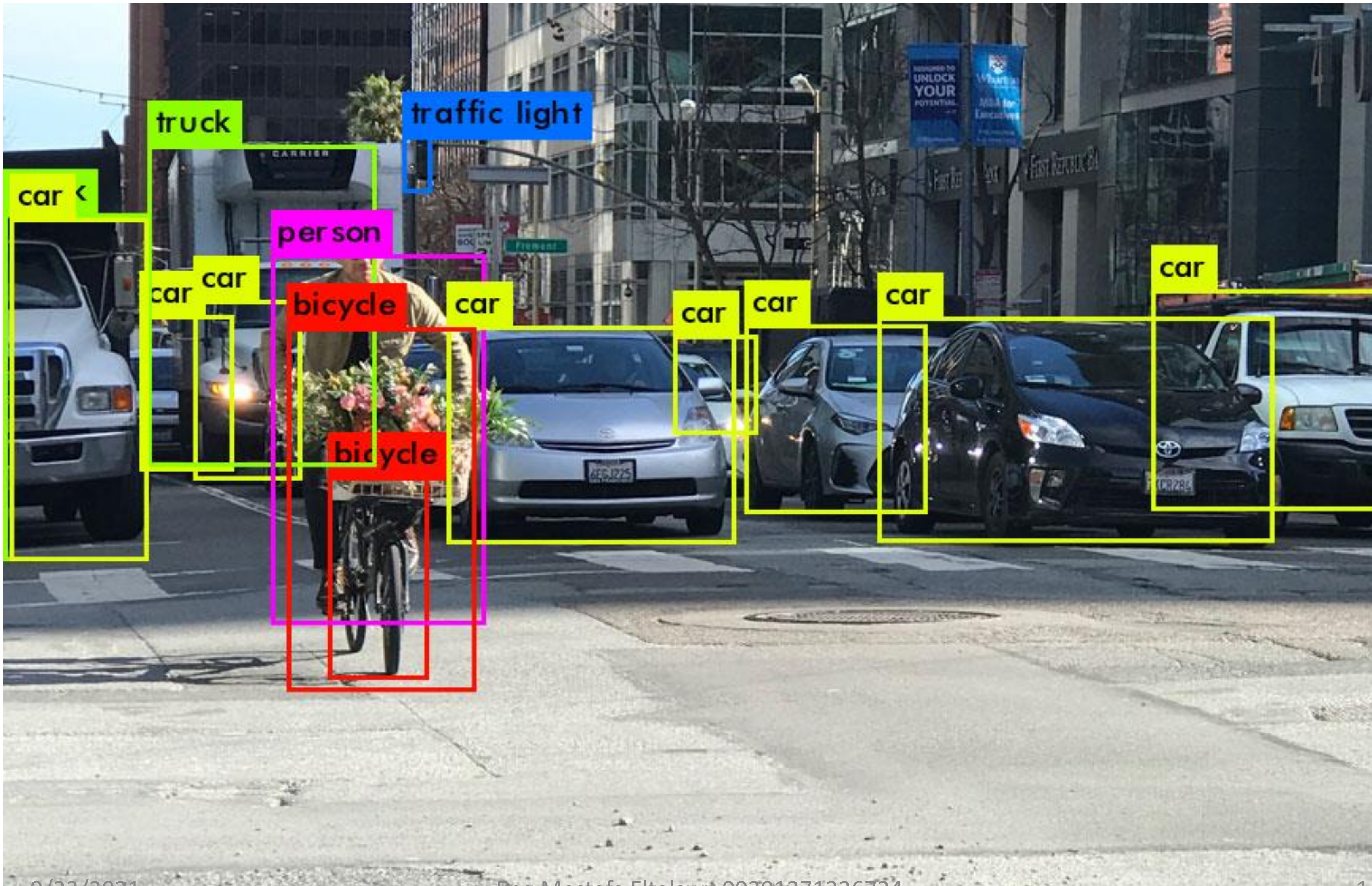
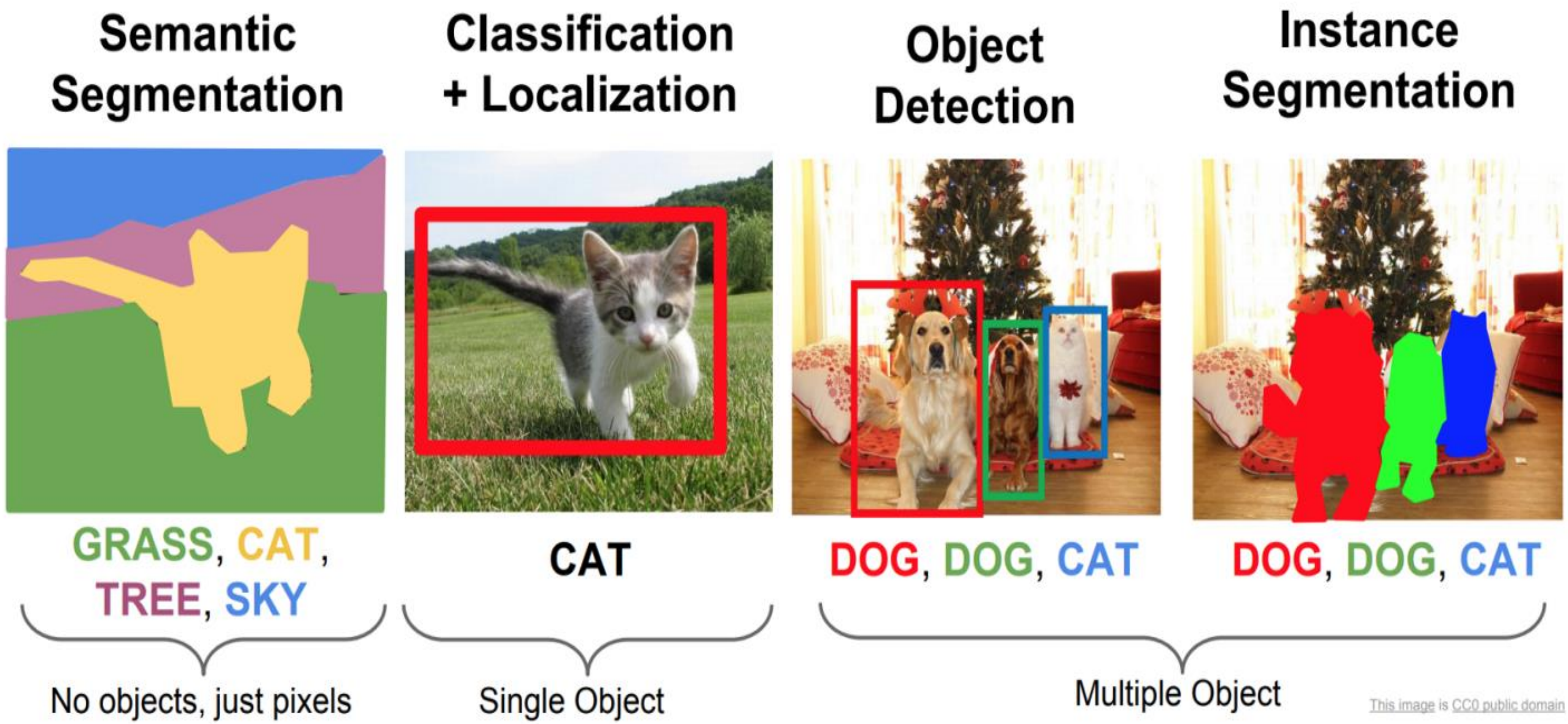
Object Detection



Introduction to object detection

- Object detection is a phenomenon in computer vision that involves the detection of various objects in digital images or videos. Some of the objects detected include people, cars, chairs, stones, buildings, and animals.
- This phenomenon seeks to answer two basic questions:
  - *What is the object?* This question seeks to identify the object in a specific image.
  - *Where is it?* This question seeks to establish the exact location of the object within the image.


Semantic Segmentation
Classification + Localization
Object Detection
Instance Segmentation
GRASS, CAT, TREE, SKY
CAT
DOG, DOG, CAT
DOG, DOG, CAT
No objects, just pixels
Single Object
Multiple Object
This image is CC0 public domain

Object Detection Algorithms

- Fast R-CNN
- Faster R-CNN
- Histogram of Oriented Gradients (HOG)
- Region-based Convolutional Neural Networks (R-CNN)
- Region-based Fully Convolutional Network (R-FCN)
- Single Shot Detector (SSD)
- Spatial Pyramid Pooling (SPP-net)
- YOLO (You Only Look Once)

YOLO (You Only Look Once)

- This is an algorithm that detects and recognizes various objects in a picture
- Object detection in YOLO is done as a regression problem and provides the class probabilities of the detected images.
- The original YOLO (You Only Look Once) was written by Joseph Redmon in a custom framework called Darknet. Darknet is a very flexible research framework written in low level languages and has produced a series of the best realtime object detectors in computer vision: YOLO, YOLOv2, YOLOv3, and now, YOLOv4.

