SCHOOL OF
HUMAN SCIENCES
& TECHNOLOGY

Object Detection

Chapter Goals

After completing this chapter, you should be able to understand :

- Understand different Object detection CNN architectures
- Understand SSD advantages compared to older object detection CNNs
- Be familiar with the Different improvements to YOLO
- Be familiar with Zero shot object detection
- Understand how to do annotation-free object detection

SCHOOL OF
HUMAN SCIENCES
& TECHNOLOGY

Object Detection

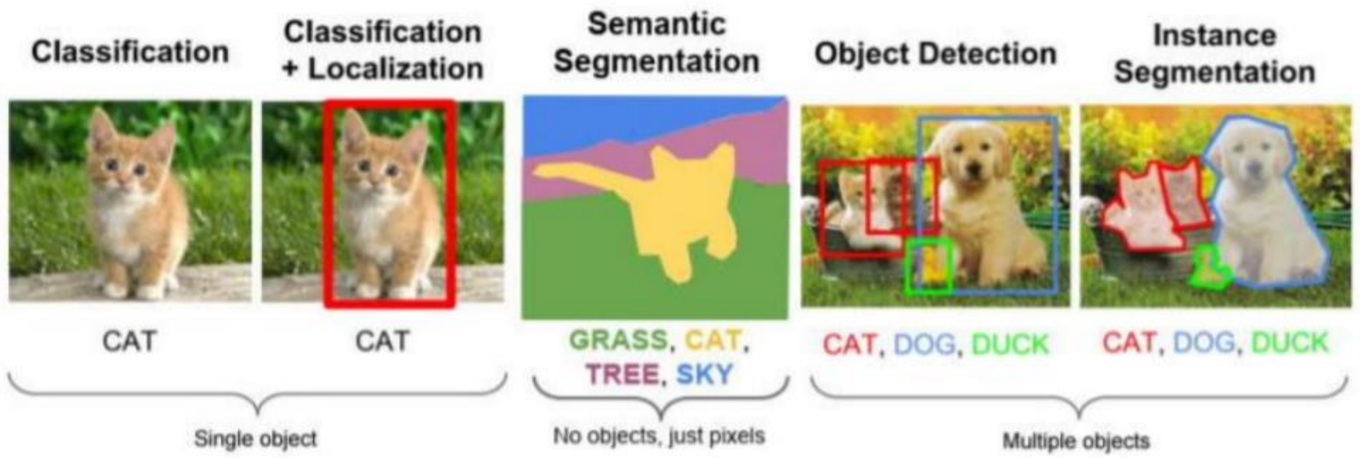
SCHOOL OF HUMAN SCIENCES & TECHNOLOGY

Object Detection Task

Input: Single RGB Image

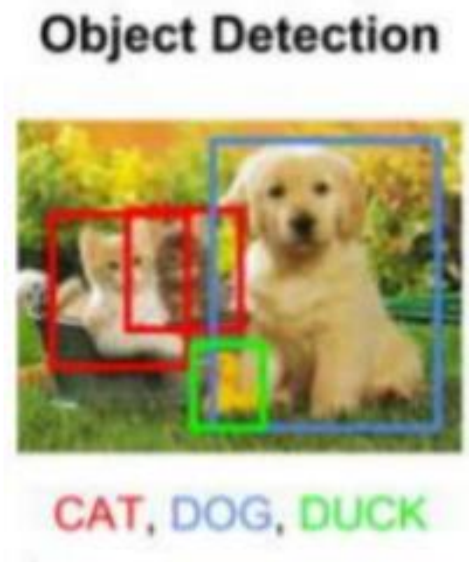
Output: A set of detected objects; For each object predict:

1. Category label (from fixed, known set of categories)
2. Bounding box (four numbers: x, y, width, height)

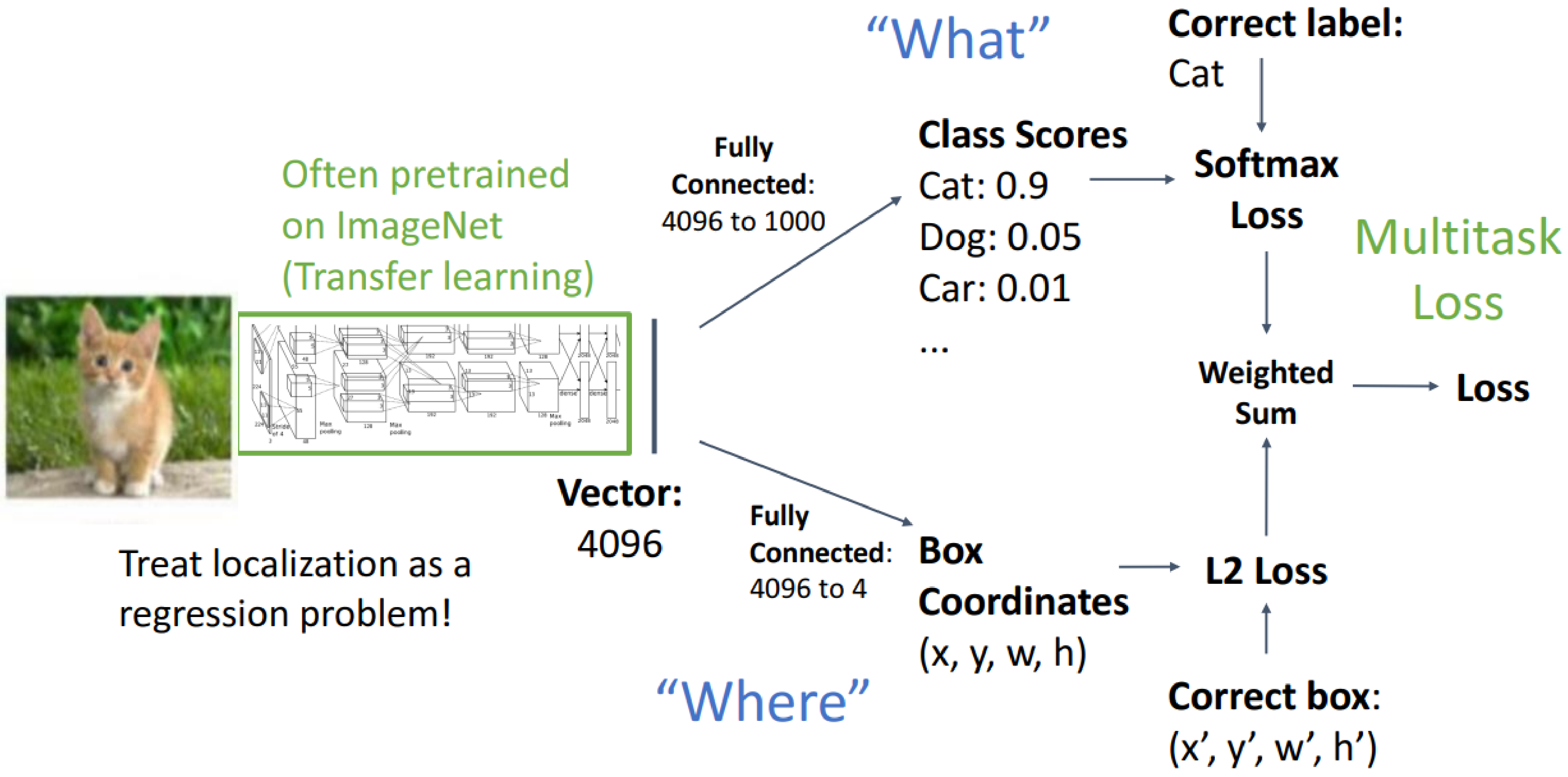


Object Detection Challenges

- **Multiple outputs:** Need to output variable numbers of objects per image
- **Multiple types of output:** Need to predict ”what” (category label) as well as “where” (bounding box)
- **Large images:** Classification works at 224x224; need higher resolution for detection,



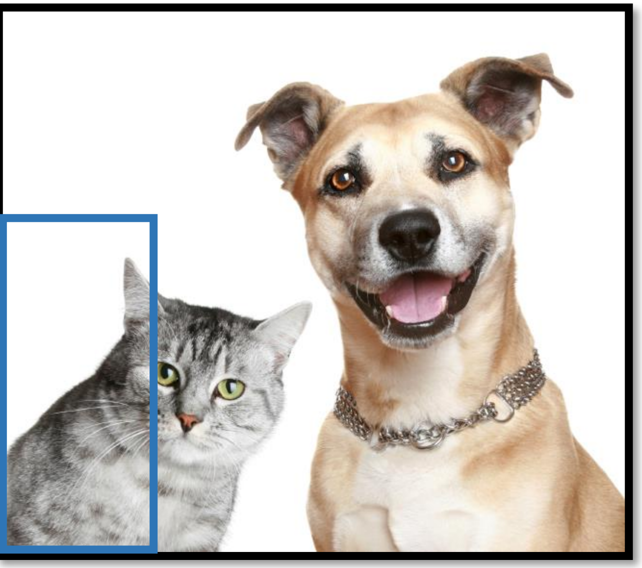
Detecting a single object

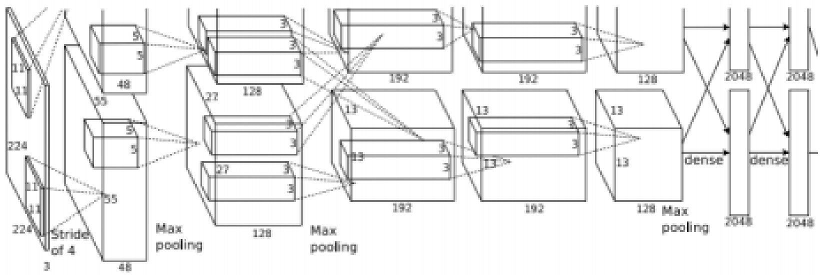

"What"
Correct label:
Cat
Often pretrained on ImageNet (Transfer learning)
Fully Connected: 4096 to 1000
Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...
Softmax Loss
Multitask Loss
Weighted Sum
Loss
Vector: 4096
Fully Connected: 4096 to 4
Box Coordinates (x, y, w, h)
L2 Loss
Correct box: (x', y', w', h')
"Where"

Treat localization as a regression problem!

Detecting multiple objects

We can think of a **sliding window** approach: Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

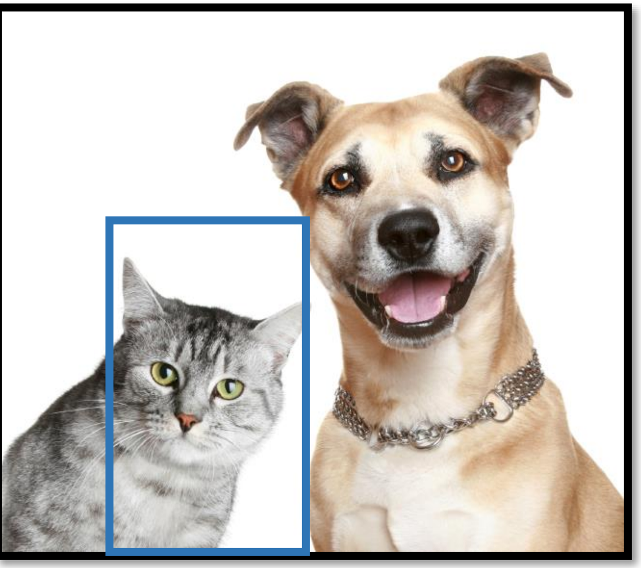


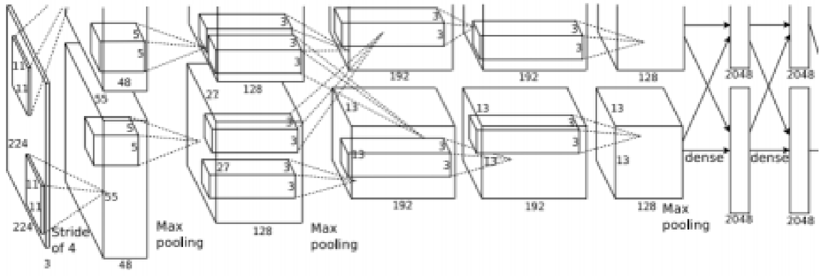


Dog? NO
Cat? NO
Background? YES

Detecting multiple objects

We can think of a **sliding window** approach: Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



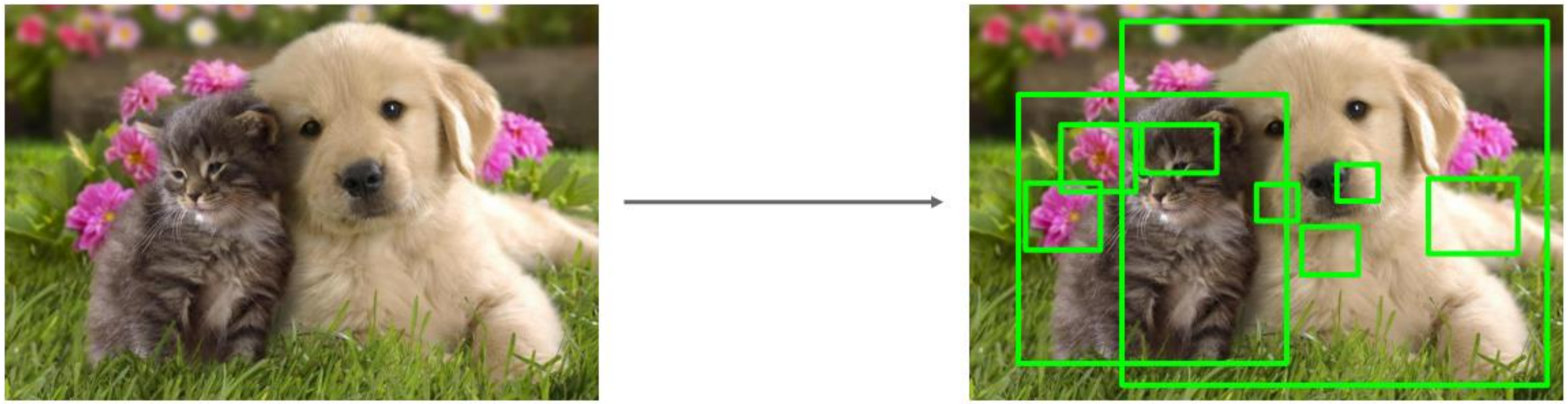


Dog? NO
Cat? YES
Background? NO

Limitation: The number of possible windows are way to high. This makes it unviable

