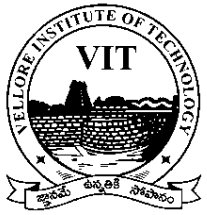


VIT-AP UNIVERSITY

Computer Vision (Course Code: 4047)

Module-4:Lecture-3: Deep Learning Classifiers and Object Detection

Gundimeda Venugopal, Professor of Practice, SCOPE

Topic Coverage

- Object Detection
  - Sliding Window based Object Detection
  - R-CNN
  - Fast R-CNN
  - Faster R-CNN
  - Mask R-CNN
  - YOLO

Object Detection

Sliding Window Approach

Object Detection: Sliding Windows Approach

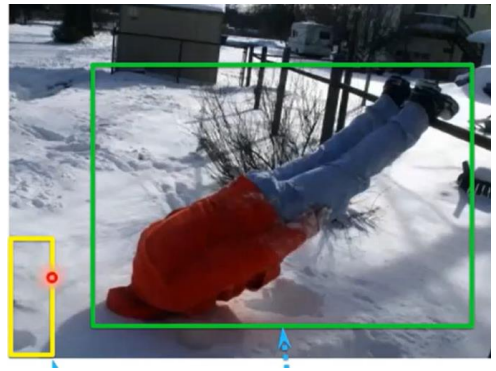
- General approach
  - **Scan all possible locations**
  - Extract features
  - Classify features
  - Post-processing

- Inspect All windows
- Size of window is fixed

- People at different sizes?
  - People nearer to the camera require a bigger bounding box
  - People who are far require a smaller bounding box

- Need to support Different aspect ratios

Objects can be of different sizes

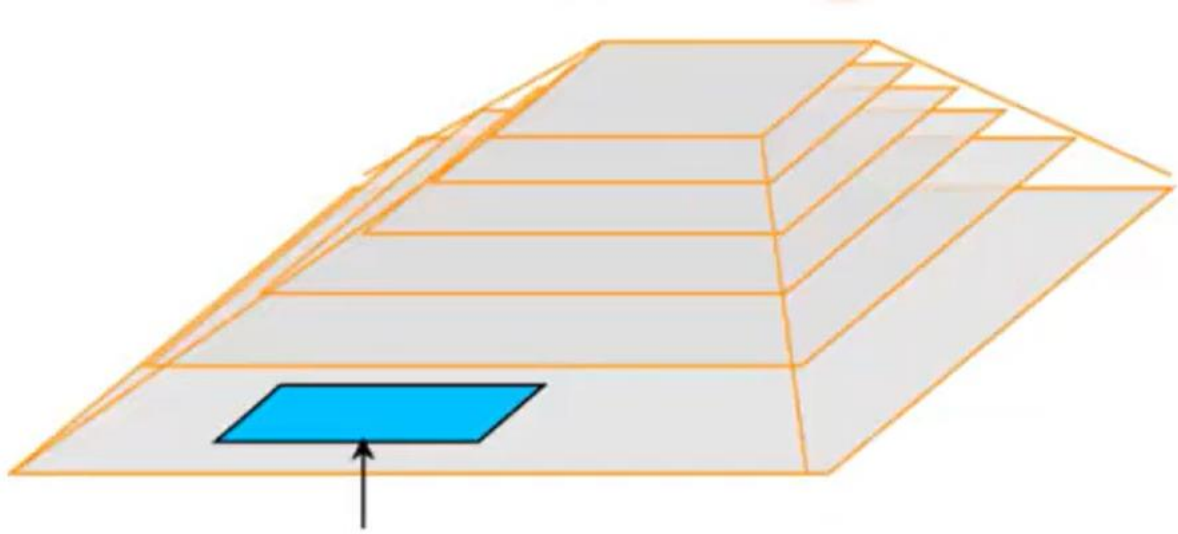


Object Detection Ideas: Addressing Scale space

Downscale the image +

Keep the bounding box size same

Scale Space Pyramid

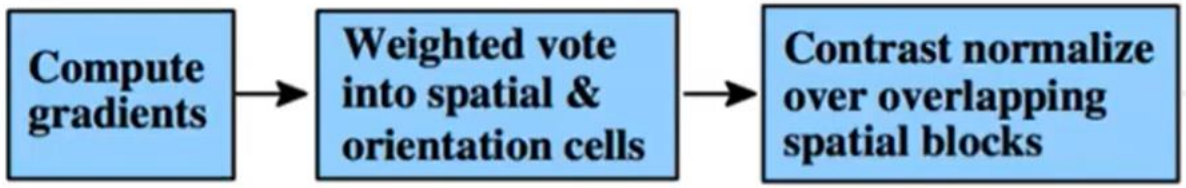


Detection Window

- Highest resolution image(bottom): Try to detect smallest scale objects
- Low resolution image (top): Try to detect the biggest objects

Object Detection: Sliding window approach

- General approach
  - Scan all possible locations
  - **Extract features** **HOG** / SIFT
  - Classify features
  - Post-processing



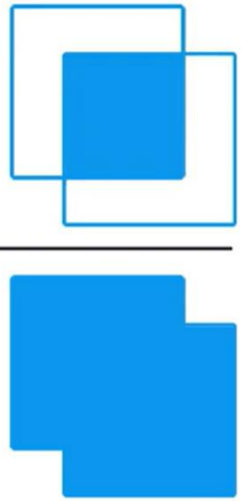
Run the Classifier at all scales
Use a threshold
Detect all positives
Compute the Confidence scores

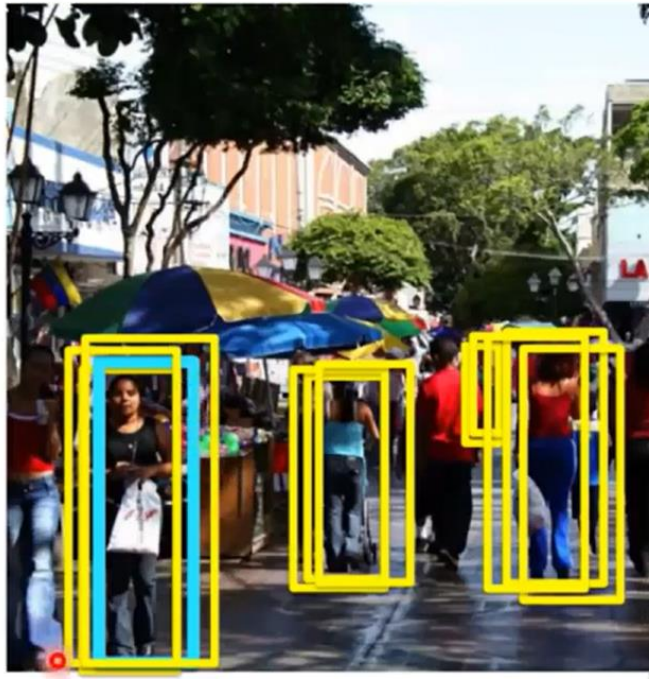
Use Non-maximum Suppression
To clean up multiple bounding boxes

Non-maximum Suppression

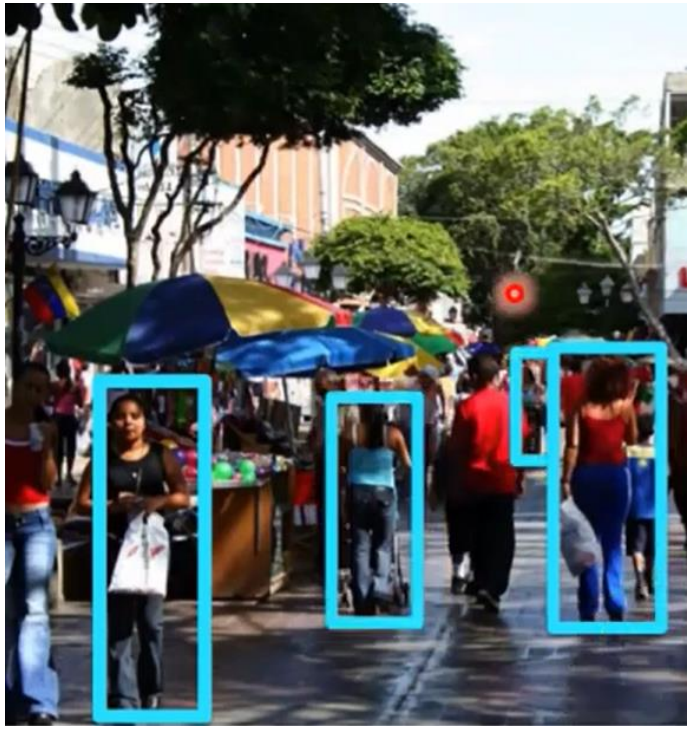
- Iterate over all detections
  - Pick the highest scoring box
- Find overlap
  - with all other boxes
- Remove boxes with high overlap
  - Threshold, usually 0.5