How the YOLO algorithm works

- YOLO algorithm works using the following three techniques:
  - Residual blocks
  - Bounding box regression
  - Intersection Over Union (IOU)

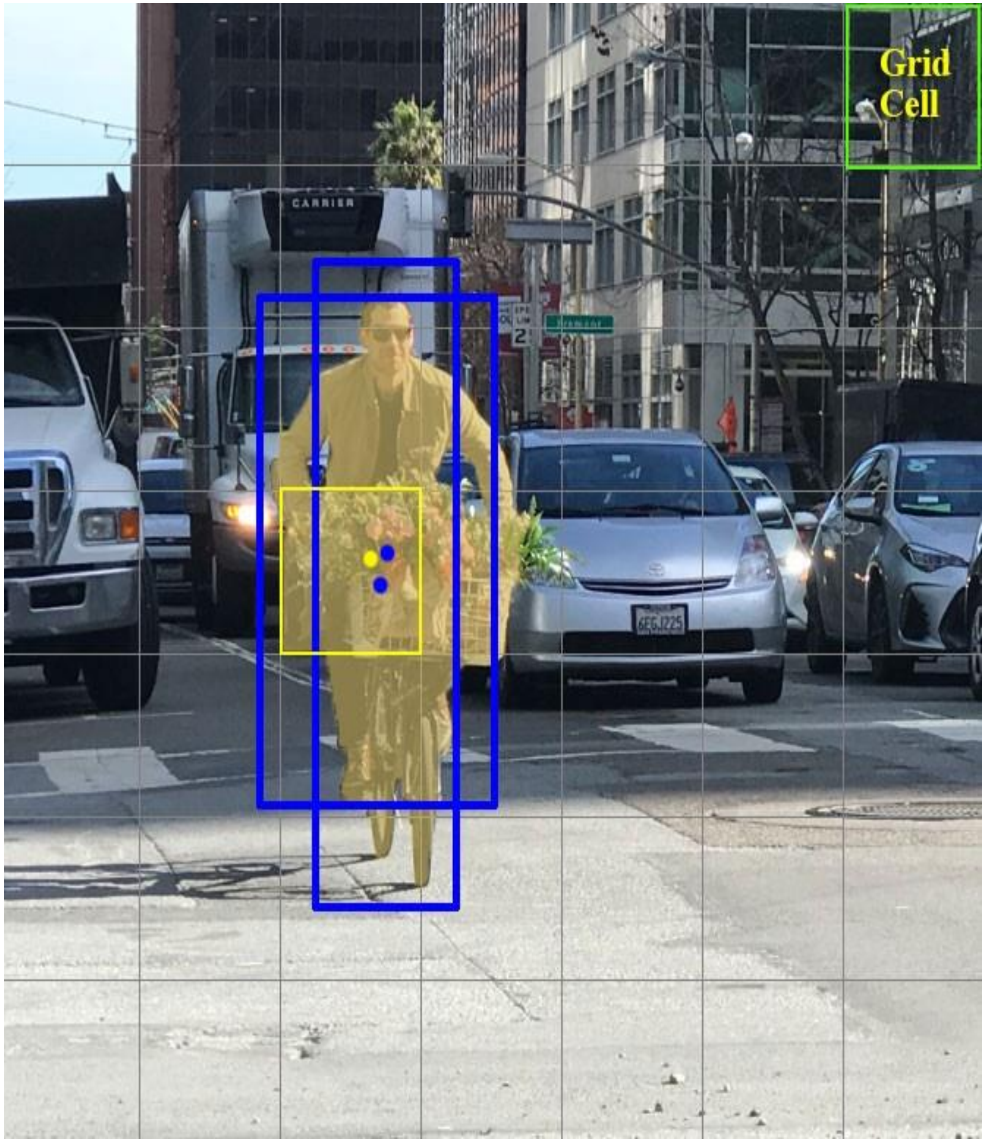
Residual blocks

- the image is divided into various grids. Each grid has a dimension of S x S.

- Every grid cell will detect objects that appear within them. For example, if an object center appears within a certain grid cell, then this cell will be responsible for detecting it.

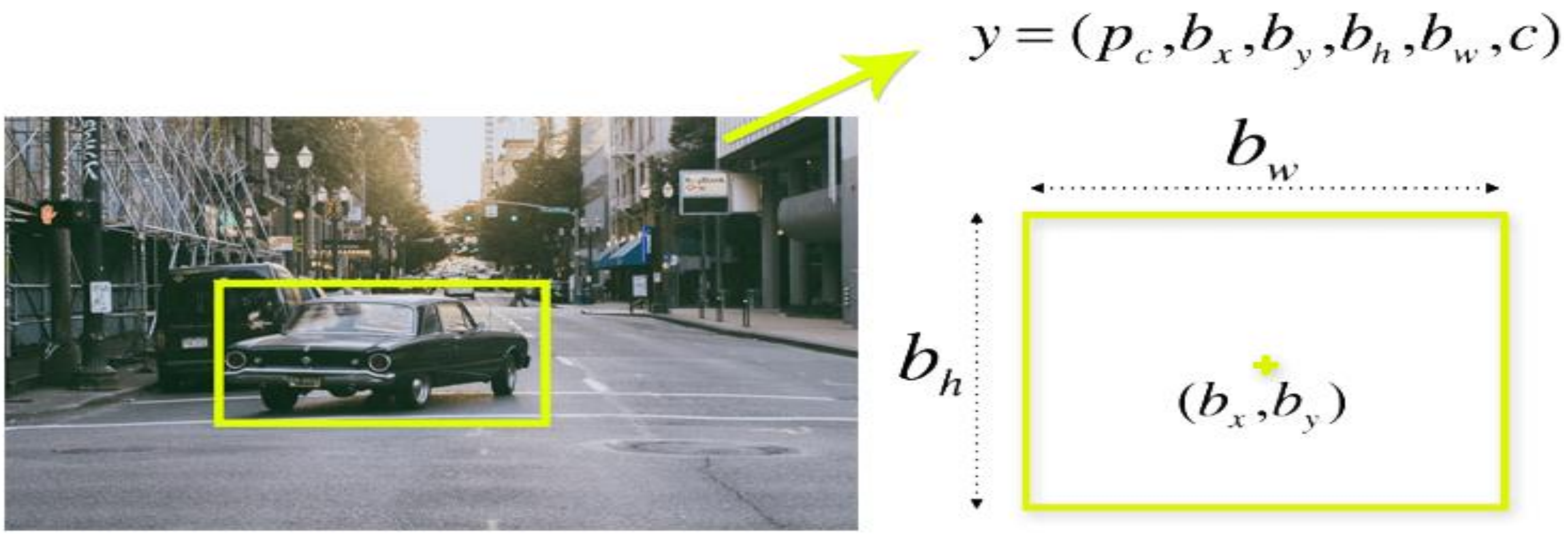
Grid cell

- For each grid cell,it predicts **B** boundary boxes and each box has one **box confidence score**,
- it detects **one** object only regardless of the number of boxes B,
- it predicts **C conditional class probabilities** (one per class for the likeliness of the object class).



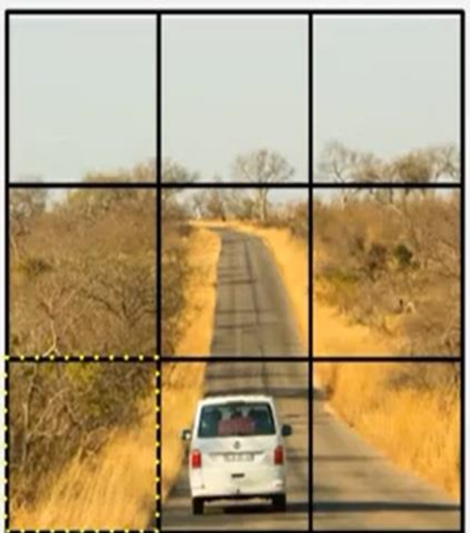
Bounding box regression

- A bounding box is an outline that highlights an object in an image.
- Every bounding box in the image consists of the following attributes:
  - Width (bw)
  - Height (bh)
  - Class (for example, person, car, etc.)- This is represented by the letter c.
  - Bounding box center (bx,by)
- YOLO uses a single bounding box regression to predict the height, width, center, and class of objects.



YOLO Steps

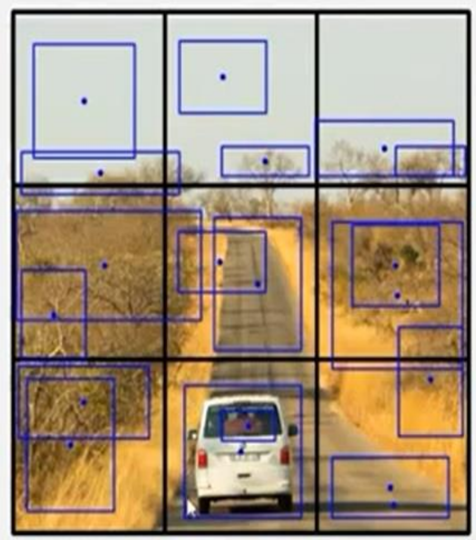
1. Divide the image into cells with an S x S grid.



$S = 3$

Cell

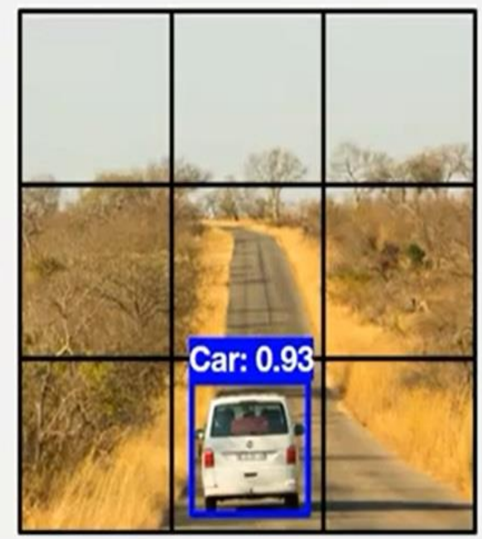
2. Each cell predicts B bounding boxes.



$B = 2$

A cell is responsible for detecting an object if the object's bounding box falls within the cell. (Notice that each cell has 2 blue dots.)