Region Proposals

- Find a small set of boxes that are likely to cover all objects
- Often based on heuristics: e.g. look for "blob-like" image regions
- Relatively fast to run; e.g. Selective Search gives 2000 region proposals in a few seconds on CPU

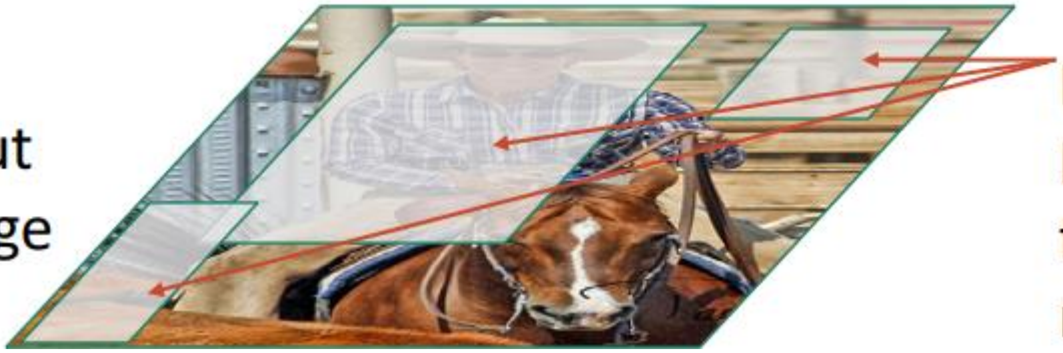


Alexe et al, "Measuring the objectness of image windows", TPAMI 2012
Uijlings et al, "Selective Search for Object Recognition", IJCV 2013
Cheng et al, "BING: Binarized normed gradients for objectness estimation at 300fps", CVPR 2014
Zitnick and Dollar, "Edge boxes: Locating object proposals from edges", ECCV 2014

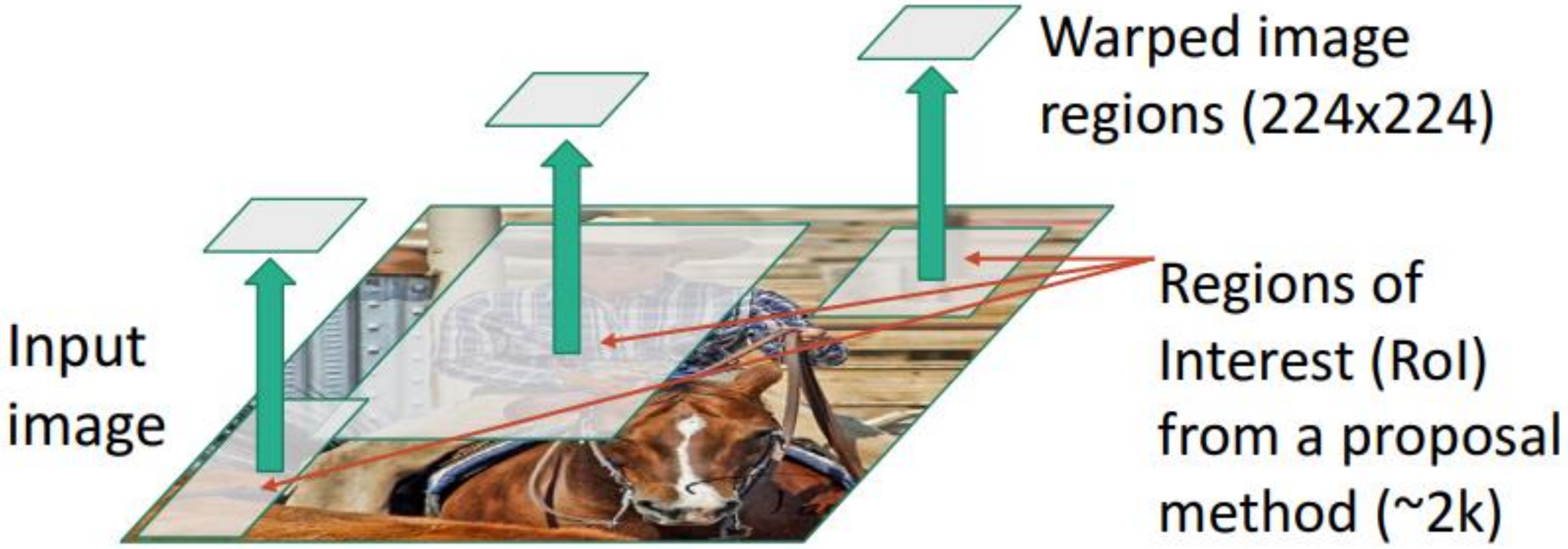
R-CNN: Region-Based CNN

Input image

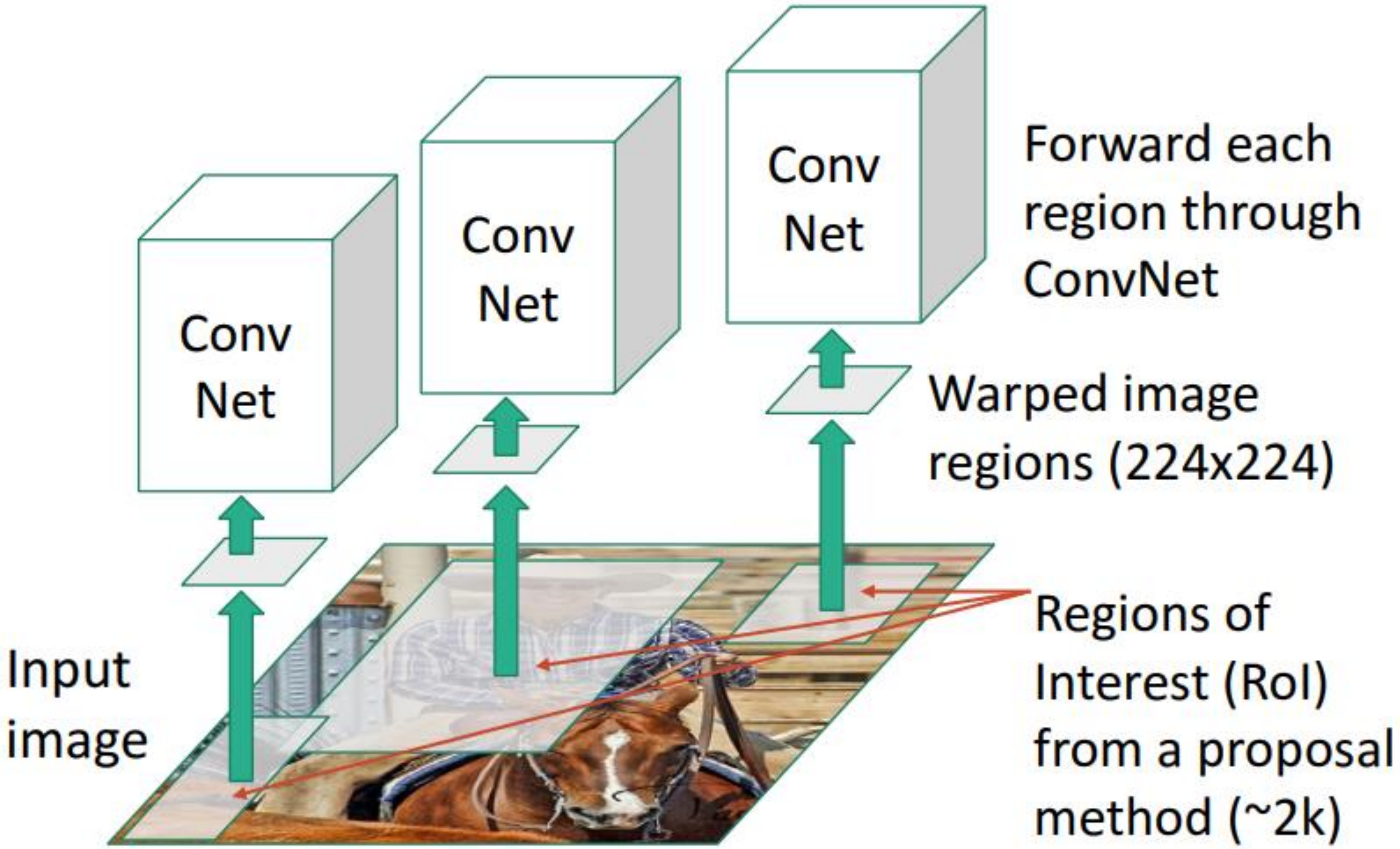
R-CNN: Region-Based CNN



R-CNN: Region-Based CNN

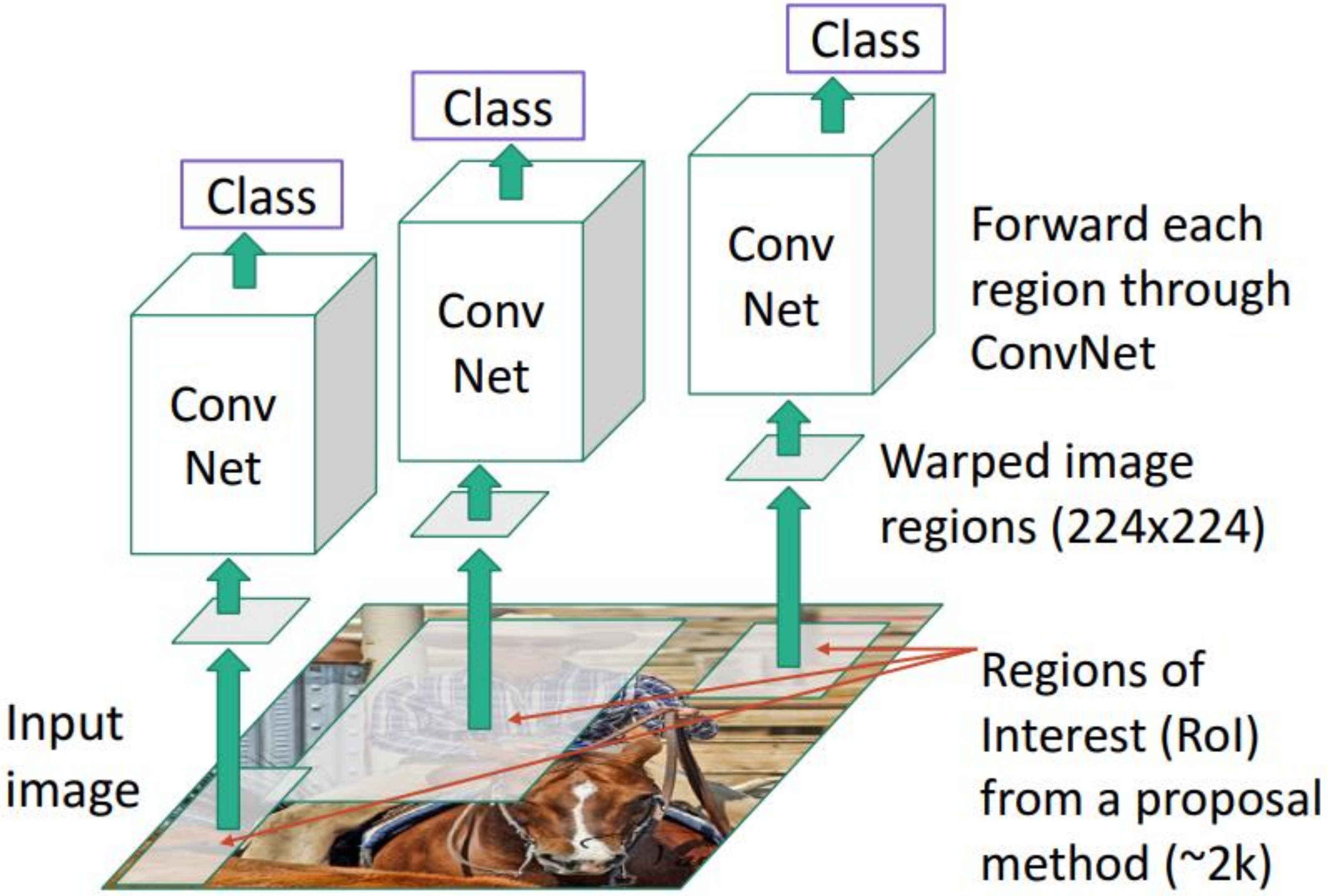

Warped image regions (224x224)
Regions of Interest (RoI) from a proposal method (~2k)
Input image

R-CNN: Region-Based CNN


Conv Net
Conv Net
Conv Net
Forward each region through ConvNet
Warped image regions (224x224)
Regions of Interest (RoI) from a proposal method (~2k)
Input image

