP is 31.4%

R-CNN

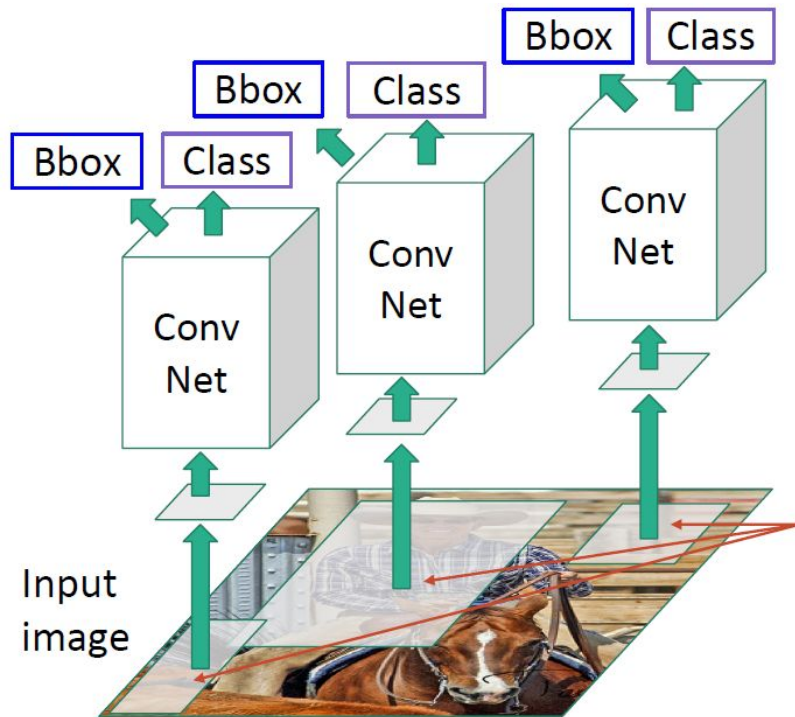
Bounding box regression



R-CNN

Inference time: for an RGB input image

1. Run region proposal method to compute ~2000 region proposals
2. Resize each region to 224x224 and run independently through CNN to predict **class scores** and **bbox transform**
3. Use scores to **filter** a subset of region proposals to output
4. Compare with ground-truth boxes

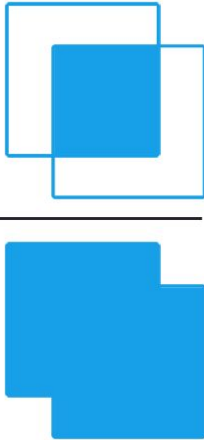


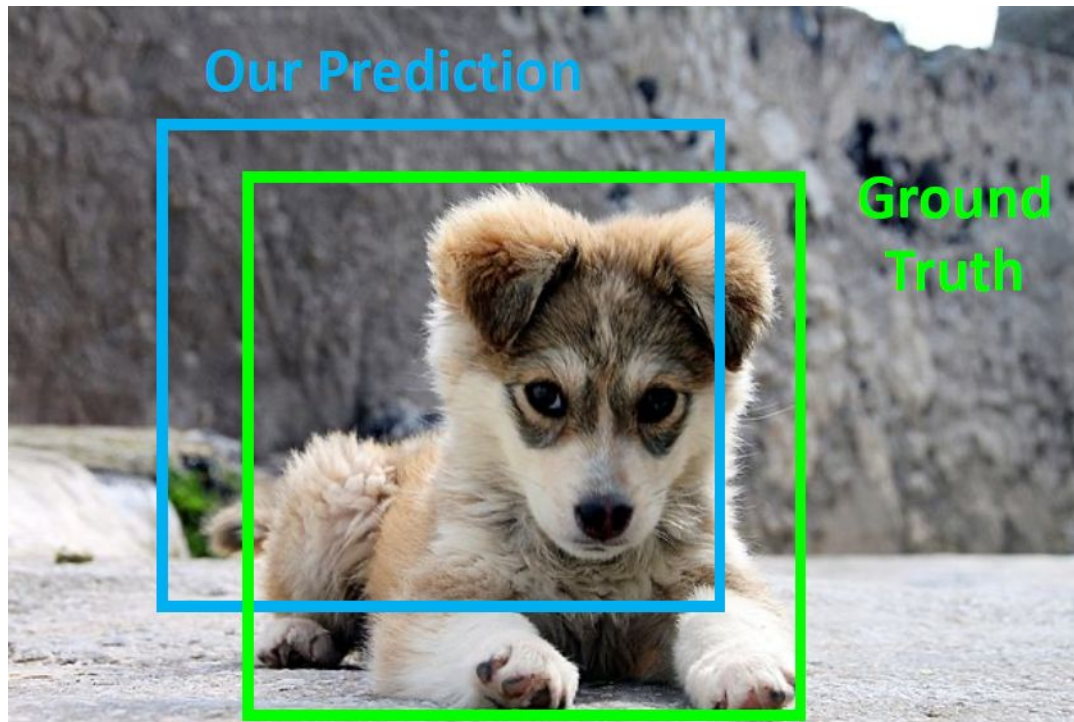
How to evaluate predictions for object detection?