3. Return bounding boxes above confidence threshold.



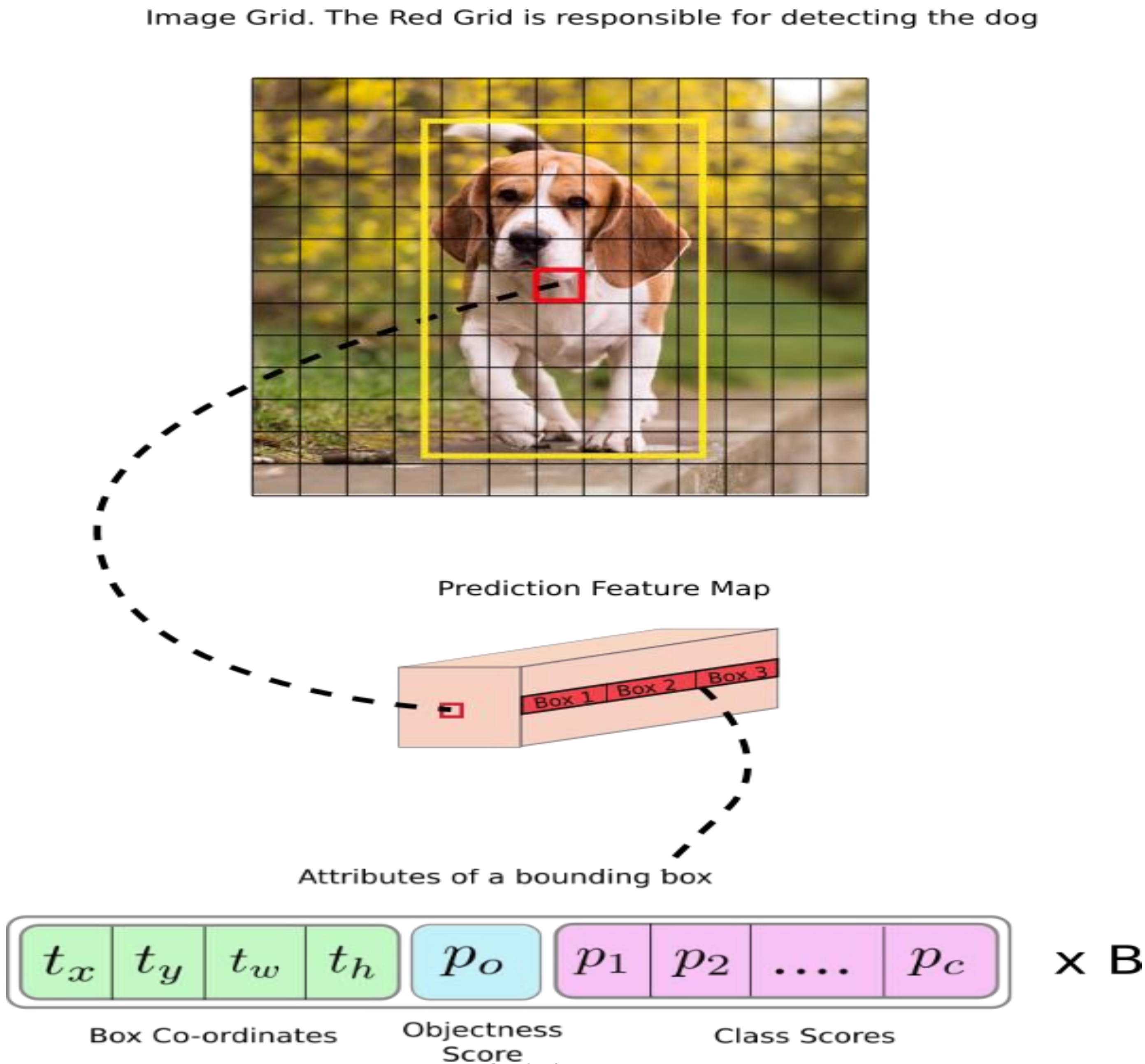
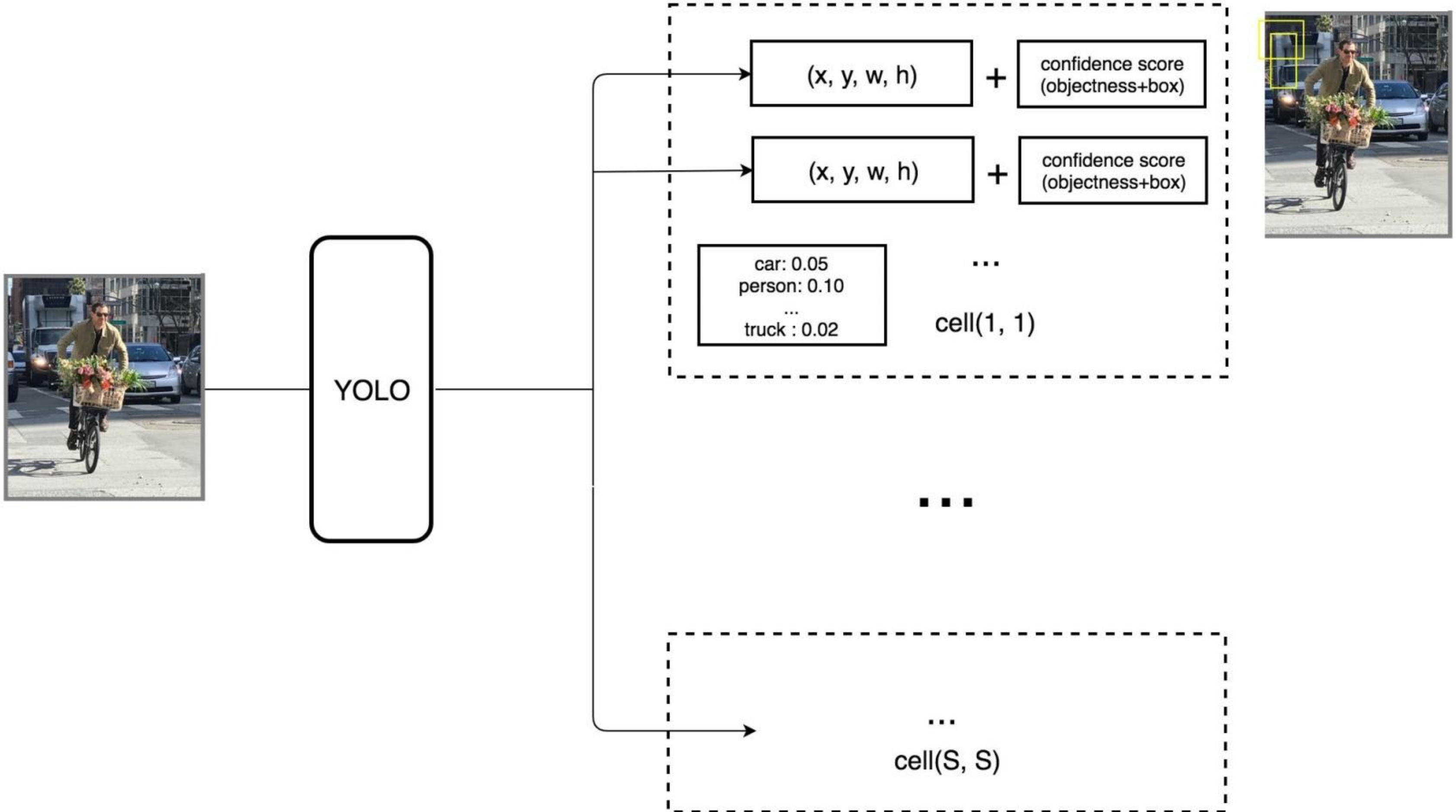
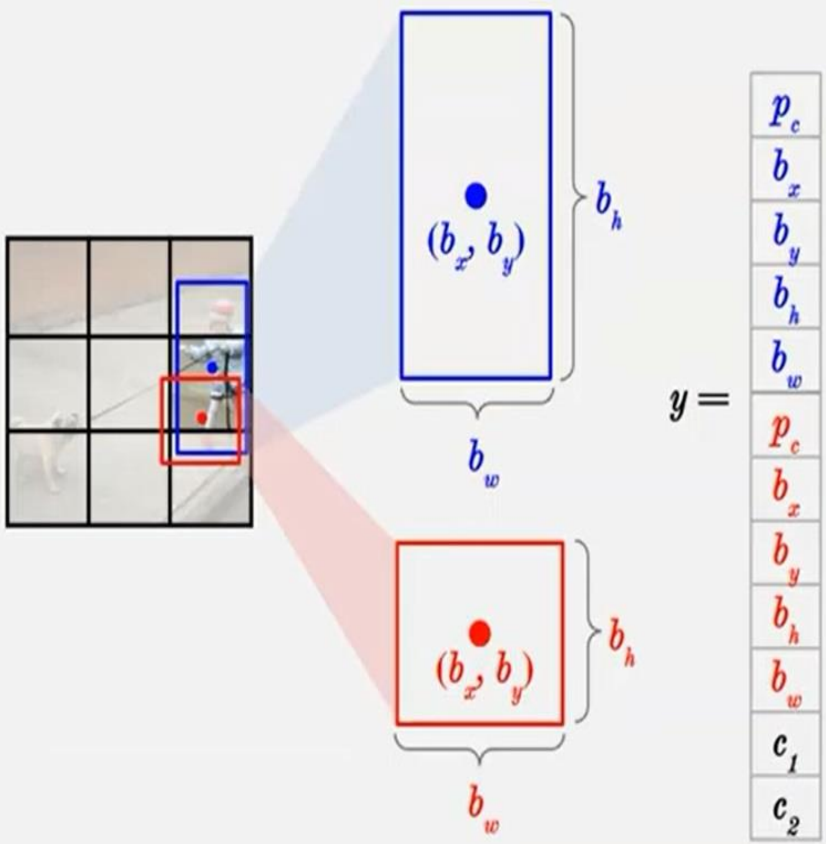


Image Grid. The Red Grid is responsible for detecting the dog
Prediction Feature Map
Box 1
Box 2
Box 3
Attributes of a bounding box
t_x t_y t_w t_h
p_o
p_1 p_2 .... p_c
x B
Box Co-ordinates
Objectness Score
Class Scores

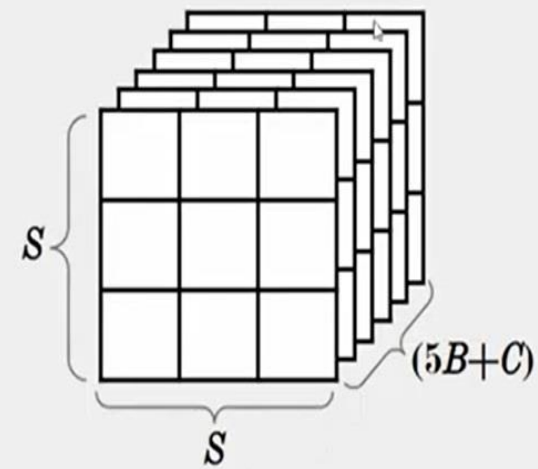

YOLO
(x, y, w, h)
+
confidence score
(objectness+box)
(x, y, w, h)
+
confidence score
(objectness+box)
car: 0.05
person: 0.10
...
truck : 0.02
...
cell(1, 1)
...
...
cell(S, S)

Encoding Multiple Bounding Boxes

What happens if we predict multiple bounding boxes per cell ($B>1$)? We simply augment y.