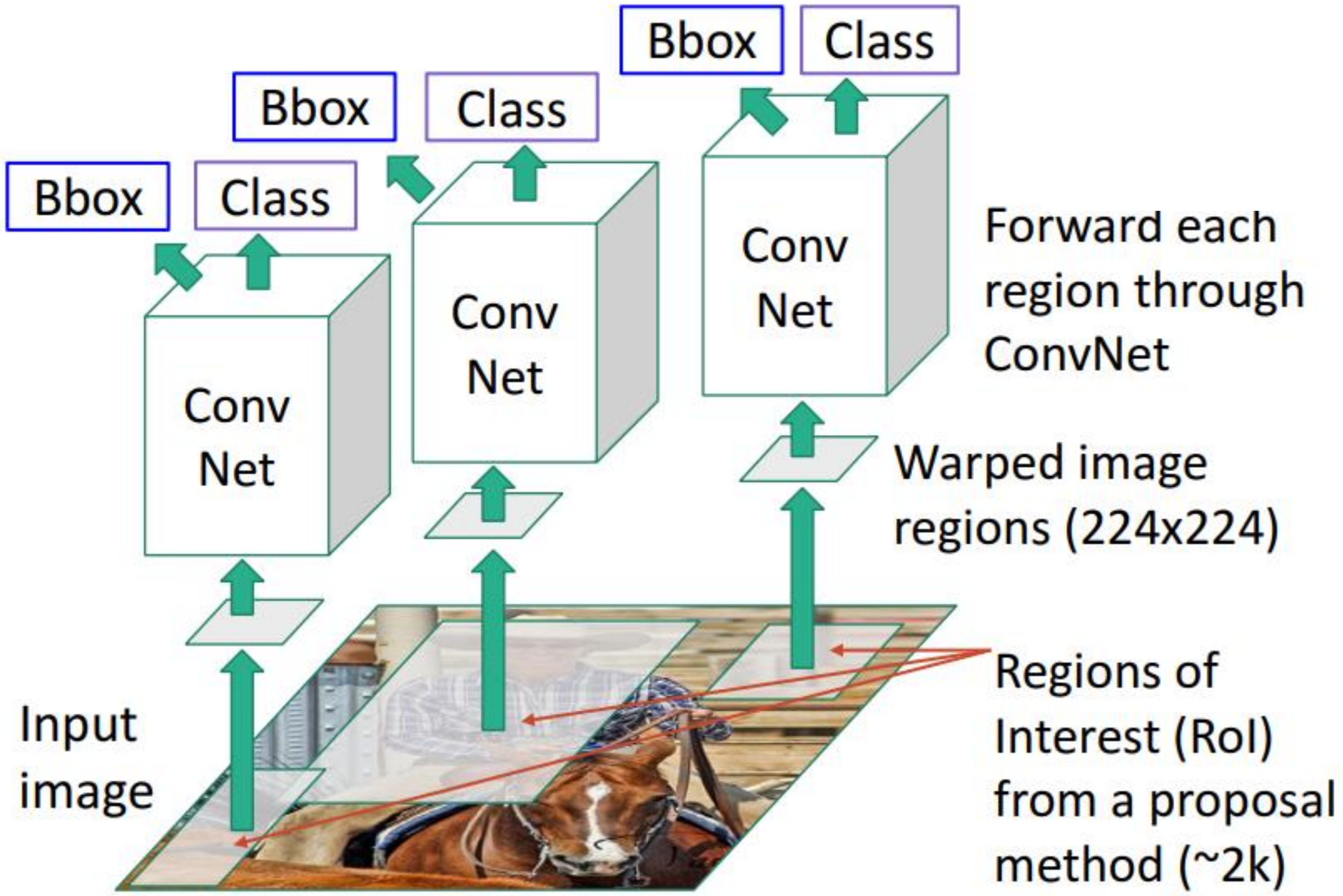
R-CNN: Region-Based CNN

Now we can classify each region


Class
Class
Class
Conv Net
Conv Net
Conv Net
Forward each region through ConvNet
Warped image regions (224x224)
Input image
Regions of Interest (RoI) from a proposal method (~2k)

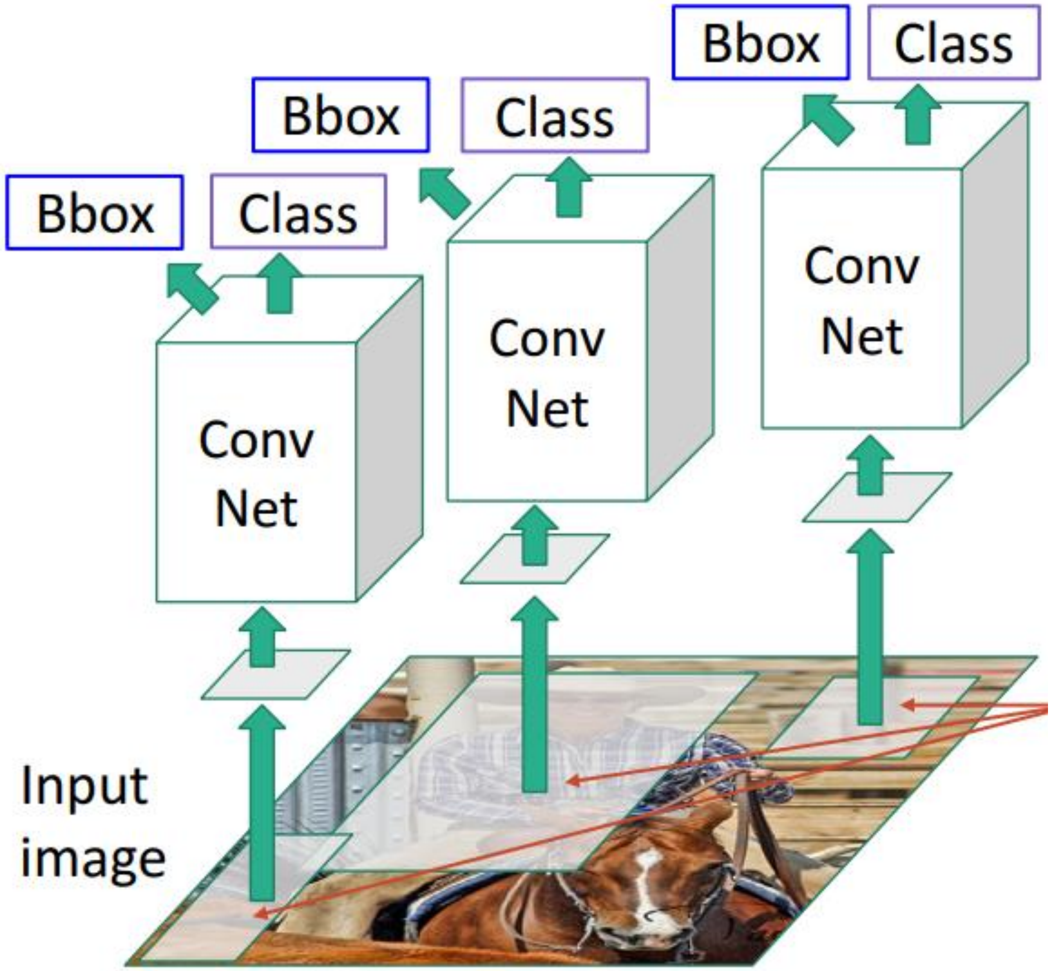
R-CNN: Region-Based CNN

Now we apply Bounding box regression:



Bbox
Class
Bbox
Class
Bbox
Class
Conv Net
Conv Net
Conv Net
Forward each region through ConvNet
Warped image regions (224x224)
Regions of Interest (RoI) from a proposal method (~2k)
Input image

R-CNN: Test-time



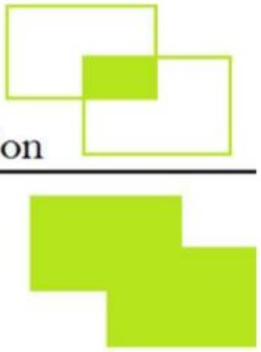
Input: Single RGB Image

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 select a subset of region proposals to output (Many choices here: threshold on background, or per-category Or take top K proposals per image?)
4. Compare with ground-truth boxes

SCHOOL OF HUMAN SCIENCES & TECHNOLOGY

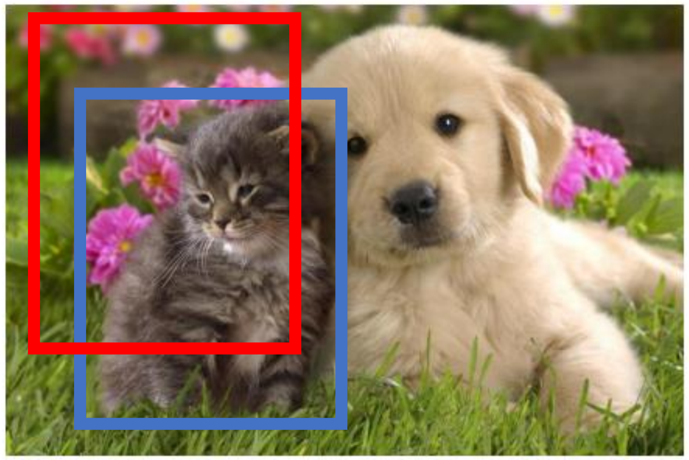
Comparing Boxes: Intersection over Union (IoU)

$$\text{IOU} = \frac{\text{Area of Intersection}}{\text{Area of Union}}$$



IOU can be computed as Area of Intersection divided over Area of Union of two boxes , so IOU must be ≥0 and ≤1 .
When predicting bounding boxes we need the find the IOU between the predicted bounding box and the ground truth box to be close to 1

Our Prediction



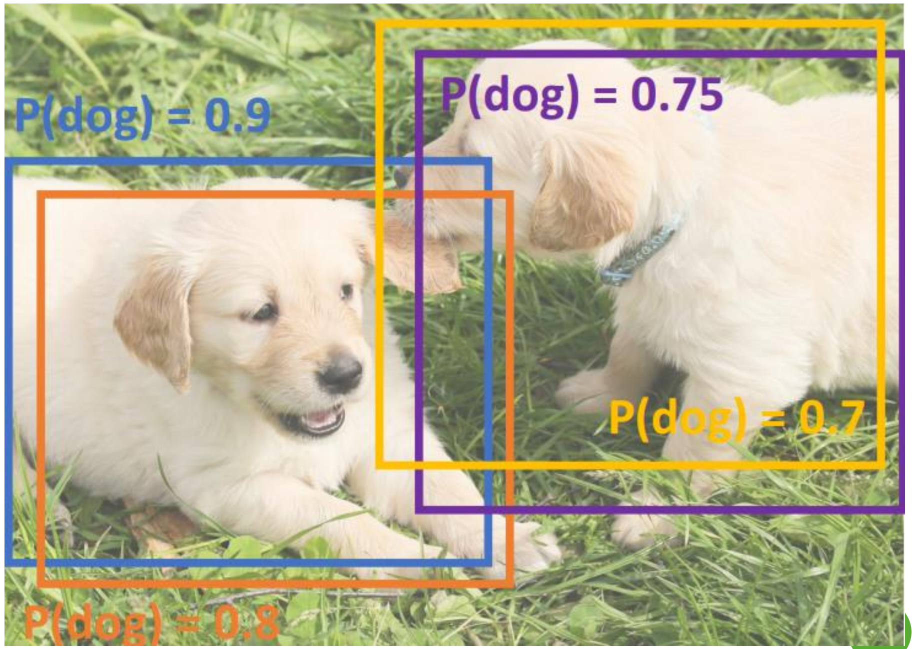
Ground truth

IoU > 0.5 is “decent”
IoU > 0.7 is “pretty good”
IoU > 0.9 is “almost perfect”

Overlapping Boxes: Non Max Suppression

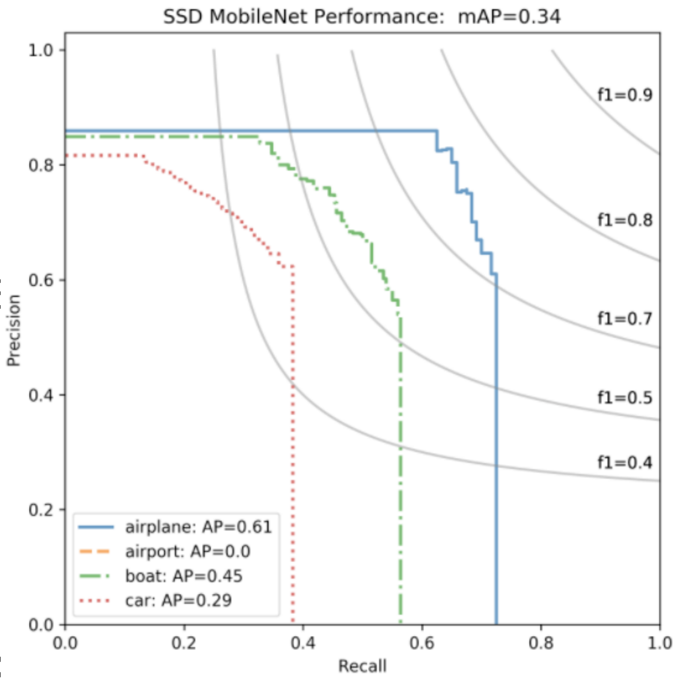
Problem: Object detectors often output many overlapping detections
Solution: Post-process raw detections using Non-Max Suppression (NMS)

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1



Evaluating Object Detectors: Mean Average Precision (mAP)