Concept: IoU

How to compare the quality of bounding box prediction?

Intersection over Union (IoU)

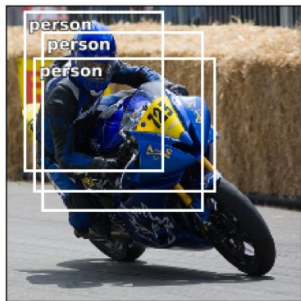
$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$




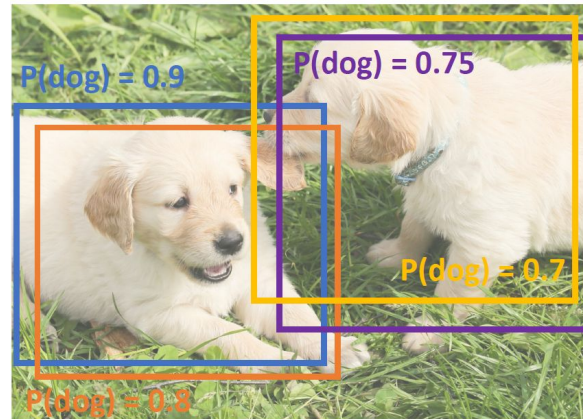
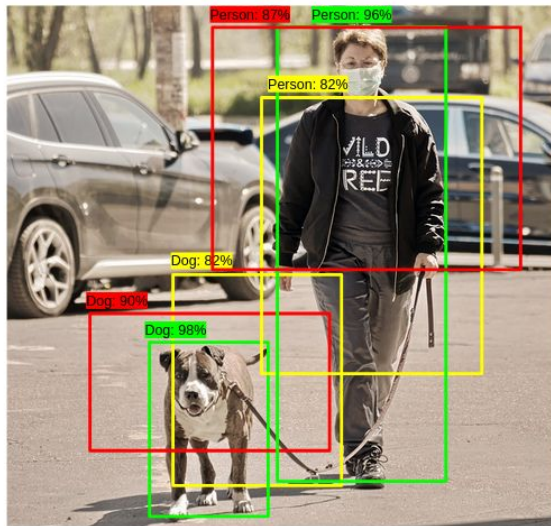
Concept: Non Max Suppression (NMS)

Model usually outputs multiple redundant boxes per object

For each class...

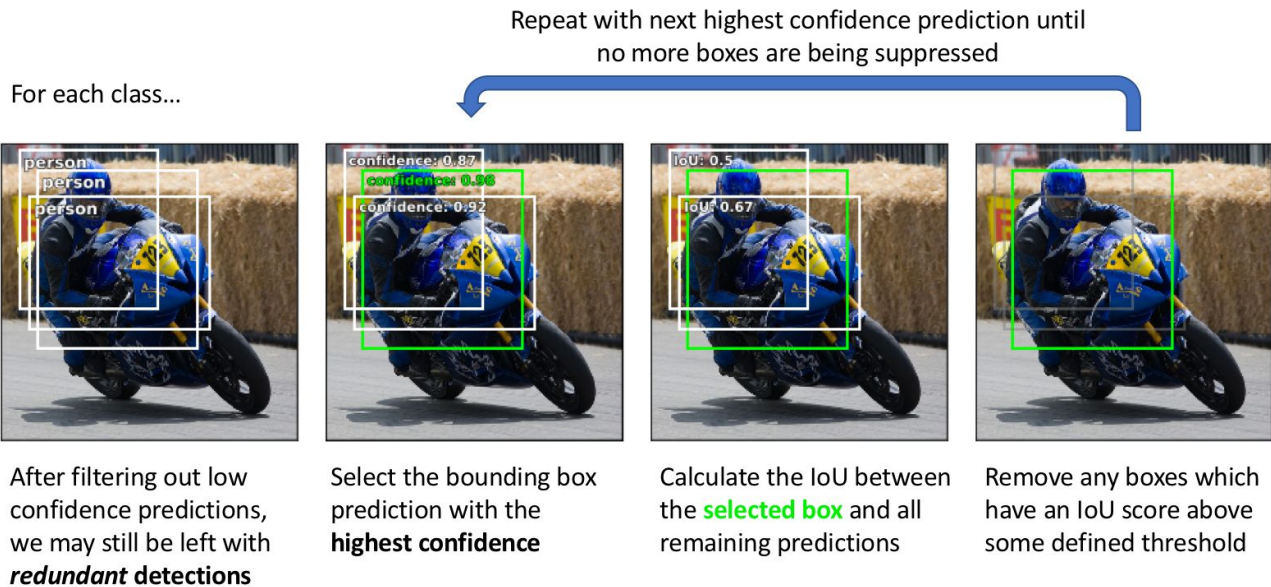


After filtering out low confidence predictions, we may still be left with **redundant detections**



Concept: Non Max Suppression (NMS)

Apply NMS



Concept: Non Max Suppression (NMS)