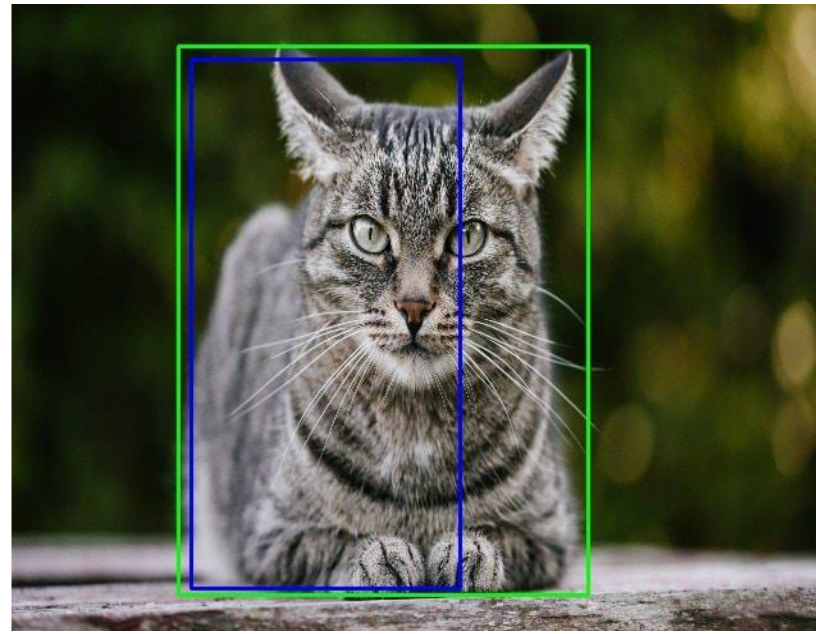
The CNN will predict a y for each cell, so the size of the output tensor (multidimensional "matrix") should be: $S \times S \times (5B+C)$

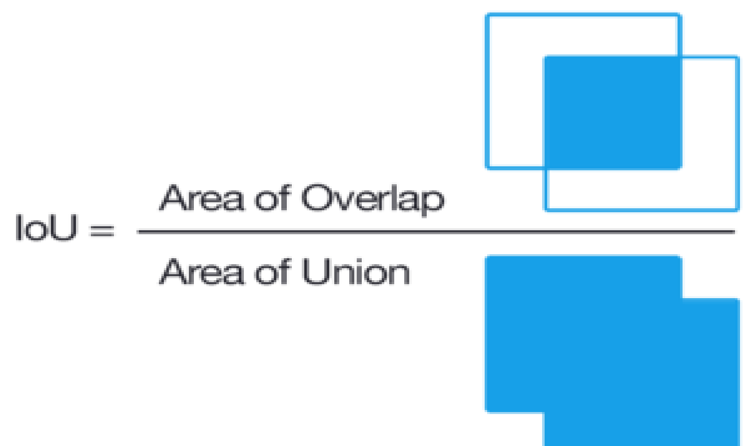


Notice that y has $5B+C$ elements (C is the number of classes).

Intersection over union (IOU)

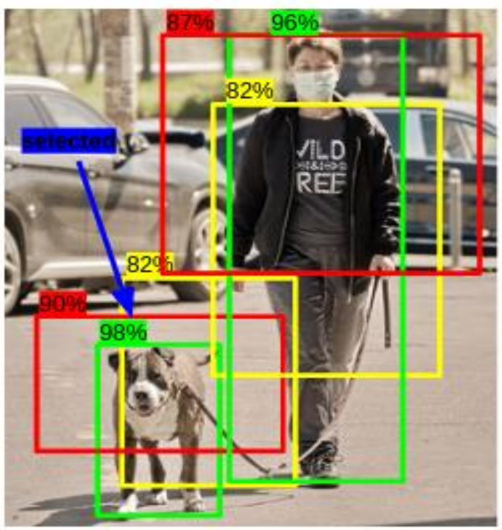
- It describes how boxes overlap. YOLO uses IOU to provide an output box that surrounds the objects perfectly.
- Each grid cell is responsible for predicting the bounding boxes and their confidence scores. The IOU is equal to 1 if the predicted bounding box is the same as the real box. This mechanism eliminates bounding boxes that are not equal to the real box.



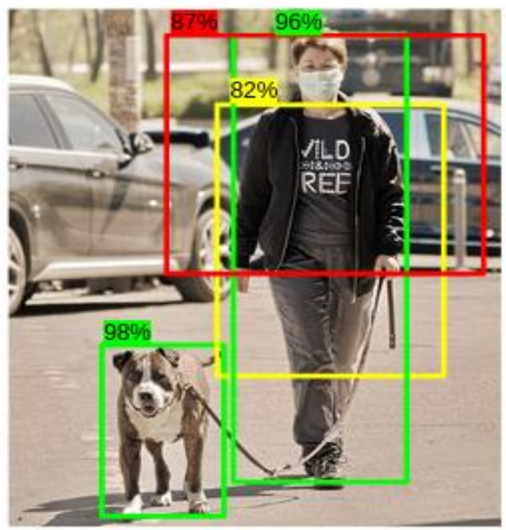


Non-maximal suppression

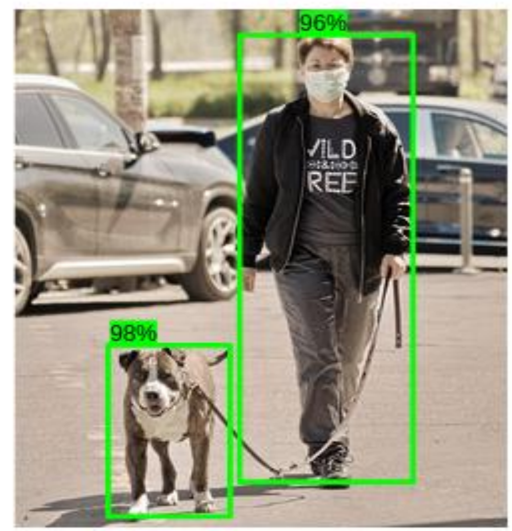
- Non-max suppression is the final step of these object detection algorithms and is used to select the most appropriate bounding box for the object.
- YOLO can make duplicate detections for the same object. To fix this, YOLO applies non-maximal suppression to remove duplications with lower confidence.