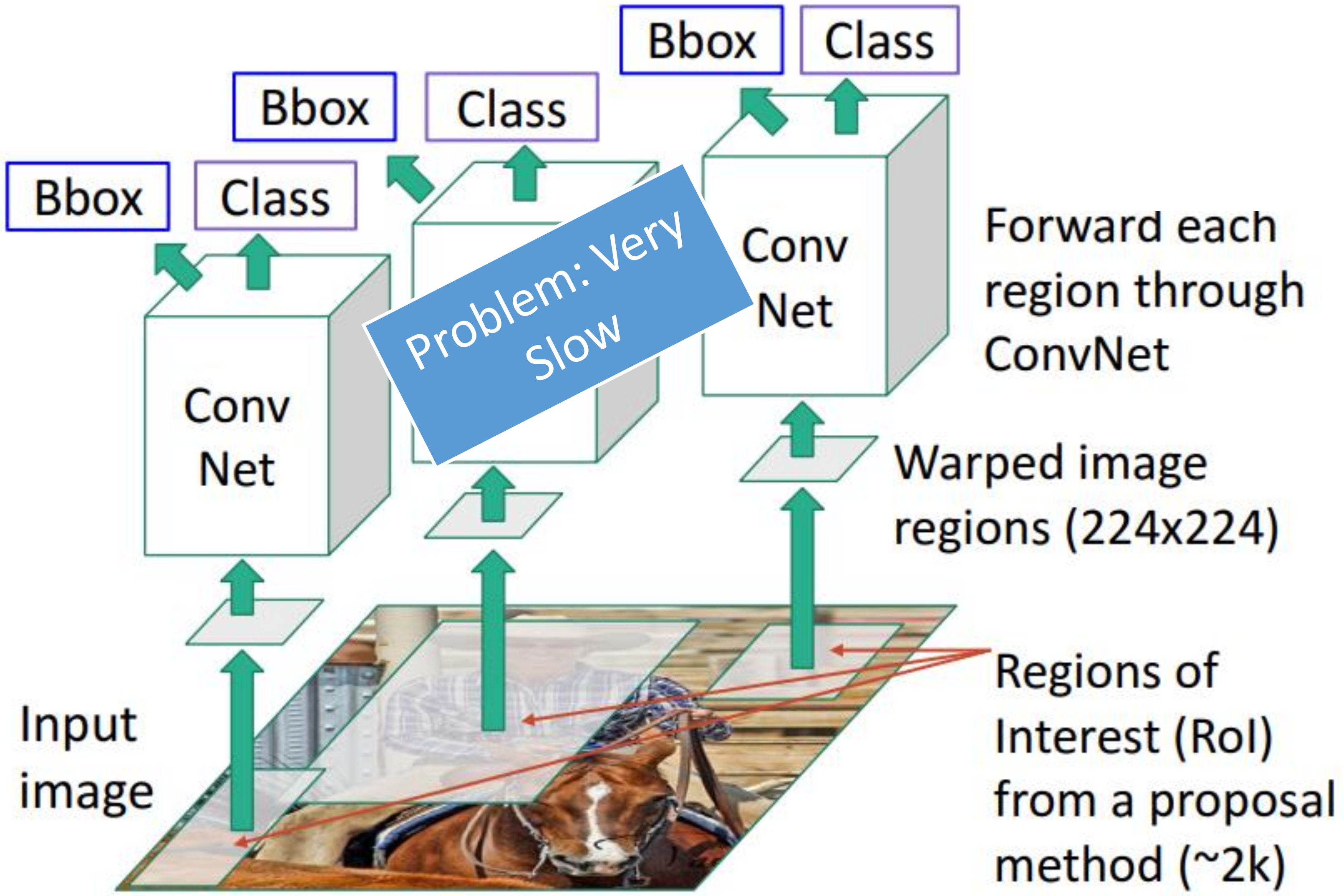
1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
   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



Beyond R-CNN

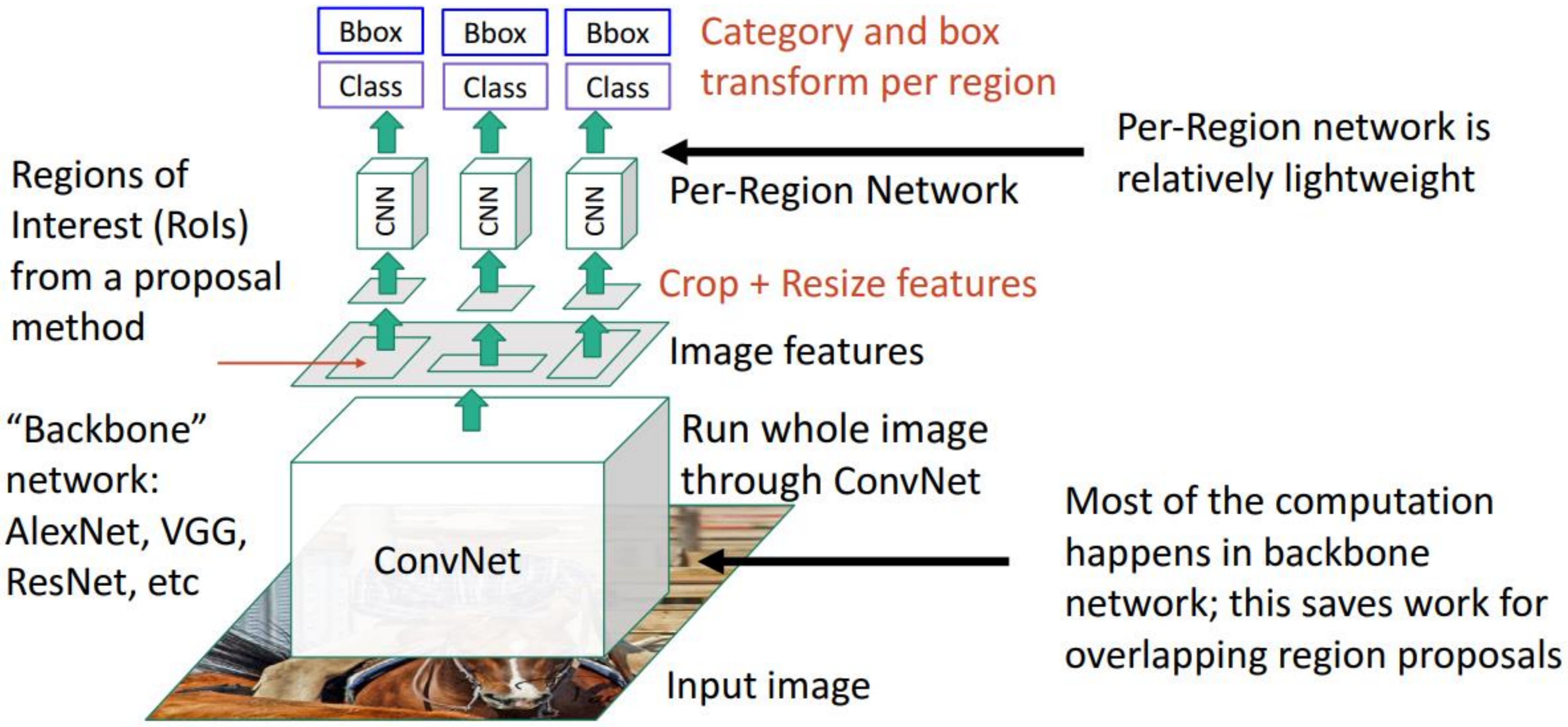
R-CNN: Region-Based CNN

Now we apply Bounding box regression:



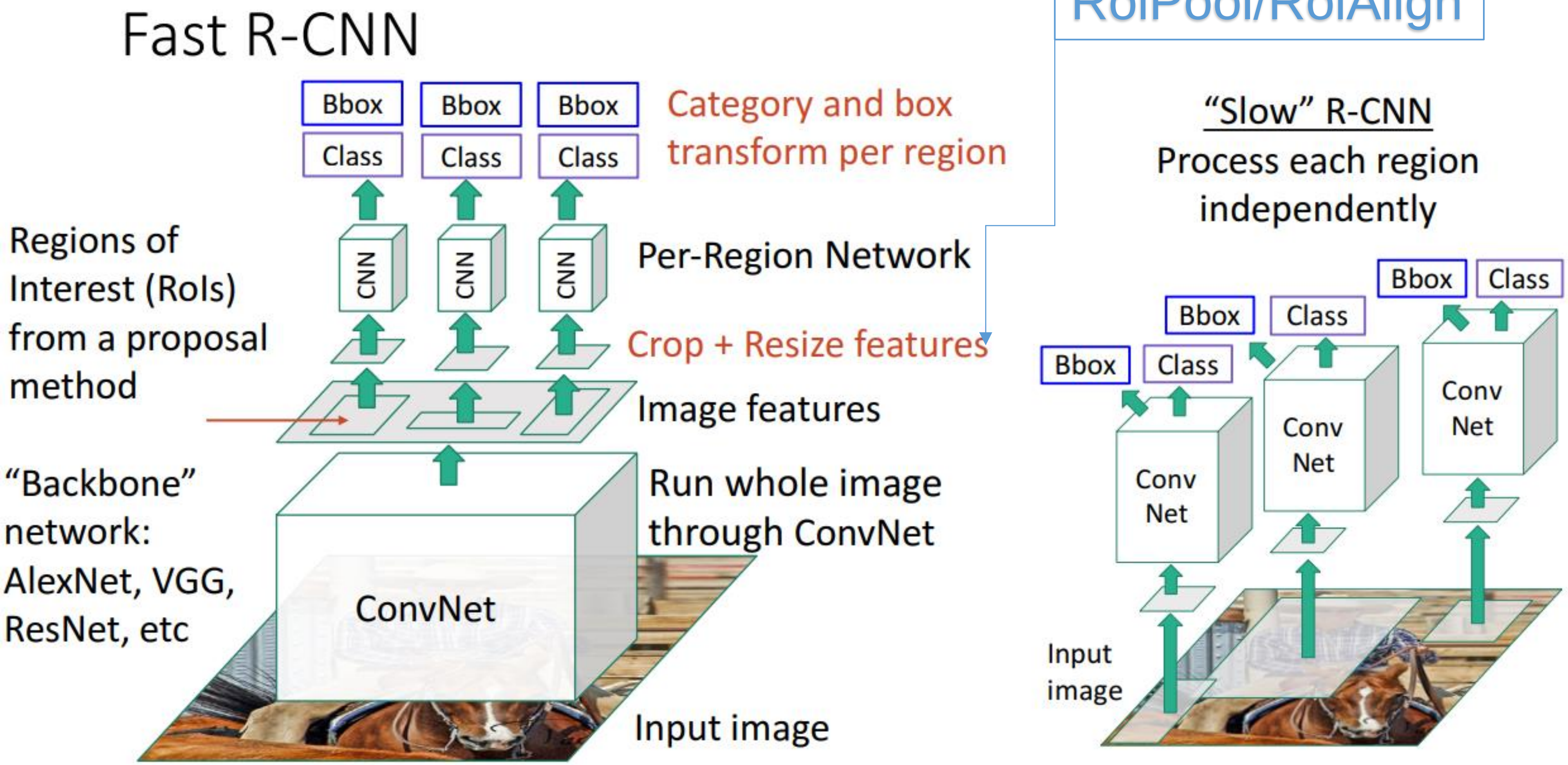
Bbox
Class
Bbox
Class
Bbox
Class
Conv Net
Conv Net
Problem: Very Slow
Forward each region through ConvNet
Warped image regions (224x224)
Input image
Regions of Interest (RoI) from a proposal method (~2k)

Fast R-CNN


Bbox
Bbox
Bbox
Class
Class
Class
Category and box transform per region
Per-Region network is relatively lightweight
CNN
CNN
CNN
Per-Region Network
Regions of Interest (RoIs) from a proposal method
Crop + Resize features
Image features
"Backbone" network: AlexNet, VGG, ResNet, etc
Run whole image through ConvNet
ConvNet
Most of the computation happens in backbone network; this saves work for overlapping region proposals
Input image