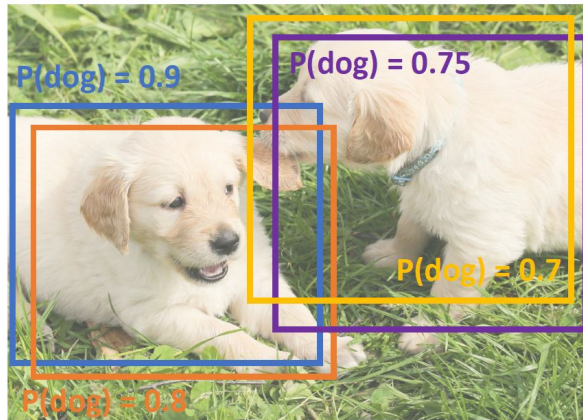
NMS Algorithm

Pseudocode

```
sorted_boxes = sort_boxes_by_confidence(boxes)
ids_to_suppress = []
```

```
for maximum_box in sorted_boxes:
    for idx, box in enumerate(boxes):
        iou = compute_iou(maximum_box, box)
        if iou > iou_threshold:
            ids_to_suppress.append(idx)
```

```
processed_boxes = np.delete(boxes, ids_to_suppress)
```



Concept: TP, FP, FN, P, R

True positive (TP): A correct detection. Detection with $\text{IoU} \geq \text{threshold}$

False positive (FP): An incorrect detection of a nonexistent object or a misplaced detection of an existing object. Detection with $\text{IoU} < \text{threshold}$

False negative (FN): An undetected ground-truth bounding box

Precision (P): $\text{TP} / (\text{TP} + \text{FP}) = \text{TP} / \text{All detections}$ $P = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{TP}}{\text{all detections}},$

Recall (R): $\text{TP} / (\text{TP} + \text{FN}) = \text{TP} / \text{All ground truths}$ $R = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{TP}}{\text{all ground truths}}.$

Concept: AP

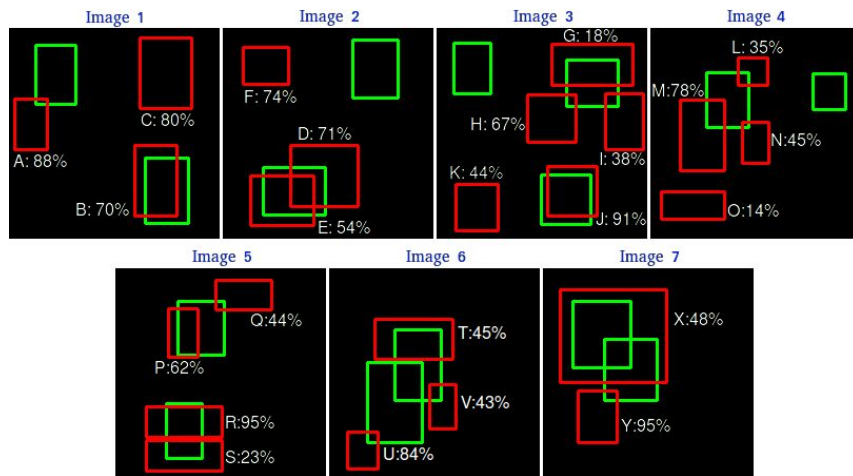
Average Precision (AP):

For *each* detection of a **single category** (highest score to lowest score)

1. If it matches some GT box with $\text{IoU} > \mathbf{0.5}$, mark it as true positive (TP) and eliminate the GT
2. Otherwise mark it as negative (FP)
3. Plot a point on PR Curve
4. Average Precision (AP) = area under PR curve

Concept: AP

There are 7 images with 15 ground truth objects represented by the green bounding boxes and 24 detected objects represented by the red bounding boxes. Each detected object has a confidence level and is identified by a letter (A,B,...,Y).

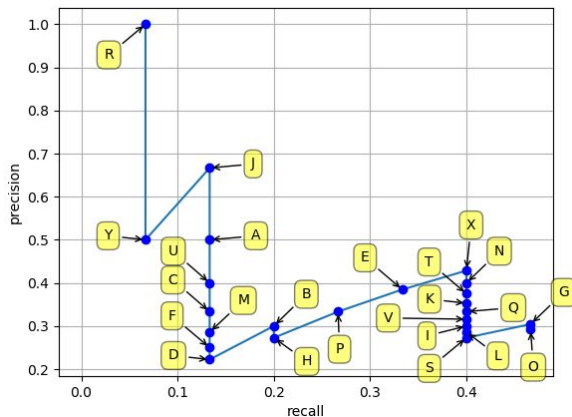


Images	Detections	Confidences	TP or FP
Image 1	A	88%	FP
Image 1	B	70%	TP
Image 1	C	80%	FP
Image 2	D	71%	FP
Image 2	E	54%	TP
Image 2	F	74%	FP
Image 3	G	18%	TP
Image 3	H	67%	FP
Image 3	I	38%	FP
Image 3	J	91%	TP
Image 3	K	44%	FP
Image 4	L	35%	FP
Image 4	M	78%	FP
Image 4	N	45%	FP
Image 4	O	14%	FP
Image 5	P	62%	TP
Image 5	Q	44%	FP
Image 5	R	95%	TP
Image 5	S	23%	FP
Image 6	T	45%	FP
Image 6	U	84%	FP
Image 6	V	43%	FP
Image 7	X	48%	TP
Image 7	Y	95%	FP

The following table shows the bounding boxes with their corresponding confidences. The last column identifies the detections as TP or FP. In this example a TP is considered if IoU 30%, otherwise it is a FP.

