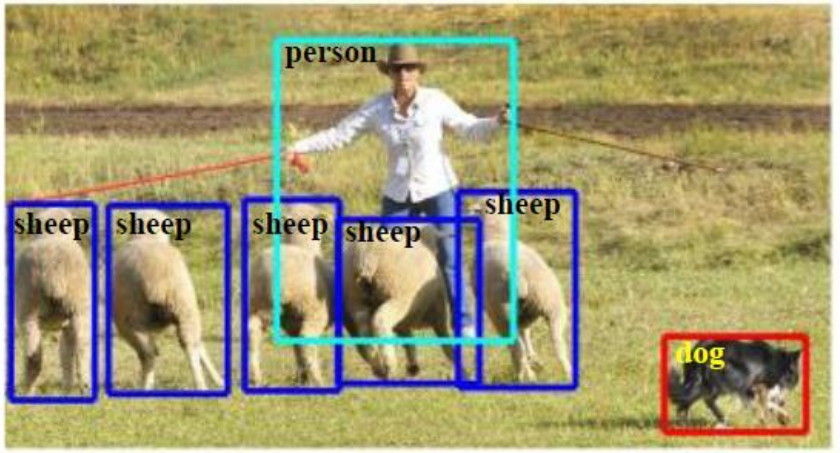
Object Detection

Outline

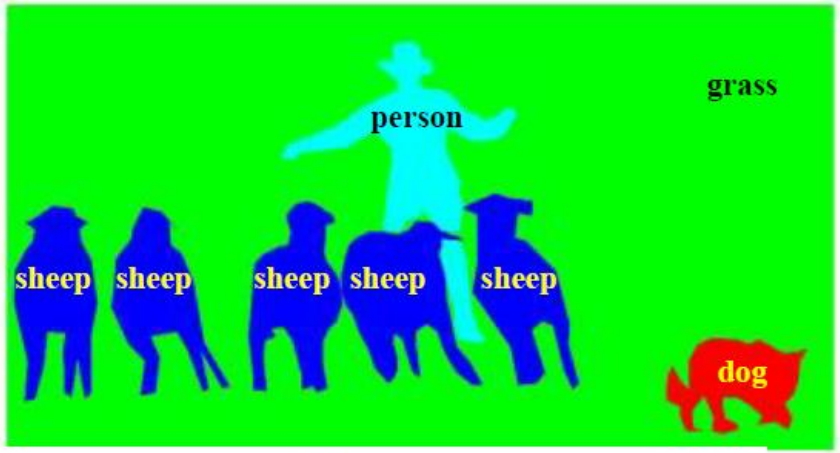
- **Introduction**
- Traditional Computer Vision Techniques:
  - Sliding Window:
  - Region proposal approach **(Selective Search)**
- Deep Learning-Based Techniques:
  - Two-Stage Detectors:
    - R-CNN (Regions with CNN features)
  - One-Stage Detectors:
    - YOLO (You Only Look Once)
- Other Detectors
- Evaluation Measures
- Benchmarks

Common CV Applications

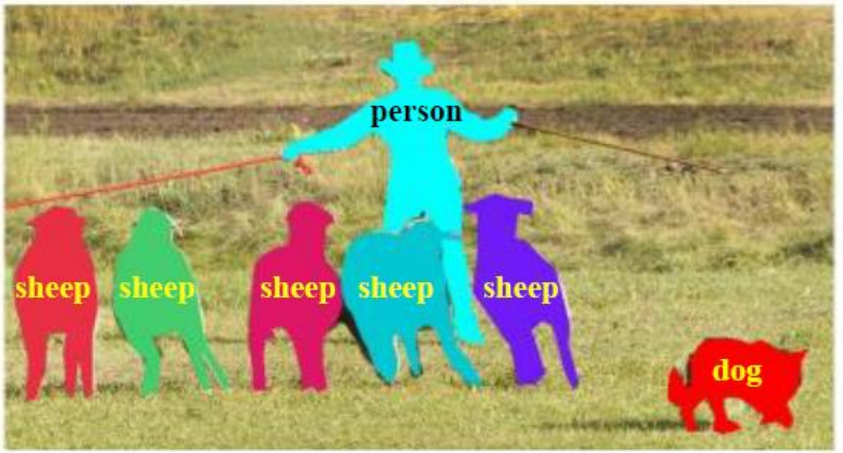
(a) Object Classification



(b) Generic Object Detection (Bounding Box)



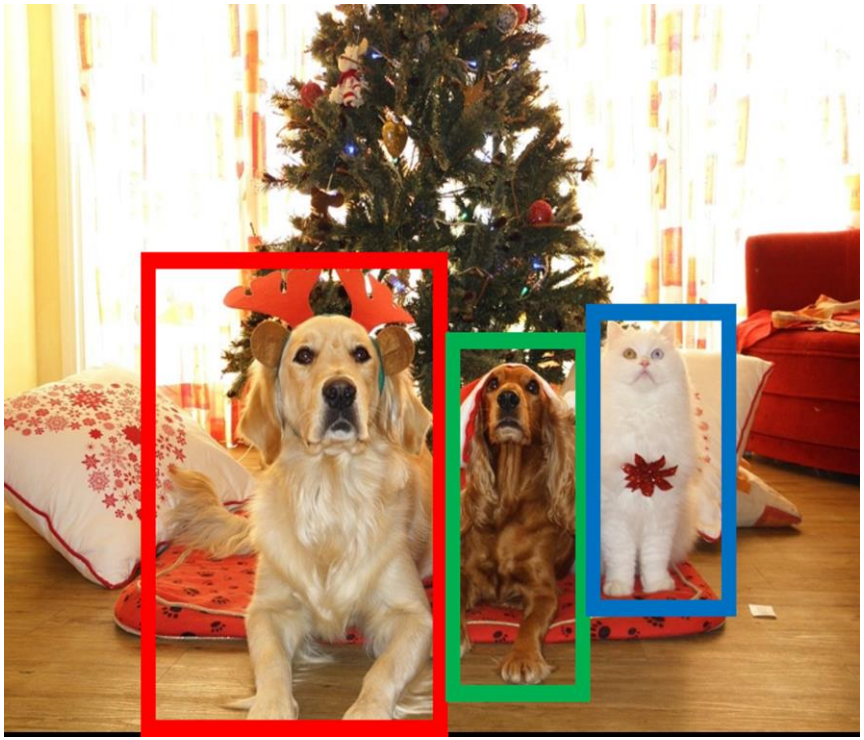
(c) Semantic Segmentation



(d) Object Instance Segmetation

Object Detection: Task Definition

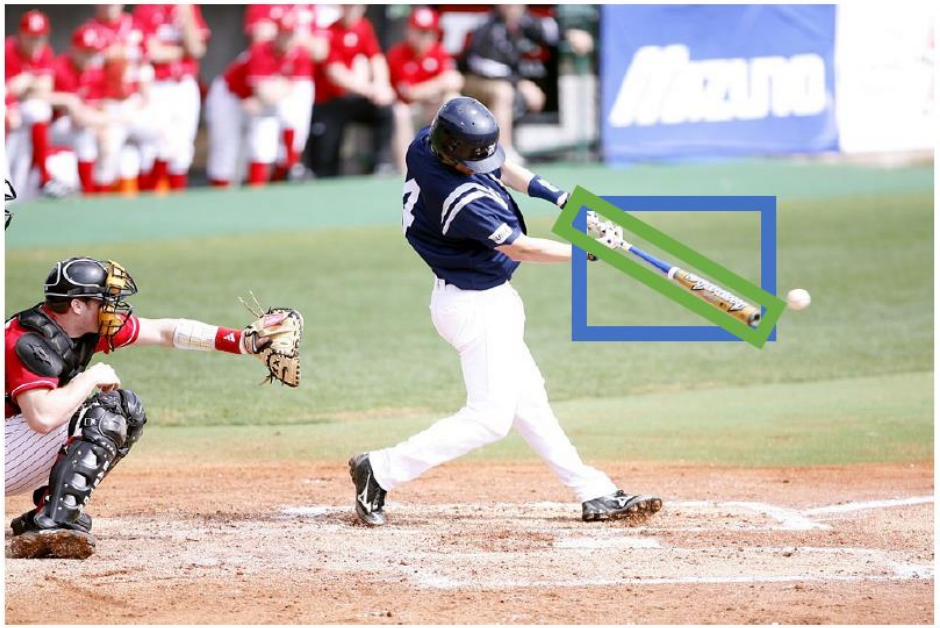
- **Problem**: Detecting and localizing generic objects from various categories, such as cars, people, etc.
  - **Input**: Single RGB Image
  - **Output**: A set of detected objects; For each object predict:
    1. Category label (from fixed, known set of categories)
    2. Bounding box (four numbers: x, y, width, height)
- **Challenges:**
  - Illumination, Viewpoint, deformations, Intra-class variability, …
  - **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, often ~800x600



Object Detection: Bounding Boxes

- Bounding boxes (usually) cover only the visible portion of the object

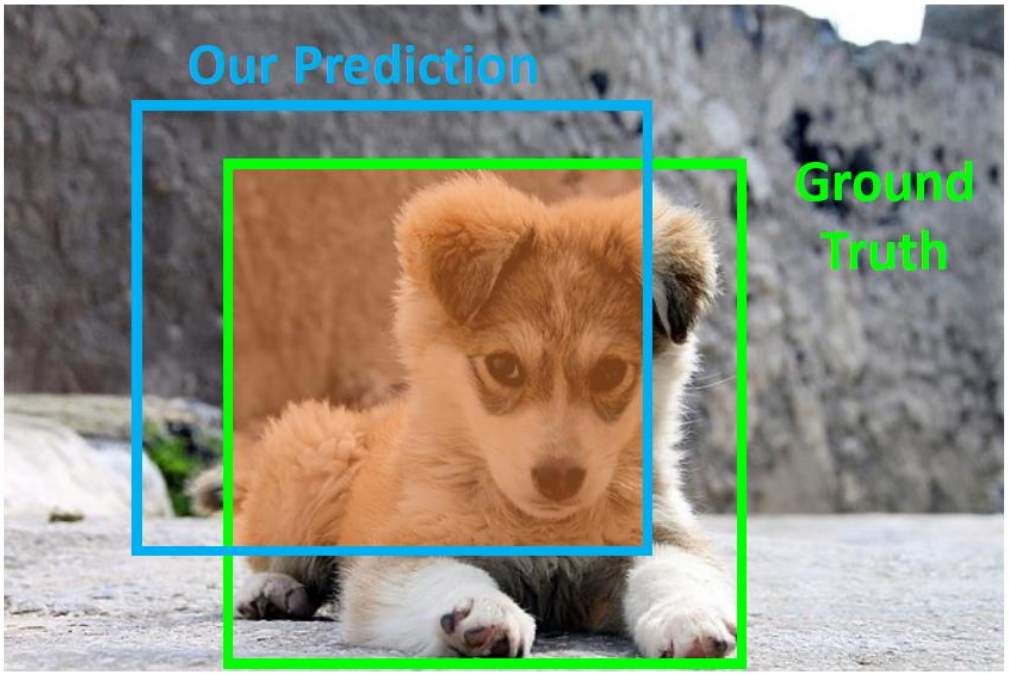
- Bounding boxes are typically *axis-aligned*
- *Oriented* boxes are much less common



Comparing Boxes: Intersection over Union (IoU)

- How can we compare our prediction to the ground-truth box?
- **Intersection over Union** (IoU) (Also called “Jaccard similarity” or “Jaccard index”):

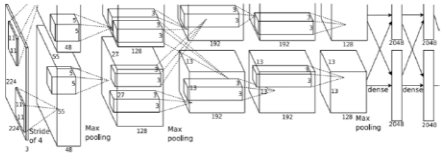
$$\frac{\textit{Area of Intersection}}{\textit{Area of Union}}$$



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

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
...

"What"

Correct label:
Cat

Softmax Loss

Fully Connected: 4096 to 4

Box Coordinates
(x, y, w, h)

"Where"

L2 Loss

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

Multitask Loss

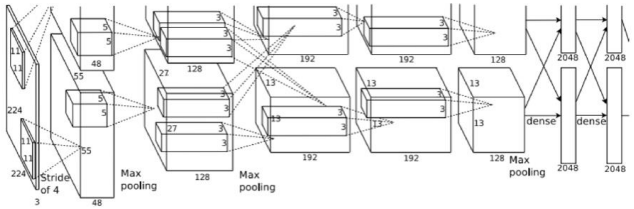
Weighted Sum → **Loss**

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

Problem: Images can have more than one object!

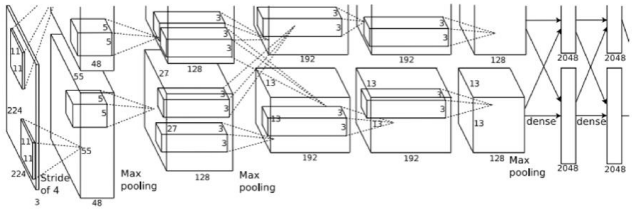
Detecting Multiple Objects

Need different numbers of outputs per image



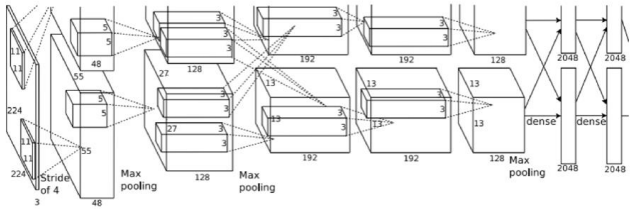
CAT: (x, y, w, h)

4 numbers



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

12 numbers



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

Many numbers!

Outline

- Introduction
- Traditional Computer Vision Techniques:
  - **Sliding Window**
  - Region proposal approach **(Selective Search)**
- Deep Learning-Based Techniques:
  - Two-Stage Detectors:
    - R-CNN (Regions with CNN features)
  - One-Stage Detectors:
    - YOLO (You Only Look Once)
- Other Detectors
- Evaluation Measures
- Benchmarks

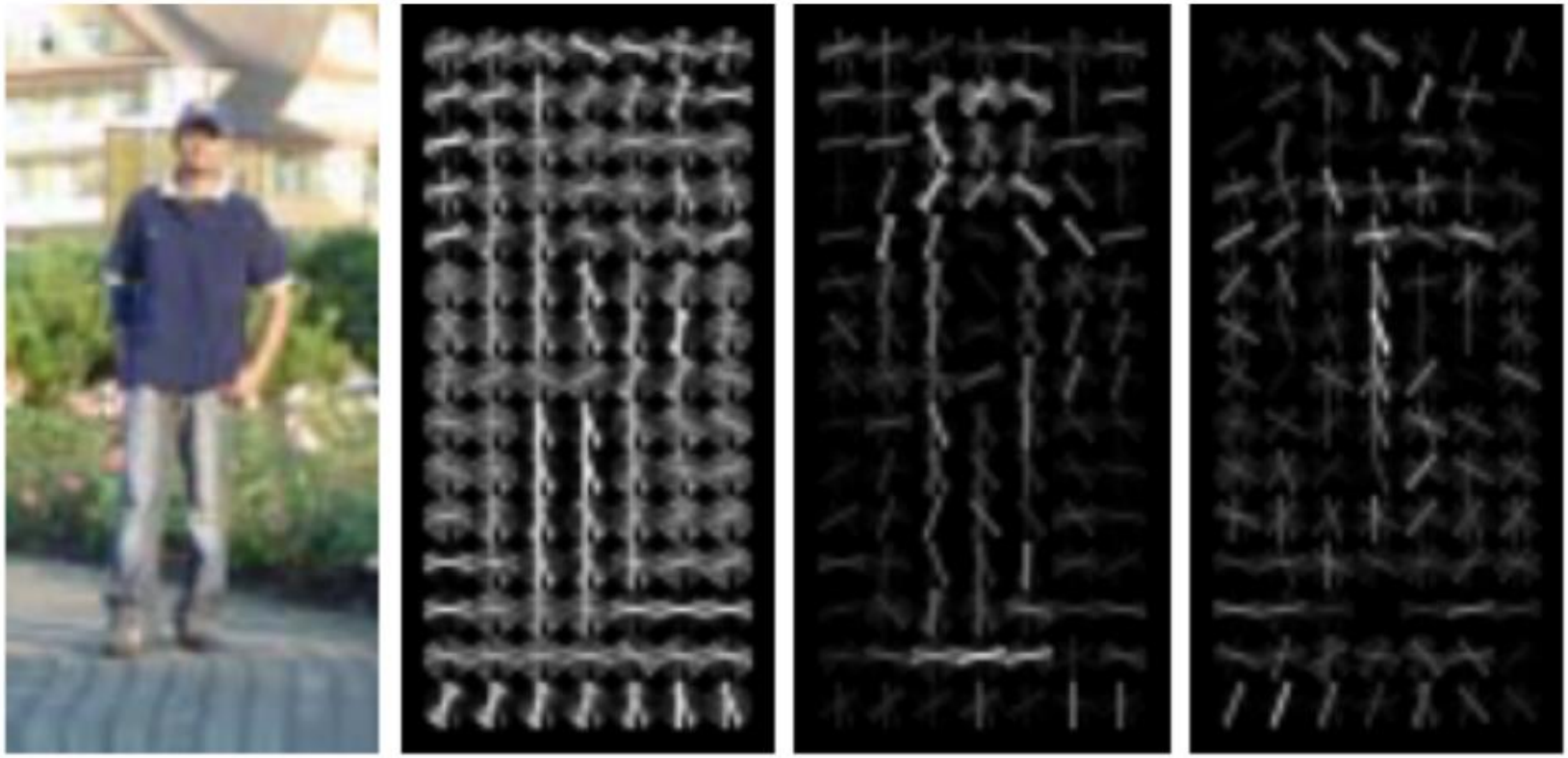
Sliding Window

- In the sliding window approach, we slide a box or window over an image to select a patch and classify each image patch covered by the window using the object recognition model.
- Not only do we need to search all possible locations in the image, we have to search at different scales.
- The problem doesn't end here. Sliding window approach is good for fixed aspect ratio objects such as faces or pedestrians.
- Number of windows is large and grows quadratically with the number of pixels, and the need to search over multiple scales and aspect ratios further increases the search space.
- The huge search space results in high computational complexity

Sliding Window with HOG method

sliding window

Recap – HOG features



Find a HOG template and use as filter

Sliding window + HOG features

- Slide through the image and check if there is an object at every location

No person here

Sliding window + hog features

- Slide through the image and check if there is an object at every location

YES!! Person match found

Sliding window + hog features

- But what if we were looking for buses?

No bus found

Sliding window + hog features

- But what if we were looking for buses?

No bus found

Sliding window + hog features

No bus found

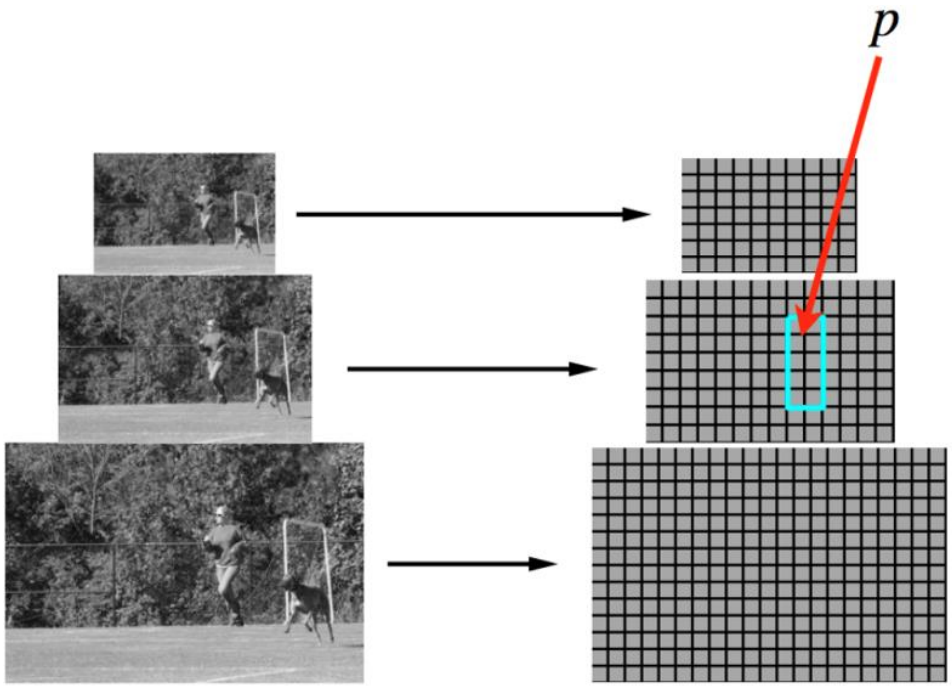
- We will never find the object we don't choose our window size wisely!