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





Final Output



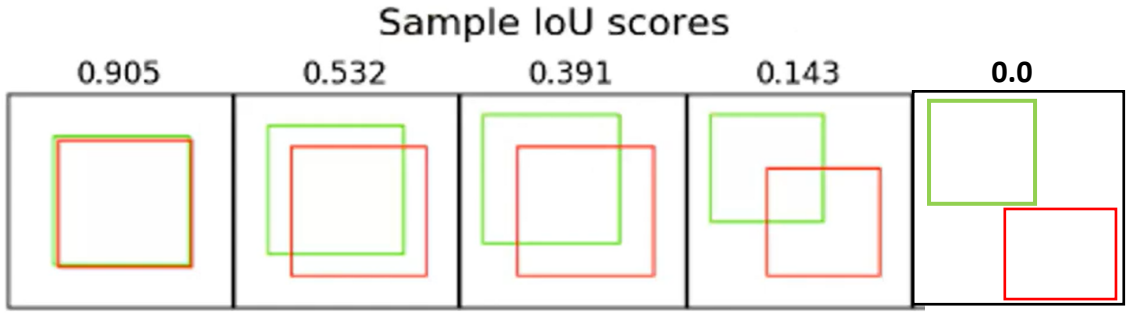
How to evaluate the Object Detection Algorithm?

How do we know if our approach is doing well?

Intersection over union (IoU)

$$= \frac{\text{size of}}{\text{size of}}$$

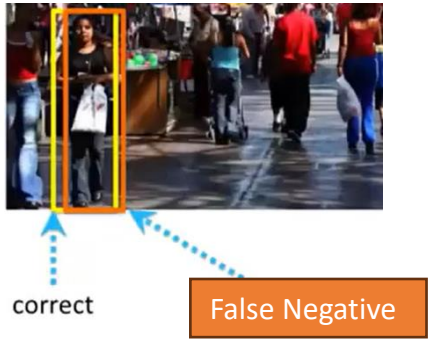
“Correct” if IoU >= 0.5



Model Evaluation

Model Evaluation is based on both:

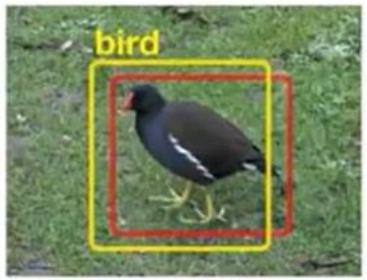
- **Class Label Prediction (classification)**
- **Accurate Bounding box prediction (regression)**



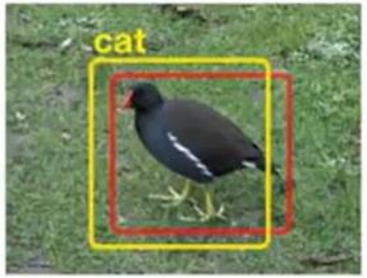
Multiple box prediction IoU >= 0.5
Only one is correct
NW Class Label =GT Label = bird
IOU(YellowP1, GT) >= IOU(OrangeP2, GT) >= 0.5
YellowP1=TP, OrangeP2=FN

- True positives
- False positives
- False negatives

TP



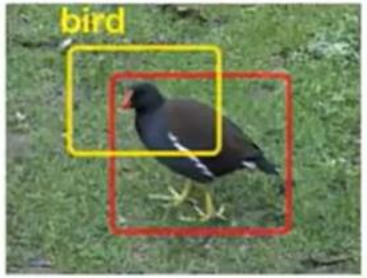
NW Class Label =GT Label = bird
IOU(YellowP, RedGT) >= 0.5



wrong class

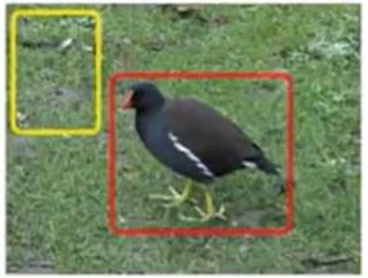
NW Class Label=cat ≠ GT Label = bird
IOU(YellowP, RedGT) >= 0.5

FP



IOU < 0.5

NW Class Label =GT Label = bird
IOU(YellowP, RedGT) < 0.5



no overlap

NW Class Label =No Label
IOU(YellowP, RedGT) = 0

FN

Nothing is present

NW Class Label =No Object
No YellowP bounding box prediction

Model Evaluation: Precision and Recall

❖Precision:

- How accurate the model is. If P objects are predicted by model (out of total N), how many objects are predicted correctly.
- Precision is the ability of a model to identify only relevant objects. It is the % of correct positive predictions among all relevant ground truths.

❖Recall

- Looks at Coverage: if N objects are presents, how many objects it is able to predict.
- Recall is the ability of a model to find all relevant cases. It is the % of true positives detected

$$precision = \frac{true\ positives}{true\ positives + false\ positives}$$

$$recall = \frac{true\ positives}{true\ positives\ + false\ negatives}$$

F1 Score = 2 . Precision . Recall / (Precision + Recall)

Detection as Classification: Challenges

Problem:

- Need to test many positions and scales
- Use a computationally demanding classifier (CNN)
- Search at different scales
- Search at different positions

Solution: Only look at a tiny subset of possible positions

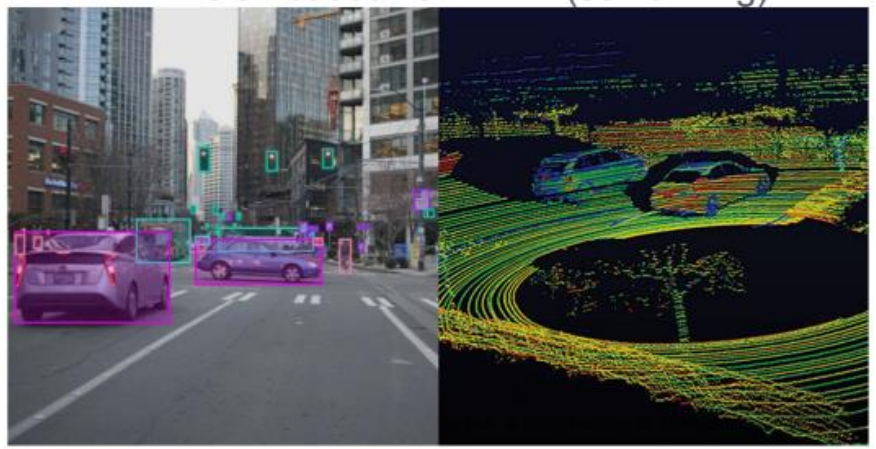
Object Detection

CNN based Approaches (R-CNN, Faster R-CNN and Mask R-CNN)