Concept: AP

- First we need to order the detections by their confidences, then we calculate the precision and recall for each accumulated detection.
- Plotting the precision and recall values we have the following Precision vs. Recall curve



Images	Detections	Confidences	TP	FP	Acc TP	Acc FP	Precision	Recall
Image 5	R	95%	1	0	1	0	1	0.0666
Image 7	Y	95%	0	1	1	1	0.5	0.0666
Image 3	J	91%	1	0	2	1	0.6666	0.1333
Image 1	A	88%	0	1	2	2	0.5	0.1333
Image 6	U	84%	0	1	2	3	0.4	0.1333
Image 1	C	80%	0	1	2	4	0.3333	0.1333
Image 4	M	78%	0	1	2	5	0.2857	0.1333
Image 2	F	74%	0	1	2	6	0.25	0.1333
Image 2	D	71%	0	1	2	7	0.2222	0.1333
Image 1	B	70%	1	0	3	7	0.3	0.2
Image 3	H	67%	0	1	3	8	0.2727	0.2
Image 5	P	62%	1	0	4	8	0.3333	0.2666
Image 2	E	54%	1	0	5	8	0.3846	0.3333
Image 7	X	48%	1	0	6	8	0.4285	0.4
Image 4	N	45%	0	1	6	9	0.4	0.4
Image 6	T	45%	0	1	6	10	0.375	0.4
Image 3	K	44%	0	1	6	11	0.3529	0.4
Image 5	Q	44%	0	1	6	12	0.3333	0.4
Image 6	V	43%	0	1	6	13	0.3157	0.4
Image 3	I	38%	0	1	6	14	0.3	0.4
Image 4	L	35%	0	1	6	15	0.2857	0.4
Image 5	S	23%	0	1	6	16	0.2727	0.4
Image 3	G	18%	1	0	7	16	0.3043	0.4666
Image 4	O	14%	0	1	7	17	0.2916	0.4666

Concept: mAP

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
3. Mean Average Precision (mAP) = average of AP for each category

Car AP = 0.65

Cat AP = 0.80

Dog AP = 0.86

mAP@**0.5** = 0.77

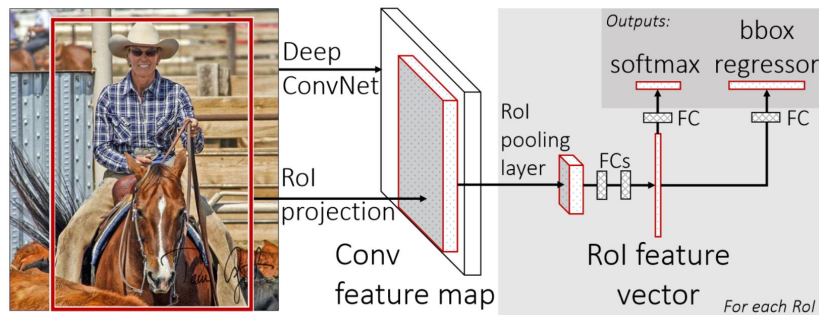
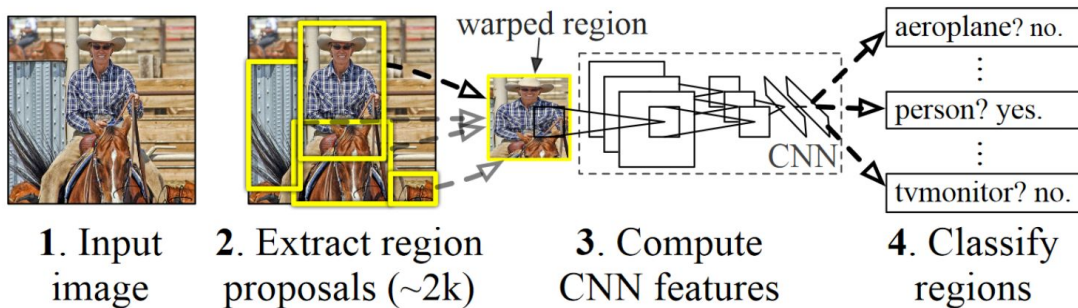
If a prediction matches some GT box with IoU > **0.5**, mark it as true positive (TP) and eliminate the GT

Region-based CNN (R-CNN)