Sliding window + hog features

We need to do multi scale sliding window

Sliding window + HoG with feature pyramid

- Sliding window technique is applied as usual over all the pyramid levels.
- The window that produces the highest similarity score out of the resizing's is used as the location of the detected object



Score of F at position p is
$F \cdot \phi(p, H)$

$\phi(p, H)$ = concatenation of HOG features from subwindow specified by p

A feature pyramid of different image resizing

Detecting Multiple Objects: Sliding Window

- **Question**: How many possible boxes are there in an image of size H x W?
  - Consider a box of size h x w:
  - Possible x positions: W – w + 1
  - Possible y positions: H – h + 1
  - Possible positions: (W – w + 1) * (H – h + 1)
  - Total possible boxes =

$$\sum_{h=1}^{H}\sum_{w=1}^{W}(W-w+1)(H-h+1)$$

$$=\frac{H(H+1)}{2}\frac{W(W+1)}{2}$$

800 x 600 image has ~58M boxes! No way we can evaluate them all

Outline

- Introduction
- Traditional Computer Vision Techniques:
  - Sliding Window:
  - **Region proposal approach (Selective Search)**
- Deep Learning-Based Techniques:
  - Two-Stage Detectors:
    - R-CNN (Regions with CNN features)
  - One-Stage Detectors:
    - YOLO (You Only Look Once)
- Other Detectors
- Evaluation Measures
- Benchmarks

Region proposal approach

- Takes an image as the input and output bounding boxes corresponding to all patches in an image that are most likely to be objects.
- These region proposals can be noisy, overlapping and may not contain the object perfectly but amongst these region proposals, there will be a proposal which will be very close to the actual object in the image.
- We can then classify these proposals using the object recognition model. The region proposals with the high probability scores are locations of the object.
- An important property of a region proposal method is to have a very **high recall**. This is just a fancy way of saying that the regions that contain the objects we are looking have to be in our list of region proposals.

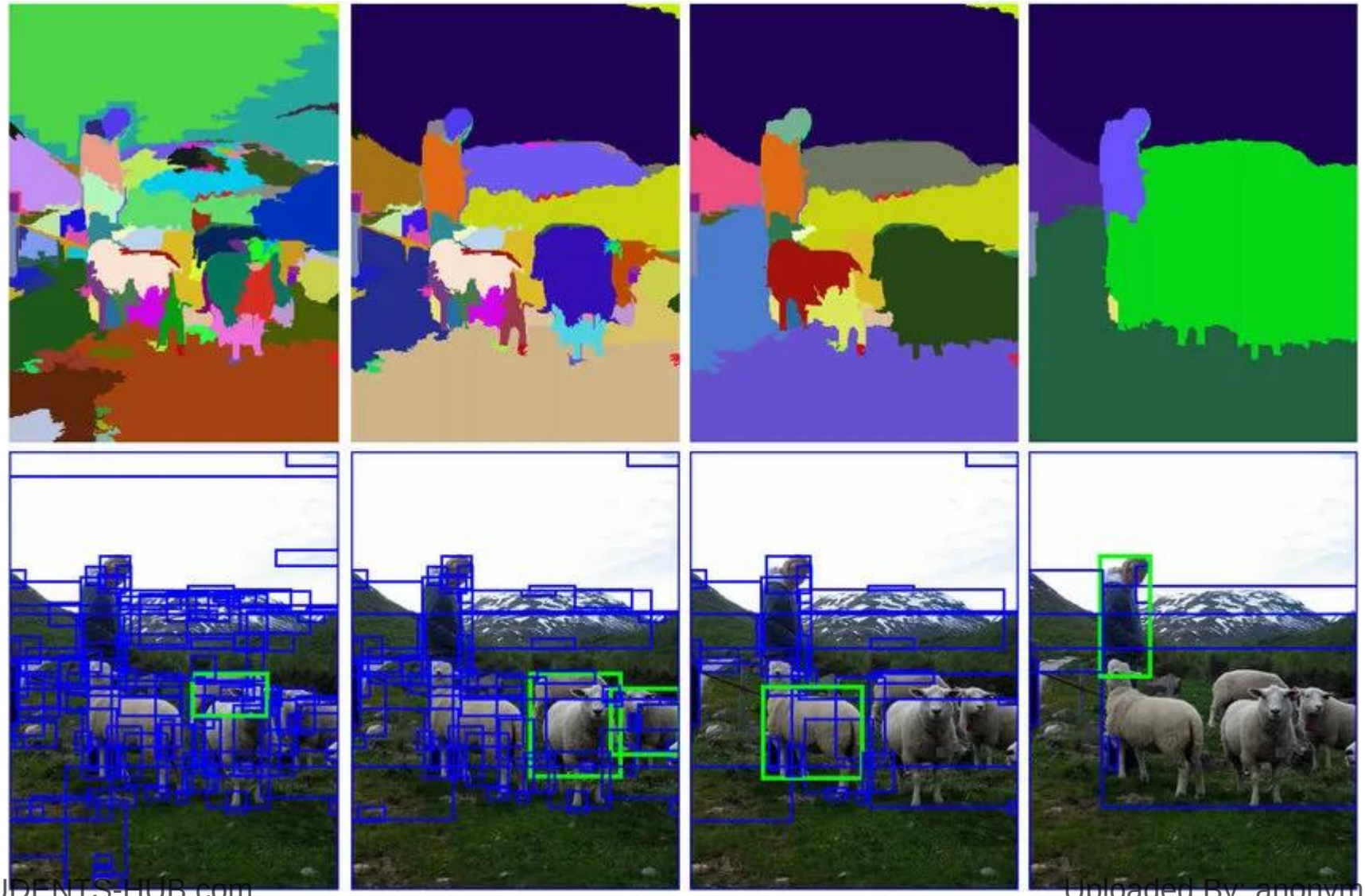
Region Proposals via Selective Search

- It aims to generate a diverse set of potential regions in an image that may contain objects.
- It create region proposals from smaller segments to larger segments in a bottom-up approach.
- Algorithm steps:
  - **Step 1:** Starts by over-segmenting the image based on intensity of the pixels using Felzenszwalb and Huttenlocher graph-based segmentation method. The goal of this step is to over-segment the image into a large number of small regions.

Region Proposals via Selective Search

- **Step 2:** From set of regions, choose two adjacent segments that are most similar based on their similarity.
  - The similarity measure often considers color, texture, size, and shape features.
- **Step 3**: Combine them into a single, larger region.
  - Use a hierarchical grouping algorithm (e.g., agglomerative clustering) based on the similarity measure.
  - The goal of this step is to merge similar segments into larger regions.
- **Step 4:** Repeat step 2. Continue grouping until a stooping criteria satisfied.
  - Common stopping criteria used in the Selective Search algorithm:
    - **Number of Region Proposals:** Set a predetermined fixed number of proposals to generate.
    - **Hierarchy Size:** Limit the size of the hierarchy or the number of levels considered.
    - **Segment Size:** Smaller segments are likely to capture fine details but may not be as meaningful.
- **Step 5**: Generate Bounding Boxes: For each group of similar regions, create a bounding box that tightly encapsulates the regions in the group. These bounding boxes represent the final region proposals.
- **Step 6:** Object Detection: Pass the selected region proposals to an object detection model (e.g., a classifier or a deep neural network) to identify and classify objects within those regions.

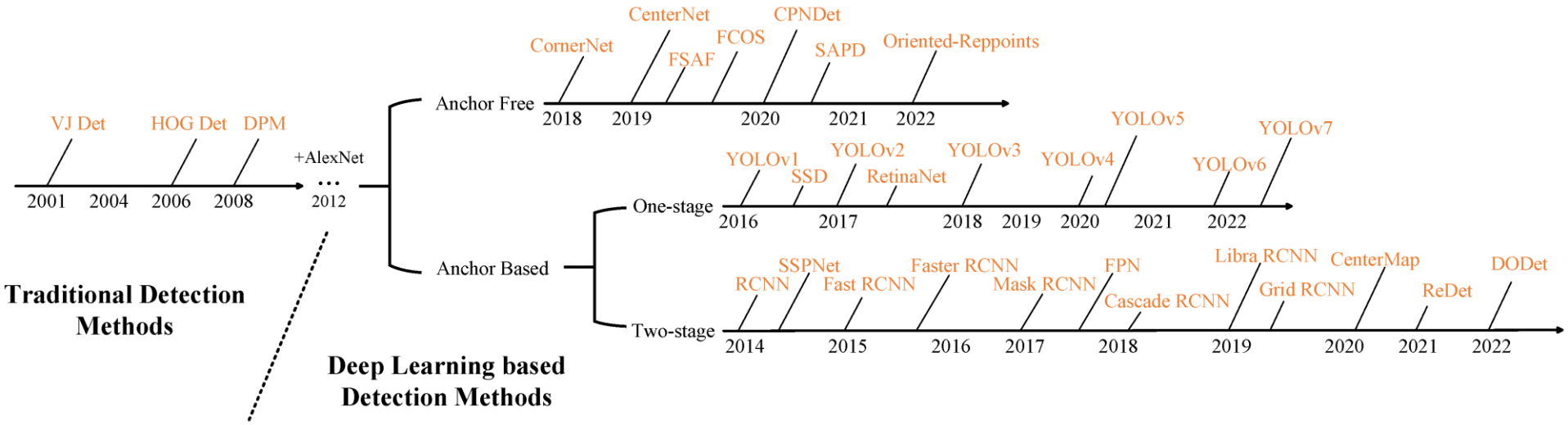
Region Proposals via Selective Search



Outline

- Introduction
- Traditional Computer Vision Techniques:
  - Sliding Window:
  - Region proposal approach **(Selective Search)**
- **Deep Learning-Based Techniques:**
  - Two-Stage Detectors:
    - R-CNN (Regions with CNN features)
  - One-Stage Detectors:
    - YOLO (You Only Look Once)
- Other Detectors
- Evaluation Measures
- Benchmarks

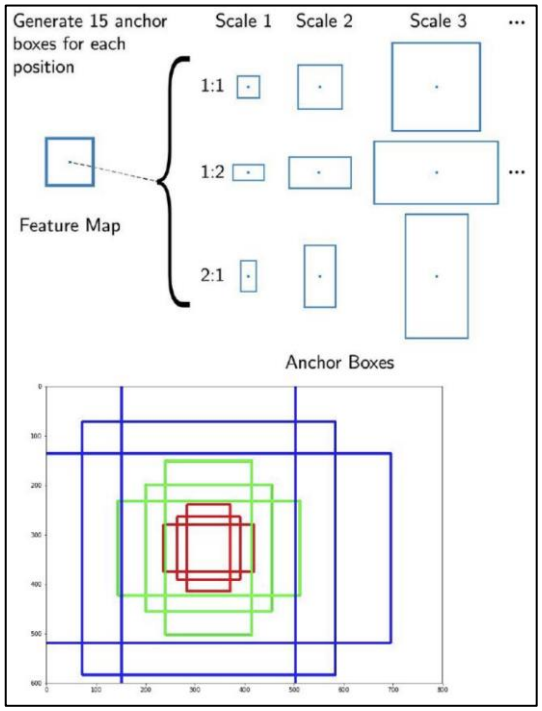
Deep Learning-Based Techniques

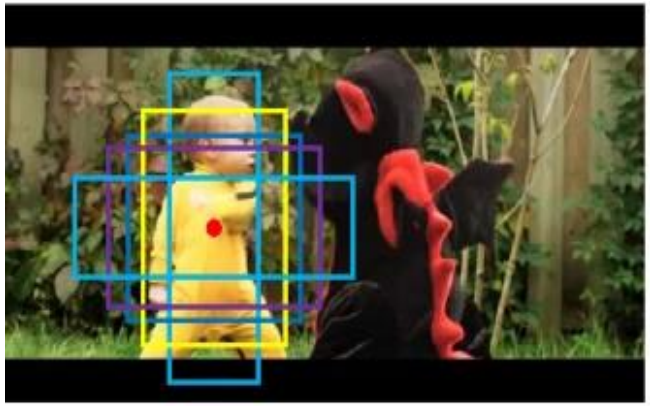


Deep Learning-Based Techniques

- **Anchor-based object detection** relies on predefined anchors, which are small rectangular boxes of various sizes and aspect ratios, to represent potential object locations.
  - The object detection model predicts the bounding box of an object relative to the anchor that best fits it.
  - Anchor boxes serve as reference templates at different scales and aspect ratios, helping the model efficiently handle objects of various sizes and shapes.
  - This approach is widely used and has proven to be effective in various object detection tasks.
  - **Advantages of Anchor-Based:**
    - Effective for handling objects of various scales and aspect ratios.
    - Provides a systematic way to propose candidate regions for object detection.
- **Anchor-free object detection**, on the other hand, does not use predefined anchors. Instead, it directly predicts the bounding box coordinates of objects, typically represented by center points and dimensions.
  - **Advantages of Anchor-free:**
    - Simplifies the model architecture and training process.
    - Avoids the need for anchor design, making the model more flexible.

Deep Learning-Based Techniques





Anchor-based

Anchor-free

- The left side is the anchor-based method which uses the fixed different ratio aspects anchors to locate the location of an object, and the right side is the anchor-free method that directly estimate the bounding box.

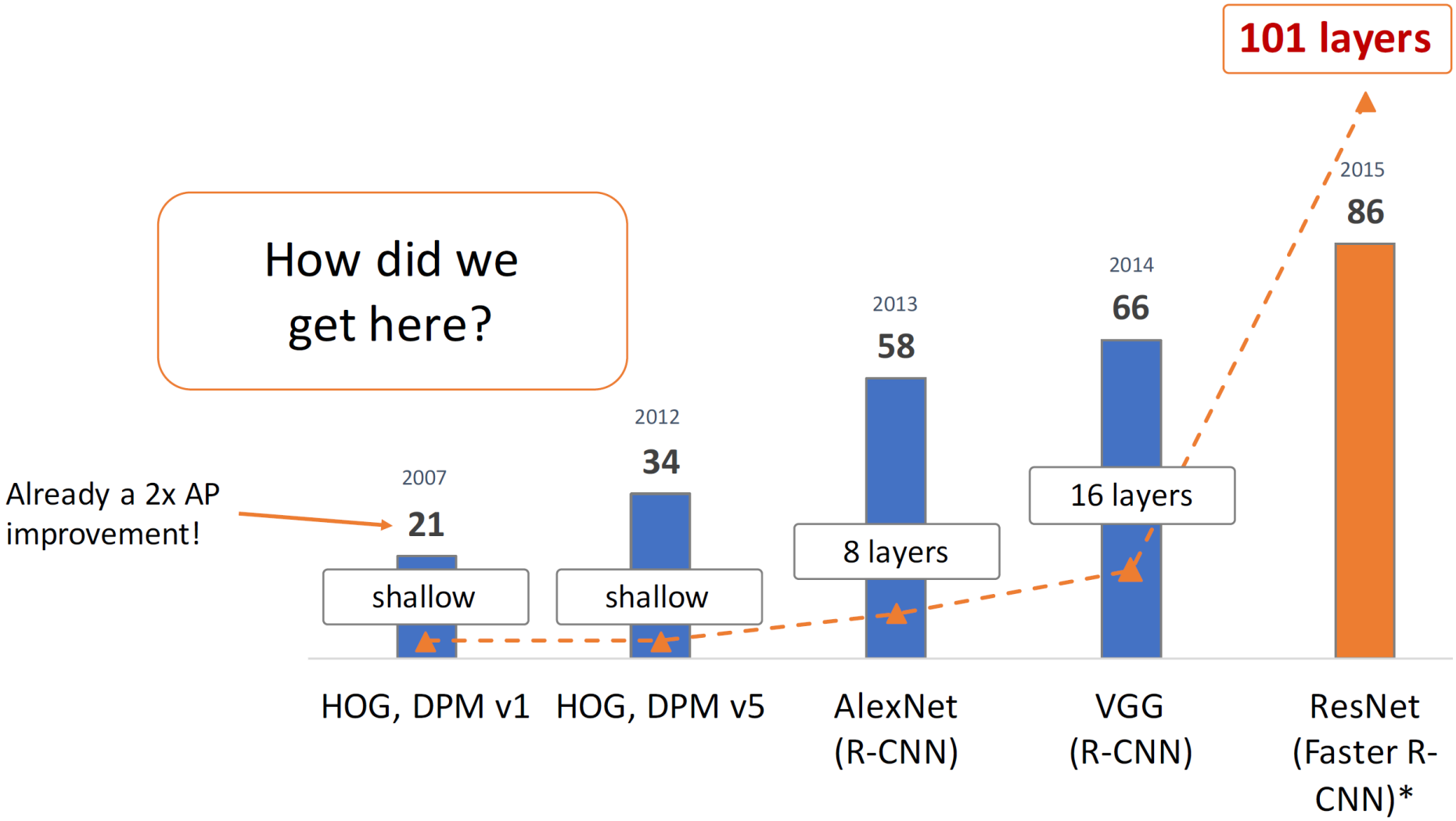
Deep Learning-Based Techniques

- **Two-Stage Object Detection**: consist of two main stages: region proposal and classification.
  1. **Region Proposal Stage:** The first stage generates a set of region proposals, which are potential locations of objects in the image. This stage typically uses a specialized algorithm, such as Selective Search or Region Proposal Network (RPN), to identify regions that are likely to contain objects.
  2. **Classification Stage:** The second stage classifies each region proposal as either an object or background. This stage typically uses a convolutional neural network (CNN) classifier to extract features from each region proposal and classify it based on those features.
- **One-Stage Object Detection:** perform both region proposal and classification in a single stage. One-stage object detection algorithms typically use a CNN to predict bounding boxes and class labels directly from the input image. The CNN is trained on a dataset of images that have been labeled with bounding boxes and class labels.

Deep Learning-Based Techniques

- Both two-stage and one-stage detectors have their strengths and trade-offs, making them suitable for different applications based on requirements such as speed, accuracy, and computational resources.
  - **Speed:** One-stage detectors are generally faster.
  - **Accuracy:** Two-stage detectors often provide higher accuracy due to the separate region proposal and classification stages.
  - **Complexity:** One-stage detectors tend to be simpler and computationally more efficient.
- If accuracy is the primary concern, then a two-stage algorithm is typically the best choice.
- One-stage object detection algorithms are often used in applications where speed is critical, such as real-time object detection in video streams.