Step 1: Selecting Bounding box with highest score

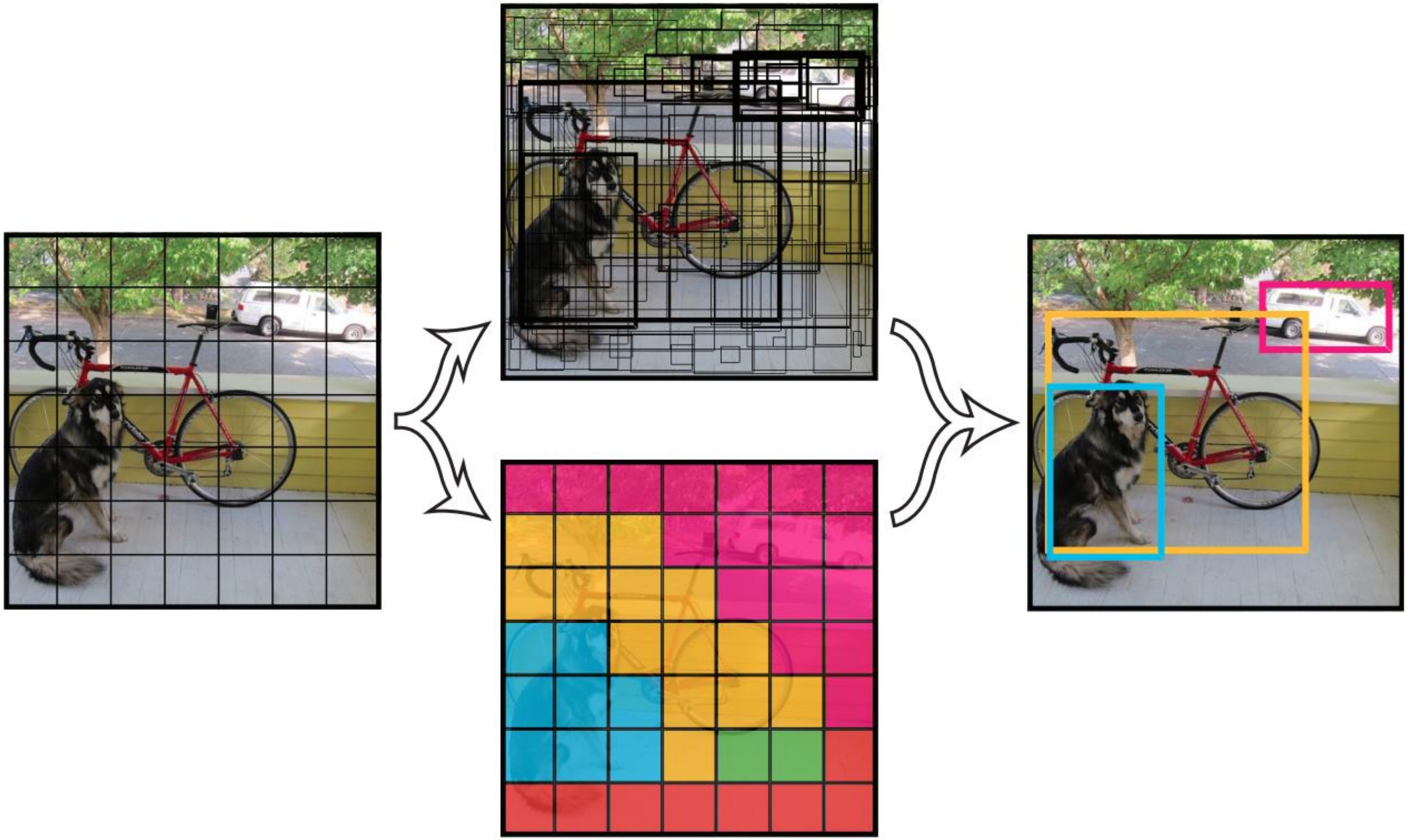


Step 3: Delete Bounding box with high overlap



Step 5: Final Output

Combination of the three techniques



Region-based Convolutional Neural Network

- R-CNN stands for Region-based Convolutional Neural Network.
- The key concept behind the R-CNN series is region proposals.
- Region proposals are used to localize objects within an image.

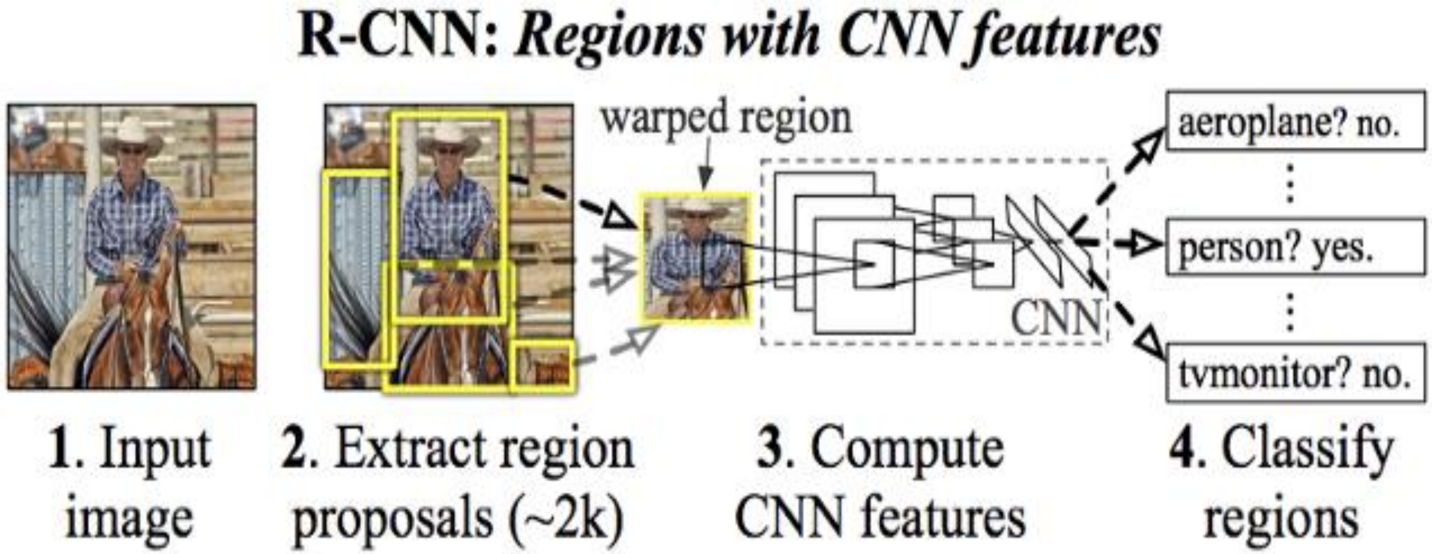
Models for object detection using regions with CNNs are based on the following three processes:

- Find regions in the image that might contain an object. These regions are called *region proposals*.
- Extract CNN features from the region proposals.
- Classify the objects using the extracted features.

Note:

There are three variants of an R-CNN.

Each variant attempts to optimize, speed up, or enhance the results of one or more of these processes.



Before passing an image through a network, we need to extract region proposals or regions of interest using an algorithm such as selective search. Then, we need to resize (wrap) all the extracted crops and pass them through a network.
Finally, a network assigns a category from C + 1, including the 'background' label, categories for a given crop.

Extract region proposals

- Selective Search is a region proposal algorithm used for object localization that groups regions together based on their pixel intensities.
- So, it groups pixels based on the hierarchical grouping of similar pixels.

Region proposal label

- After we extract our region proposal, we also have to label them for training.
- Therefore, we label all the proposals having IOU of at least 0.5 with any of the ground-truth bounding boxes with their corresponding classes.
- However, all other region proposals that have an IOU of less than 0.3 are labeled as background. Thus, the rest of them are simply ignored.

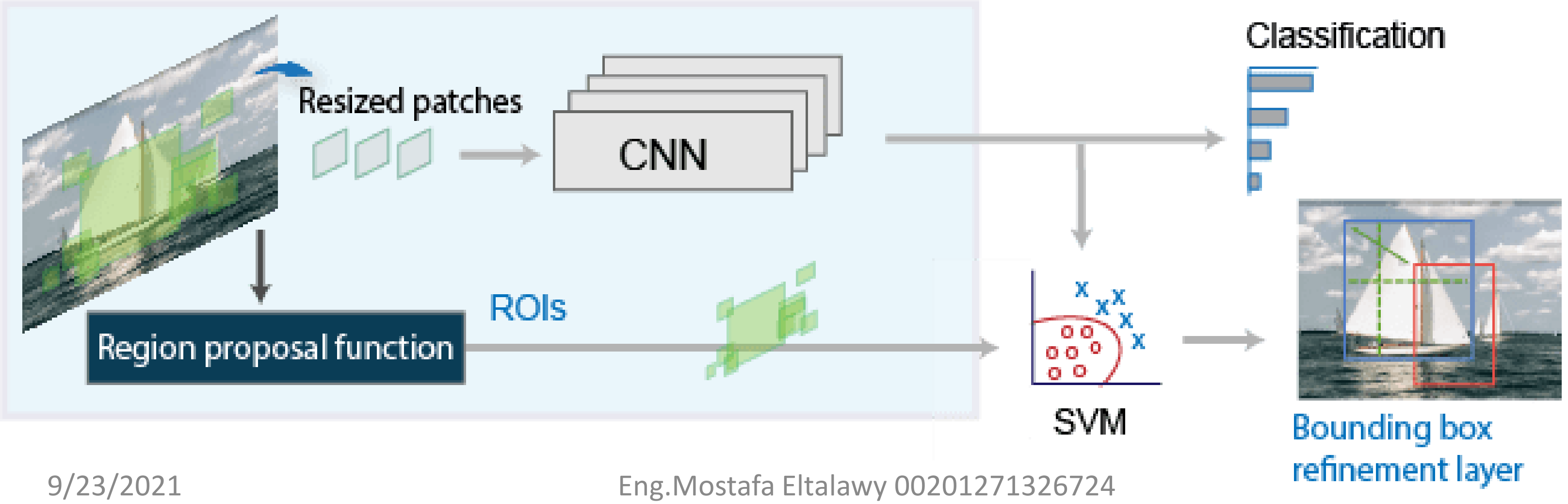
Bounding-box regression

- x, y are center coordinates. whereas w, h are width and height respectively.
- G and P stand for ground-truth bounding box and region proposal respectively.
- It is important to note that the bounding box loss is only calculated for positive samples.

$$t_x = (G_x - P_x)/P_w \quad (6)$$
$$t_y = (G_y - P_y)/P_h \quad (7)$$
$$t_w = \log(G_w/P_w) \quad (8)$$
$$t_h = \log(G_h/P_h). \quad (9)$$

R-CNN

- The R-CNN ,first generates region proposals using an algorithm
- The proposal regions are cropped out of the image and resized.
- Then, the CNN classifies the cropped and resized regions.
- Finally, the region proposal bounding boxes are refined by a support vector machine (SVM) that is trained using CNN features.