Problem

Very slow because computing CNN feature maps for $\sim 2k$ region proposals

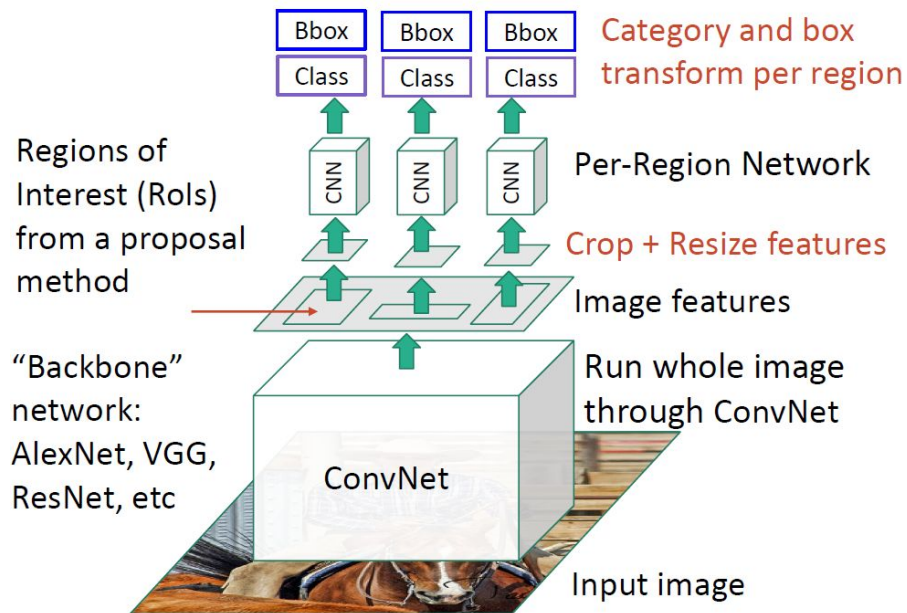
Workaround: what if instead of passing each region proposals through a CNN, we first compute the feature map for the whole image and use projected region proposals onto the feature map?



Fast R-CNN

Fast R-CNN

- An input image and multiple regions of interest (Rois) are input into a fully convolutional network.
- Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs).
- The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets.
- The architecture is trained end-to-end with a multi-task loss.