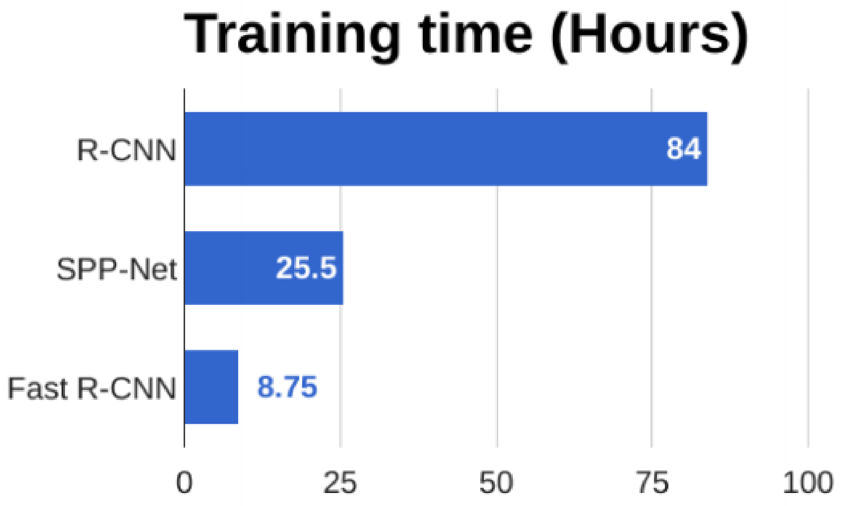
Fast R-CNN

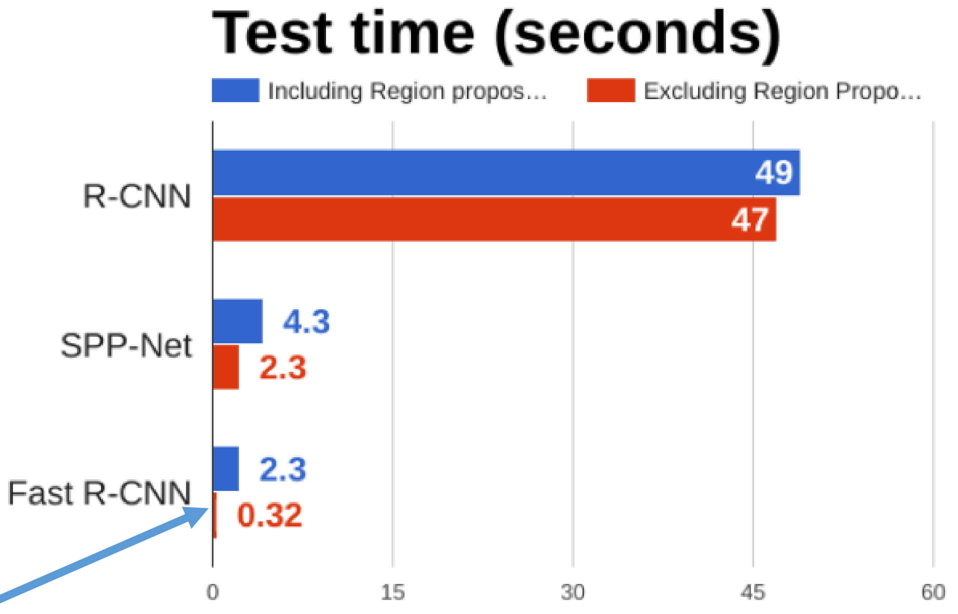
RoiPool/RoiAlign



Fast R-CNN
Bbox
Class
Category and box transform per region
CNN
Per-Region Network
Crop + Resize features
Image features
Regions of Interest (RoIs) from a proposal method
"Backbone" network: AlexNet, VGG, ResNet, etc
ConvNet
Run whole image through ConvNet
Input image
"Slow" R-CNN
Process each region independently
Conv Net
Input image

R-CNN vs Fast R-CNN





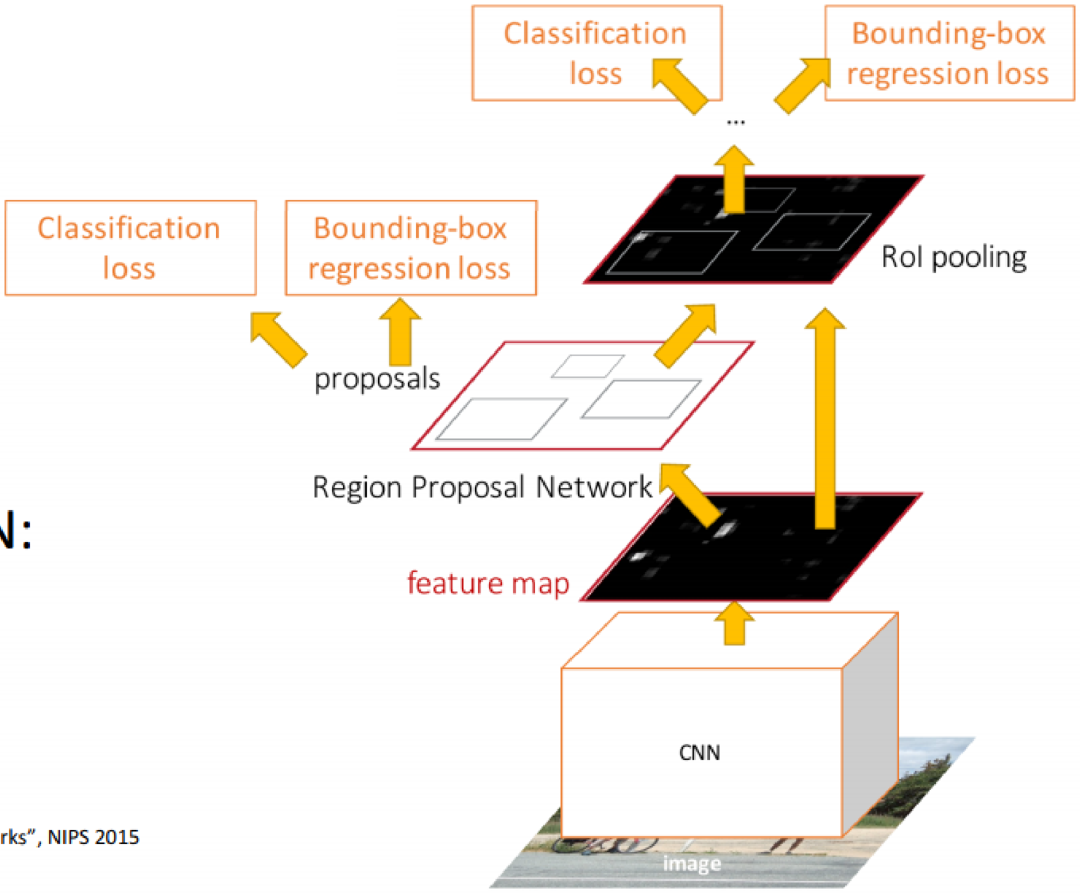
There is still space for improvement, particularly for region proposals

What if we try learning the regions with a CNN instead of Selective search?

Faster R-CNN: CNN for Region Proposals

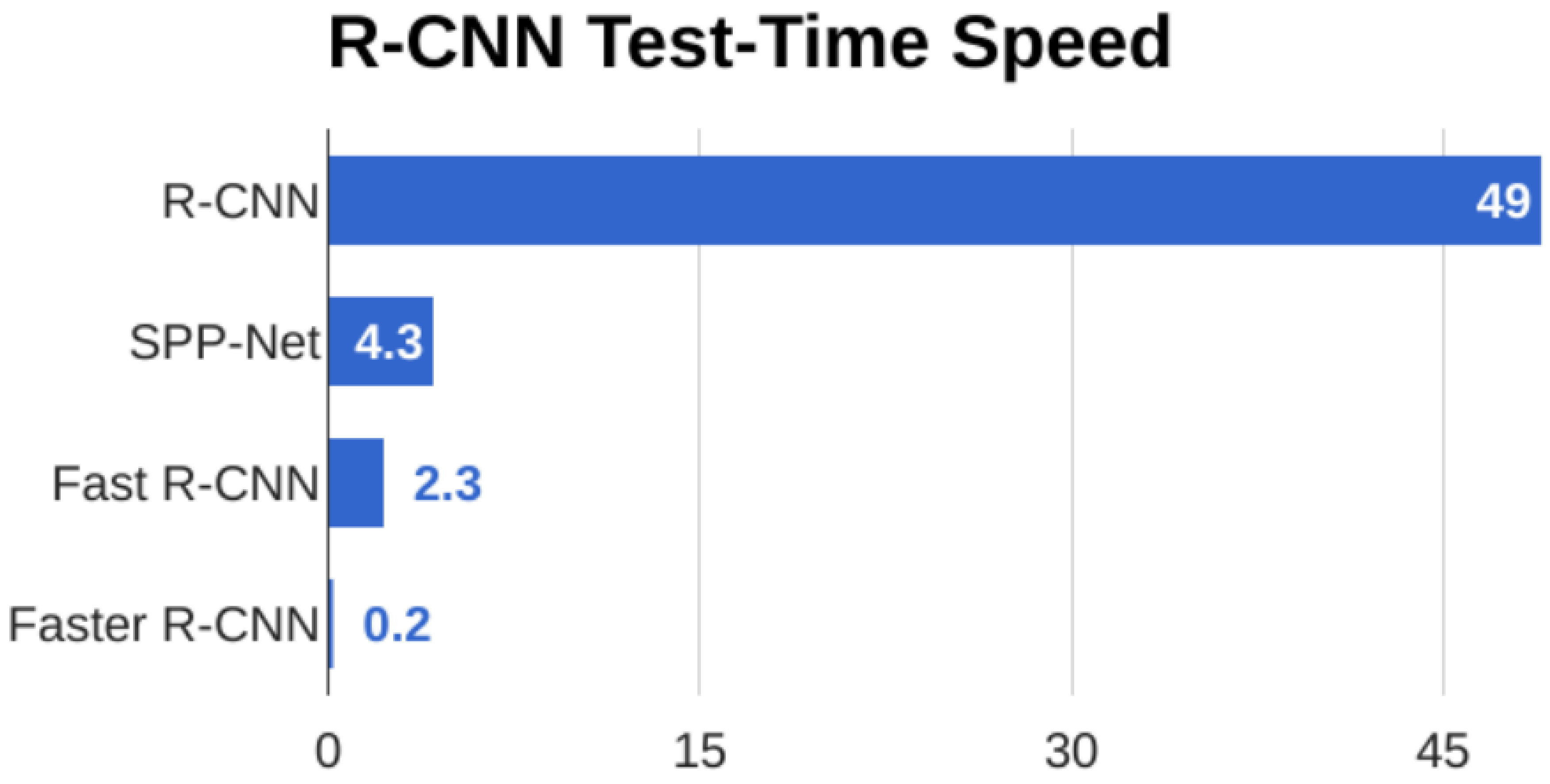
Insert **Region Proposal Network (RPN)** to predict proposals from features

Otherwise same as Fast R-CNN: Crop features for each proposal, classify each one



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

Faster R-CNN: CNN for Region Proposals

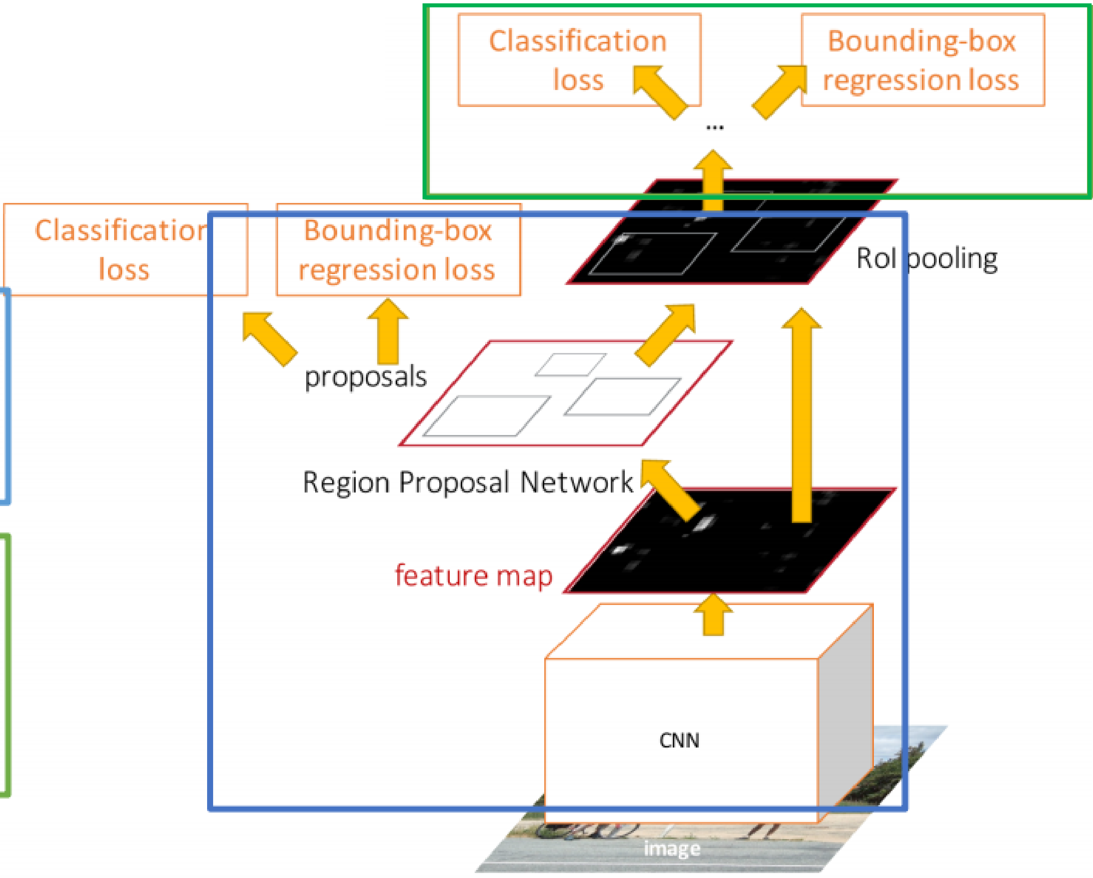


Faster R-CNN: CNN for Region Proposals

Faster R-CNN is a
Two-stage object detector

First stage: Run once per image
- Backbone network
- Region proposal network

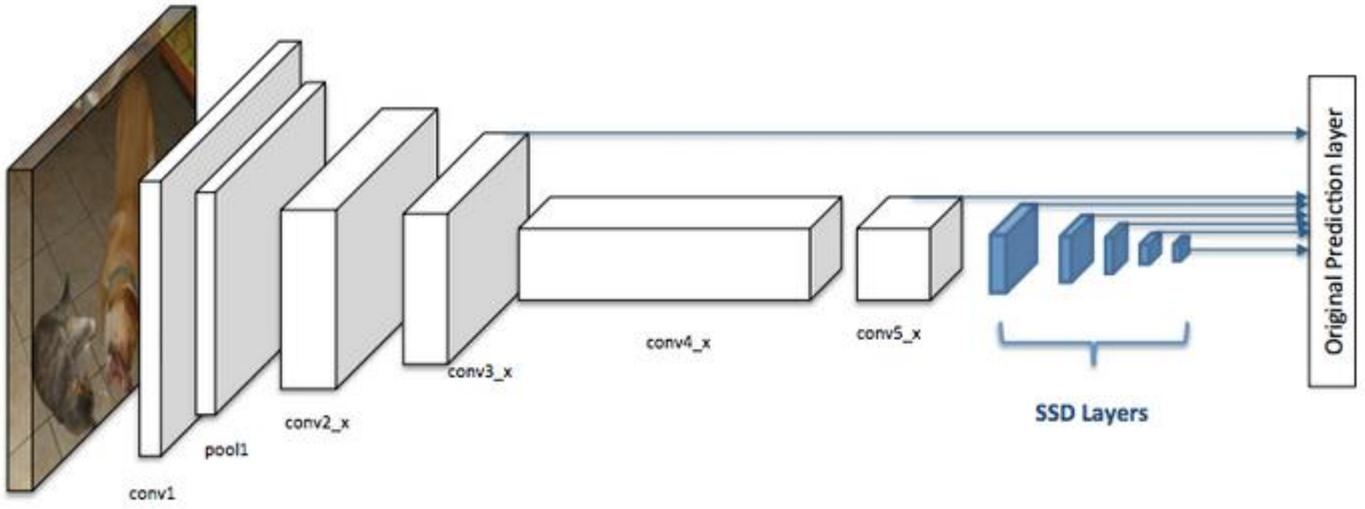
Second stage: Run once per region
- Crop features: RoI pool / align
- Predict object class
- Prediction bbox offset