Importance of Object Detection for Robotics

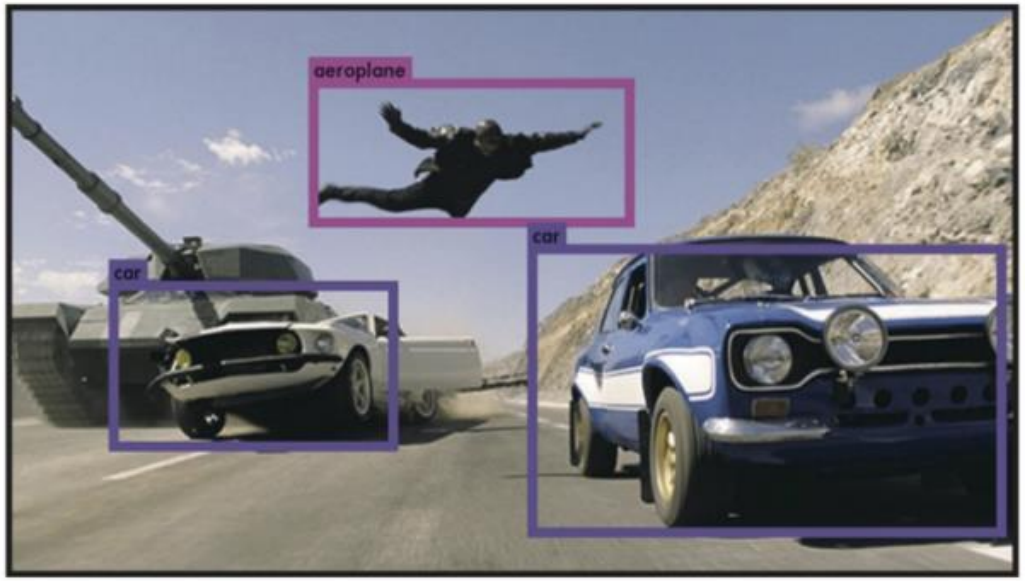
- Visual modality is very powerful
- Humans are able to detect objects and do perception using just this modality in real time (not needing radar)
- If we want responsive robot systems that work in real time (without specialized sensors) almost real time vision based object detection can help greatly

Vision based vs LIDAR (self driving)



Formal Problem Setting: what is given and what is needed

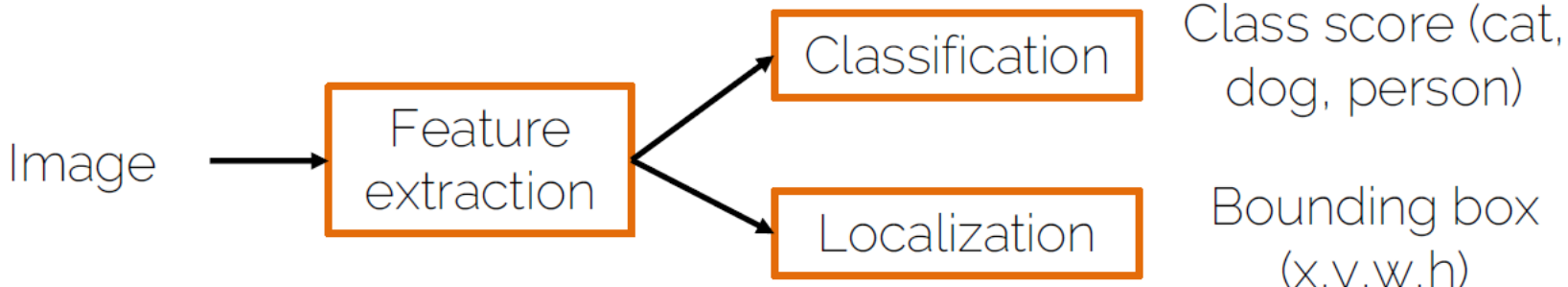
- Given an image generate bounding boxes, one for each detectable object in image
- For each bounding box, output 5 predictions: x, y, w, h, confidence. Also output class



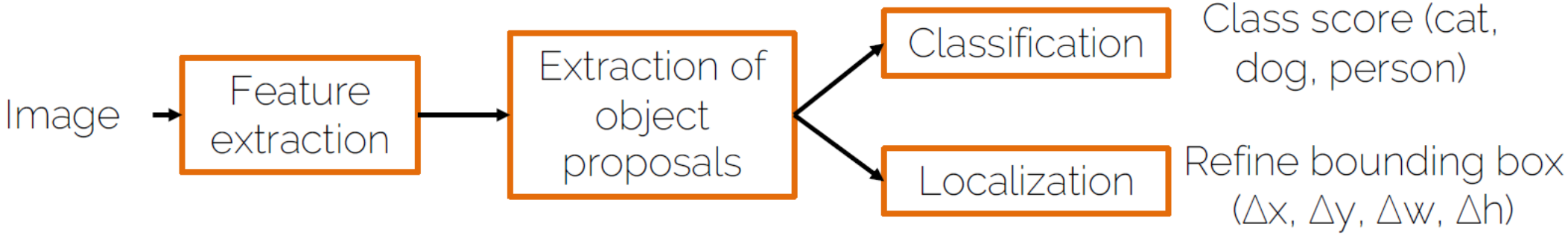
- x, y (coordinates for center of bounding box)
- w,h (width and height)
- confidence (probability bounding box has object)
- class (classification of object in bounding box)

Types of Object Detectors

- One-stage detectors



- Two-stage detectors



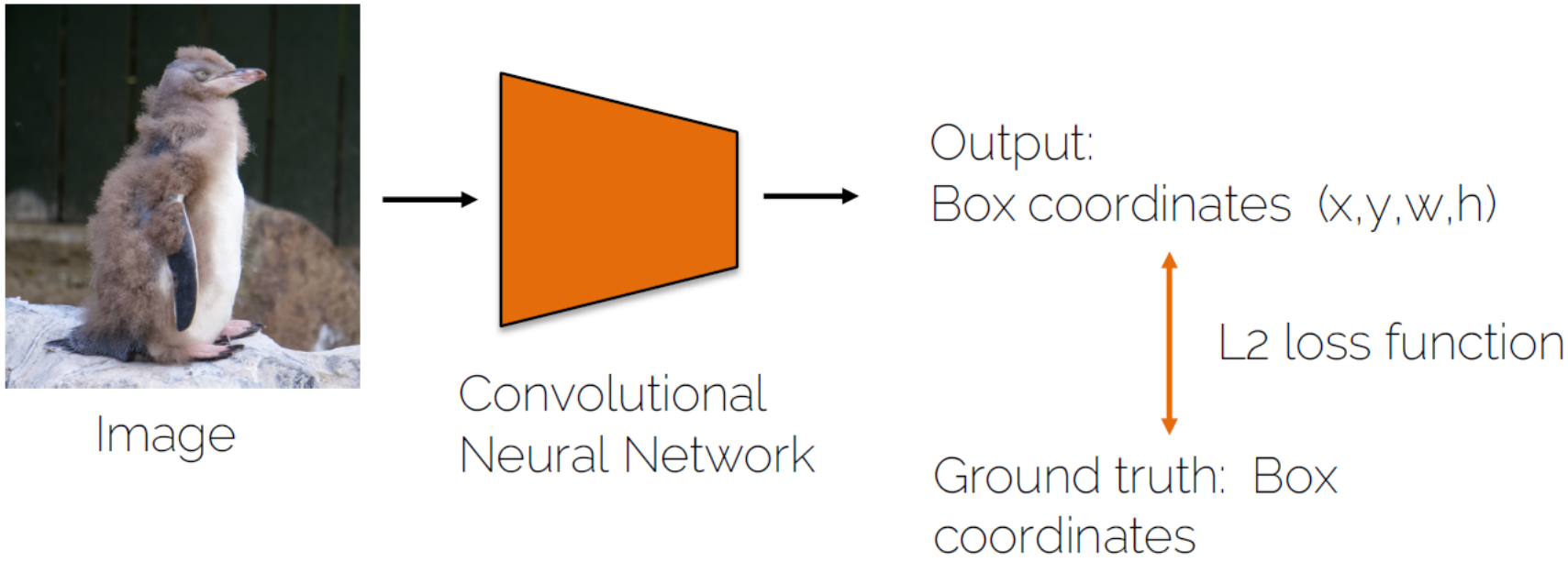
Types of Object Detectors

- One-stage detectors
  - YOLO, SSD, RetinaNet
  - CenterNet, CornerNet, ExtremeNet

- Two-stage detectors
  - R-CNN, Fast R-CNN, Faster R-CNN 
  - SPP-Net, R-FCN, FPN