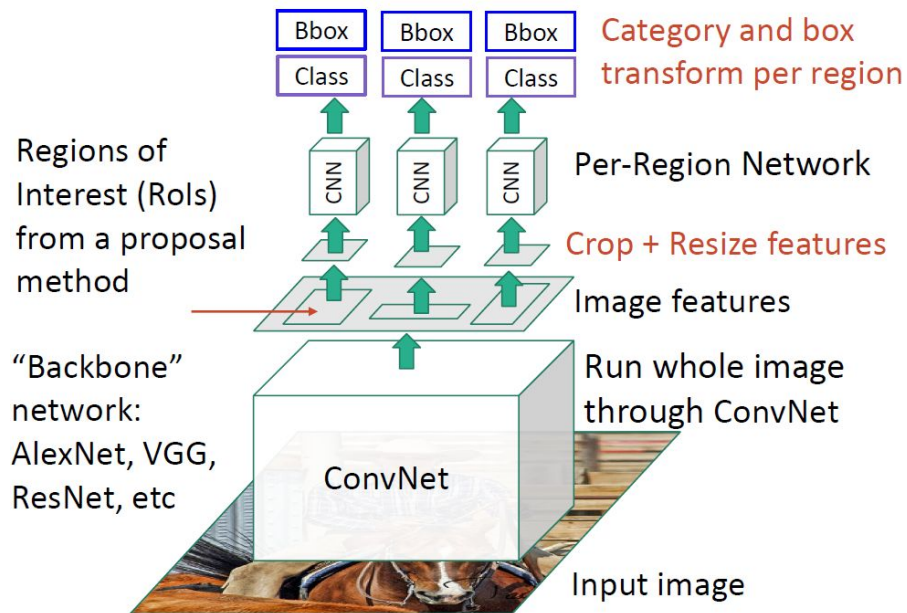


Fast R-CNN

Performance

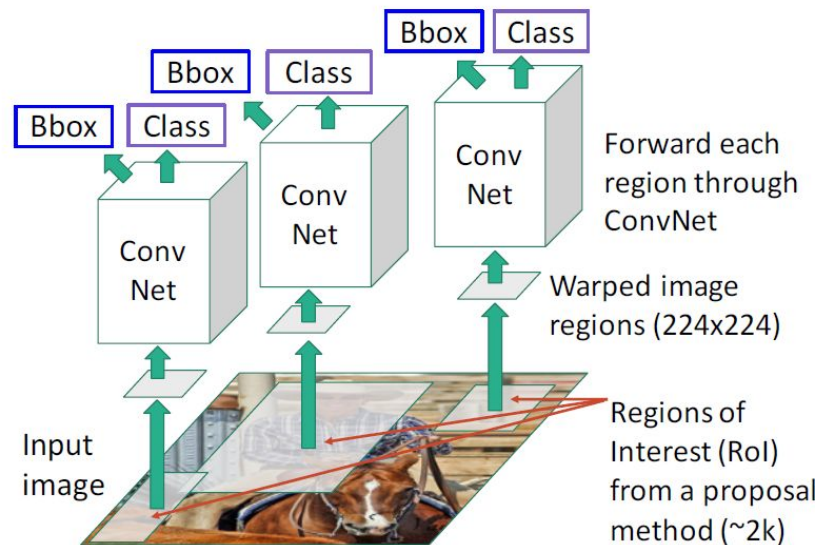
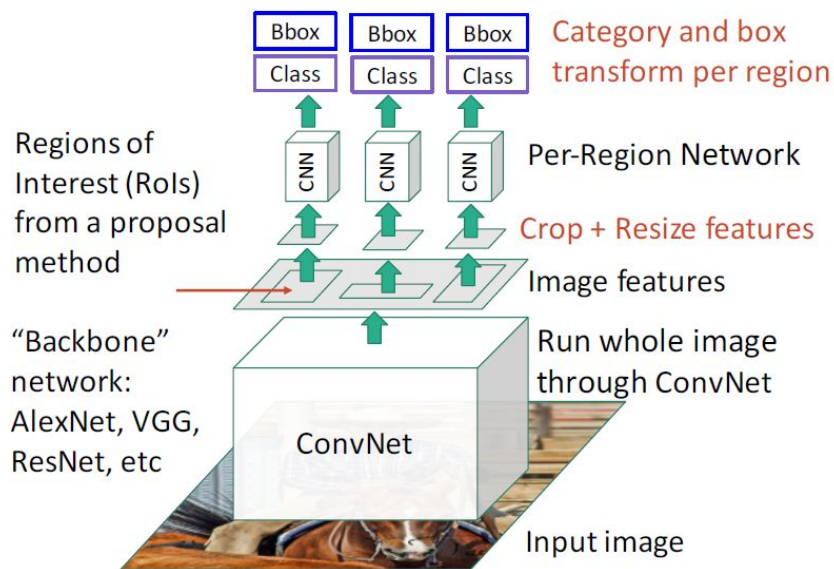
Fast R-CNN achieves the top result on VOC12 with a mAP of 65.7%

R-CNN achieves a mAP of 66.0% on the MS COCO dataset.



Fast R-CNN Vs R-CNN

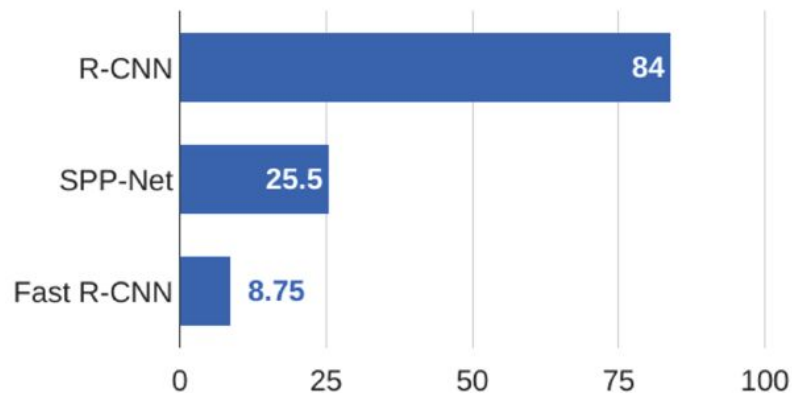
Architecture comparison



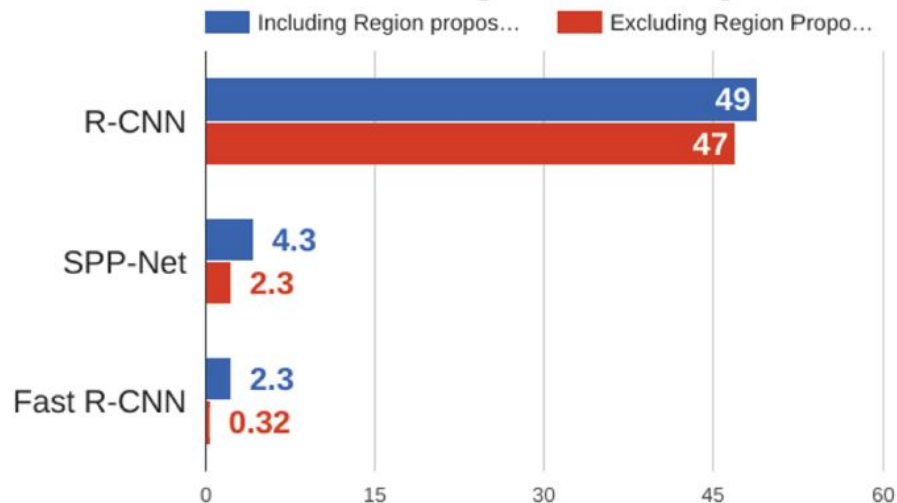
Fast R-CNN Vs R-CNN

Performance comparison

Training time (Hours)