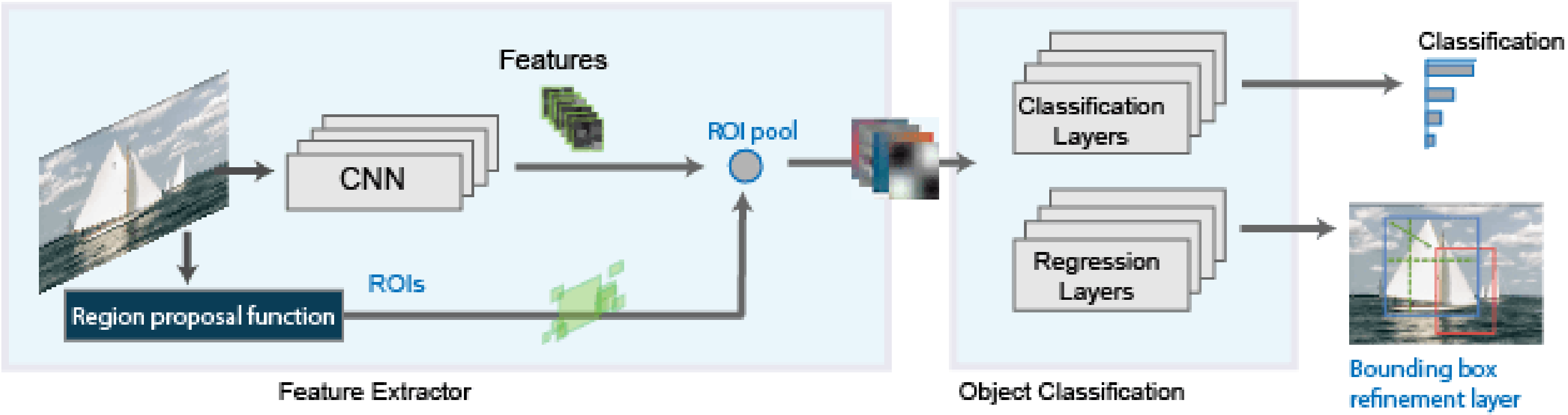
Model Workflow

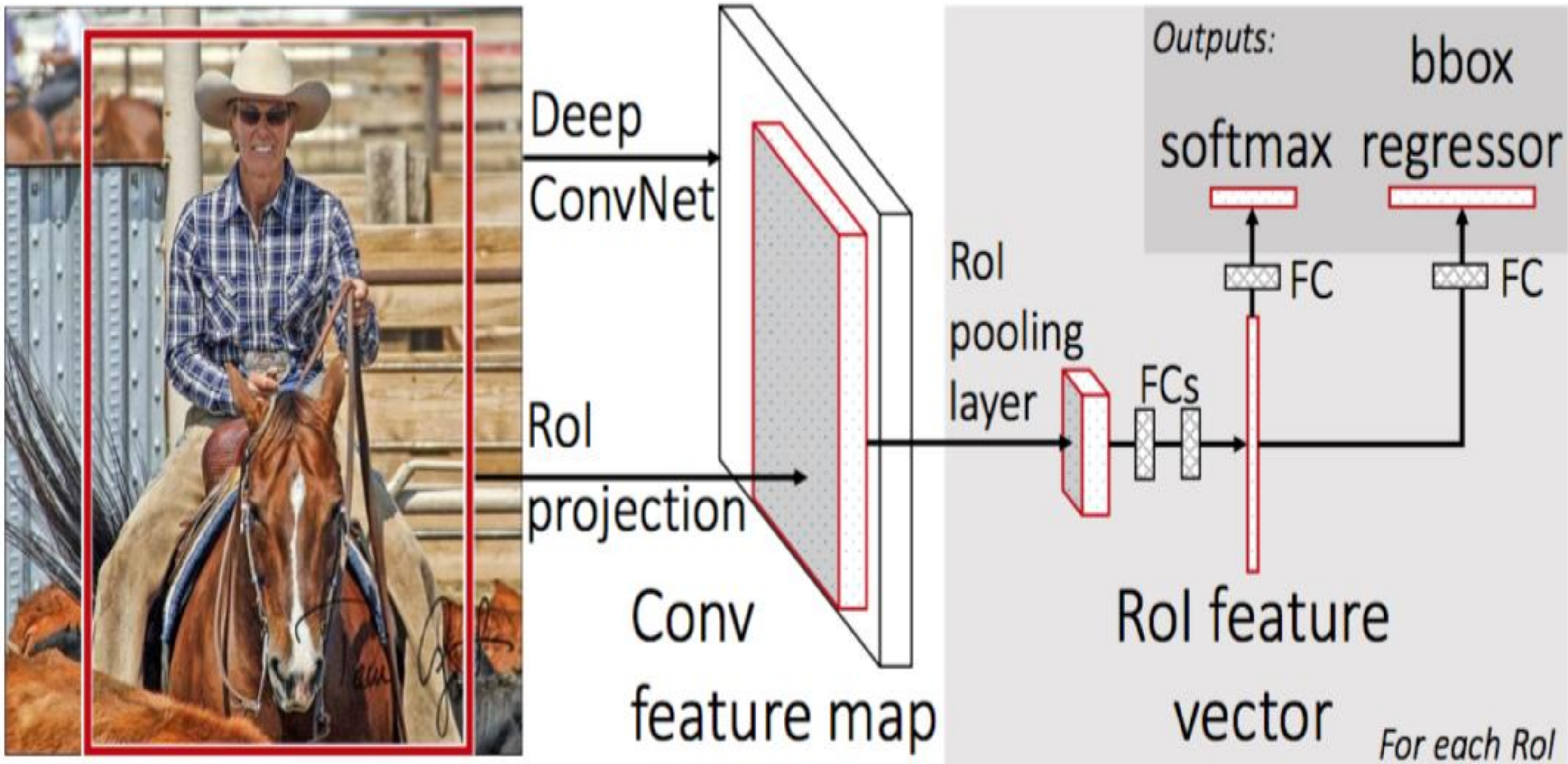
- Pre-train a CNN network on image classification tasks; for example, VGG or ResNet trained on ImageNet dataset. The classification task involves N classes.
- Propose category-independent regions of interest by selective search (~2k candidates per image). Those regions may contain target objects and they are of different sizes.
- Region candidates are warped to have a fixed size as required by CNN.
- Continue fine-tuning the CNN on warped proposal regions for K + 1 classes; The additional one class refers to the background (no object of interest). In the fine-tuning stage, we should use a much smaller learning rate and the mini-batch oversamples the positive cases because most proposed regions are just background.

- Given every image region, one forward propagation through the CNN generates a feature vector. This feature vector is then consumed by a binary SVM trained for each class independently.
- The positive samples are proposed regions with IoU (intersection over union) overlap threshold >= 0.3, and negative samples are irrelevant others.
- To reduce the localization errors, a regression model is trained to correct the predicted detection window on bounding box correction offset using CNN features.

Fast R-CNN

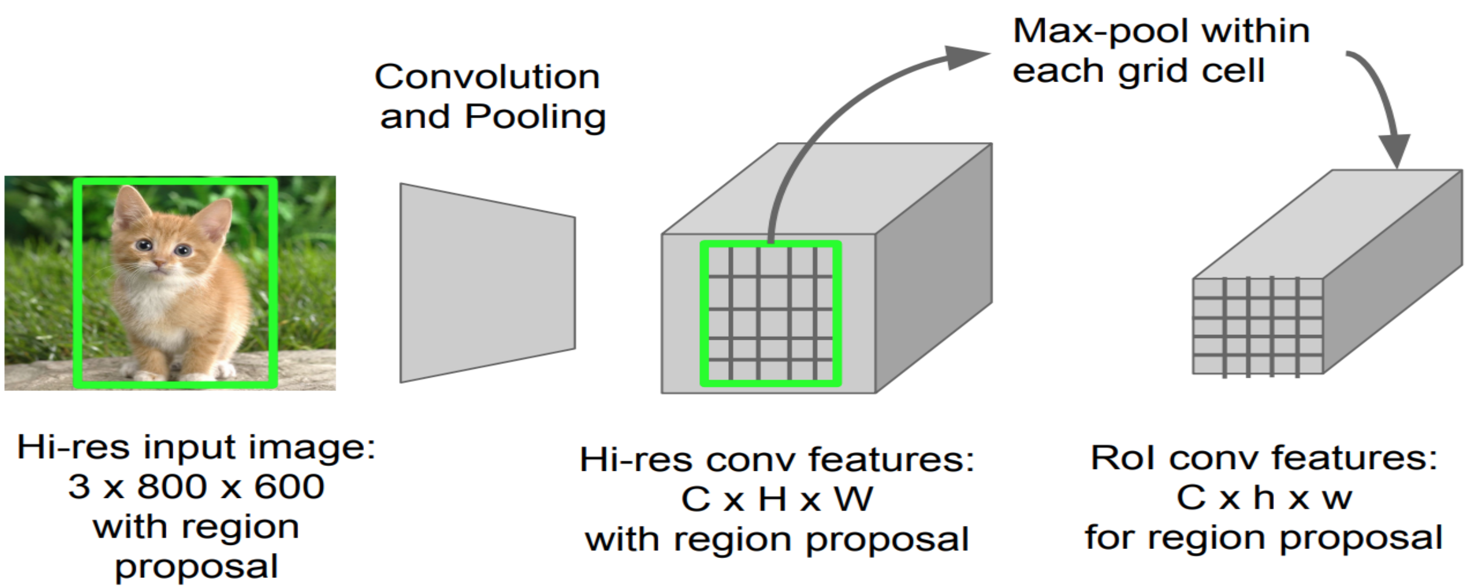
- the Fast R-CNN uses an algorithm to generate region proposals.
- Unlike the R-CNN, which crops and resizes region proposals, the Fast R-CNN processes the entire image.
- Whereas an R-CNN must classify each region, Fast R-CNN pools CNN features corresponding to each region proposal.
- Fast R-CNN is more efficient than R-CNN, because in the Fast R-CNN, the computations for overlapping regions are shared.




Deep
ConvNet
RoI
projection
Conv
feature map
RoI
pooling
layer
FCs
RoI feature
vector
Outputs:
softmax
bbox
regressor
FC
FC
For each RoI

RoI Pooling

- It is a type of max pooling to convert features in the projected region of the image of any size, h x w, into a small fixed window, H x W. The input region is divided into H x W grids, approximately every subwindow of size h/H x w/W. Then apply max-pooling in each grid.