Object Detection - Past vs. Present



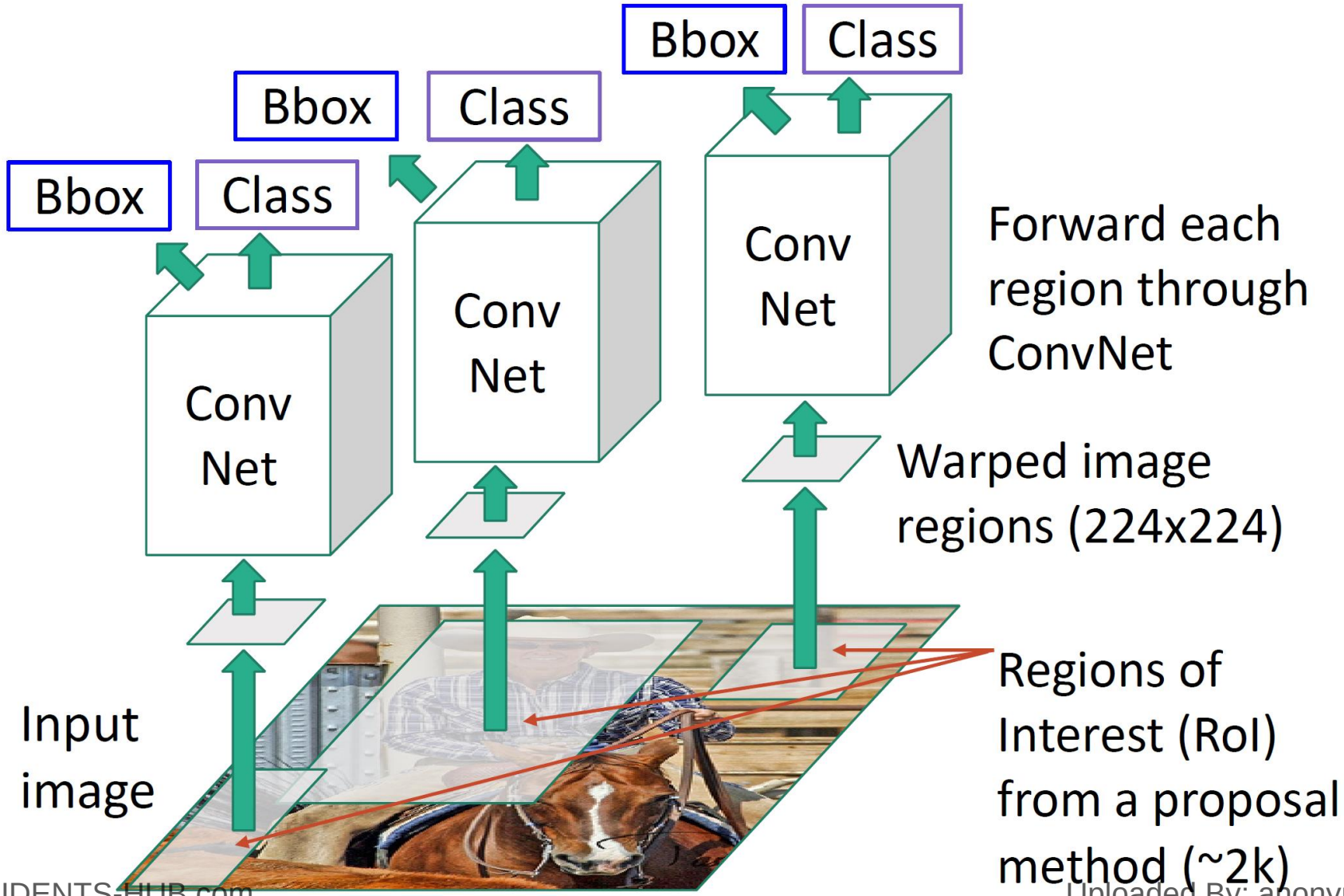
Outline

- Introduction
- Traditional Computer Vision Techniques:
  - Sliding Window:
  - Region proposal approach **(Selective Search)**
- Deep Learning-Based Techniques:
  - **Two-Stage Detectors:**
    - **R-CNN (Regions with CNN features)**
  - One-Stage Detectors:
    - YOLO (You Only Look Once)
- Other Detectors
- Evaluation Measures
- Benchmarks

R-CNN (Regions with CNN features)

- R-CNN (Region-based Convolutional Neural Network) is a two-stage object detection framework that combines region proposals with convolutional neural networks.
- R-CNN consists of four main steps:
  - **Region Proposal**: R-CNN uses a region proposal algorithm (such as selective search) to generate candidate regions in an image that are likely to contain objects.
  - **Feature Extraction**: Each proposed region is cropped and warped, and the features within these regions are extracted using a pre-trained Convolutional Neural Network (CNN).
  - **Object Classification**: The CNN-extracted features are used to train Support Vector Machines (SVMs) for object classification, determining the presence of objects and their respective classes.
  - **Bounding Box Regression**: A separate regression model is trained to refine the coordinates of the proposed bounding boxes, improving the localization accuracy of the detected objects. This ensures that the bounding boxes tightly enclose the detected objects.

R-CNN (Regions with CNN features)



R-CNN: Bounding Box Regression

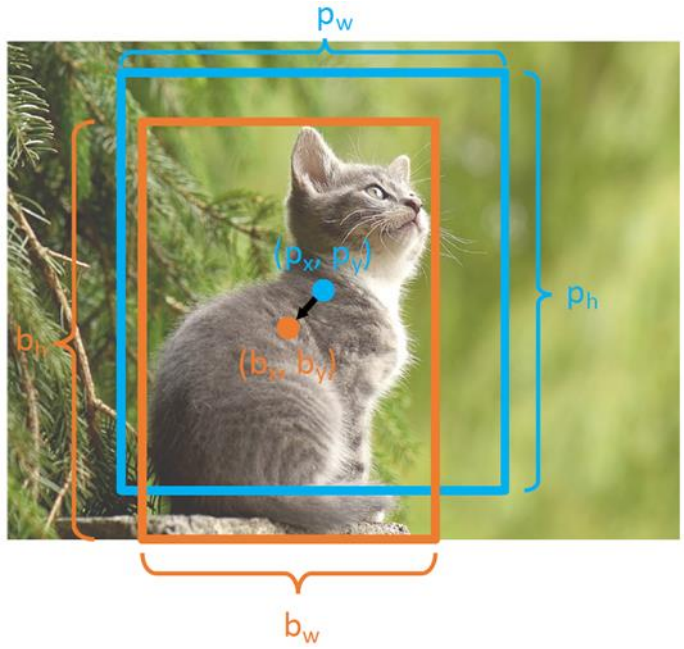
- Bounding Box regression is a metric for measuring how well predicted bounding boxes captures objects.
- It aims to refine the predicted bounding boxes generated by the region proposal stage to accurately enclose the detected objects (ground truth bounding boxes).
- Bounding Box Regression utilizes machine learning regression technique to learn the offsets between the predicted bounding boxes and the true object locations.
- The target labels for regression are the corrections needed for the coordinates of the proposed bounding boxes to match the coordinates of the ground truth bounding boxes.

R-CNN: Bounding Box Regression

- Bounding box regression: Predict "transform" to correct the Bbox RoI: 4 numbers (t_x, t_y, t_h, t_w)

Consider a region proposal with center (p_x, p_y), width p_w, height p_h

Model predicts a transform (t_x, t_y, t_w, t_h) to correct the region proposal



The output box is defined by:

$b_x = p_x + p_w t_x$ Shift center by amount relative to proposal size

$b_y = p_y + p_h t_y$

$b_w = p_w \exp(t_w)$ Scale proposal; exp ensures that scaling factor is > 0

$b_h = p_h \exp(t_h)$

R-CNN: Bounding Box Regression

Consider a region proposal with center (p_x, p_y), width p_w, height p_h

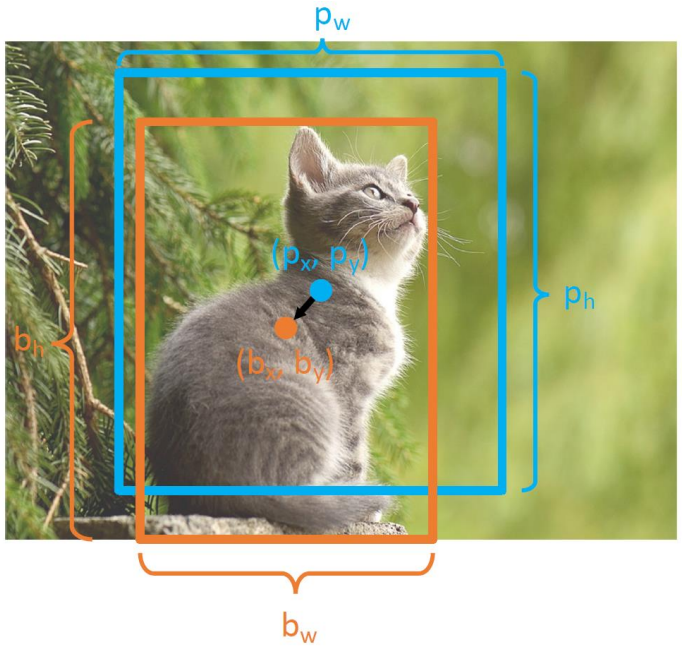
Model predicts a transform (t_x, t_y, t_w, t_h) to correct the region proposal

The output box is:

$$b_x = p_x + p_w t_x$$
$$b_y = p_y + p_h t_y$$
$$b_w = p_w \exp(t_w)$$
$$b_h = p_h \exp(t_h)$$

When transform is 0, output = proposal

L2 regularization encourages leaving proposal unchanged



R-CNN: Bounding Box Regression

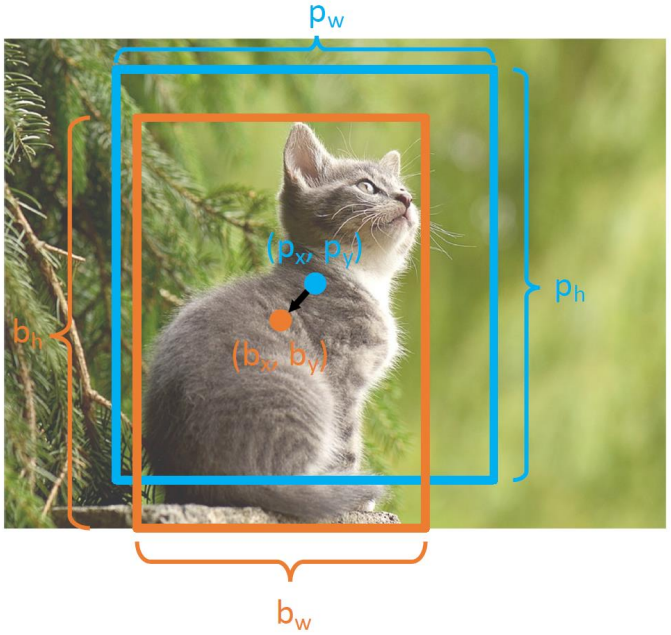
Consider a region proposal with center (p_x, p_y), width p_w, height p_h

Model predicts a transform (t_x, t_y, t_w, t_h) to correct the region proposal

The output box is:

$$b_x = p_x + p_w t_x$$
$$b_y = p_y + p_h t_y$$
$$b_w = p_w \exp(t_w)$$
$$b_h = p_h \exp(t_h)$$

Scale / Translation invariance: Transform encodes *relative* difference between proposal and output; important since CNN doesn't see absolute size or position after cropping



R-CNN: Bounding Box Regression

Consider a region proposal with center (p_x, p_y), width p_w, height p_h

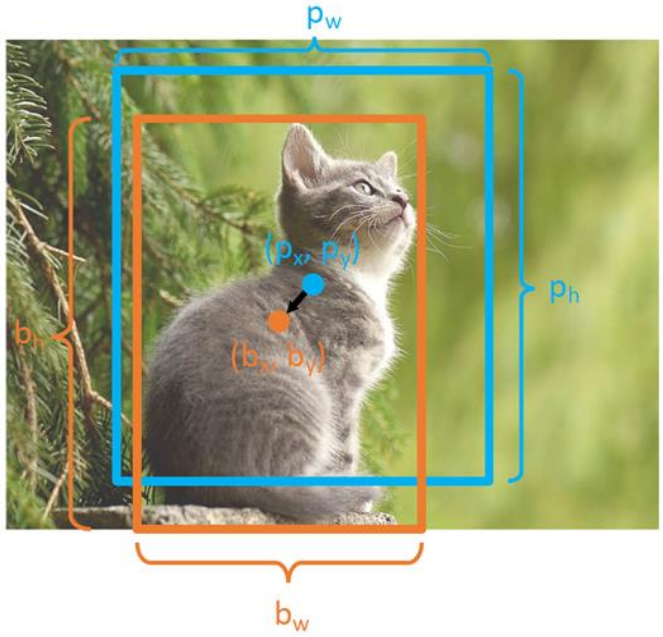
Model predicts a transform (t_x, t_y, t_w, t_h) to correct the region proposal

The output box is:

$$b_x = p_x + p_w t_x$$
$$b_y = p_y + p_h t_y$$
$$b_w = p_w \exp(t_w)$$
$$b_h = p_h \exp(t_h)$$

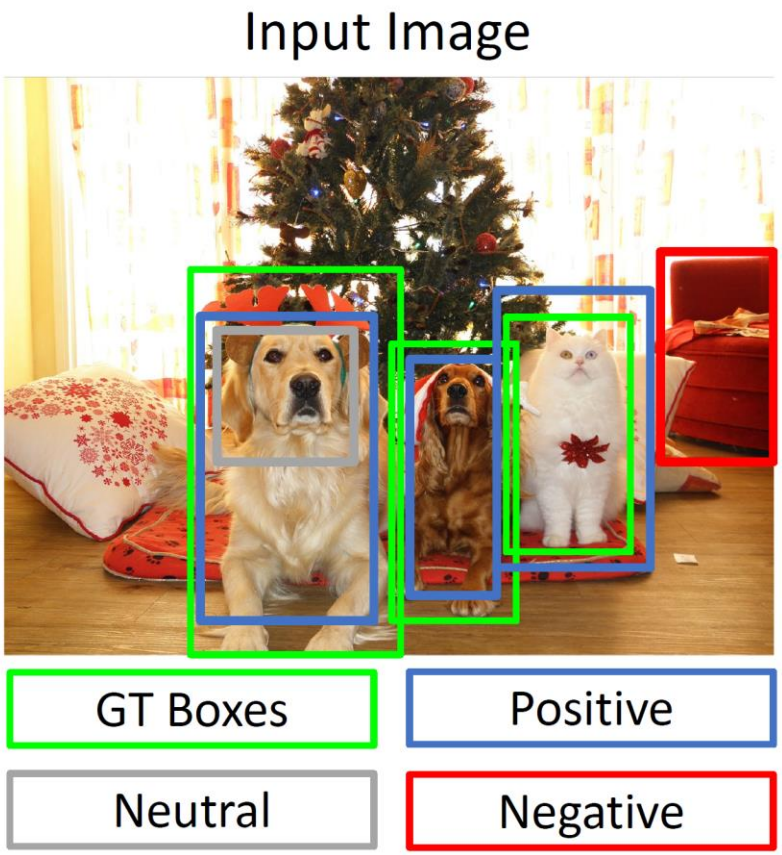
Given proposal and target output, we can solve for the transform the network should output:

$$t_x = (b_x - p_x)/p_w$$
$$t_y = (b_y - p_y)/p_h$$
$$t_w = \log(b_w/p_w)$$
$$t_h = \log(b_h/p_h)$$



R-CNN: Bounding Box Regression

- R-CNN Training



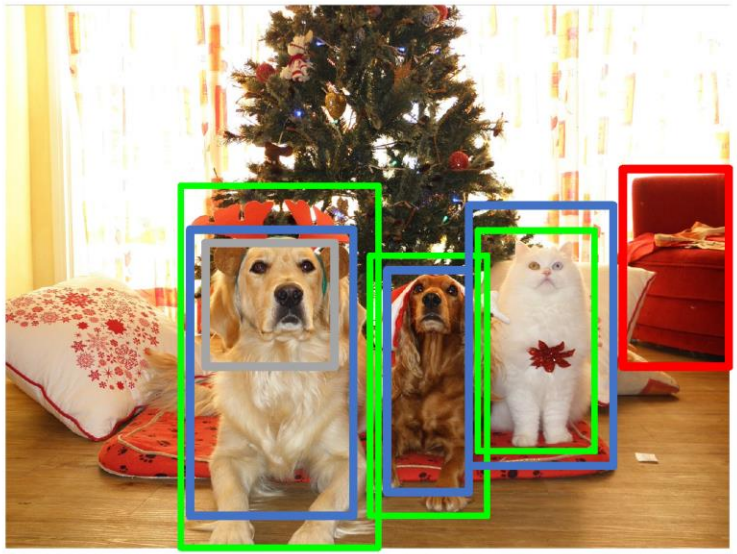
Categorize each region proposal as positive, negative, or neutral based on overlap with ground-truth boxes:

Positive: > 0.5 IoU with a GT box
Negative: < 0.3 IoU with all GT boxes
Neutral: between 0.3 and 0.5 IoU with GT boxes

R-CNN: Bounding Box Regression

R-CNN Training

Input Image

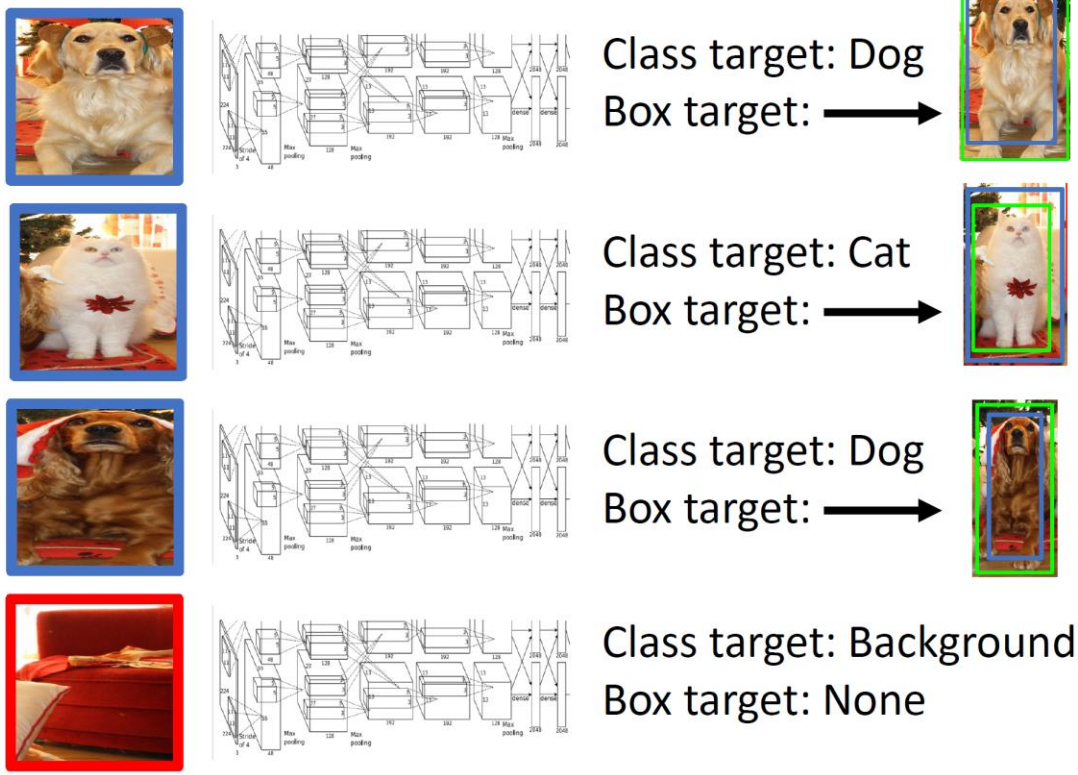


GT Boxes | Positive

Neutral | Negative

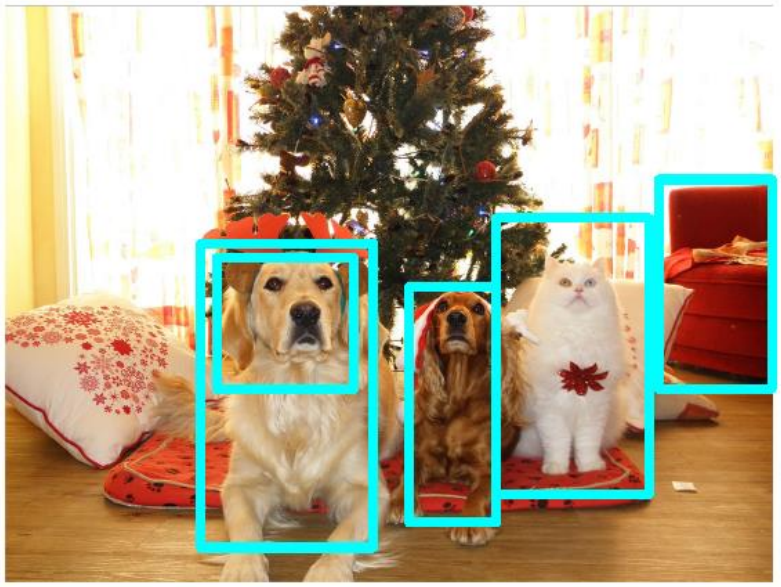
Crop pixels from each positive and negative proposal, resize to 224 x 224

Run each region through CNN
Positive regions: predict class and transform
Negative regions: just predict class



R-CNN Test-Time

Input Image



Region Proposals

1. Run proposal method
2. Run CNN on each proposal to get class scores, transforms
3. Threshold class scores to get a set of detections

2 problems:

- CNN often outputs overlapping boxes
- How to set thresholds?