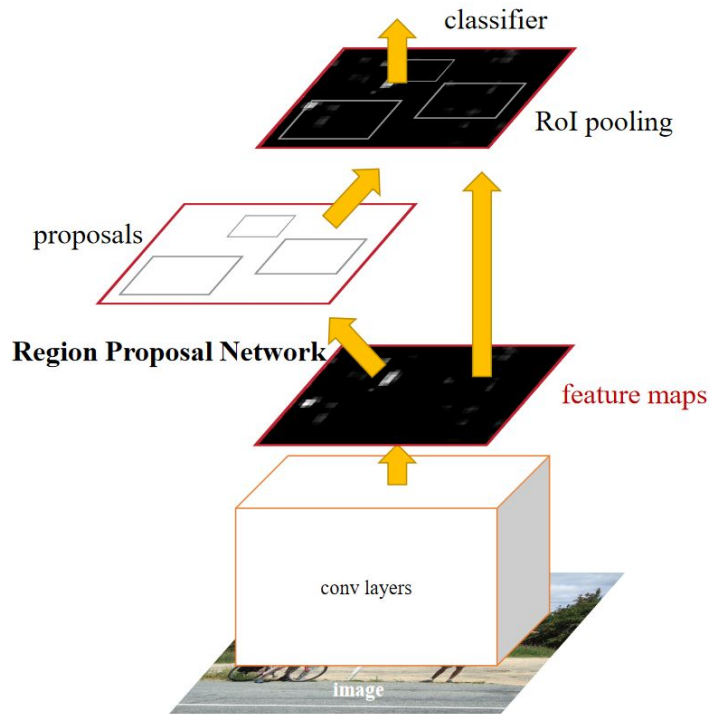
Test time (seconds)



Faster R-CNN

Faster R-CNN

1. Regions of interest are generated using the region proposal network (RPN). To generate Rols, the RPN uses convolutional layers.
2. The second part of Faster R-CNN is classification. It outputs the final bounding boxes and accepts two inputs—the list of Rols from the previous step (RPN), and a feature volume computed from the input image.



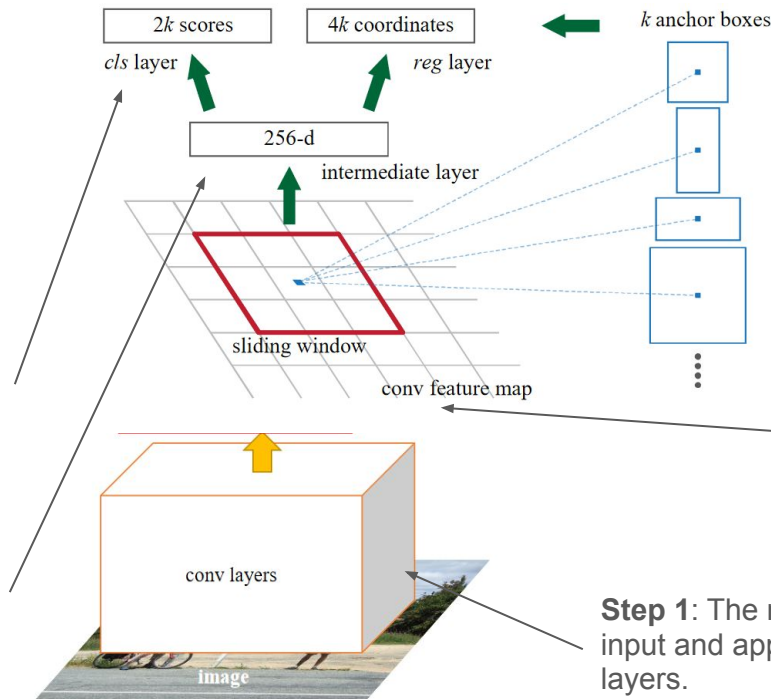
Faster R-CNN

RPN Stage

- **Input:** RGB Image
- **Output:** list of Rols

Step 4: Two sibling 1×1 convolutional layers compute the objectness scores and the bounding box coordinates. There are two objectness scores for each of the k bounding boxes.

Step 3: At each position in the feature volume, each sliding window is mapped to a lower-dimensional feature vector (256-d for ZF and 512-d for VGG) for k anchor boxes



Step 5: Postprocessing to keep only regions with highest confidence (~300)

Step 2: It outputs a feature volume. A convolutional filter is applied over the feature volume.

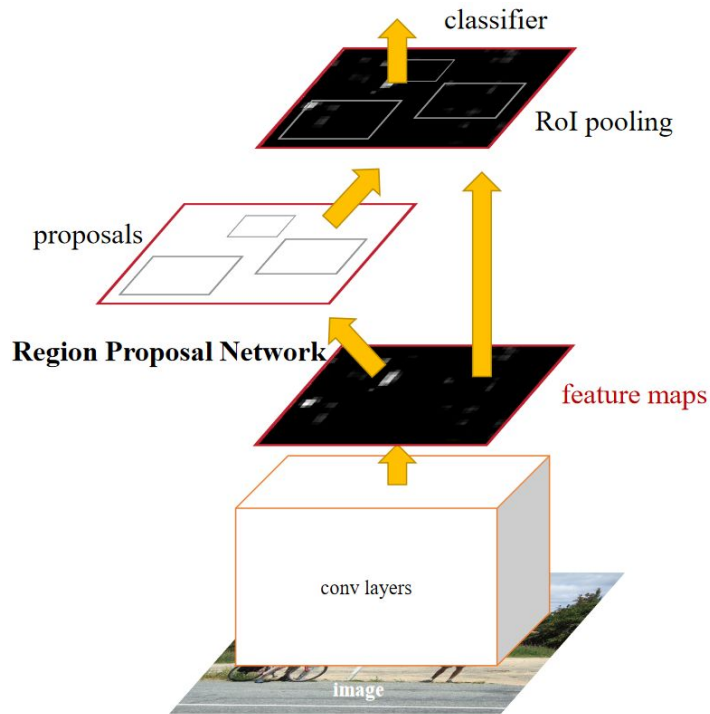
Step 1: The network accepts an image as input and applies several convolutional layers.