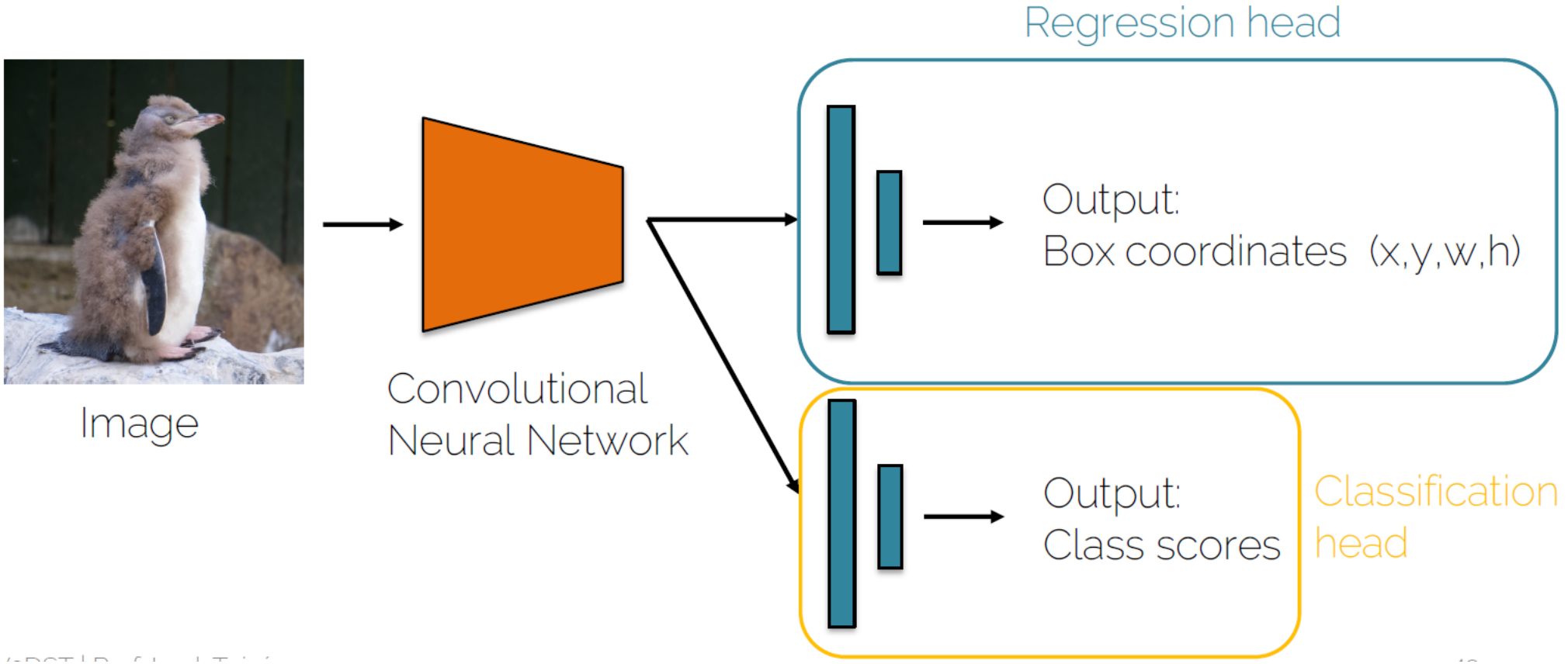
Localization

- Bounding box regression

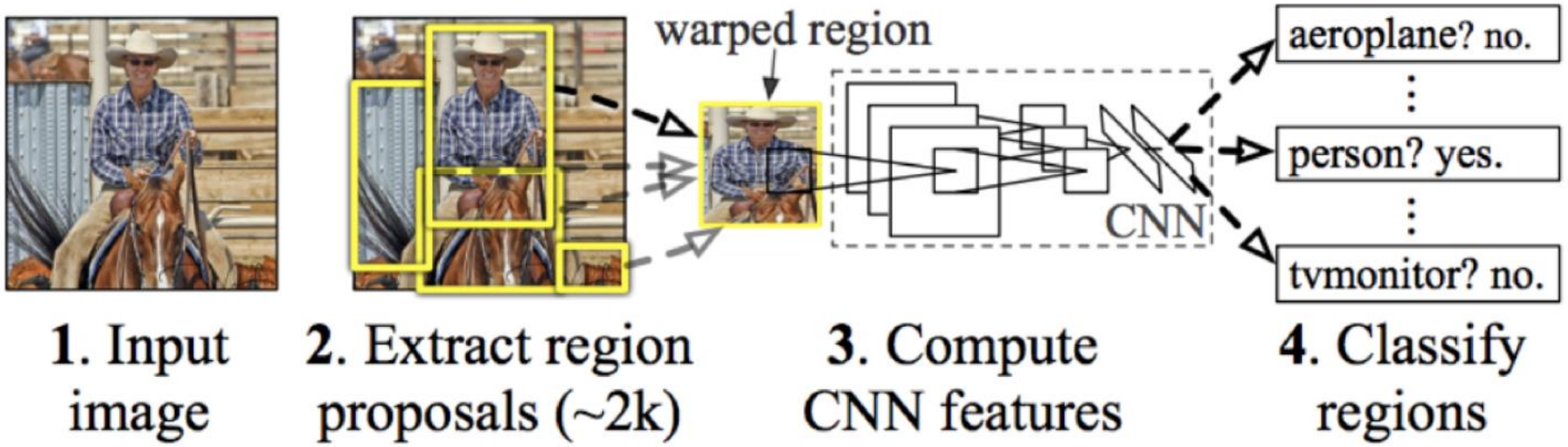


Localization and Classification

- Bounding box regression



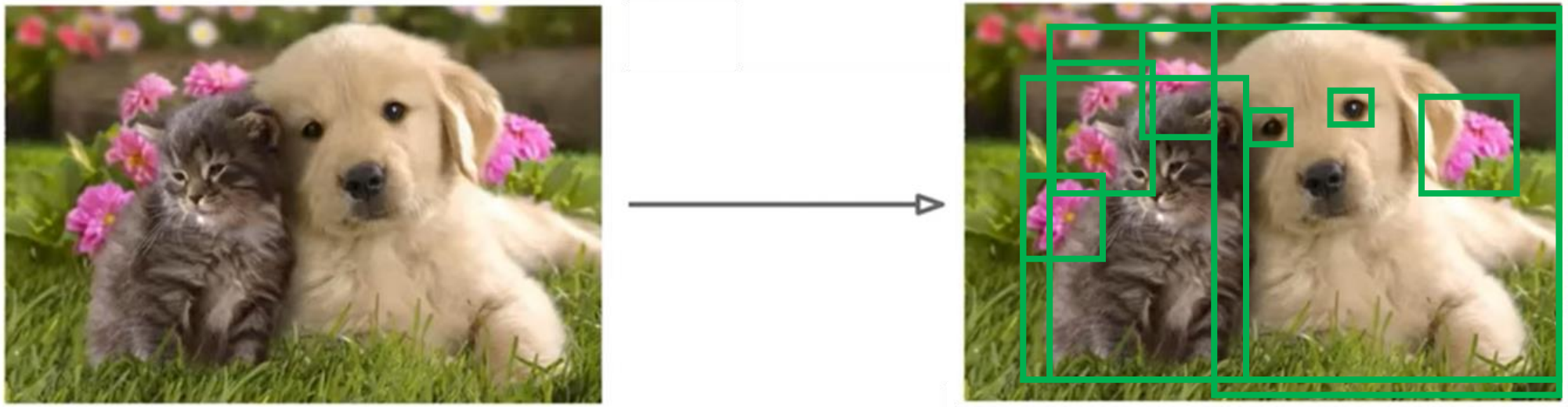
R-CNN



Girschick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014

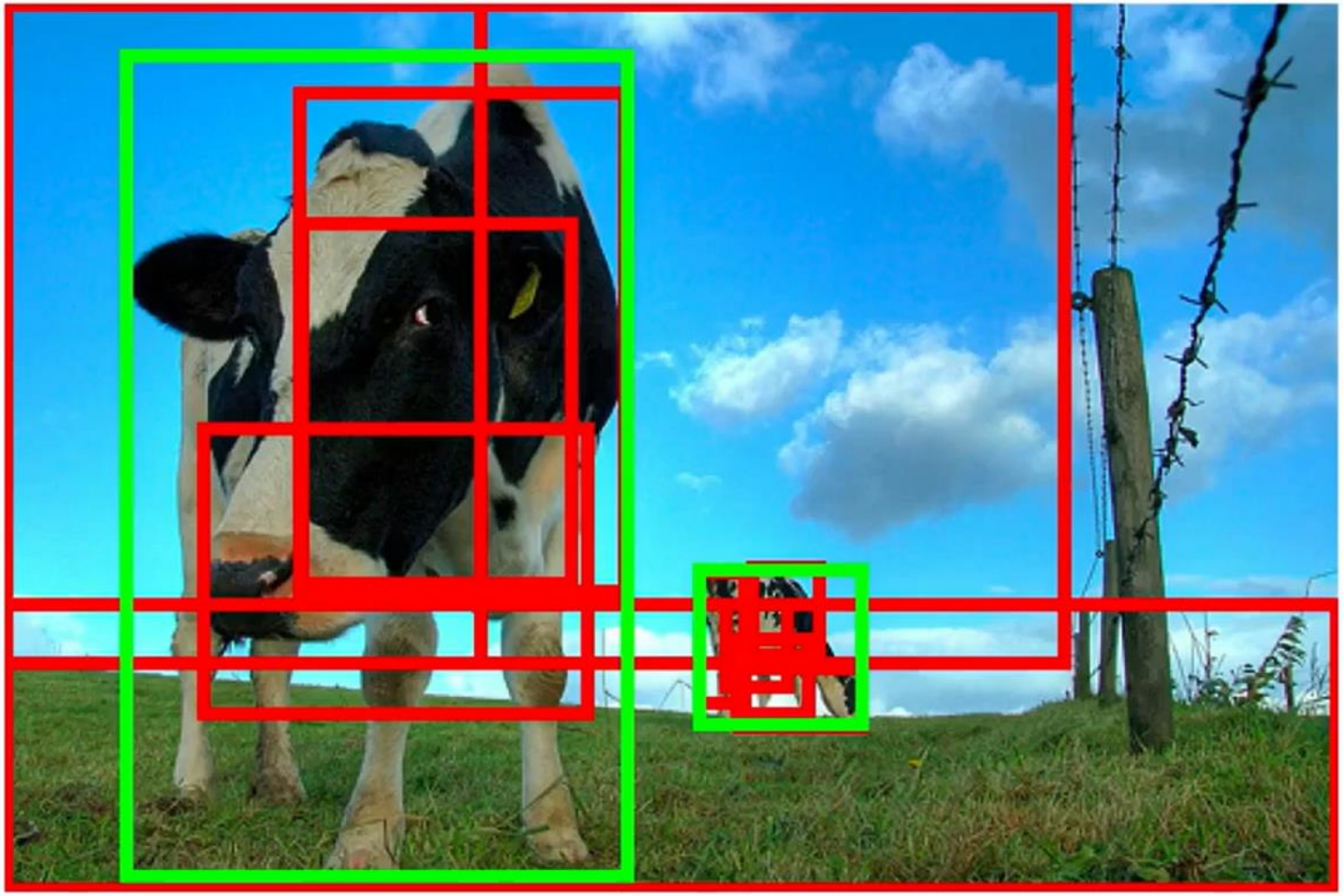
R-CNN: Region Proposals

- Find “blobby” image regions that are likely to contain objects
- “Class-agnostic” object detector
- Look for “blob-like” regions



Idea: Use an Algorithm to subsample the search space to identify places where potential objects are present.

Region Proposals: Selective Search Algorithm



Designed to have high recall, but low precision

equivalent to…

Return many false positive regions, but we are quite certain that they contain the objects of interest

Uijlings, Jasper RR, et al. "Selective search for object recognition." *International journal of computer vision* 104 (2013): 154-171.

R-CNN: Region Proposals → Selective Search Algorithm

Bottom up segmentation , merging regions at multiple scales

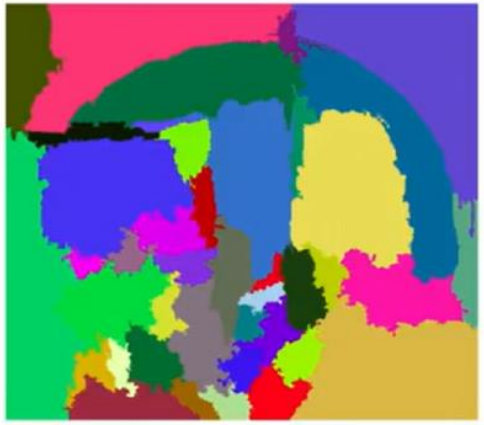
Based on
- Color
- Texture
- Size
- shape