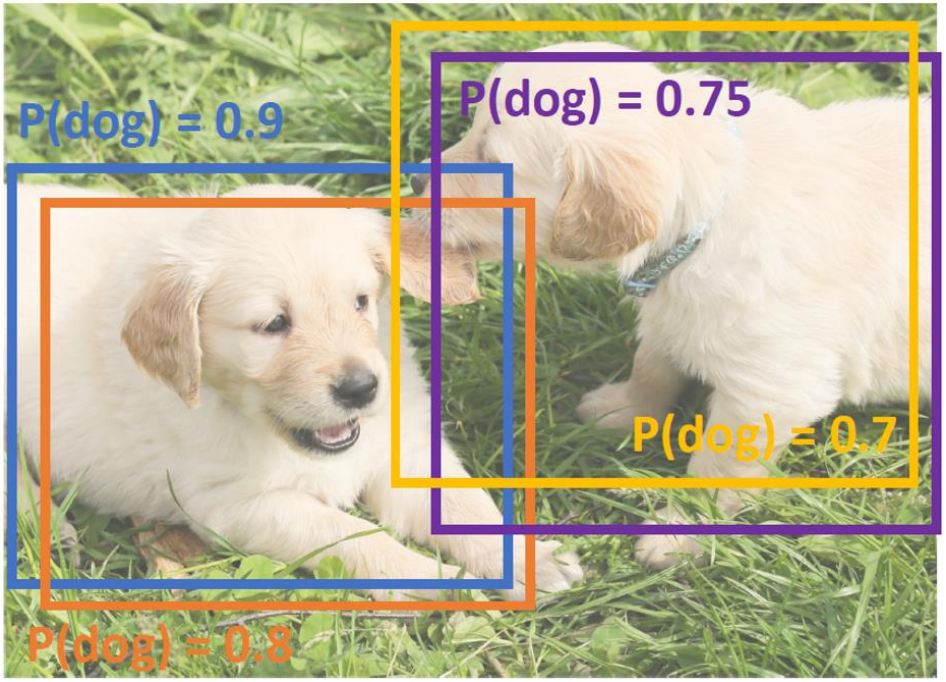
Overlapping Boxes: Non-Max Suppression (NMS)

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

IoU(■, ■) = **0.78**
IoU(■, ■) = 0.05
IoU(■, ■) = 0.07



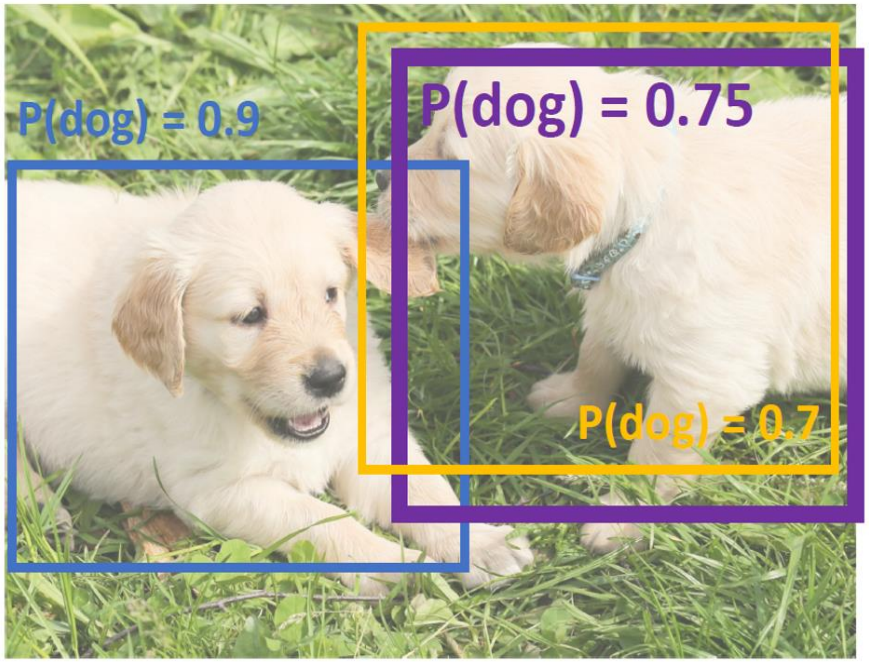
Overlapping Boxes: Non-Max Suppression (NMS)

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

IoU(■, ■) = **0.74**

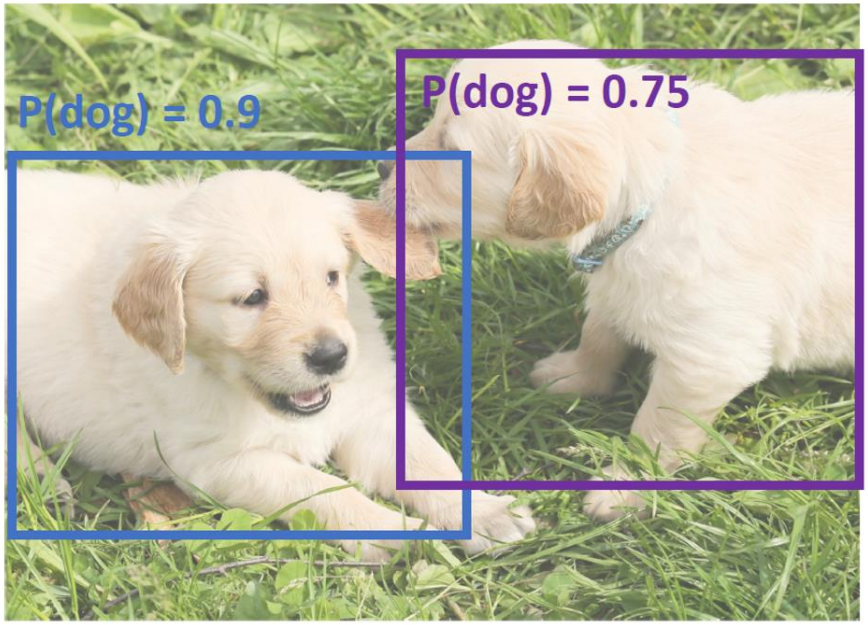


Overlapping Boxes: Non-Max Suppression (NMS)

Problem: Object detectors often output many overlapping detections:

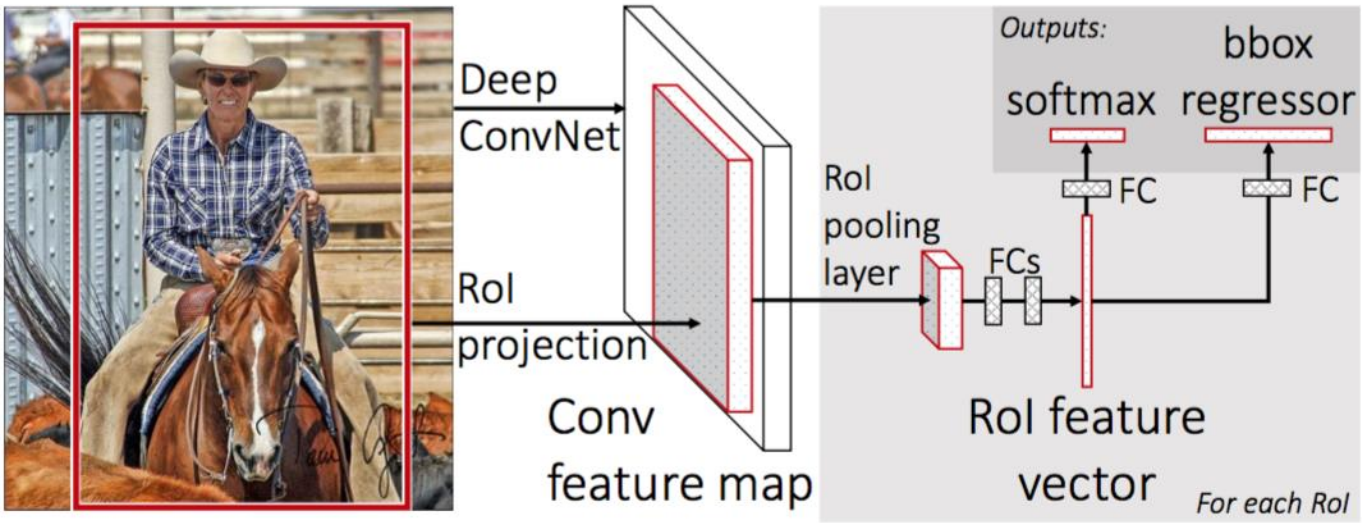
Solution: Post-process raw detections using **Non-Max Suppression (NMS)**

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



Fast R-CNN

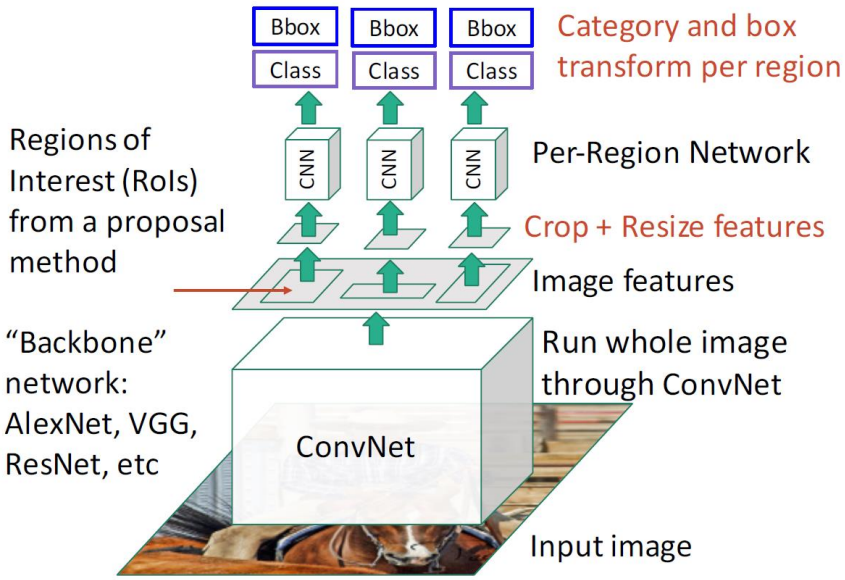
- Instead of extracting CNN feature vectors independently for each region proposal, this model aggregates them into one CNN forward pass over the entire image and the region proposals share this feature matrix.
- Then the same feature matrix is branched out to be used for learning the object classifier and the bounding-box regressor. In conclusion, computation sharing speeds up R-CNN.



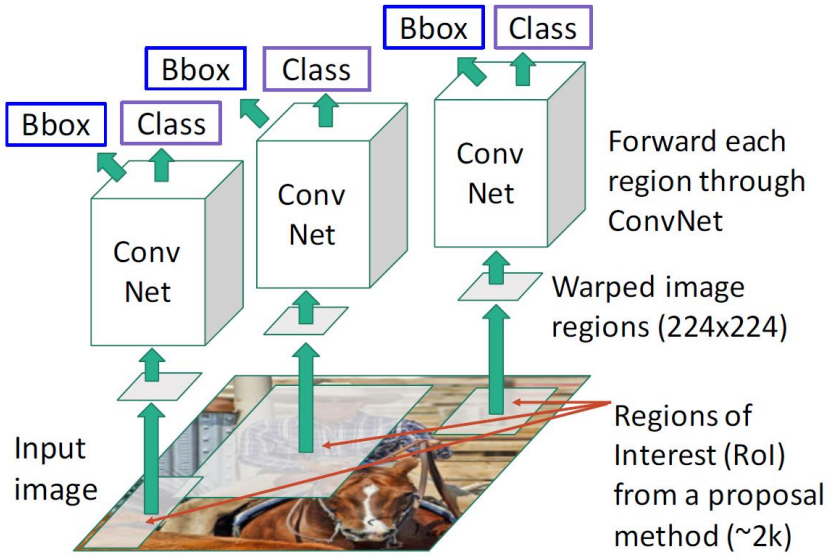
Fast R-CNN is much faster in both training and testing time. However, the improvement is not dramatic because the region proposals are generated separately by another model and that is very expensive.

Fast R-CNN vs “Slow” R-CNN

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

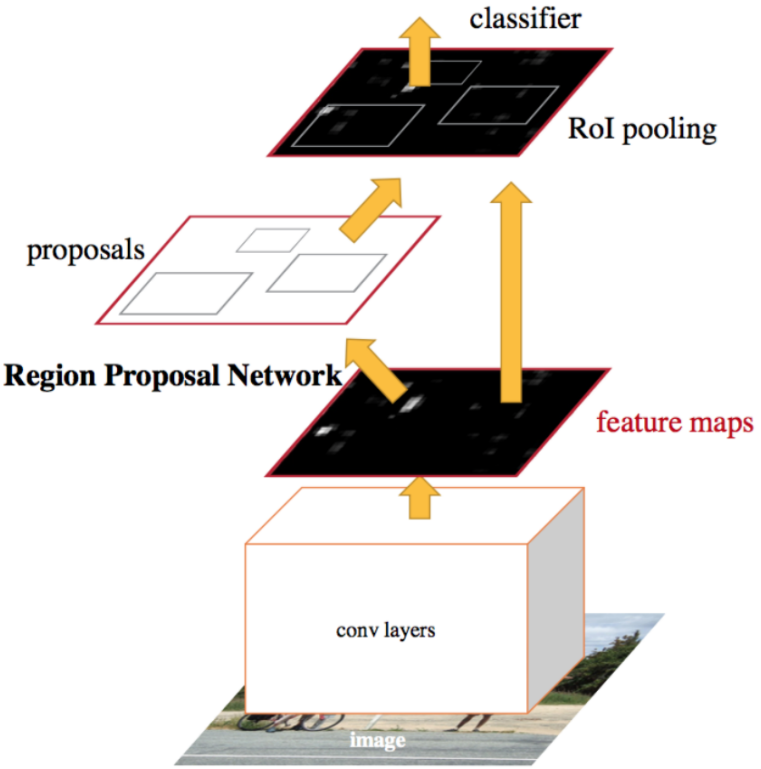


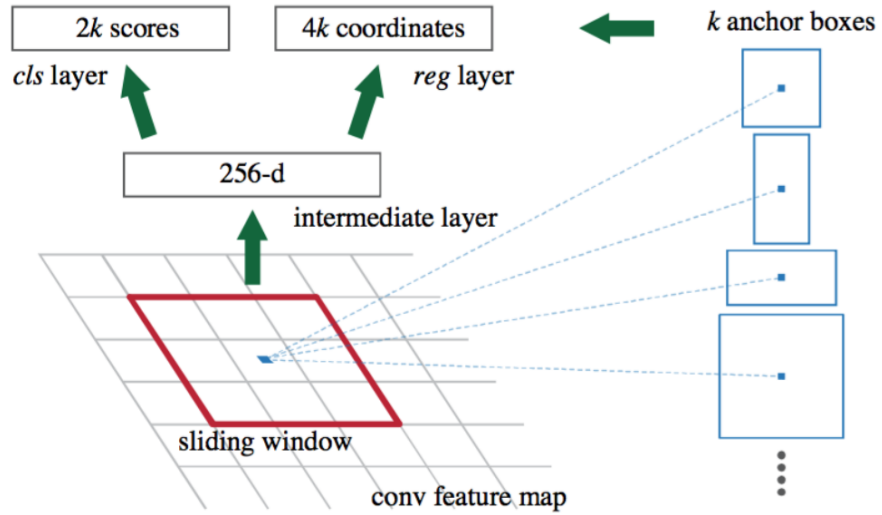
“Slow” R-CNN: Apply differentiable cropping to shared image features



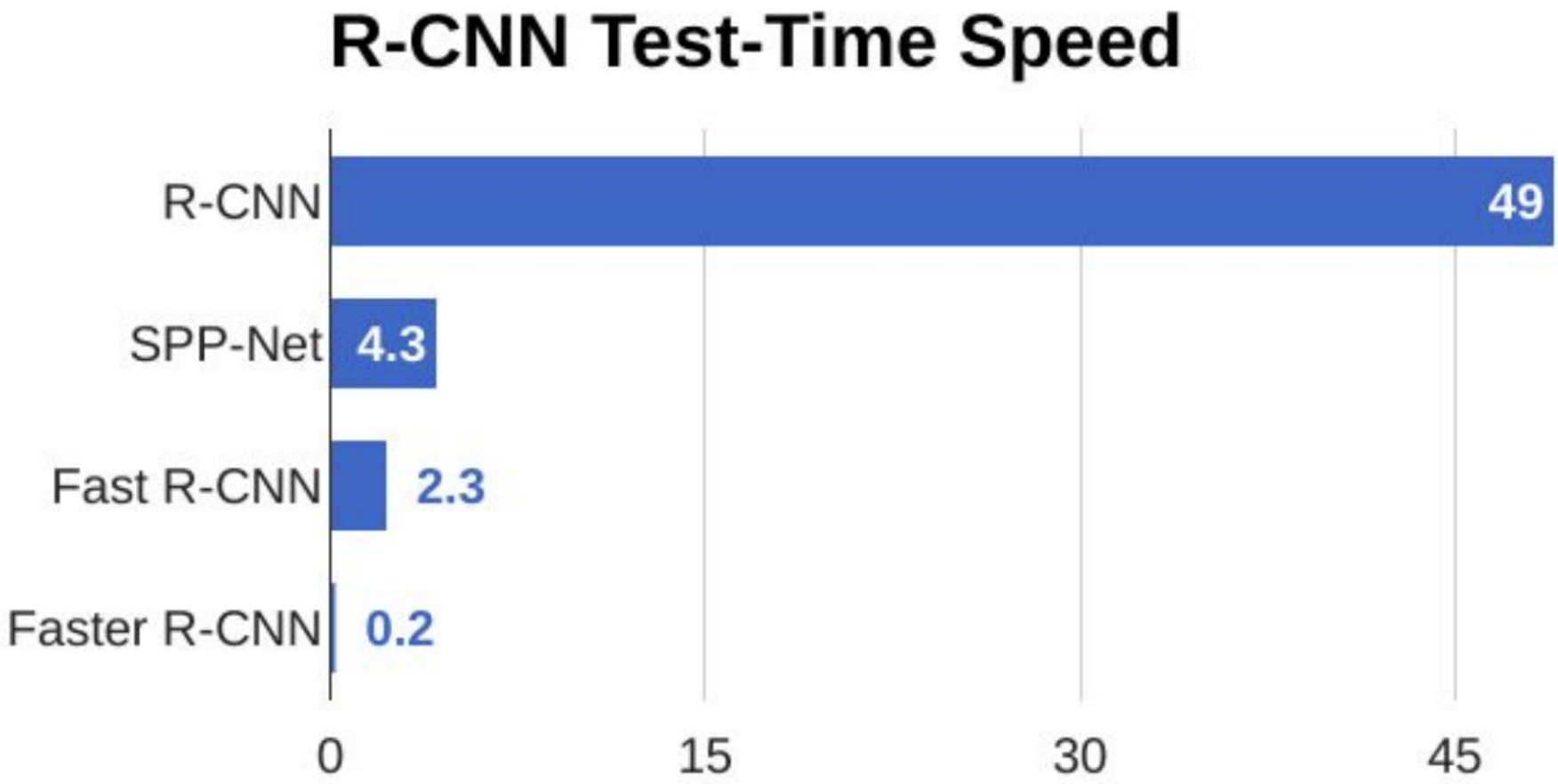
Faster R-CNN

- An intuitive speedup solution is to integrate the region proposal algorithm into the CNN model. **Faster R-CNN** is doing exactly this: construct a single, unified model composed of RPN (region proposal network) and fast R-CNN with shared convolutional feature layers.