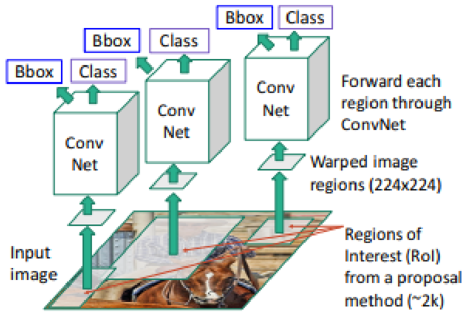
Question: Do we really need two stages?

Single Stage Object Detection

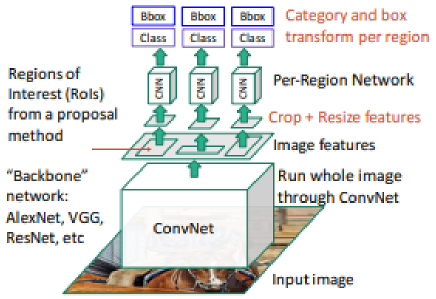
SSD has two components: a **backbone** model and **SSD head**.
Backbone model usually is a pre-trained image classification network as a feature extractor.
The ***SSD*** *head* is just one or more convolutional layers added to this backbone and the outputs are interpreted as the bounding boxes and classes of objects in the spatial location of the final layers' activations.
In the figure below, the first few layers (white boxes) are the backbone, the last few layers (blue boxes) represent the SSD head.



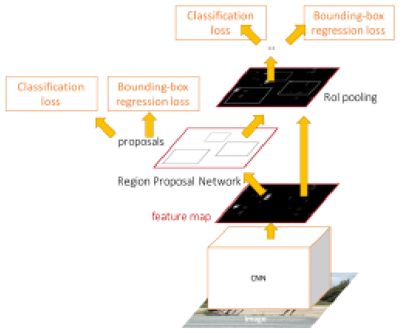
“Slow” R-CNN: Run CNN independently for each region


Bbox
Class
Bbox
Class
Bbox
Class
Conv Net
Conv Net
Conv Net
Forward each region through ConvNet
Warped image regions (224x224)
Regions of Interest (RoI) from a proposal method (~2k)
Input image

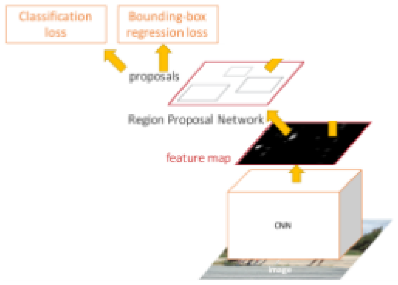
Fast R-CNN: Apply differentiable cropping to shared image features


Bbox
Class
Bbox
Class
Bbox
Class
Category and box transform per region
CNN
CNN
CNN
Per-Region Network
Regions of Interest (RoIs) from a proposal method
Crop + Resize features
Image features
“Backbone” network: AlexNet, VGG, ResNet, etc
ConvNet
Run whole image through ConvNet
Input image

Faster R-CNN: Compute proposals with CNN



Single-Stage: Fully convolutional detector



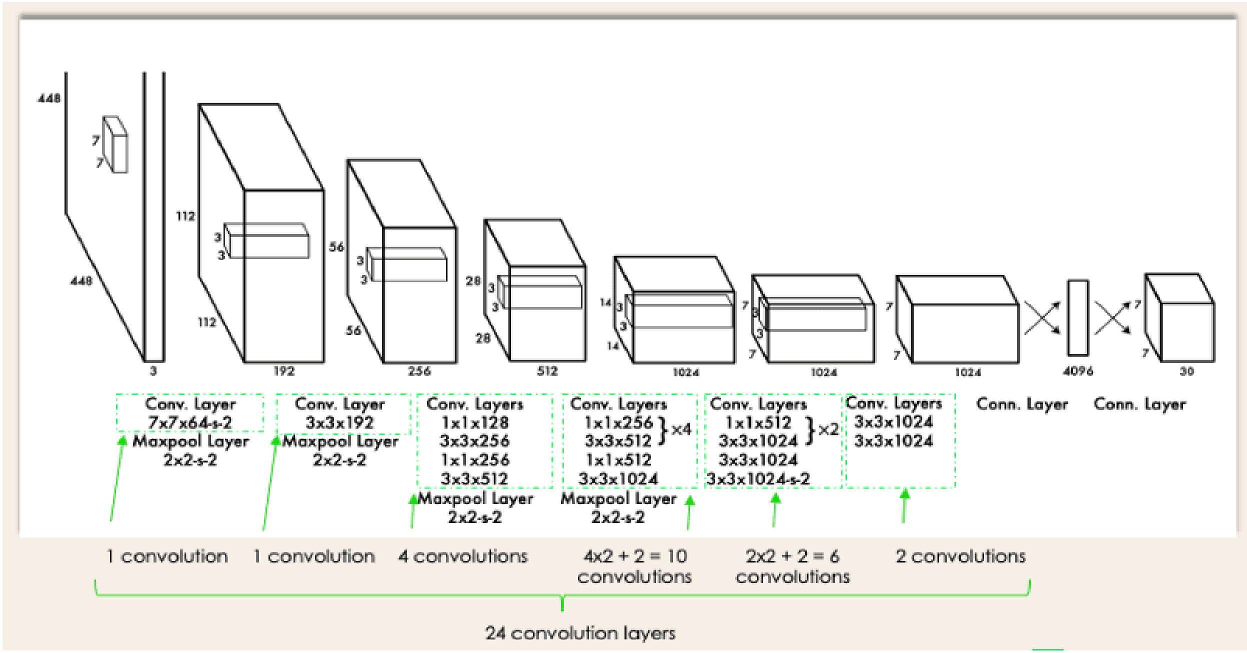
SCHOOL OF
HUMAN SCIENCES
& TECHNOLOGY

You Only Look Once

YOLO architecture

YOLO As illustrated below, has overall 24 convolutional layers, four max-pooling layers, and two fully connected layers.
The architecture works as follows:
•Resizes the input image into 448x448 before going through the convolutional network.
•A 1x1 convolution is first applied to reduce the number of channels, which is then followed by a 3x3 convolution to generate a cuboidal output.
•The activation function under the hood is ReLU, except for the final layer, which uses a linear activation function.

- Some additional techniques, such as batch normalization and dropout, respectively regularize the model and prevent it from overfitting.

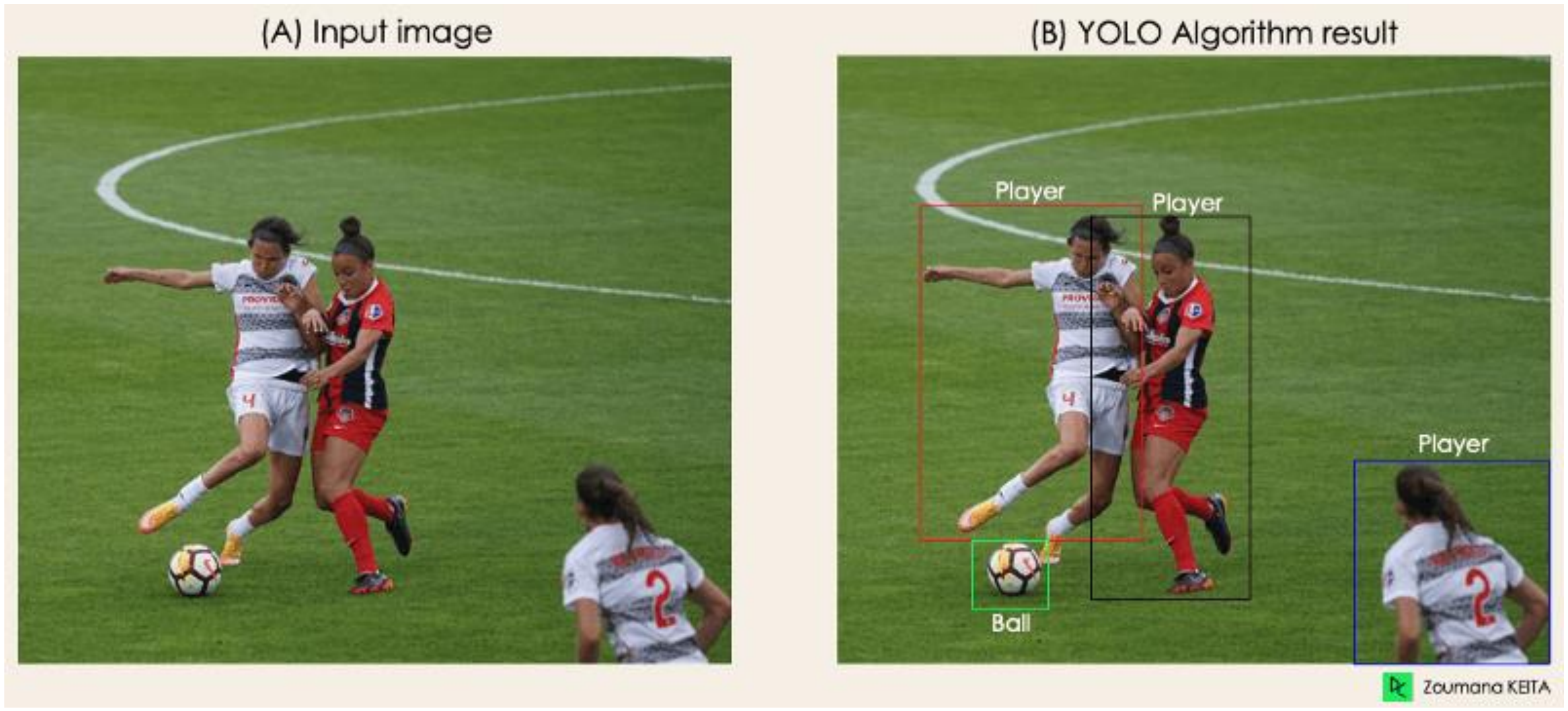


448
7
7
448
3
112
3
3
112
192
56
3
3
56
256
28
3
3
28
512
14
3
3
14
1024
7
3
3
7
1024
7
7
1024
4096
7
7
30
Conv. Layer 7x7x64-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x192 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x128 3x3x256 1x1x256 3x3x512 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x256 3x3x512 }×4 1x1x512 3x3x1024 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x512 3x3x1024 }×2 3x3x1024 3x3x1024-s-2
Conv. Layers 3x3x1024 3x3x1024
Conn. Layer
Conn. Layer
1 convolution
1 convolution
4 convolutions
4x2 + 2 = 10 convolutions
2x2 + 2 = 6 convolutions
2 convolutions
24 convolution layers