Seed regions

Fine Level

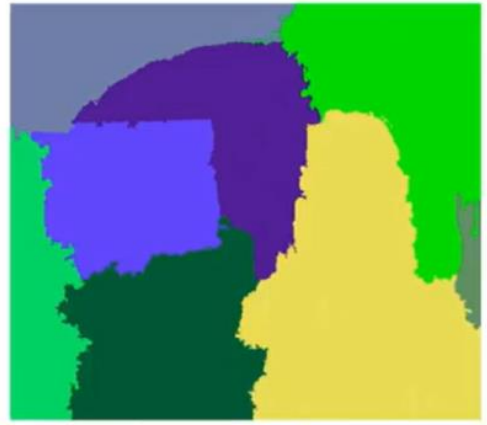
Iteration 1

Coarse Level

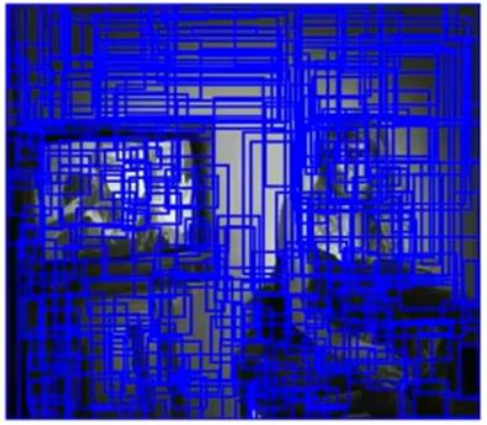


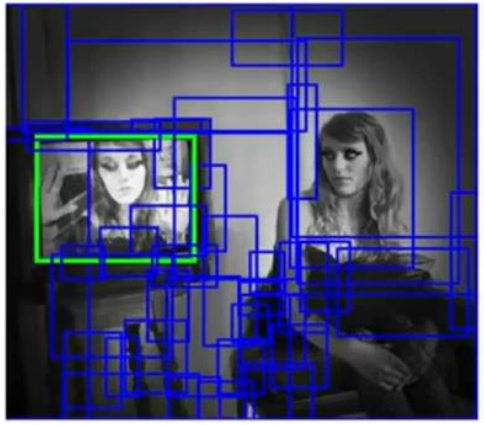
Iteration 2

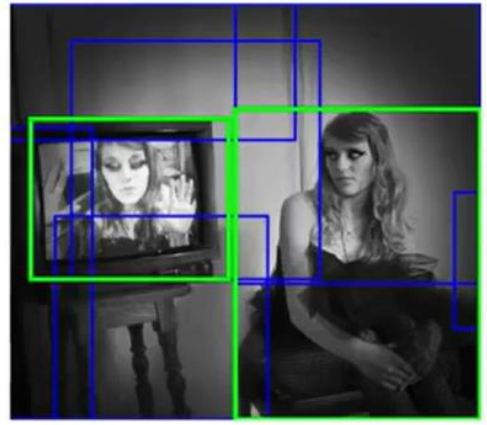
Even Coarse Level



Convert regions (super pixels) to boxes

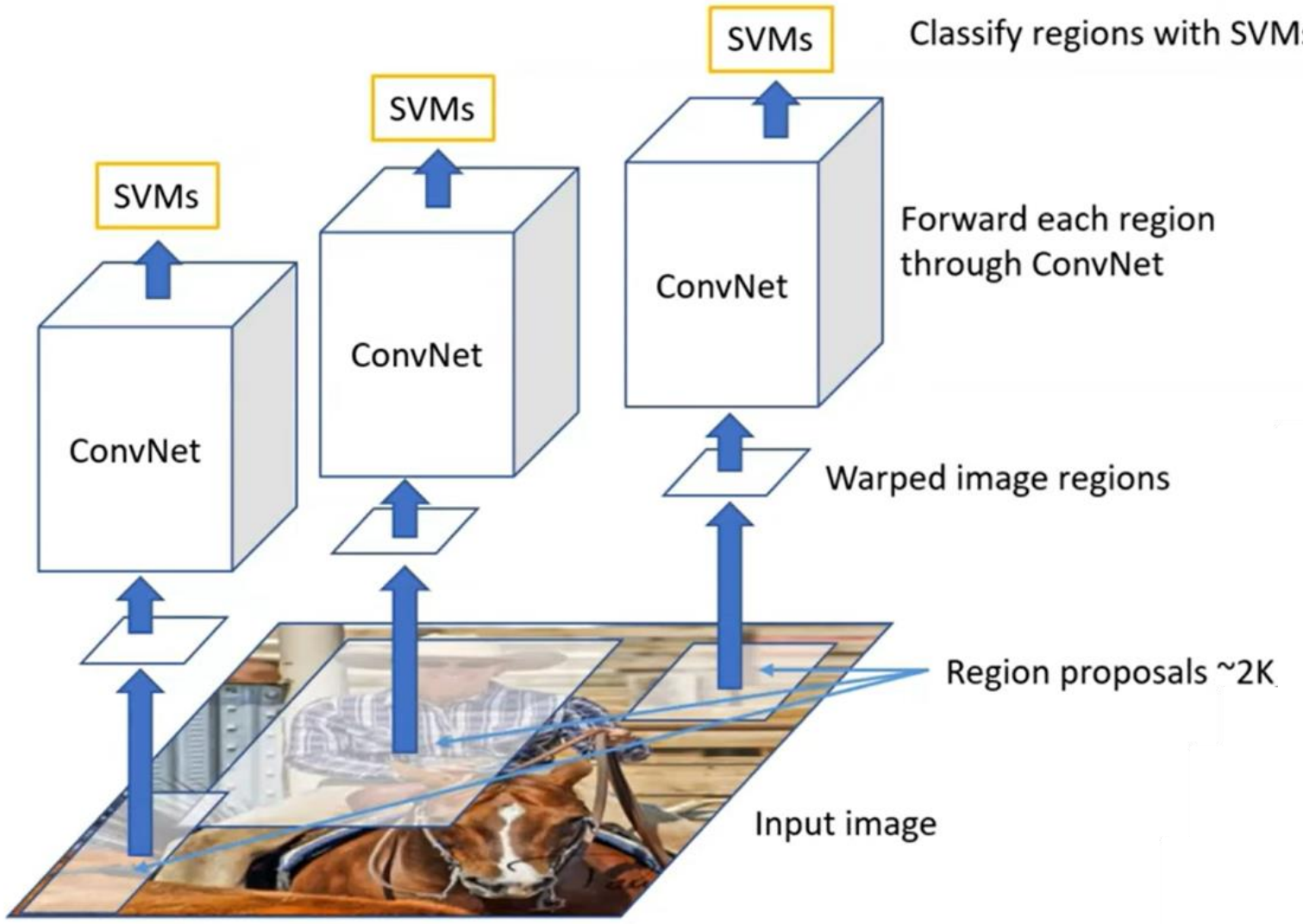






Uijlings, Jasper RR, et al. "Selective search for object recognition." *International journal of computer vision* 104 (2013): 154-171.