Comparison



Outline

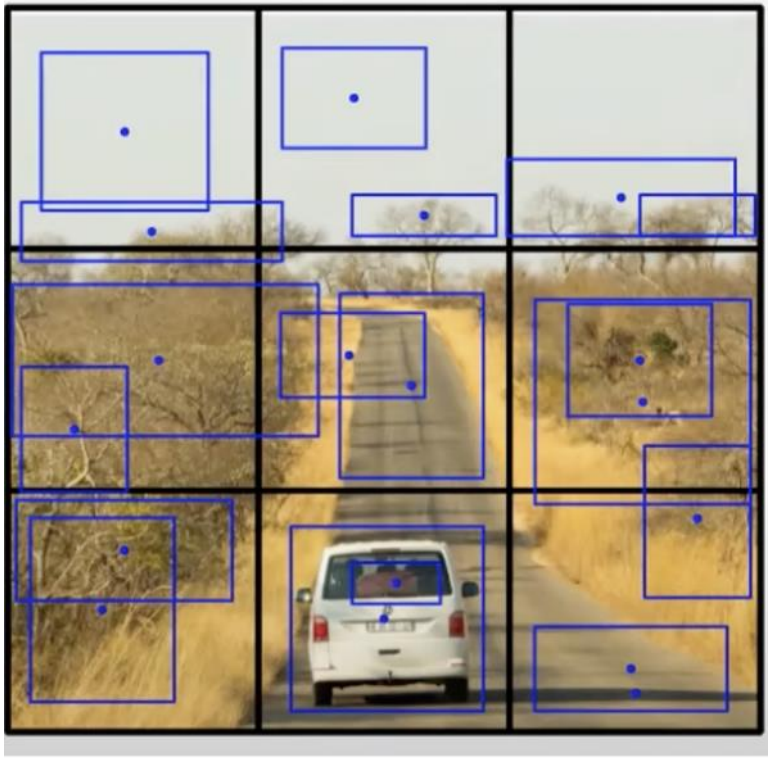
- Introduction
- Traditional Computer Vision Techniques:
  - Sliding Window:
  - Region proposal approach **(Selective Search)**
- Deep Learning-Based Techniques:
  - Two-Stage Detectors:
    - R-CNN (Regions with CNN features)
  - **One-Stage Detectors:**
    - **YOLO (You Only Look Once)**
- Other Detectors
- Evaluation Measures
- Benchmarks

YOLO (You Only Look Once)

- The R-CNN family of techniques primarily use regions to localize the objects within the image. The network does not look at the entire image, only at the parts of the images which have a higher chance of containing an object.
- The YOLO framework (You Only Look Once) on the other hand, deals with object detection in a different way. It takes the entire image in a single instance and predicts the bounding box coordinates and class probabilities for these boxes.
- You Only Look Once (YOLO) is a state-of-the-art, real-time object detection algorithm introduced in 2015
- The authors frame the object detection problem as a regression problem instead of a classification task by spatially separating bounding boxes and associating probabilities to each of the detected images using a single convolutional neural network (CNN).

How does the YOLO Framework Function?

- First, image is split into a SxS grid
- For each grid square, generate B bounding boxes
- For each bounding box, there are 5 predictions
- Image classification and localization are applied on each cell grid.
- 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.

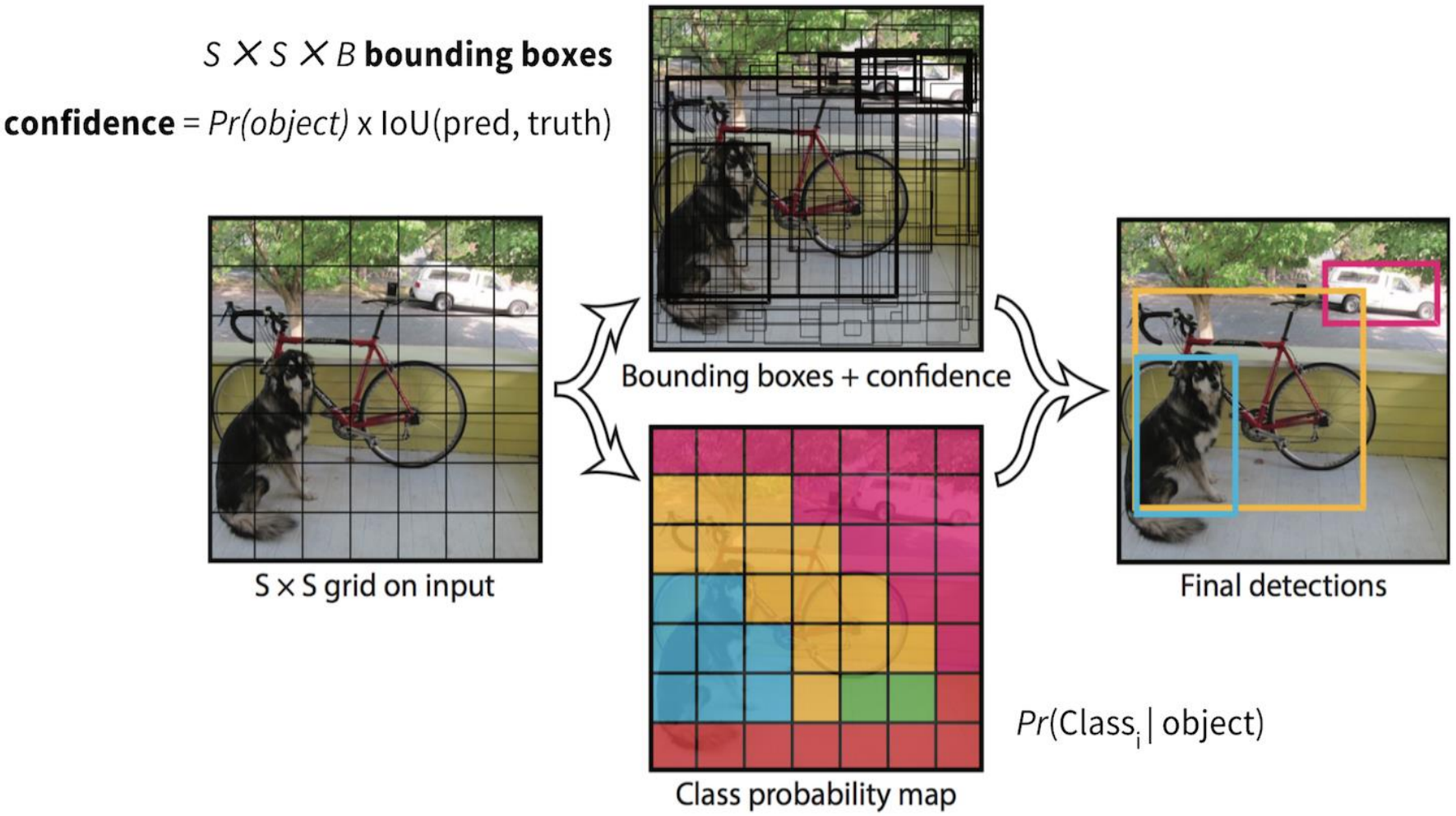


S = 3, B = 2

How does the YOLO Framework Function?

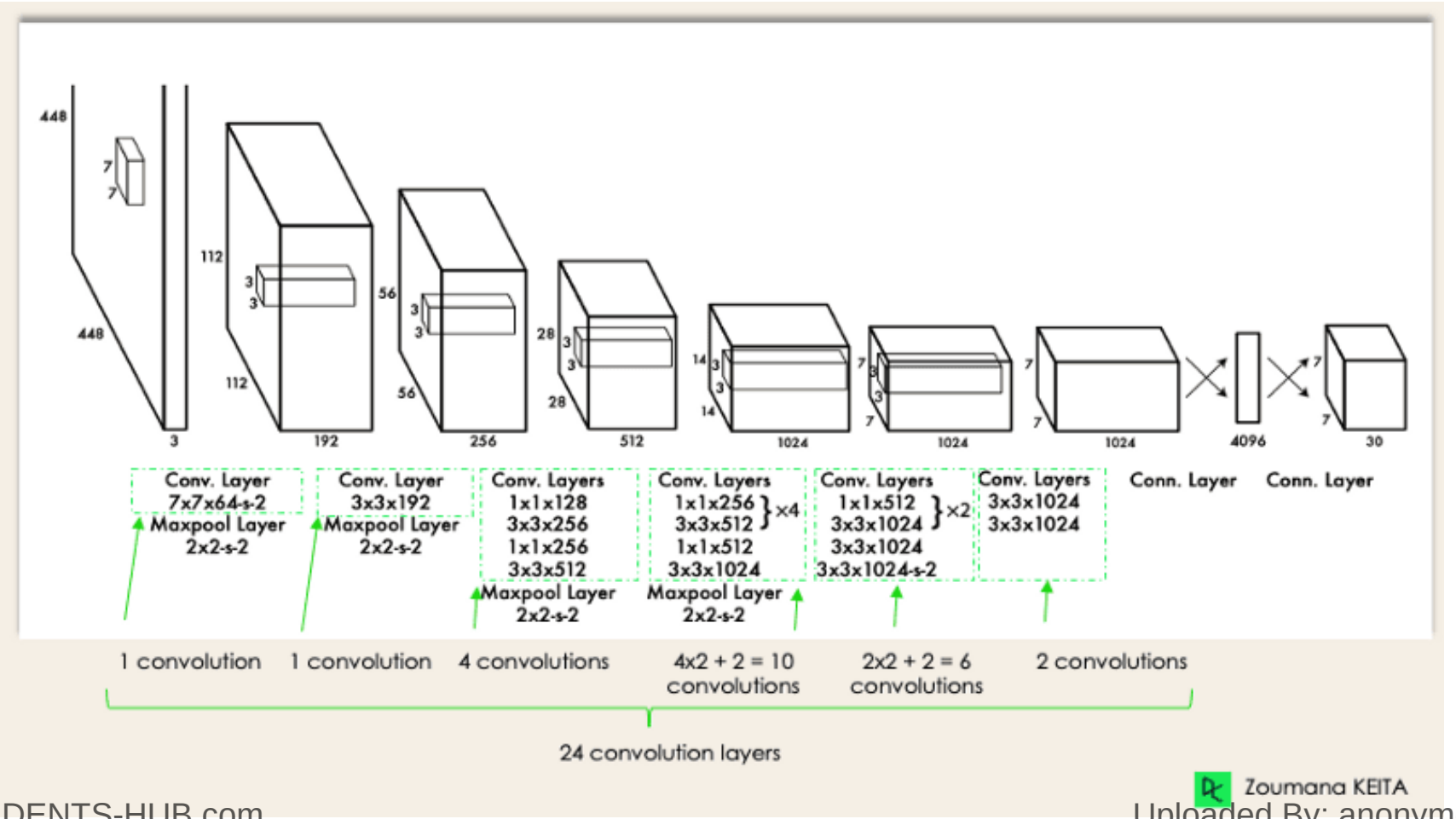
- YOLO predicts both bounding box parameters and class probabilities for each grid cell in a single forward pass through the network.
- **Bounding Box Predictions:**
  - For each grid cell, YOLO predicts bounding box parameters. These parameters include the coordinates (x, y) of the box's center, its width (w), and its height (h).
- **Class Predictions:**
  - YOLO predicts class probabilities for each bounding box within a grid cell. The model assigns a probability score for each possible class that the object might belong to (e.g., person, car, dog).
- **Confidence Score:**
  - Each bounding box is associated with a confidence score. This score reflects the model's confidence that the predicted box contains an object.

How does the YOLO Framework Function?



YOLO Architecture

- YOLO architecture is similar to GoogleNet. As illustrated below, it has overall 24 convolutional layers, four max-pooling layers, and two fully connected layers.



YOLO Architecture

- 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.

YOLO Training

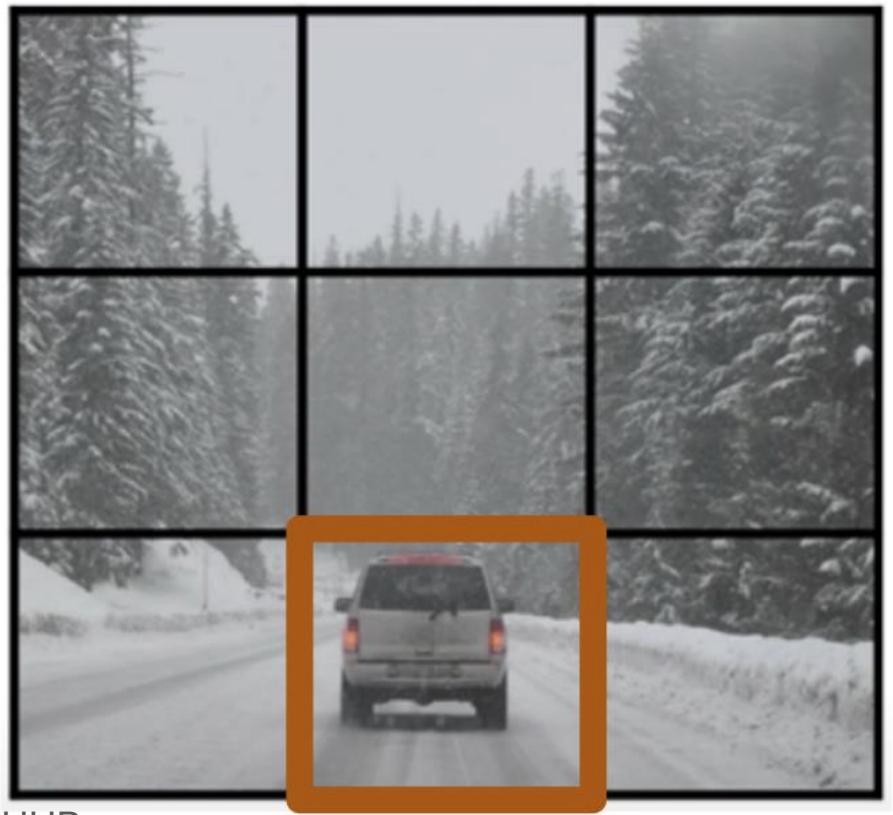
- We need to pass the labelled data to the model in order to train it.
- Suppose we have divided the image into a grid of size 3 X 3 and there are a total of 2 classes which we want the objects to be classified into.
- YOLO determines the attributes of the bounding boxes using a single regression module in the following format, where Y is the final vector representation for each bounding box.

  Y = [pc, bx, by, bh, bw, c1, c2]

- Here,
  - pc corresponds to the probability score of the grid containing an object.
  - 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.

YOLO Training

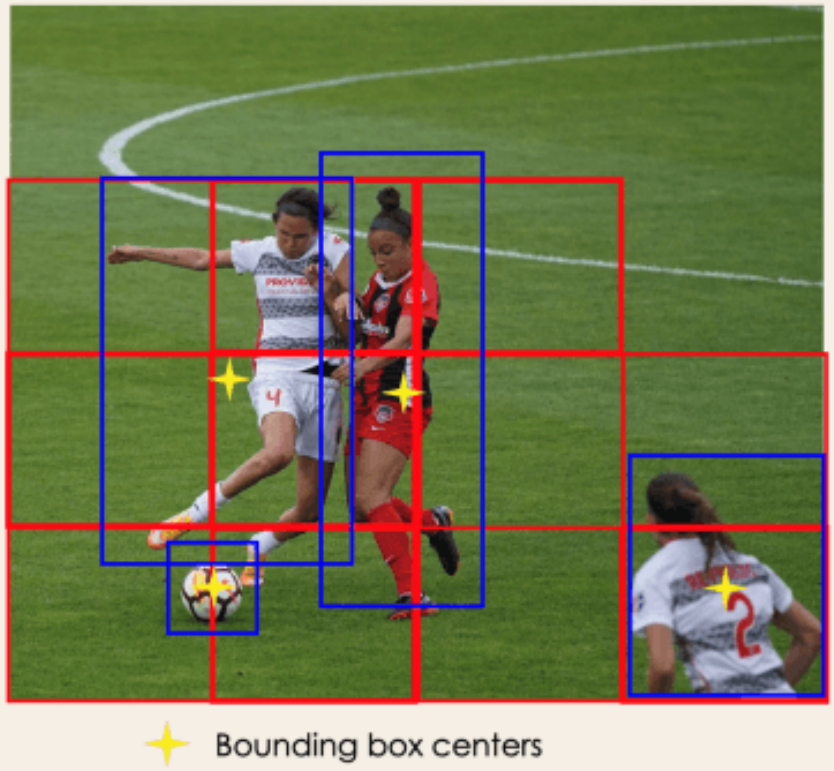
- YOLO is a regression algorithm. What is X? What is Y?
  - X is simple, just an image width (in pixels) * height (in pixels) * RGB values
  - Y is a tensor of size S * S * (B * 5 + C)
  - B*5 + C term represents the predictions + class predicted distribution for a grid block



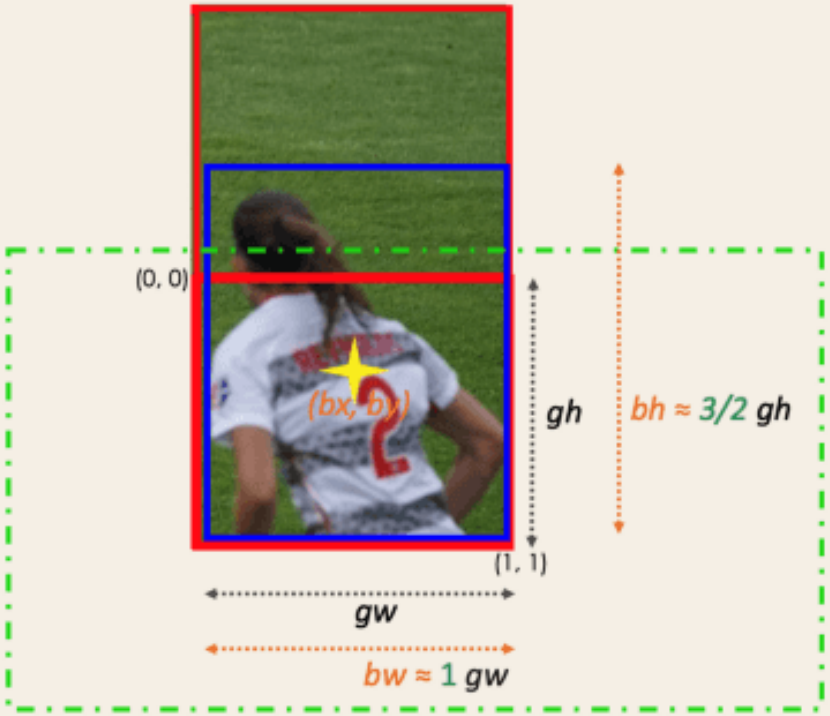
GT label example:

1
b_x_1
b_y_1
b_h_1
b_w_1
0
?
?
?
?
c_1 = 1
c_2 = 0

How to Encode Bounding Boxes?