SCHOOL OF
HUMAN SCIENCES
& TECHNOLOGY

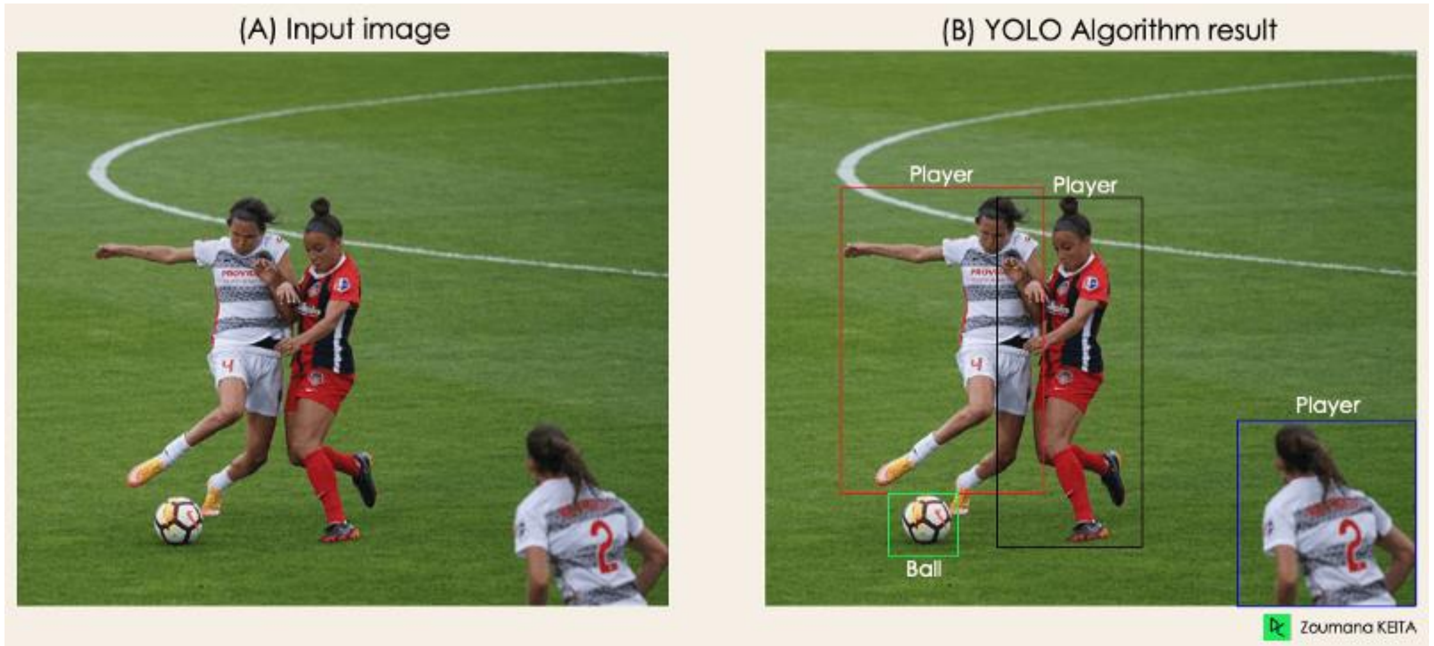
How Does YOLO Object Detection Work?

Imagine you built a YOLO application that detects players and soccer balls from a given image.



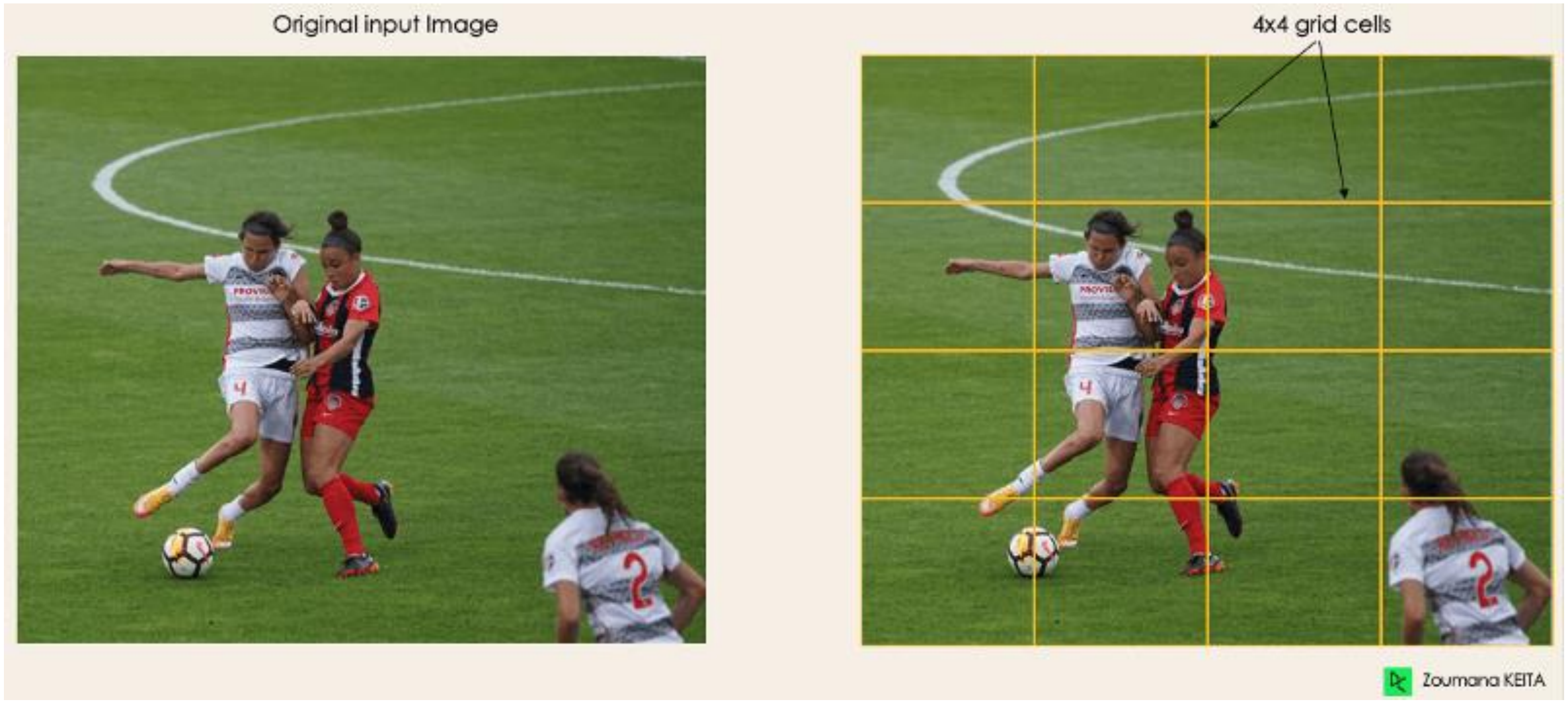
How Does YOLO Object Detection Work?

The algorithm works based on the following four approaches:
•Residual blocks
•Bounding box regression
•Intersection Over Unions or IOU for short
•Non-Maximum Suppression.



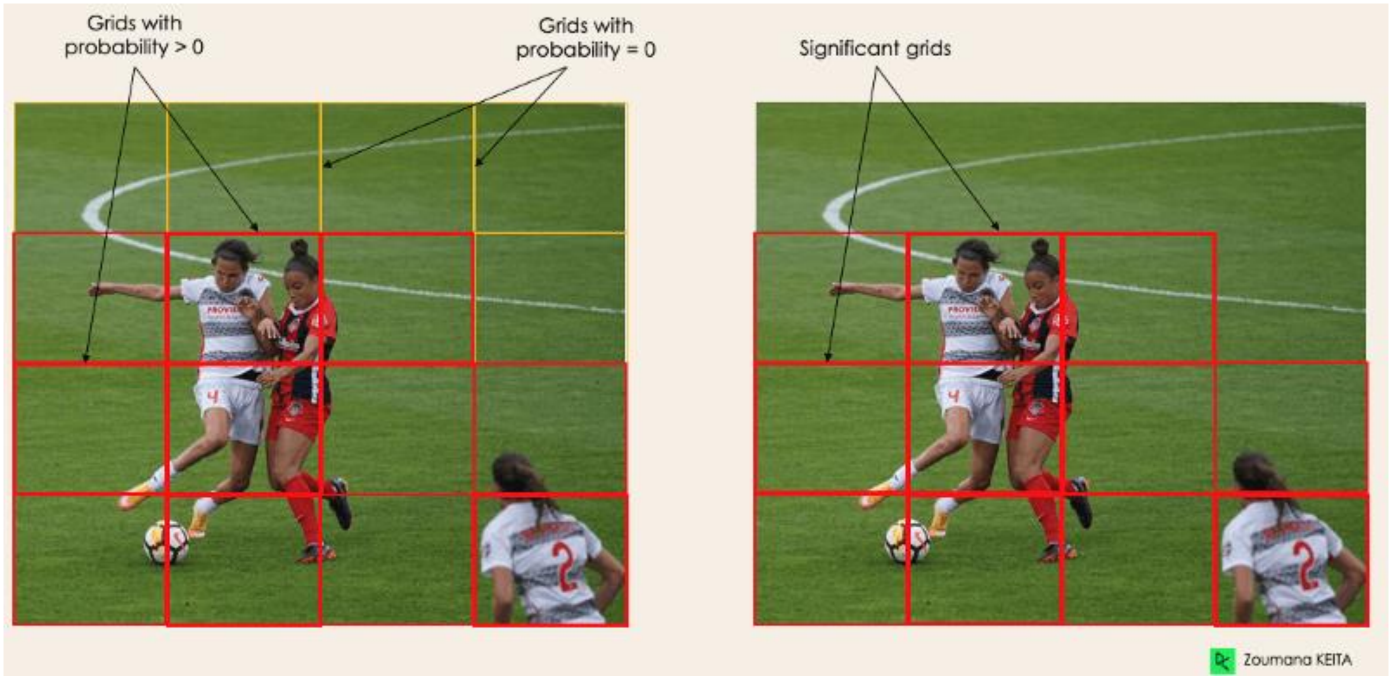
YOLO - Residual blocks

This first step starts by dividing the original image into NxN grid cells of equal shape, where N in our case is 4 shown on the image on the right. Each cell in the grid is responsible for localizing and predicting the class of the object that it covers, along with the probability/confidence value.



YOLO - Bounding box regression

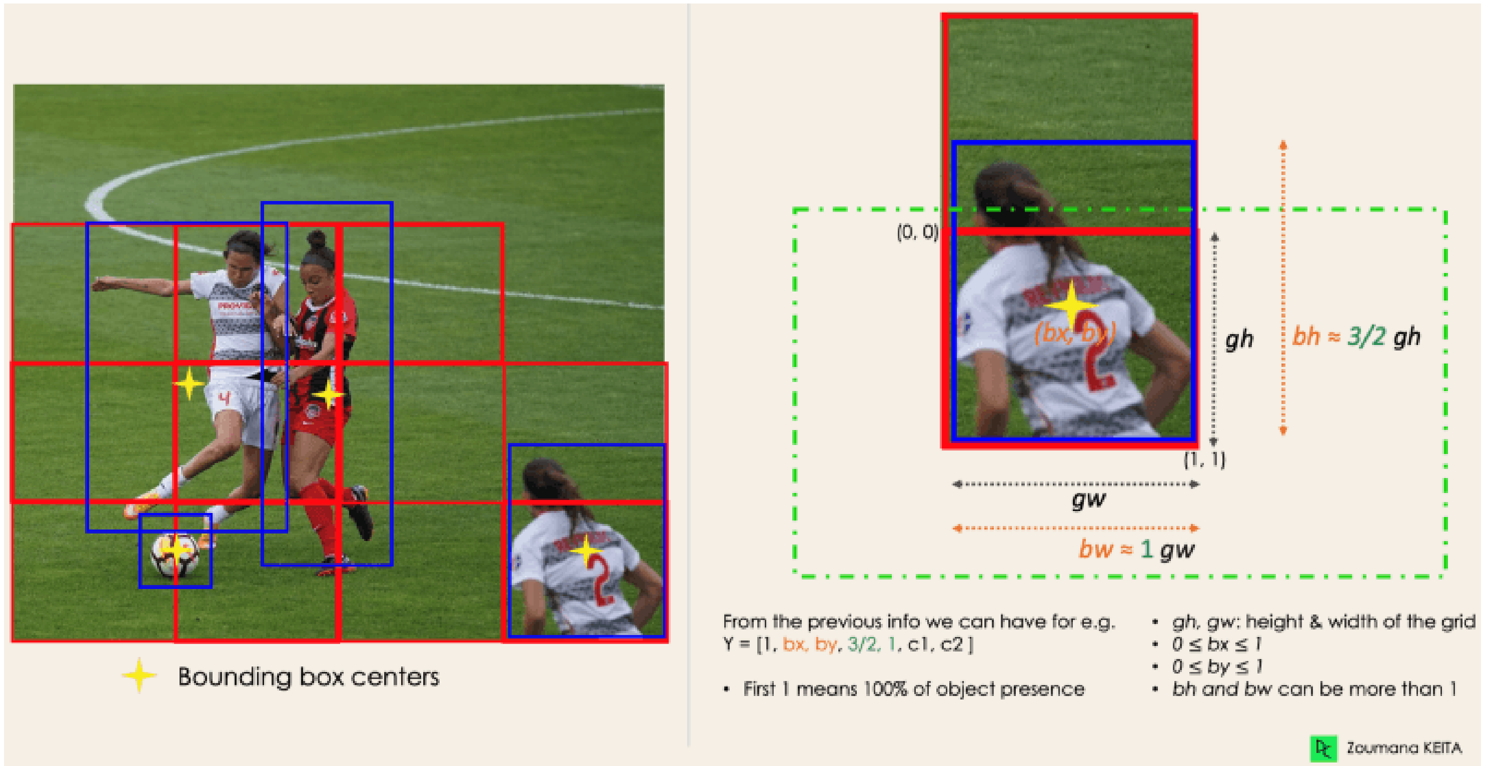
Now we determine the bounding boxes which correspond to rectangles highlighting all the objects in the image.
YOLO determines the attributes of these bounding boxes using a single regression module in the following format, where Y is the final vector representation for each bounding box.
Y = [pc, bx, by, bh, bw, c1, c2]
This is especially important during the training phase of the model.
•pc corresponds to the probability score of the grid containing an object.