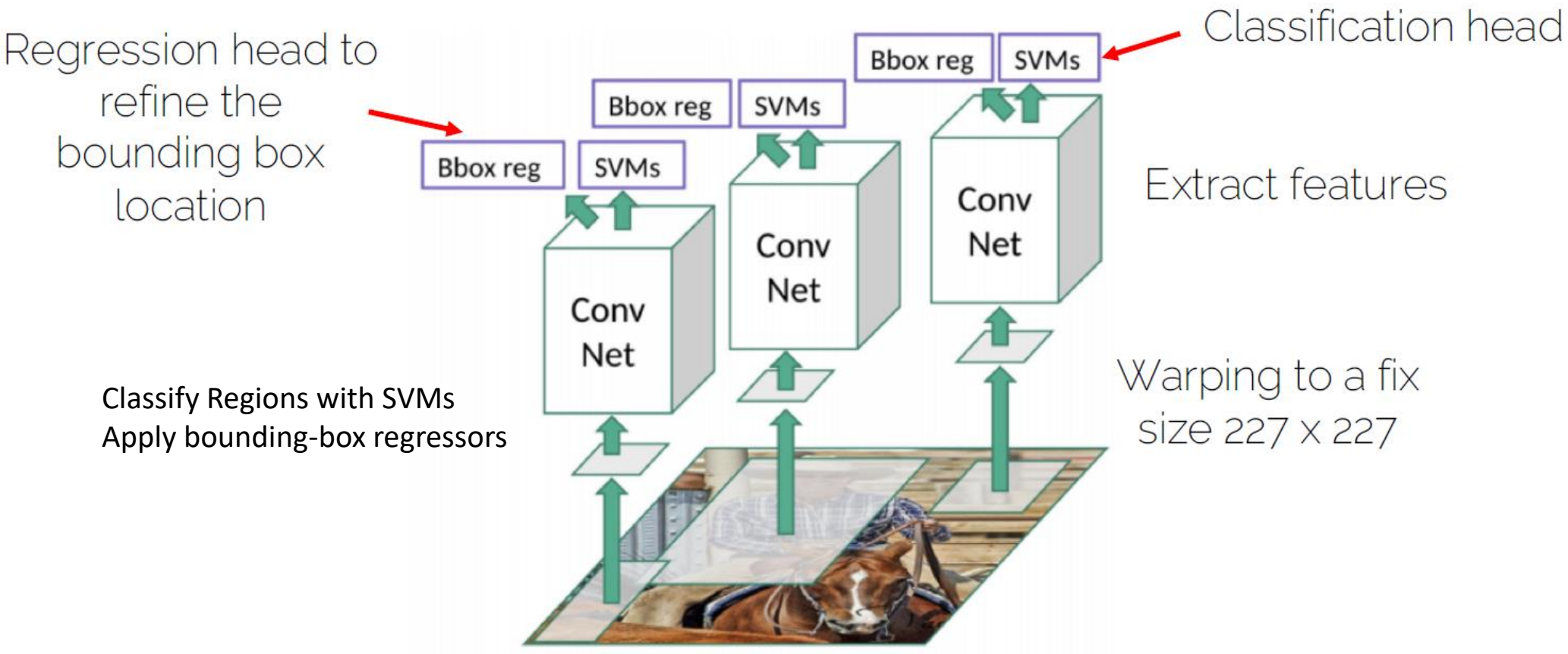
R-CNN: Region Proposals + CNN features



Source: R. Girshick

R. Girshick, J. Donahue, T. Darrell, and J. Malik, Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, CVPR 2014.

R-CNN: Region Proposals + CNN features

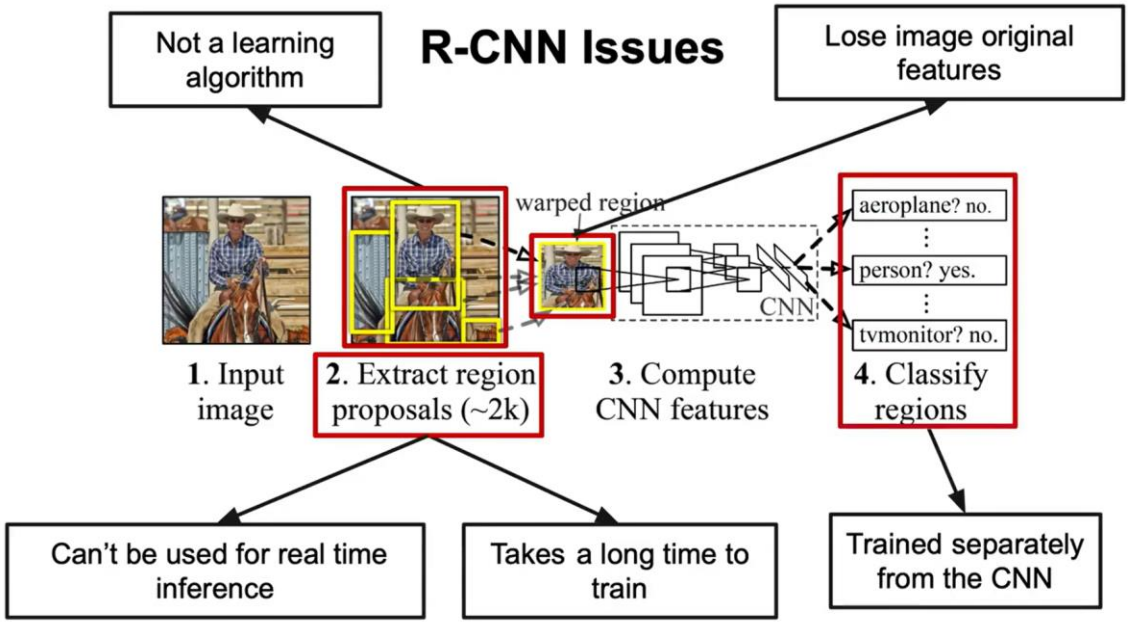


Girschick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014

R-CNN Training Scheme

- Training scheme:
  - 1. Pre-train the CNN on ImageNet
  - 2. Finetune the CNN on the number of classes the detector is aiming to classify (softmax loss)
  - 3. Train a linear Support Vector Machine classifier to classify image regions. One SVM per class! (hinge loss)
  - 4. Train the bounding box regressor (L2 loss)

R-CNN Pros and Cons



- PROS:
  - The pipeline of proposals, feature extraction and SVM classification is well-known and tested. Only features are changed (CNN instead of HOG).
  - CNN summarizes each proposal into a 4096 vector (much more compact representation compared to HOG)
  - Leverage transfer learning: the CNN can be pre-trained for image classification with C classes. One needs only to change the FC layers to deal with Z classes.

- CONS: Let us try to solve this first
  - Slow! 47s/image with VGG16 backbone. One considers around 2000 proposals per image, they need to be warped and forwarded through the CNN.
  - Training is also slow and complex
  - The object proposal algorithm is fixed. Feature extraction and SVM classifier are trained separately → not exploiting learning to its full potential.

Fast R-CNN

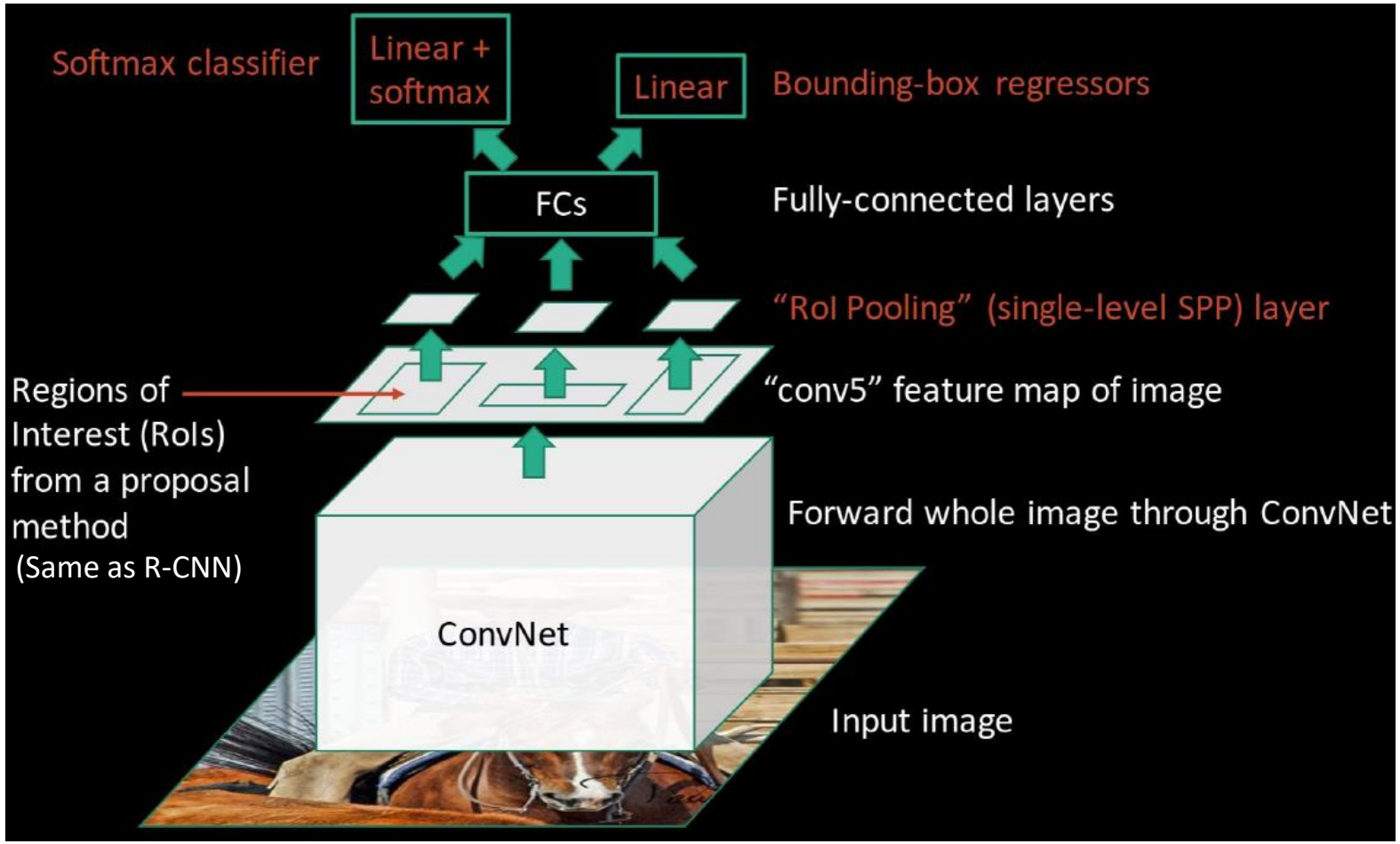
Solution:

- Fast R-CNN
  - Single forward pass for each image
  - No separate classifier
  - End-to-end training