Bounding box centers



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

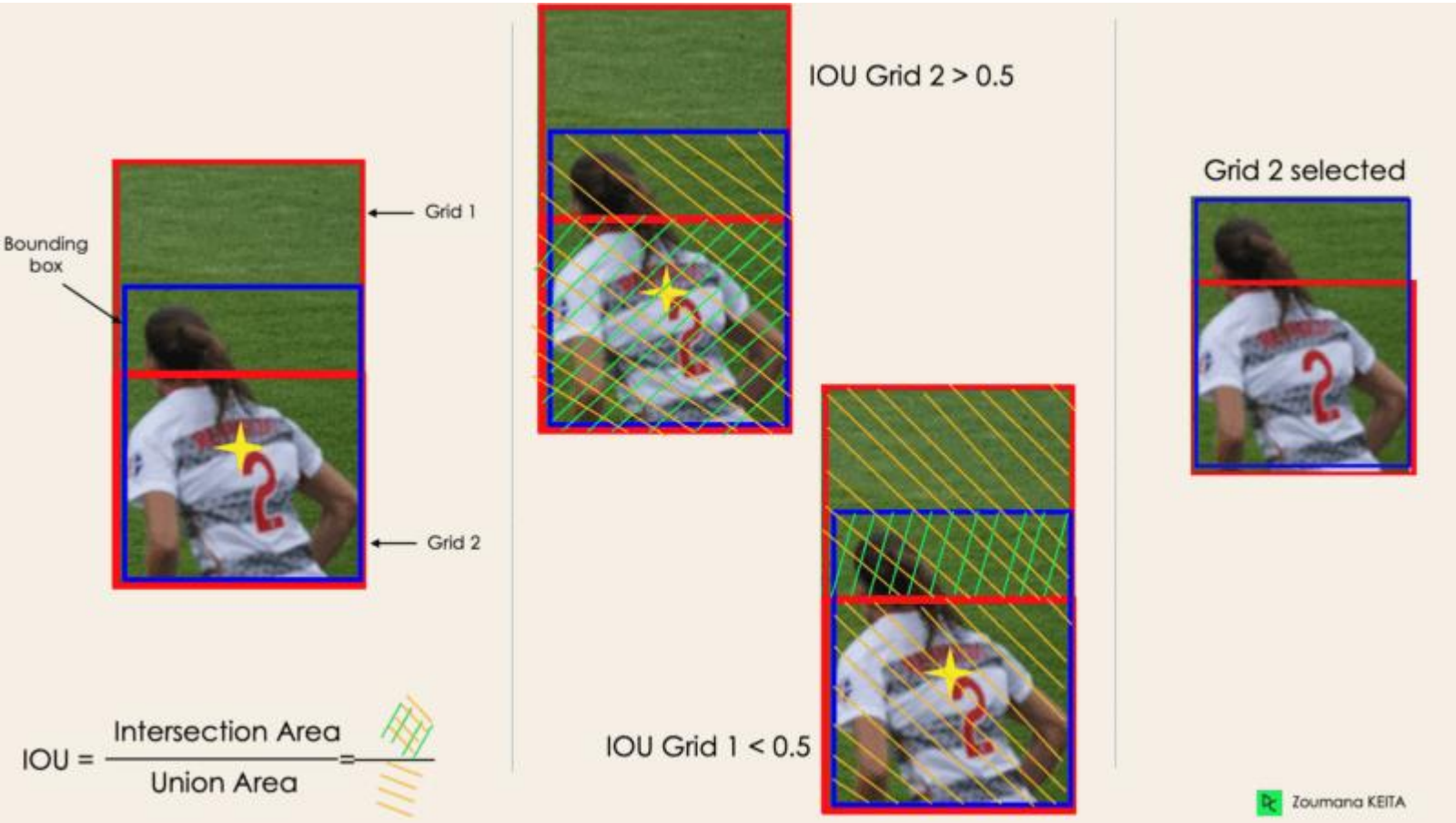
- *gh, gw*: height & width of the grid
- $0 \leq bx \leq 1$
- $0 \leq by \leq 1$
- *bh and bw* can be more than 1

Zoumana KEITA

Intersection Over Unions or IOU

- Most of the time, a single object in an image can have multiple grid box candidates for prediction, even though not all of them are relevant.
- The goal of the IOU (a value between 0 and 1) is to discard such grid boxes to only keep those that are relevant.
- Here is the logic behind it:
  - 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.
- Intuitively, the more you increase the threshold, the better the predictions become.

Intersection Over Unions or IOU

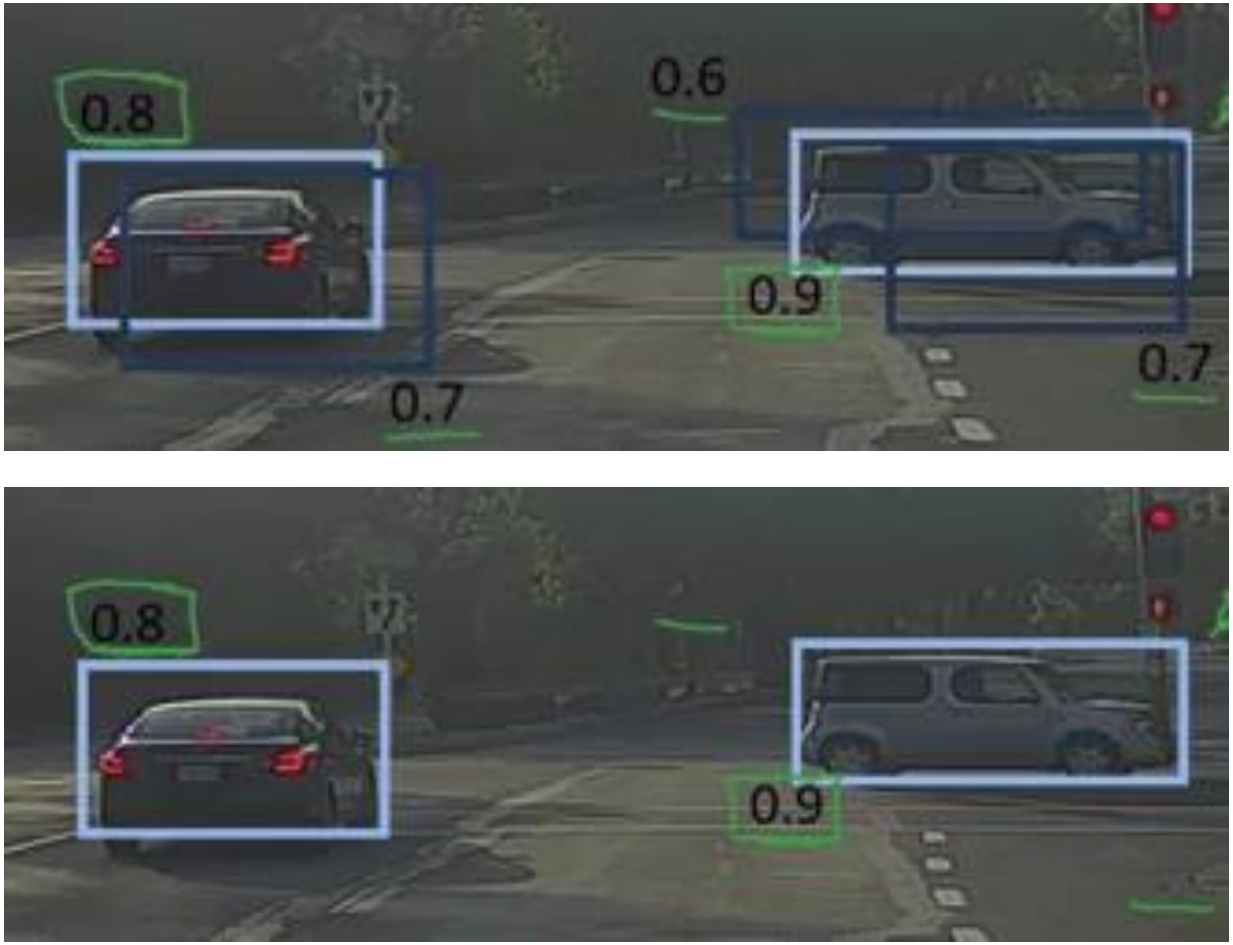


We can observe that the object originally had two grid candidates, then only "Grid 2" was selected at the end.

Non-Max Suppression or NMS

- Setting a threshold for the IOU is not always enough because an object can have multiple boxes with IOU beyond the threshold, and leaving all those boxes might include noise.
- Non-Max suppression algorithm:
  1. Discard all the boxes having probabilities less than or equal to a pre-defined threshold (say, 0.5)
  2. For the remaining boxes:
     1. Pick the box with the highest probability and take that as the output prediction
     2. Discard any other box which has IoU greater than the threshold with the output box from the above step
  3. Repeat step 2 until all the boxes are either taken as the output prediction or discarded

Non-Max Suppression or NMS



YOLO Objective Function

- For YOLO, we need to minimize the following loss
- Sum squared error is used

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

Coordinate Loss: Minimize the difference between x,y,w,h pred and x,y,w,h ground truth. ONLY IF object exists in grid box and if bounding box is resp for pred

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

Confidence Loss: Loss based on confidence ONLY IF there is object

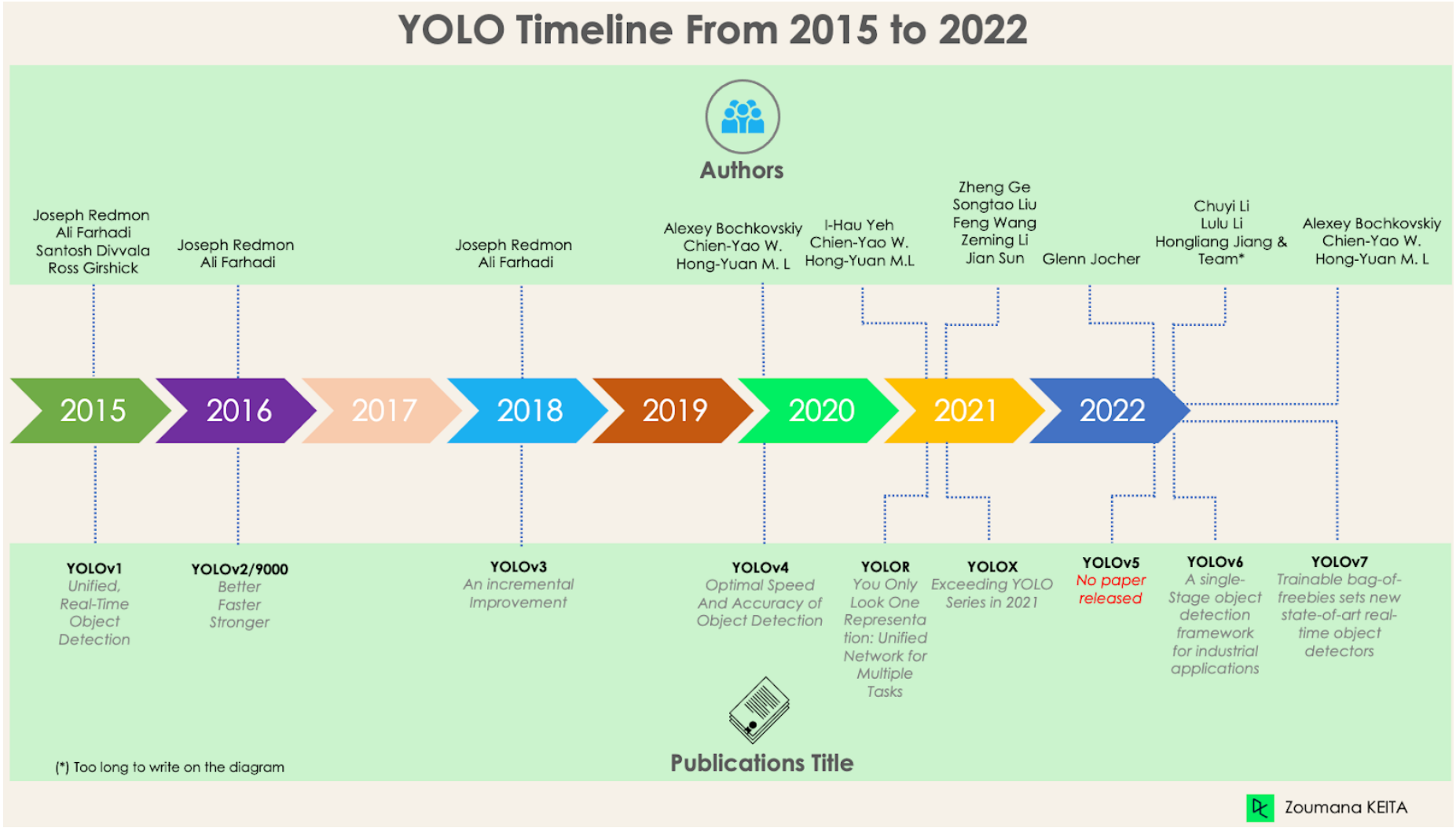
$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

No Object Loss based on confidence if there is no object

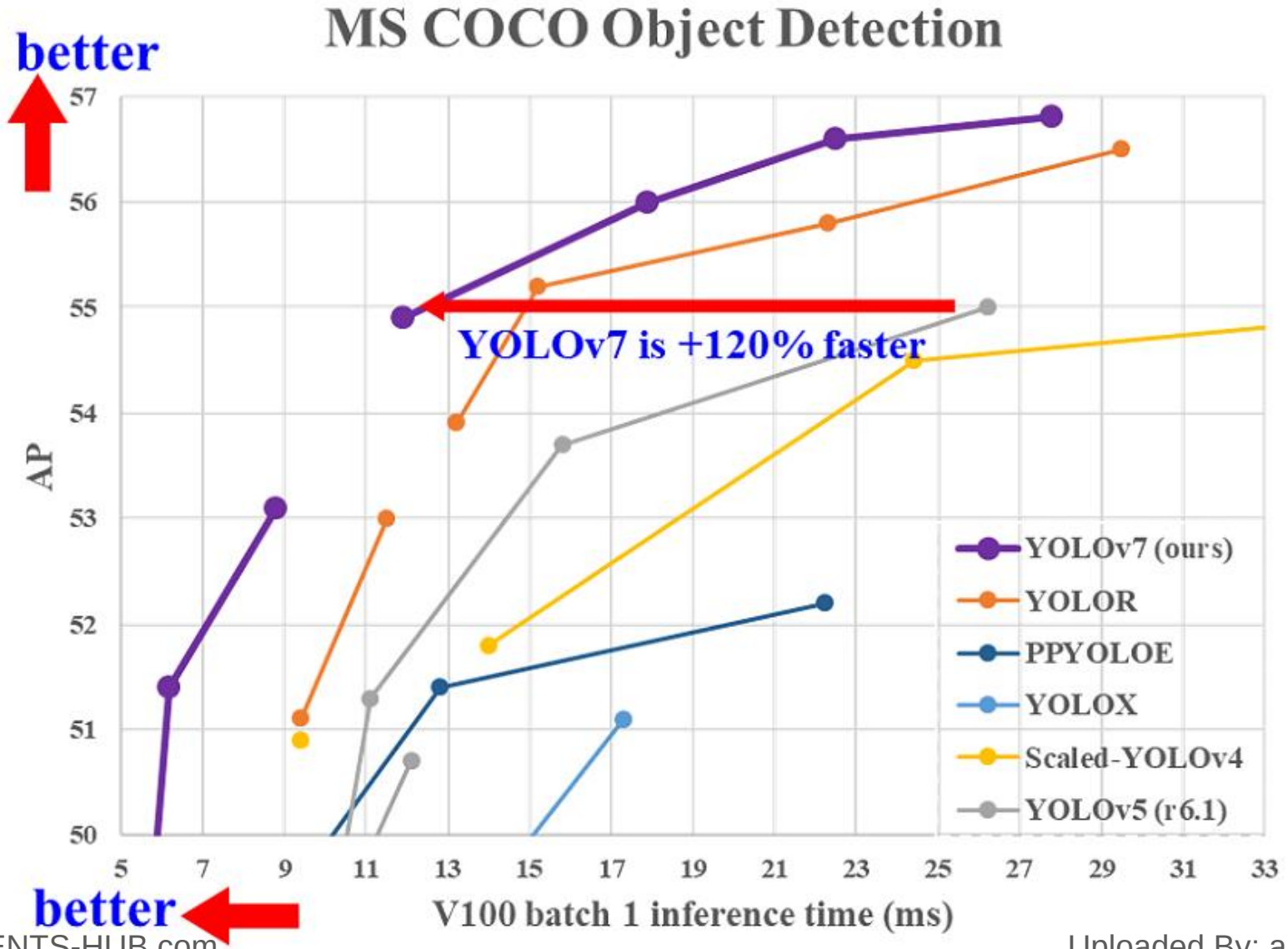
$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