SCHOOL OF HUMAN SCIENCES & TECHNOLOGY

YOLO - Bounding box regression

•bx, by are the x and y coordinates of the center of the bounding box with respect to the enveloping grid cell.
•bh, bw correspond to the height and the width of the bounding box with respect to the enveloping grid cell.
•c1 and c2 correspond to the two classes Player and Ball. We can have as many classes as your use case requires.
To understand, let's pay closer attention to the player on the bottom right.



From the previous info we can have for e.g.
Y = [1, bx, by, 3/2, 1, c1, c2]

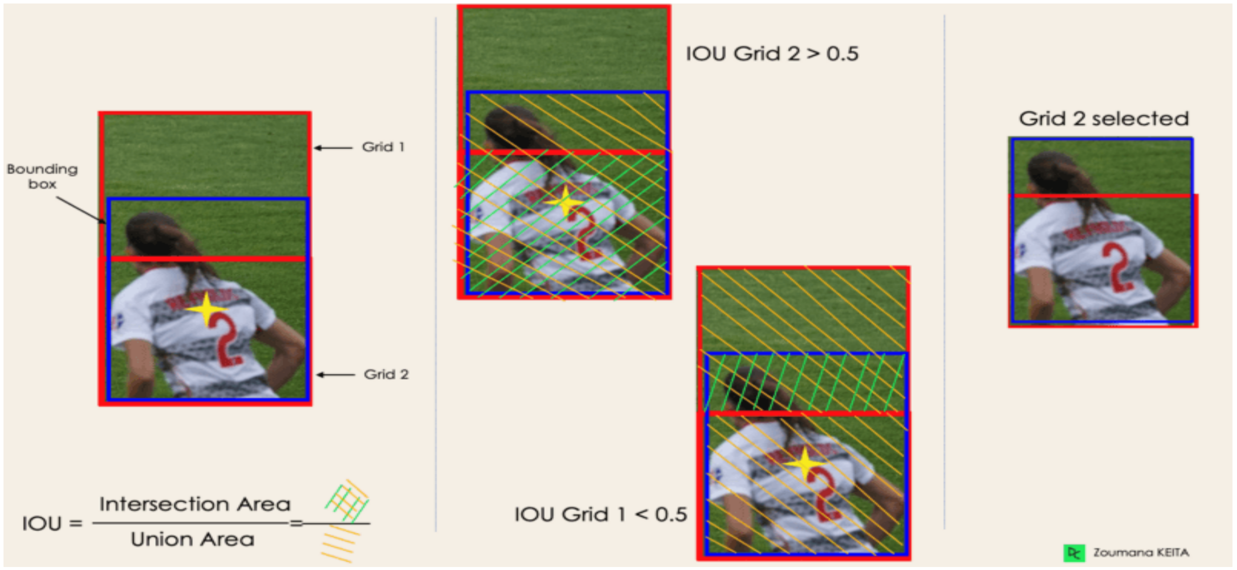
- First 1 means 100% of object presence

- gh, gw: height & width of the grid
- $0 \leq bx \leq 1$
- $0 \leq by \leq 1$
- bh and bw can be more than 1

SCHOOL OF HUMAN SCIENCES & TECHNOLOGY

YOLO - Intersection Over Unions or IOU

•The user defines its IOU selection threshold, which can be, for instance, 0.5.
•Then YOLO computes the IOU of each grid cell which is the Intersection area divided by the Union Area.
•Finally, it ignores the prediction of the grid cells having an IOU ≤ threshold and considers those with an IOU > threshold.
Below is an illustration of applying the grid selection process to the bottom left object. We can observe that the object originally had two grid candidates, then only “Grid 2” was selected at the end.



Last step of YOLO is where we can use NMS to keep only the boxes with the highest probability score of detection.

SCHOOL OF
HUMAN SCIENCES
& TECHNOLOGY

Beyond YOLO
Zero Shot Object Detection

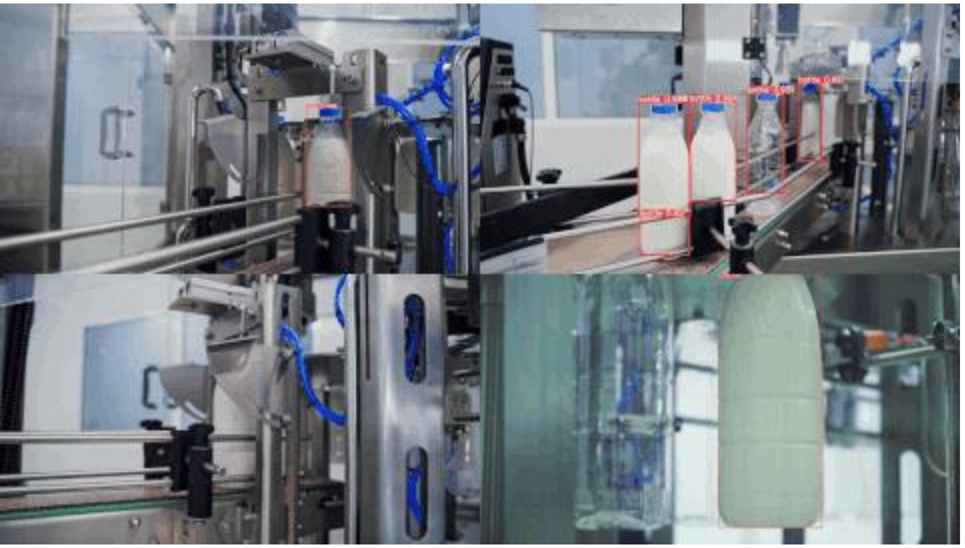
What is Zero-Shot Object Detection?

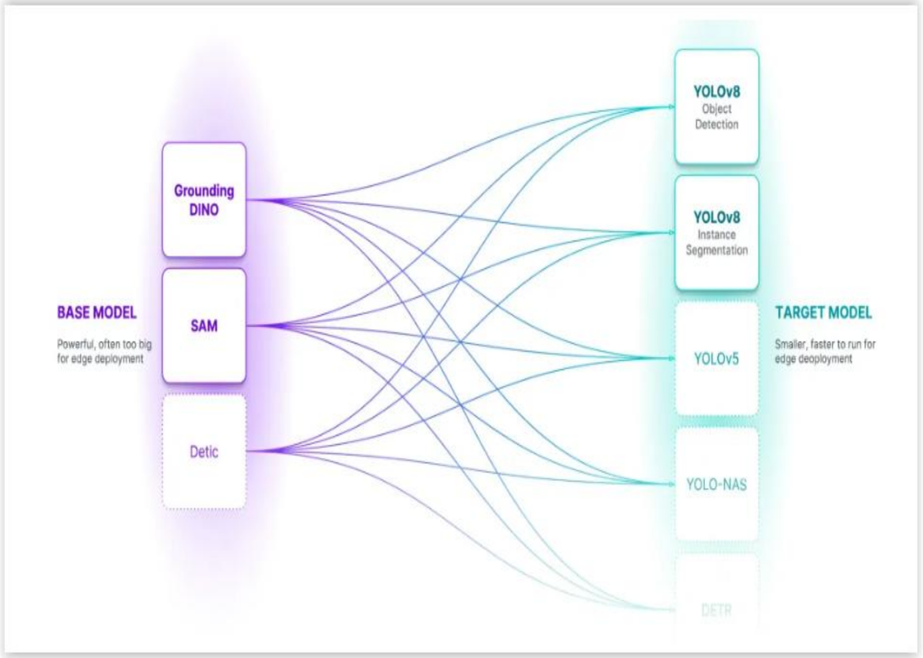
- Zero-shot object detection models identify the location of objects in images.
- Zero-shot object detection models accept one or more text prompts (i.e. "car") and will aim to identify the location of all the objects of interest you have listed.
- Zero-shot object detection models are designed to identify a wide range of images, examples of which include **Grounding DINO**, **OWL-ViT**, and **OWLv2**.

- Although Grounding DINO and other zero-shot models are powerful, they are large and require significant compute resources to run compared to fine-tuned models. As a result, many state-of-the-art zero-shot models are impractical to run at large scale, in real-time, or on the edge.
- Zero-shot models can be run across images for analysis, or you can use zero-shot models to automatically label data for use in training a smaller, fine-tuned model.
- The Autodistill framework enables you to do this in a few lines of code. You can use a model like Grounding DINO or OWLv2 to label data, then train a smaller model like **YOLOv8** on your labeled dataset. The resultant model is suitable for real-time edge deployment, unlike large zero-shot models.

Annotation-free Object detection

To use autodistill, you input unlabeled data into a Base Model which uses an Ontology to label a Dataset that is used to train a Target Model which outputs a Distilled Model fine-tuned to perform a specific Task.




BASE MODEL
Powerful, often too big for edge deployment
Grounding DINO
SAM
Detic
YOLOv8 Object Detection
YOLOv8 Instance Segmentation
YOLOv5
YOLO-NAS
DETR
TARGET MODEL
Smaller, faster to run for edge deoployment

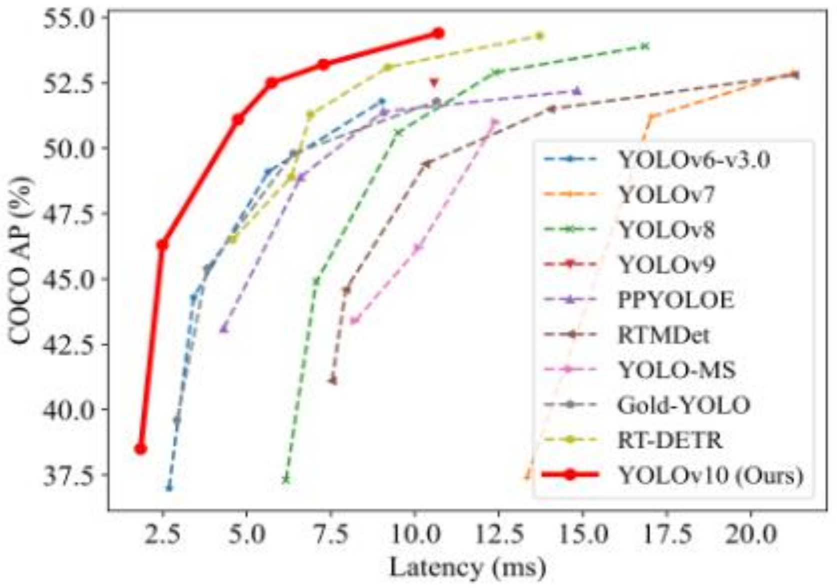
YOLO v10

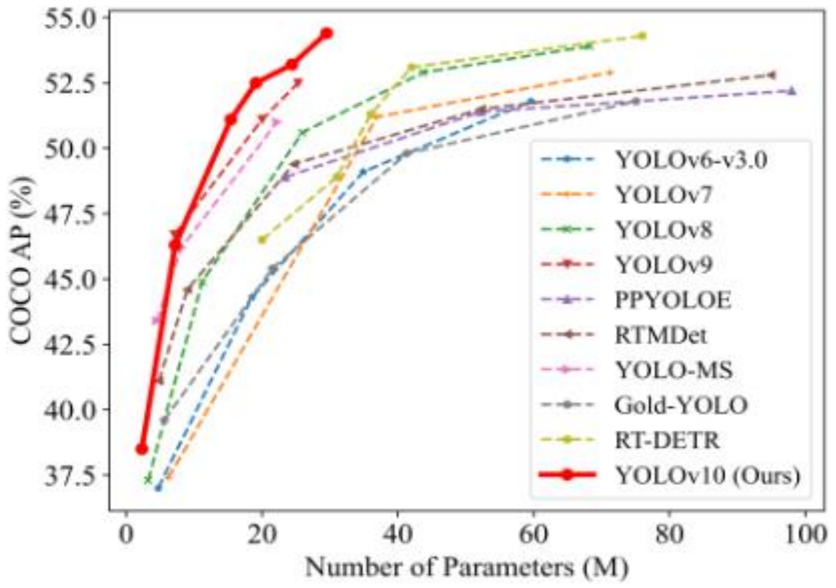
Key Features

1.**NMS-Free Training**: Utilizes consistent dual assignments to eliminate the need for NMS, reducing inference latency.

2.**Holistic Model Design**: Comprehensive optimization of various components from both efficiency and accuracy perspectives, including lightweight classification heads, spatial-channel decoupled down sampling, and rank-guided block design.

3.**Enhanced Model Capabilities**: Incorporates large-kernel convolutions and partial self-attention modules to improve performance without significant computational cost.





Annotation formats

COCO:

```
annotation{

"id" : int,

"image_id": int,

"category_id": int,

"segmentation": RLE or [polygon],

"area": float,

"bbox": [x,y,width,height],

"iscrowd": 0 or 1,

}

categories[{

"id": int,

"name": str,

"supercategory": str,

}]
```

Pascal VOC:

```
<annotation>
  <folder>Train</folder>
  <filename>01.png</filename>
  <path>/path/Train/01.png</path>
  <source>
    <database>Unknown</database>
  </source>
  <size>
    <width>224</width>
    <height>224</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>36</name>
    <pose>Frontal</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <occluded>0</occluded>
    <bndbox>
      <xmin>90</xmin>
      <xmax>190</xmax>
      <ymin>54</ymin>
      <ymax>70</ymax>
    </bndbox>
  </object>
</annotation>
```

YOLO:

```
<object-class> <x> <y> <width> <height>
```

```
0 45 55 29 67
1 99 83 28 44
```

Chapter Summary

- We saw different Object detection CNN architectures
- We reviewed SSD advantages compared to older object detection CNNs
- We visited the Different improvements to YOLO
- We gor familiar with Zero shot object detection
- We understood how to do annotation-free object detection