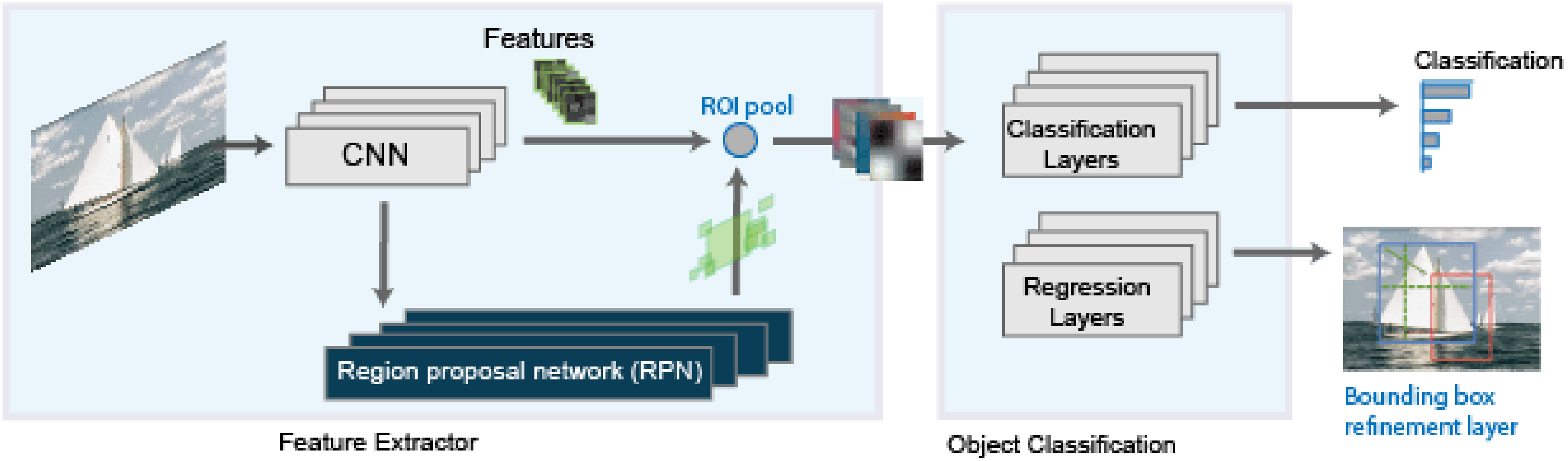
Model Workflow

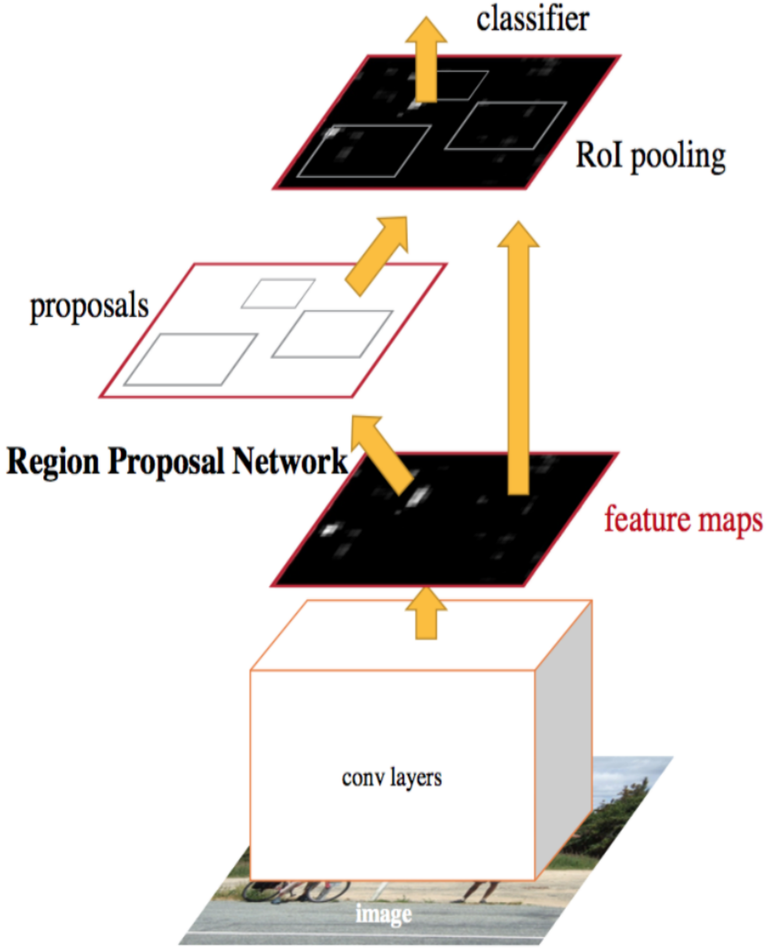
- First, pre-train a convolutional neural network on image classification tasks.
- Propose regions by selective search (~2k candidates per image).
- After the pre-trained CNN:
  - Replace the last max pooling layer of the pre-trained CNN with a RoI pooling layer. The RoI pooling layer outputs fixed-length feature vectors of region proposals. Sharing the CNN computation makes a lot of sense, as many region proposals of the same images are highly overlapped.
  - Replace the last fully connected layer and the last softmax layer (K classes) with a fully connected layer and softmax over K + 1 classes.

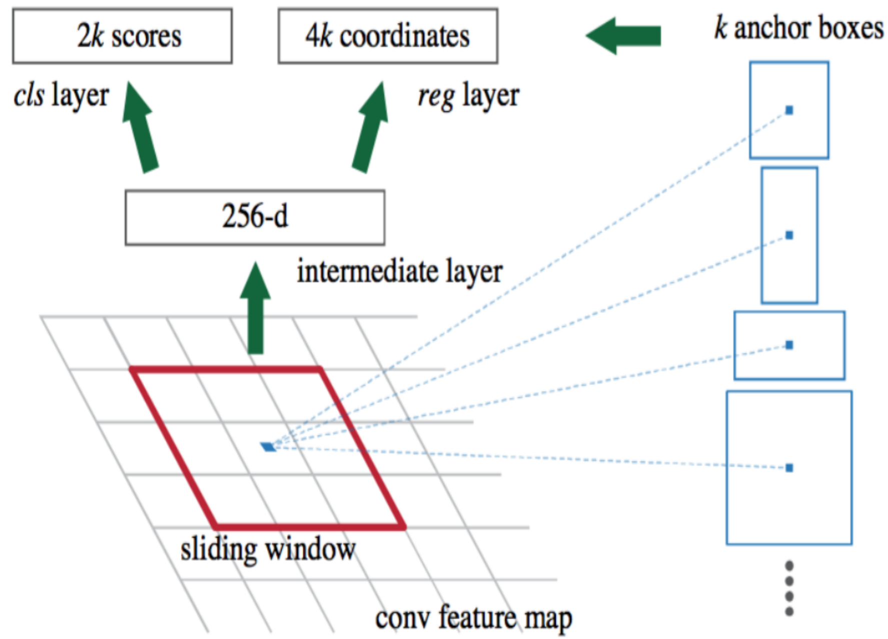
- Finally the model branches into two output layers:
  - A softmax estimator of K + 1 classes (same as in R-CNN, +1 is the “background” class), outputting a discrete probability distribution per RoI.
  - A bounding-box regression model which predicts offsets relative to the original RoI for each of K classes.

Faster R-CNN

- The Faster R-CNN adds a region proposal network (RPN) to generate region proposals directly in the network instead of using an external algorithm
- The RPN uses Anchor Boxes for Object Detection.
- Generating region proposals in the network is faster and better tuned to your data.




classifier
RoI pooling
proposals
Region Proposal Network
feature maps
conv layers
image


2k scores
4k coordinates
k anchor boxes
cls layer
reg layer
256-d
intermediate layer
sliding window
conv feature map

Model Workflow

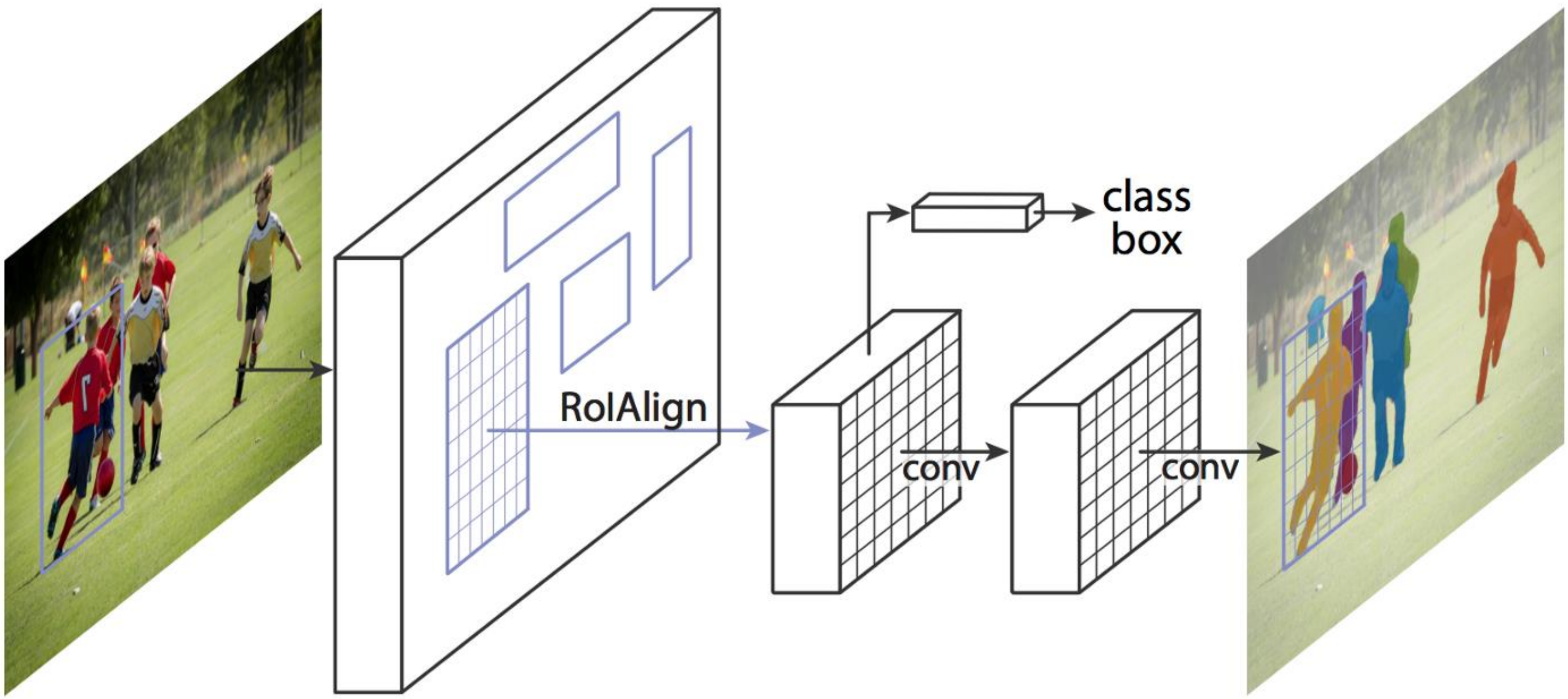
- Pre-train a CNN network on image classification tasks.
- Fine-tune the RPN (region proposal network) end-to-end for the region proposal task, which is initialized by the pre-train image classifier. Positive samples have IoU (intersection-over-union) > 0.7, while negative samples have IoU < 0.3.
  - Slide a small n x n spatial window over the conv feature map of the entire image.
  - At the center of each sliding window, we predict multiple regions of various scales and ratios simultaneously. An anchor is a combination of (sliding window center, scale, ratio). For example, 3 scales + 3 ratios => k=9 anchors at each sliding position.