Class loss, minimize loss between true class of object in grid box

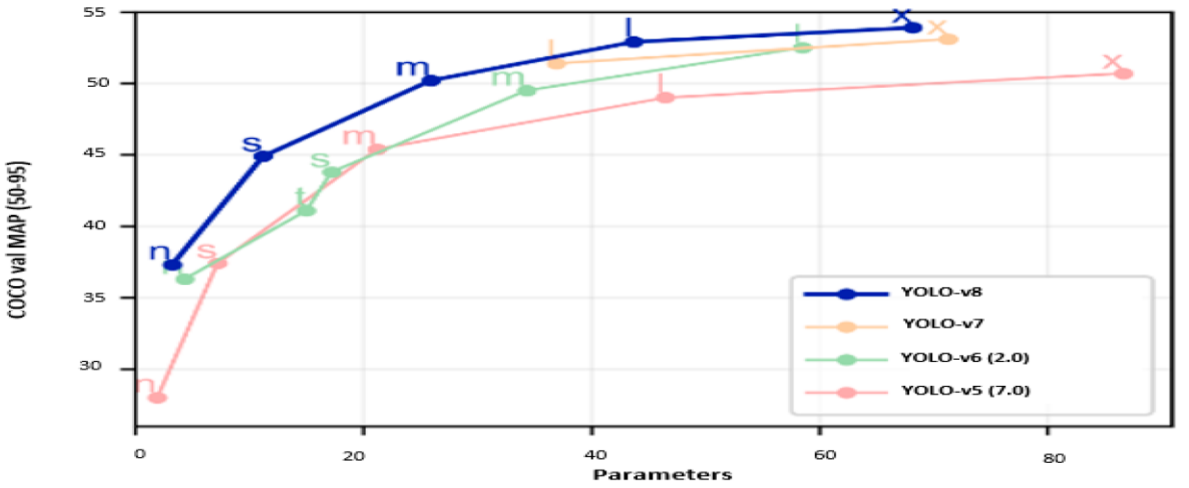
YOLO Generations

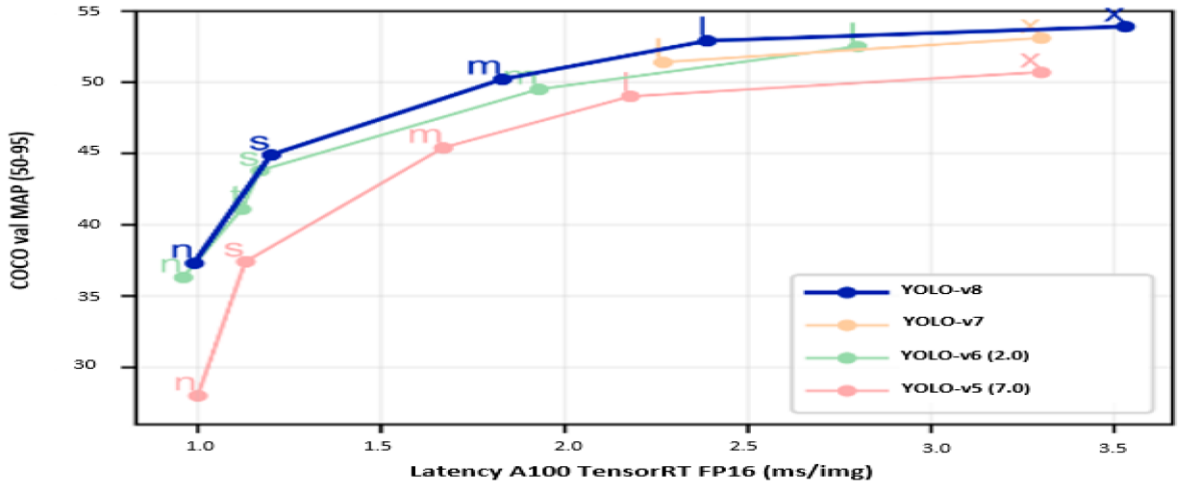


Some Comparisons



Some Comparisons





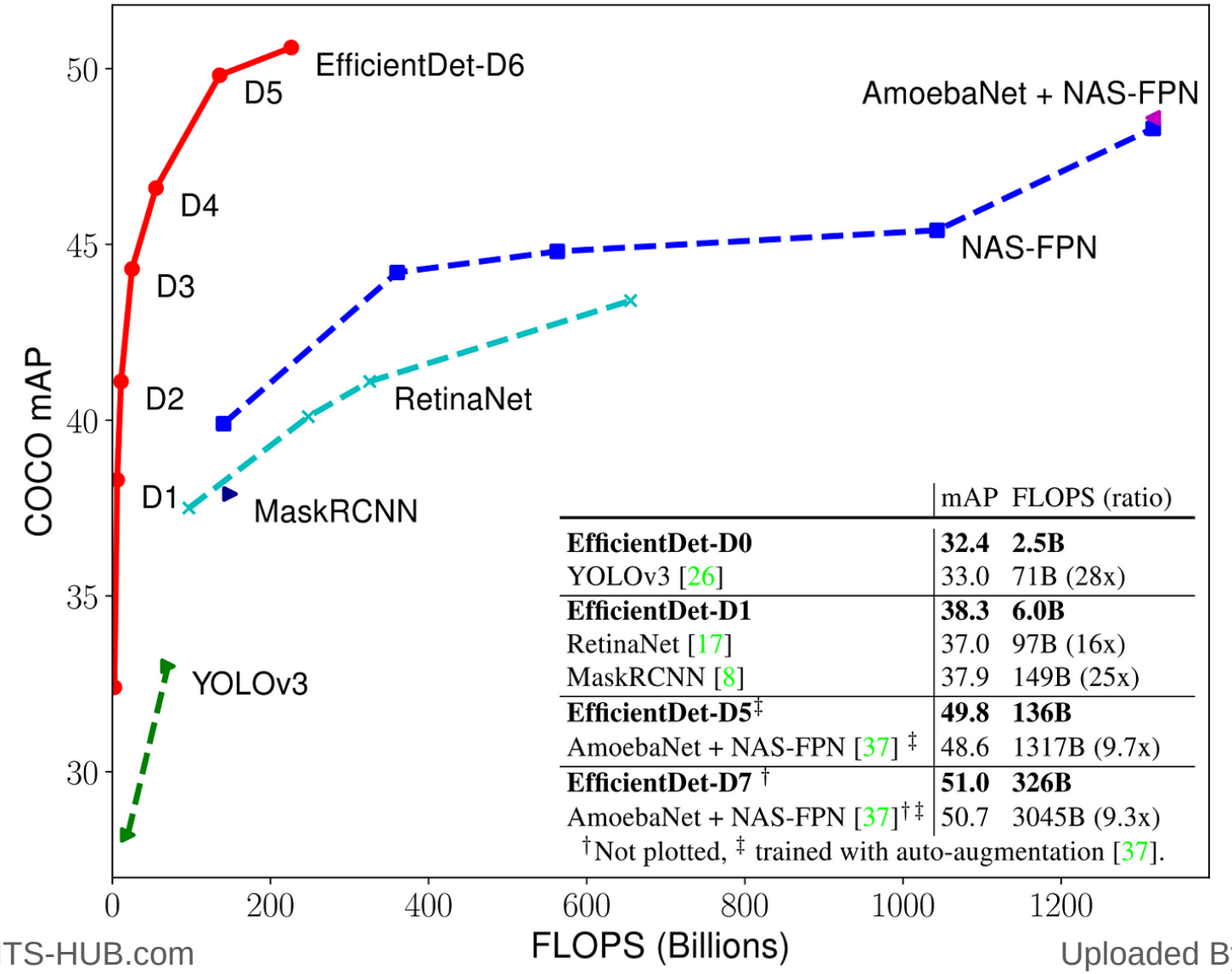
Outline

- Introduction
- Traditional Computer Vision Techniques:
  - Sliding Window:
  - Region proposal approach **(Selective Search)**
- Deep Learning-Based Techniques:
  - Two-Stage Detectors:
    - R-CNN (Regions with CNN features)
  - One-Stage Detectors:
    - YOLO (You Only Look Once)
- **Other Detectors**
- Evaluation Measures
- Benchmarks

Other Detectors

- **Mask R-CNN**
  - An extension of Faster R-CNN, Mask R-CNN,
  - Includes an additional branch for instance segmentation.
  - It simultaneously predicts object masks along with bounding boxes and class scores.
- **SSD (Single Shot Multibox Detector)**
  - Similar to YOLO, SSD is a single-shot object detector that predicts bounding boxes and class scores at multiple scales.
- **RetinaNet**
  - Addressing the issue of class imbalance in object detection
  - It is known for its strong performance on datasets with a large number of background samples.
- **EfficientDet**
  - EfficientDet focuses on balancing model efficiency and accuracy by using a compound scaling method to optimize the model's depth, width, and resolution.
- **Cascade R-CNN**
  - An extension of Faster R-CNN that aims to improve the accuracy of object detection by using a cascade structure.
  - It refines the bounding box predictions in a cascaded manner.
- **CenterNet**
  - Directly predicts object centers.
  - It achieves high accuracy with a single-stage architecture.

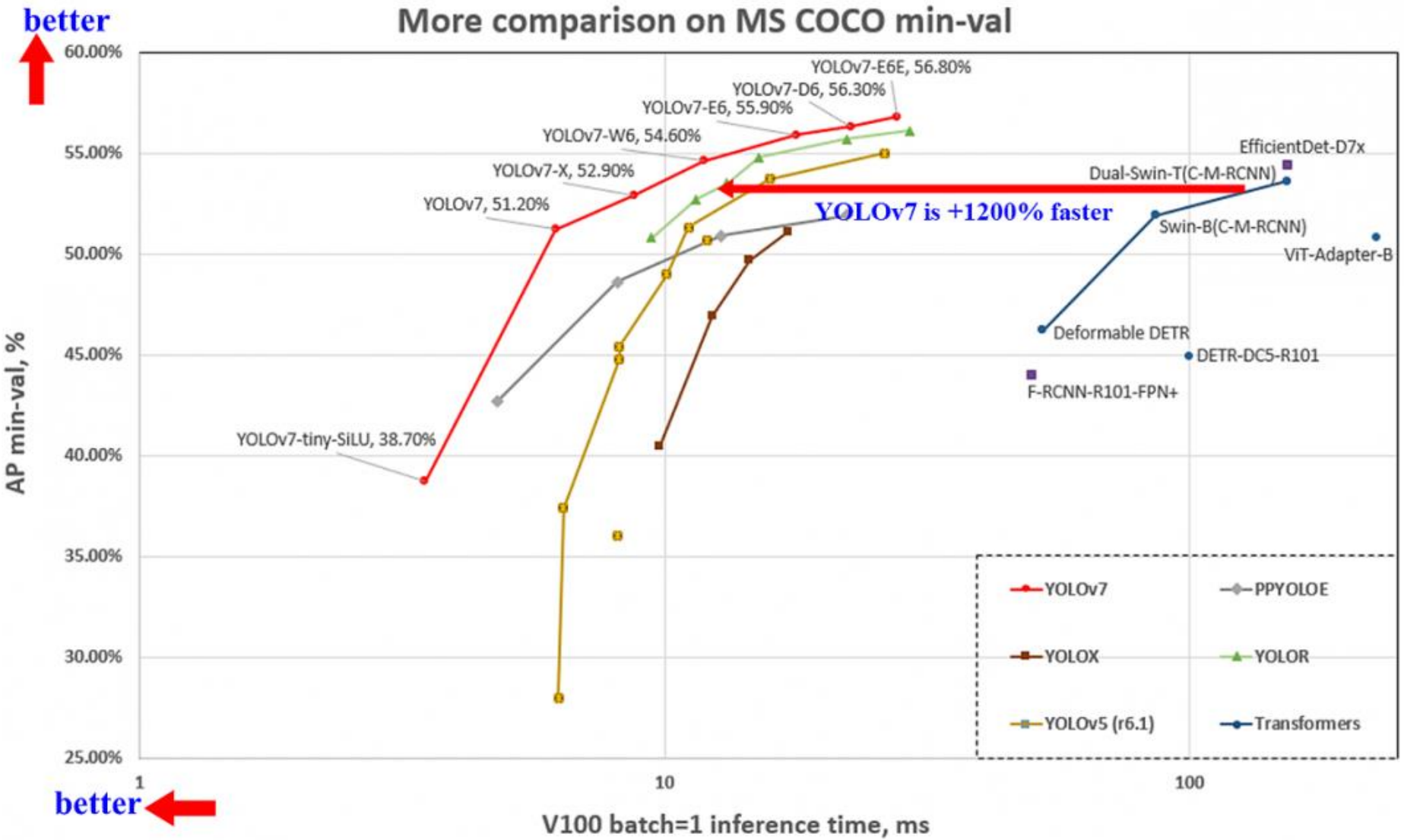
Detectors Performance



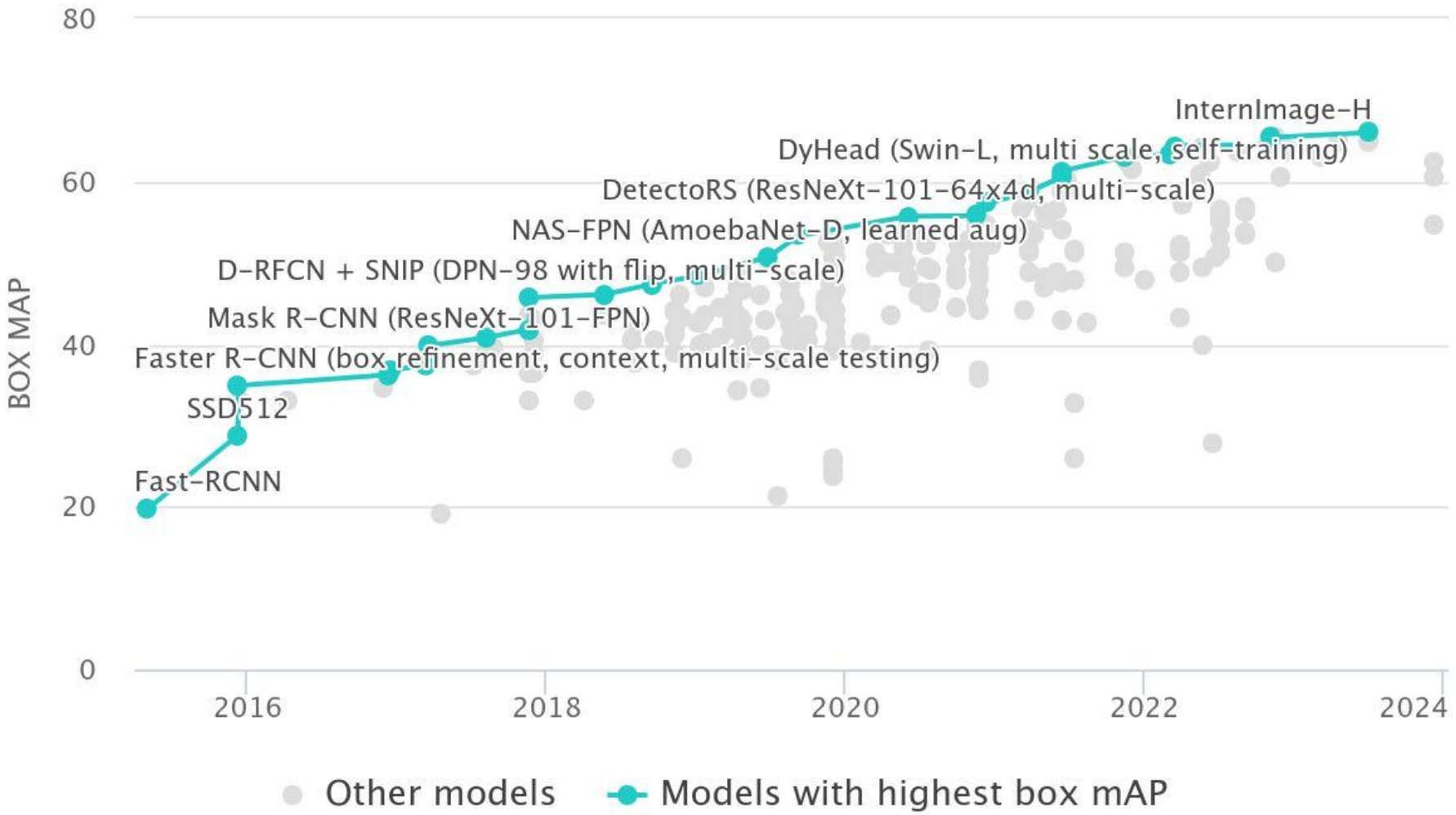
	mAP	FLOPS (ratio)
EfficientDet-D0	**32.4**	**2.5B**
YOLOv3 [26]	33.0	71B (28x)
EfficientDet-D1	**38.3**	**6.0B**
RetinaNet [17]	37.0	97B (16x)
MaskRCNN [8]	37.9	149B (25x)
EfficientDet-D5‡	**49.8**	**136B**
AmoebaNet + NAS-FPN [37] ‡	48.6	1317B (9.7x)
EfficientDet-D7 †	**51.0**	**326B**
AmoebaNet + NAS-FPN [37]†‡	50.7	3045B (9.3x)

†Not plotted, ‡ trained with auto-augmentation [37].

Detectors Performance



Detectors Performance



https://paperswithcode.com/sota/object-detection-on-coco

Outline

- Introduction
- Traditional Computer Vision Techniques:
  - Sliding Window:
  - Region proposal approach **(Selective Search)**
- Deep Learning-Based Techniques:
  - Two-Stage Detectors:
    - R-CNN (Regions with CNN features)
  - One-Stage Detectors:
    - YOLO (You Only Look Once)
- Other Detectors
- **Evaluation Measures**
- Benchmarks

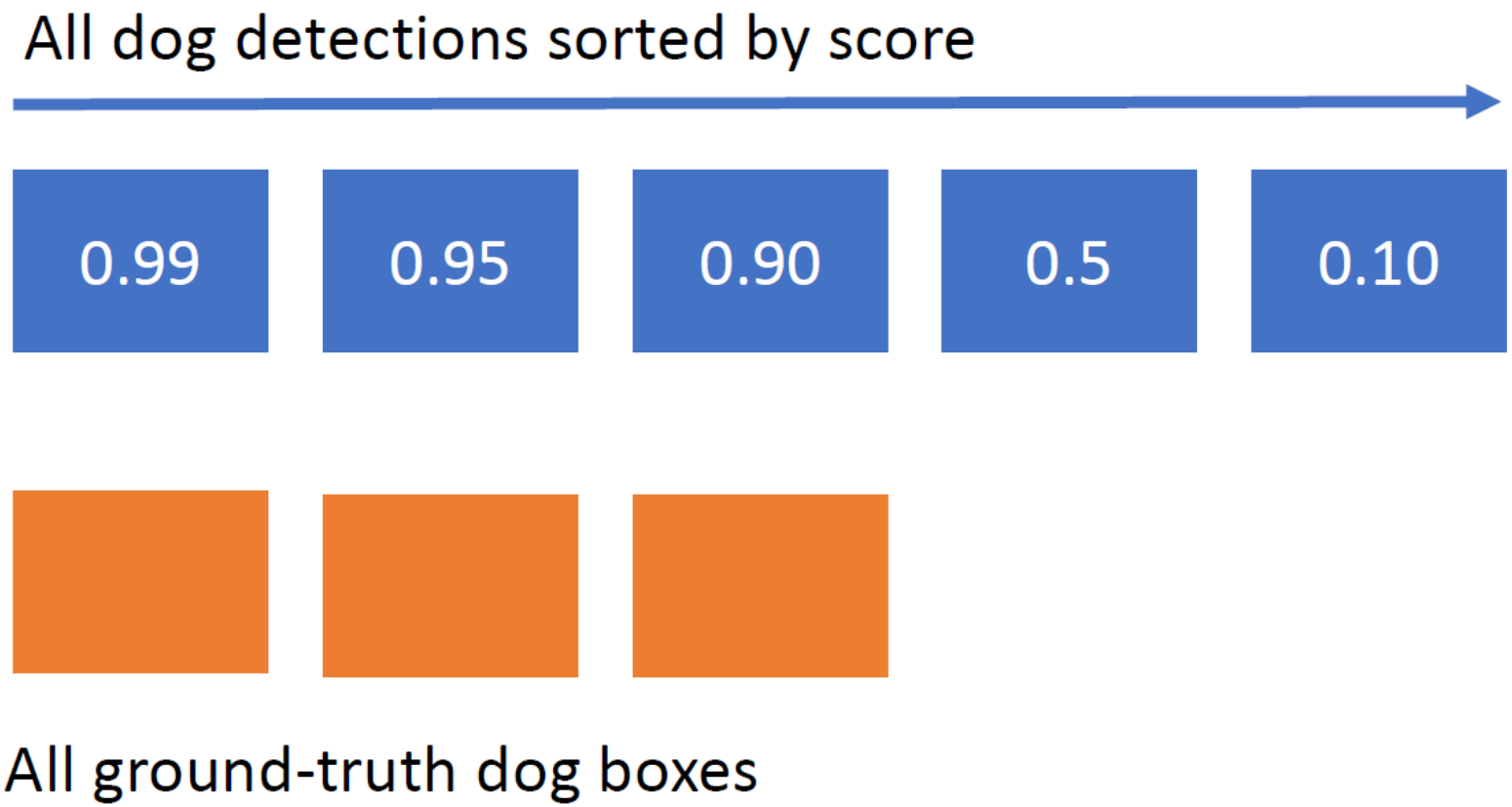
Evaluating Object Detectors: Mean Average Precision (mAP)

- Mean Average Precision (mAP) is a commonly used metric for evaluating the performance of object detectors in computer vision tasks, particularly in the context of object detection.
- mAP is calculated based on:
  - **Precision and Recall**
    - Precision: Precision is the ratio of true positive predictions to the total number of positive predictions (true positives + false positives). It measures the accuracy of positive predictions.
    - Recall: Recall is the ratio of true positive predictions to the total number of actual positive instances (true positives + false negatives). It measures the ability of the model to capture all positive instances.
  - **Intersection over Union (IoU)**
    - IoU is a measure of the overlap between the predicted bounding box and the ground truth bounding box.
  - **Precision-Recall Curve**
    - The precision-recall curve is generated by varying the confidence threshold of the model and calculating precision and recall at each threshold.
  - **Average Precision (AP)**
    - AP is calculated by computing the area under the precision-recall curve. It represents the average precision across all recall levels.
  - **Mean Average Precision (mAP)**
    - mAP is the mean of the average precision values calculated for each class. It provides an overall assessment of the detector's performance across different object categories.