Faster R-CNN

Classification Stage

- **Input:** list of Rols + Conv feature map
- **Output:** Class score + bbox refinement

1. Accept the feature maps and the Rols from the RPN step.
2. Resize each Rol to make it fit the input of the fully connected layers.
3. Apply the fully connected layer. It is very similar to the final layers of any convolutional network. We obtain a feature vector.
4. Apply two different convolutional layers. One handles the classification (called cls) and the other handles the refinement of the Rol (called rgs).

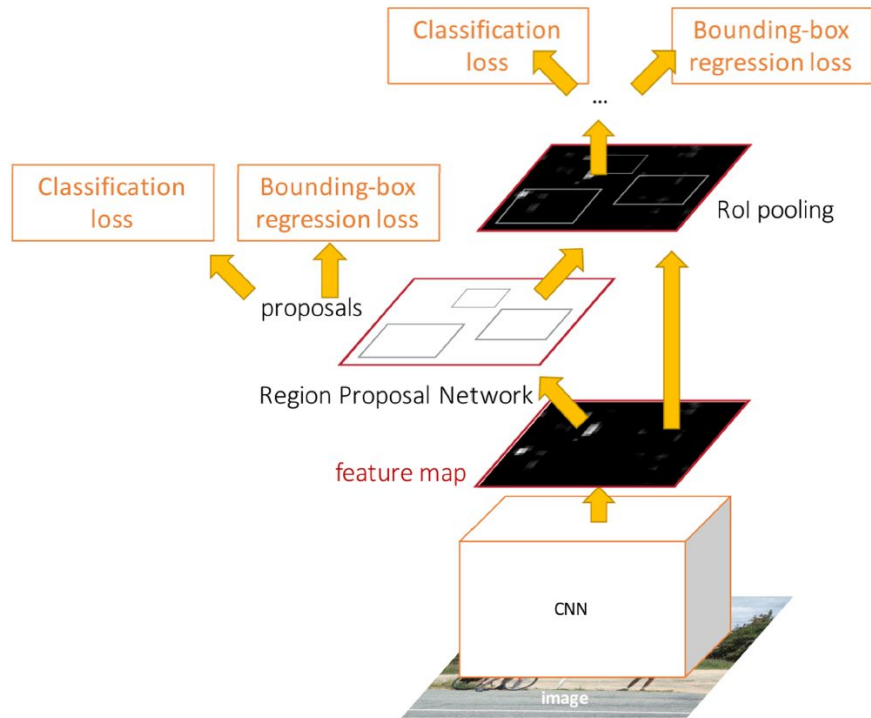


Faster R-CNN

Training

Jointly train with 4 losses:

- 1. RPN classification:** anchor box is object / not an object
- 2. RPN regression:** predict transform from anchor box to proposal box
- 3. Object classification:** classify proposals as background / object class
- 4. Object regression:** predict transform from proposal box to object box



R-CNN, Fast R-CNN & Faster R-CNN

Algorithm	Features	Prediction time / image	Limitations
CNN	Divides the image into multiple regions and then classify each region into various classes.	-	Needs a lot of regions to predict accurately and hence high computation time.
RCNN	Uses selective search to generate regions. Extracts around 2000 regions from each image.	40-50 seconds	High computation time as each region is passed to the CNN separately also it uses three different model for making predictions.
Fast RCNN	Each image is passed only once to the CNN and feature maps are extracted. Selective search is used on these maps to generate predictions. Combines all the three models used in RCNN together.	2 seconds	Selective search is slow and hence computation time is still high.
Faster RCNN	Replaces the selective search method with region proposal network which made the algorithm much faster.	0.2 seconds	Object proposal takes time and as there are different systems working one after the other, the performance of systems depends on how the previous system has performed.

Resources

1. <https://cs231n.github.io/convolutional-networks/>
2. <https://web.eecs.umich.edu/~justincj/teaching/eecs498/FA2020/>
3. https://www.tensorflow.org/api_docs/python/tf/keras
4. Deep Learning with Python Book by François Chollet
5. <https://www.deeplearningbook.org/>
6. **Hands-on Computer Vision with TensorFlow 2 by Eliot Andres & Benjamin Planche (Packt Pub.)**