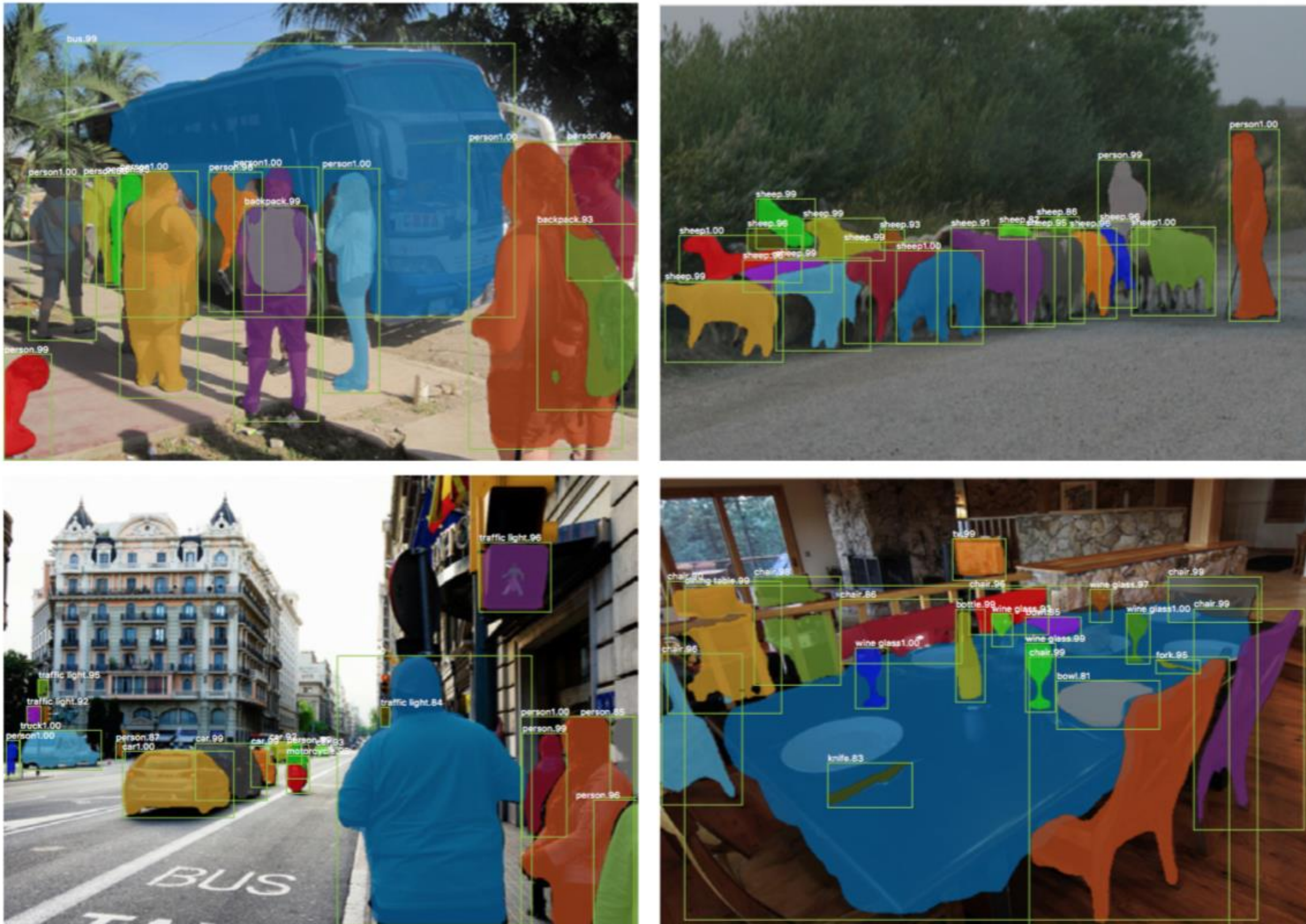
- Train a Fast R-CNN object detection model using the proposals generated by the current RPN
- Then use the Fast R-CNN network to initialize RPN training. While keeping the shared convolutional layers, only fine-tune the RPN-specific layers. At this stage, RPN and the detection network have shared convolutional layers!
- Finally fine-tune the unique layers of Fast R-CNN
- Step 4-5 can be repeated to train RPN and Fast R-CNN alternatively if needed.

Mask R-CNN

- Mask R-CNN extends Faster R-CNN to pixel-level image segmentation.
- The key point is to decouple the classification and the pixel-level mask prediction tasks.
- Based on the framework of Faster R-CNN, it added a third branch for predicting an object mask in parallel with the existing branches for classification and localization.
- The mask branch is a small fully-connected network applied to each RoI, predicting a segmentation mask in a pixel-to-pixel manner.

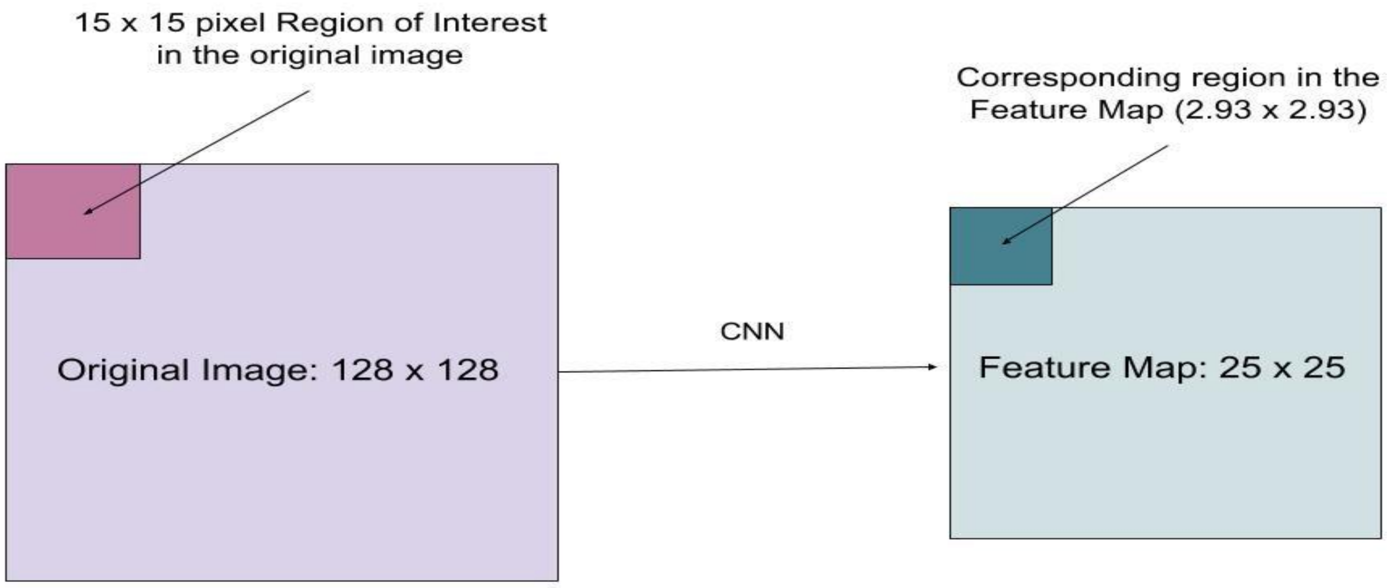
Because pixel-level segmentation requires much more fine-grained alignment than bounding boxes, mask R-CNN improves the RoI pooling layer (named "RoIAlign layer") so that RoI can be better and more precisely mapped to the regions of the original image.





RoIAlign

The RoIAlign layer is designed to fix the location misalignment caused by quantization in the RoI pooling. RoIAlign removes the hash quantization, for example, by using x/16 instead of [x/16], so that the extracted features can be properly aligned with the input pixels. Bilinear interpolation is used for computing the floating-point location values in the input.



Summary of Models in the R-CNN family