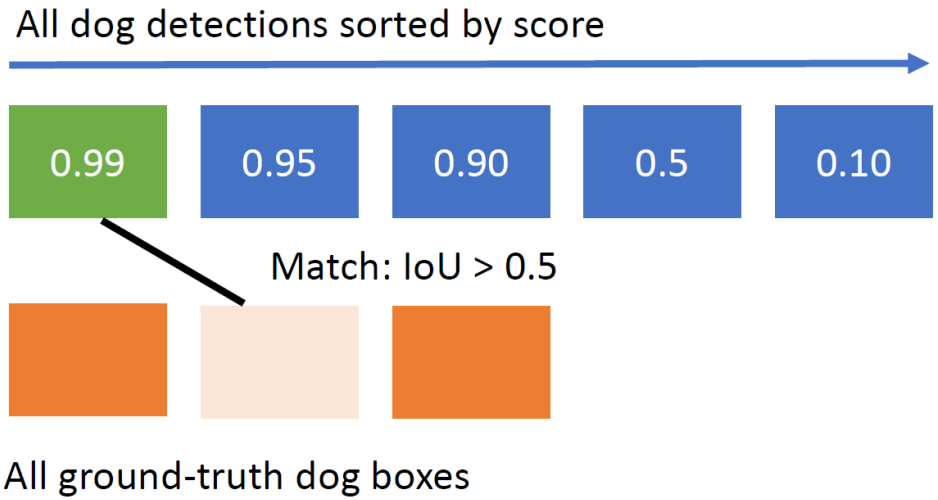
mAP: Step-by-step

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

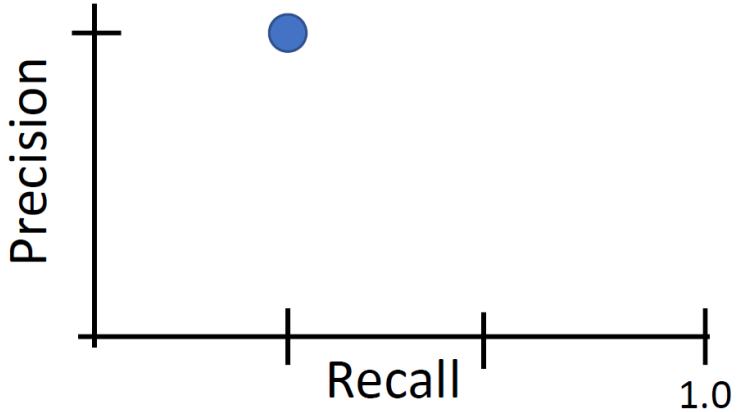
Evaluating Object Detectors: Mean Average Precision (mAP)



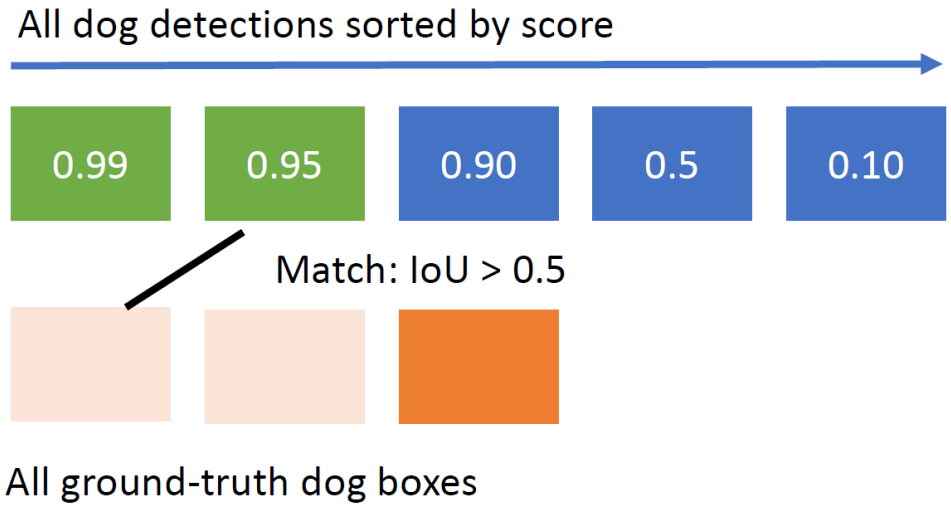
Evaluating Object Detectors: Mean Average Precision (mAP)



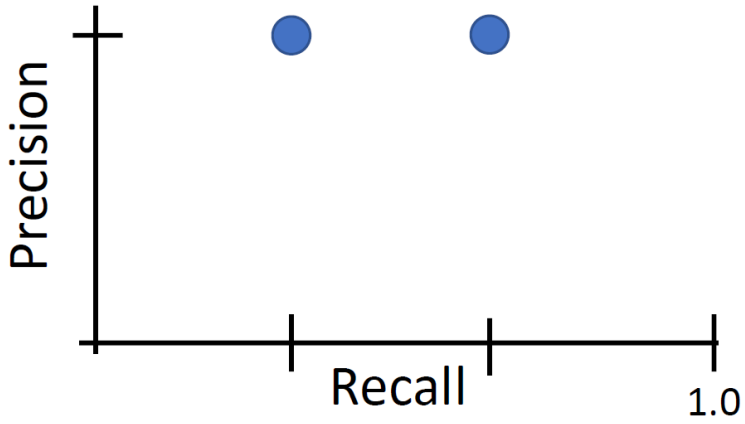
Precision = 1/1 = 1.0
Recall = 1/3 = 0.33



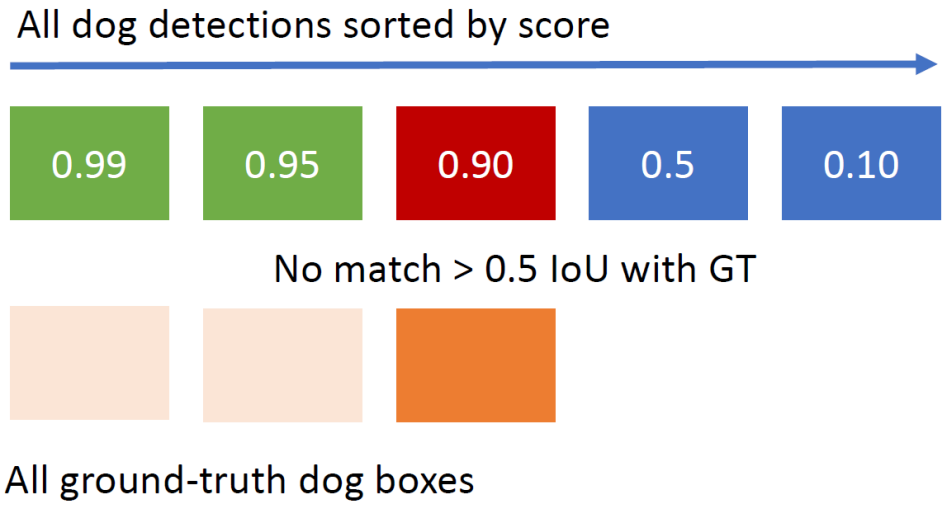
Evaluating Object Detectors: Mean Average Precision (mAP)



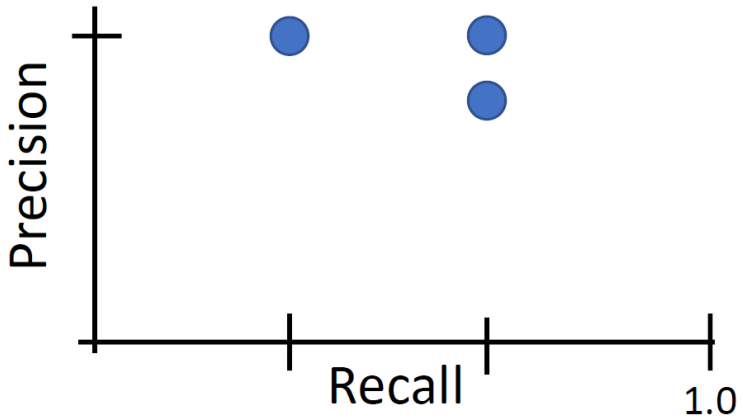
Precision = 2/2 = 1.0
Recall = 2/3 = 0.67



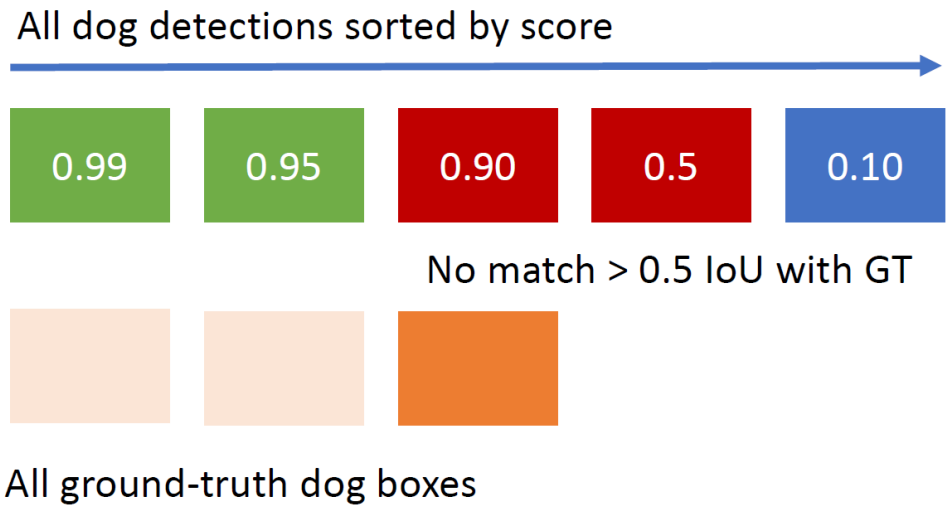
Evaluating Object Detectors: Mean Average Precision (mAP)



Precision = 2/3 = 0.67
Recall = 2/3 = 0.67

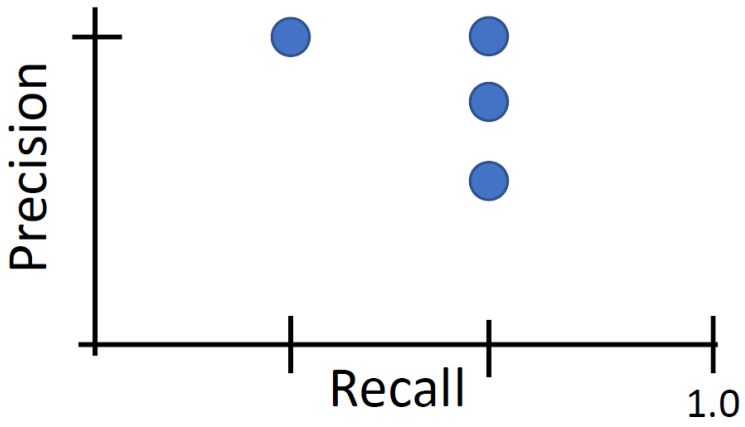


Evaluating Object Detectors: Mean Average Precision (mAP)

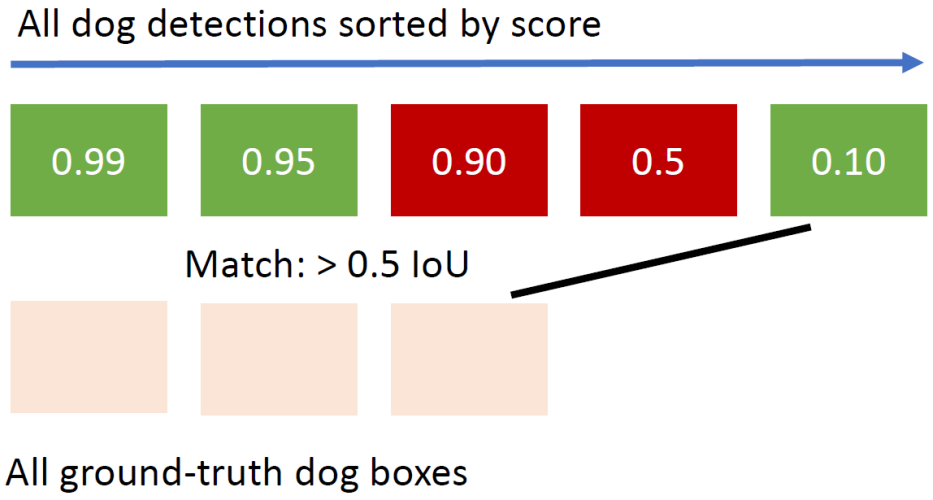


Precision = 2/4 = 0.5

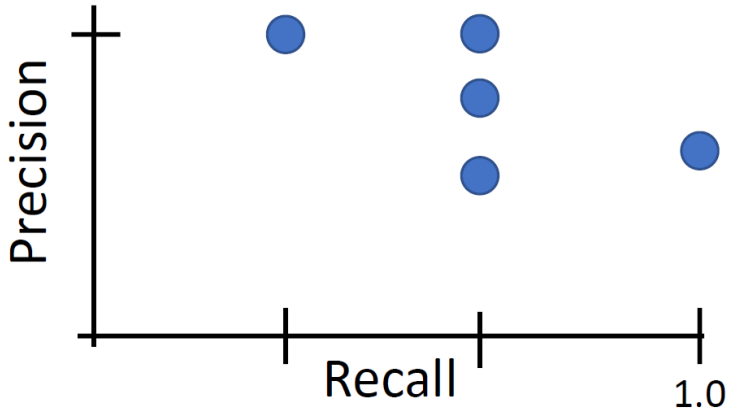
Recall = 2/3 = 0.67



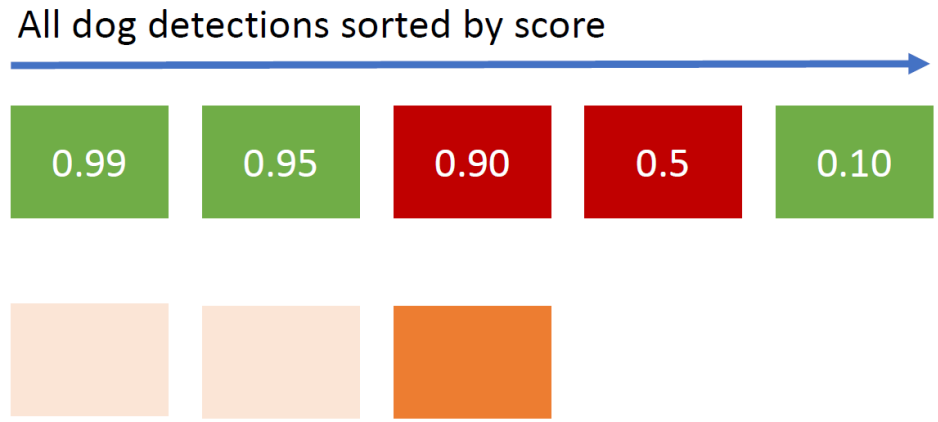
Evaluating Object Detectors: Mean Average Precision (mAP)

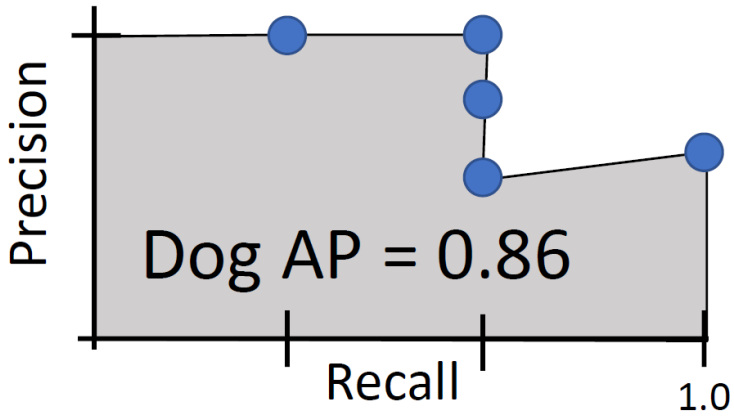


Precision = 3/5 = 0.6
Recall = 3/3 = 1.0



Evaluating Object Detectors: Mean Average Precision (mAP)





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

Car AP = 0.65

Cat AP = 0.80

Dog AP = 0.86

mAP@0.5 = 0.77

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

mAP@0.5 = 0.77
mAP@0.55 = 0.71
mAP@0.60 = 0.65
…
mAP@0.95 = 0.2

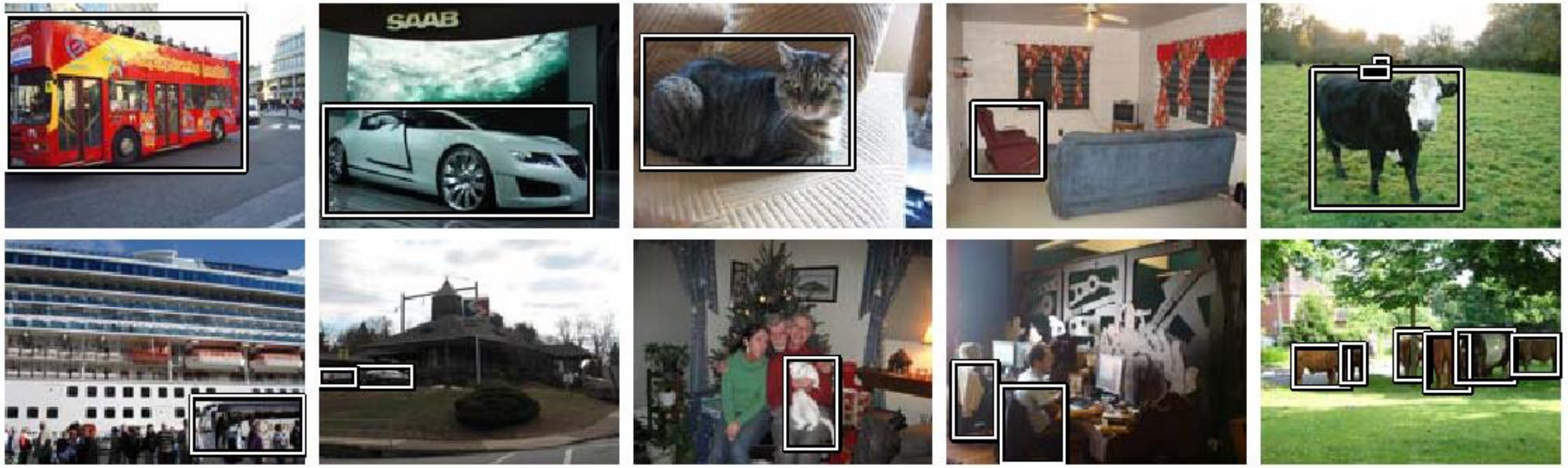
COCO mAP = 0.4

Object Detection Benchmarks

- MS COCO (Microsoft Common Objects in Context)
  - One of the largest and most widely used object detection benchmarks.
  - Consists of 328,000 images with annotations for 80 object categories.
  - Provides evaluation metrics such as Average Precision (AP) and mean Average Precision (mAP).

Object Detection Benchmarks

- PASCAL VOC Challenge
  - This benchmark is older than COCO, but it is still widely used.
  - Consists of about 20,000 images with annotations for 20 object categories.
  - Evaluation metrics include mAP.



Object Detection Benchmarks

- **KITTI Vision Benchmark Suite**
  - This benchmark focuses on object detection in autonomous driving scenarios.
  - Consists of 80,000 images with annotations for 3 object categories (cars, pedestrians, and cyclists).
  - Provides metrics like precision, recall, and F1 score.

Acknowledgement

- The material in these slides are based on:
  - Digital Image Processing: Rafael C. Gonzalez, and Richard
  - Forsythe and Ponce: Computer Vision: A Modern Approach
  - Rick Szeliski's book: Computer Vision: Algorithms and Applications
  - cs131@ Stanford University
  - cs131n@ Stanford University
  - CS198-126@ University of California, Berkely
  - CAP5415@ University of Central Florida
  - CSW182 @ University of California, Berkely
  - Deep Learning Lecture Series @UCL
  - EECS 498.008 @ University of Michigan
  - CSE576 @ Washington University
  - 11-785@ Carnegie Mellon University
  - CSCI1430@ Brown University
  - Computer Vision@ Bonn University
  - ICS 505@ KFUPM
  - Digital Image Processing@ University of Jordan