Object Detection

Fast R-CNN

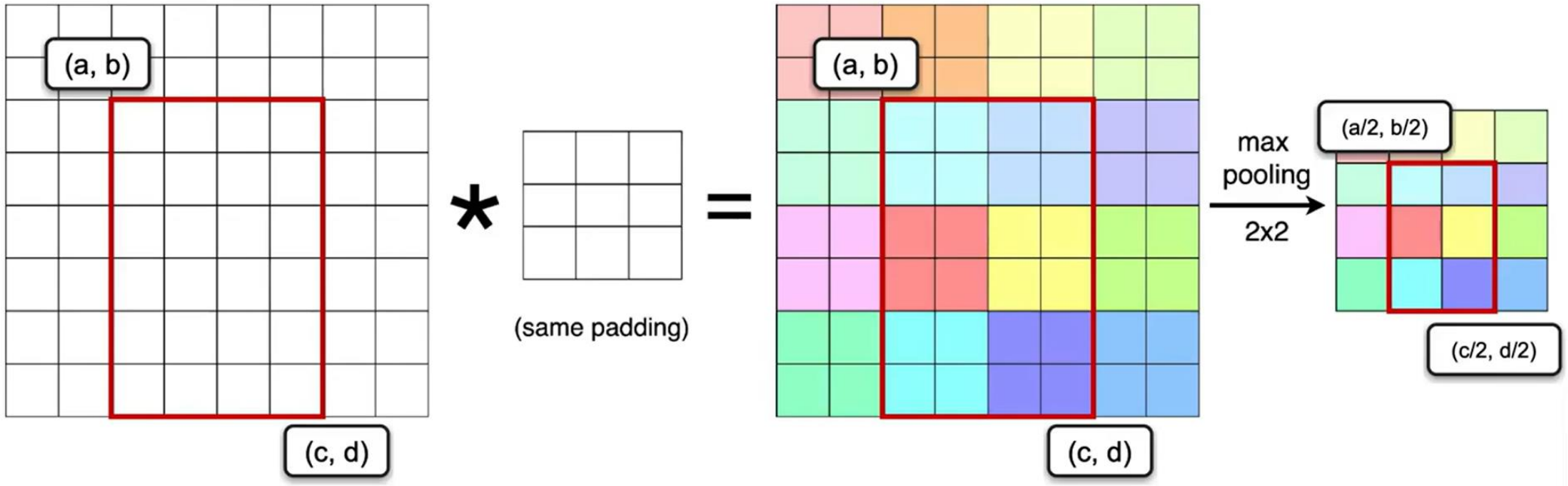
Fast R-CNN



- In R-CNN, Regional proposals are generated using the Selective Search Algorithm.
- Different Size proposals are warped to a common size and the CNN Feature extractor is run for each proposal to extract features for the respective Proposal regions.
- In Fast R-CNN, which is introduced in 2015, an improvement is made to run the CNN only once to generate the entire Feature map.
- Run the Regional proposal algorithm (same as R-CNN) on the input image and project the resulting regions coordinates into the corresponding coordinates in the feature map.

Fast R-CNN: Region Projection

- Projecting the Region Proposal regions coordinates into the corresponding coordinates in the feature map.



Input Image

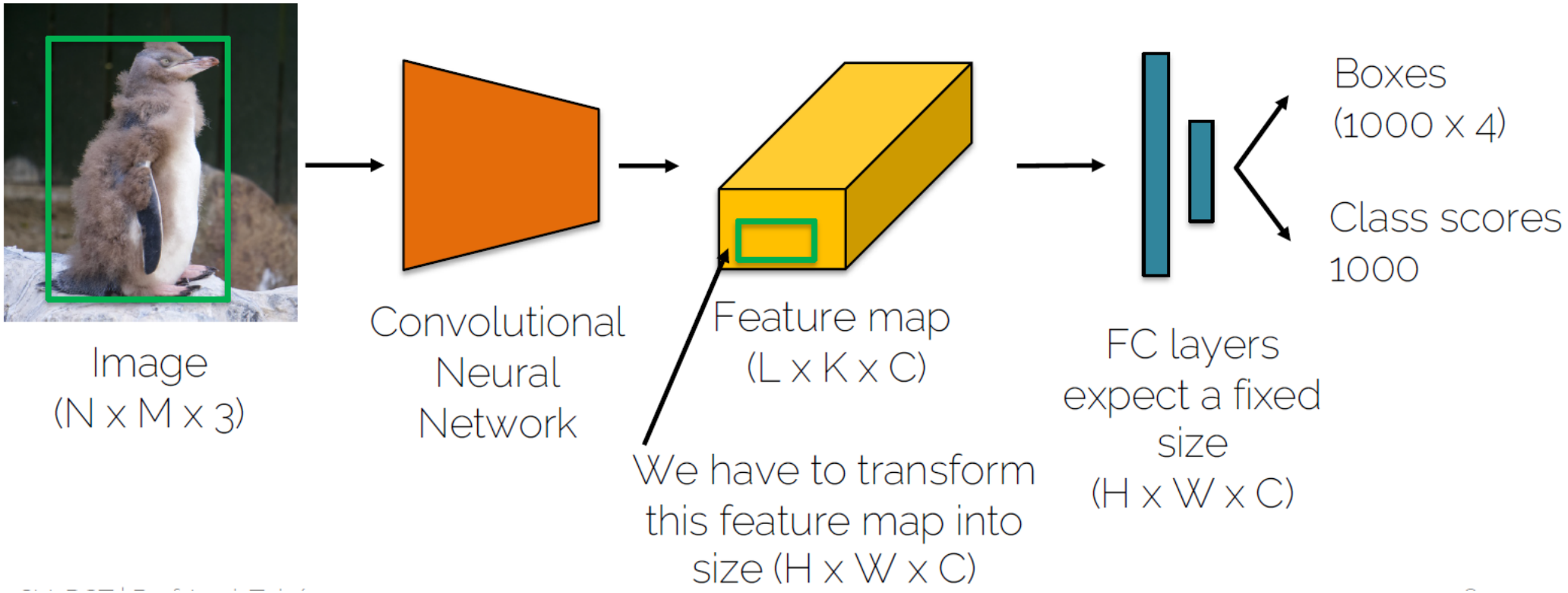
Feature Map (a)

Feature Map (b)

Extrapolate these two operations and keep track of where in the feature map each region proposal is mapped

Fast R-CNN: ROI Pooling

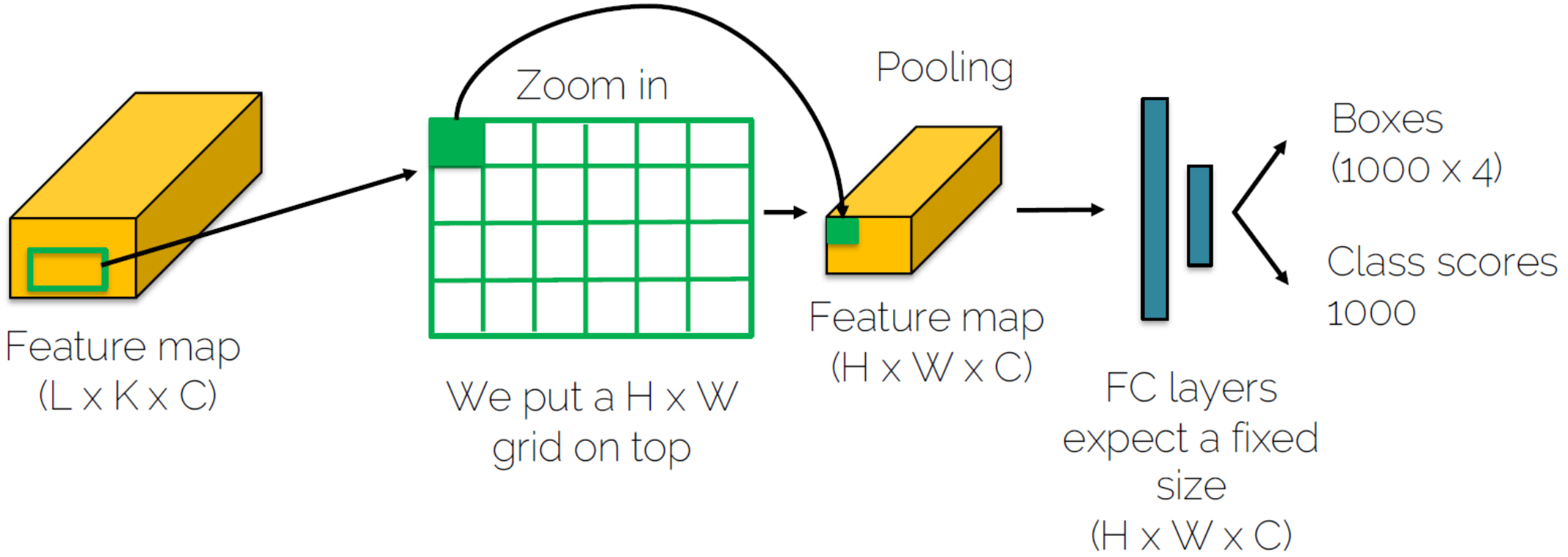
- Region of Interest Pooling



Height (H), Width (W), #Number of Channels (C)

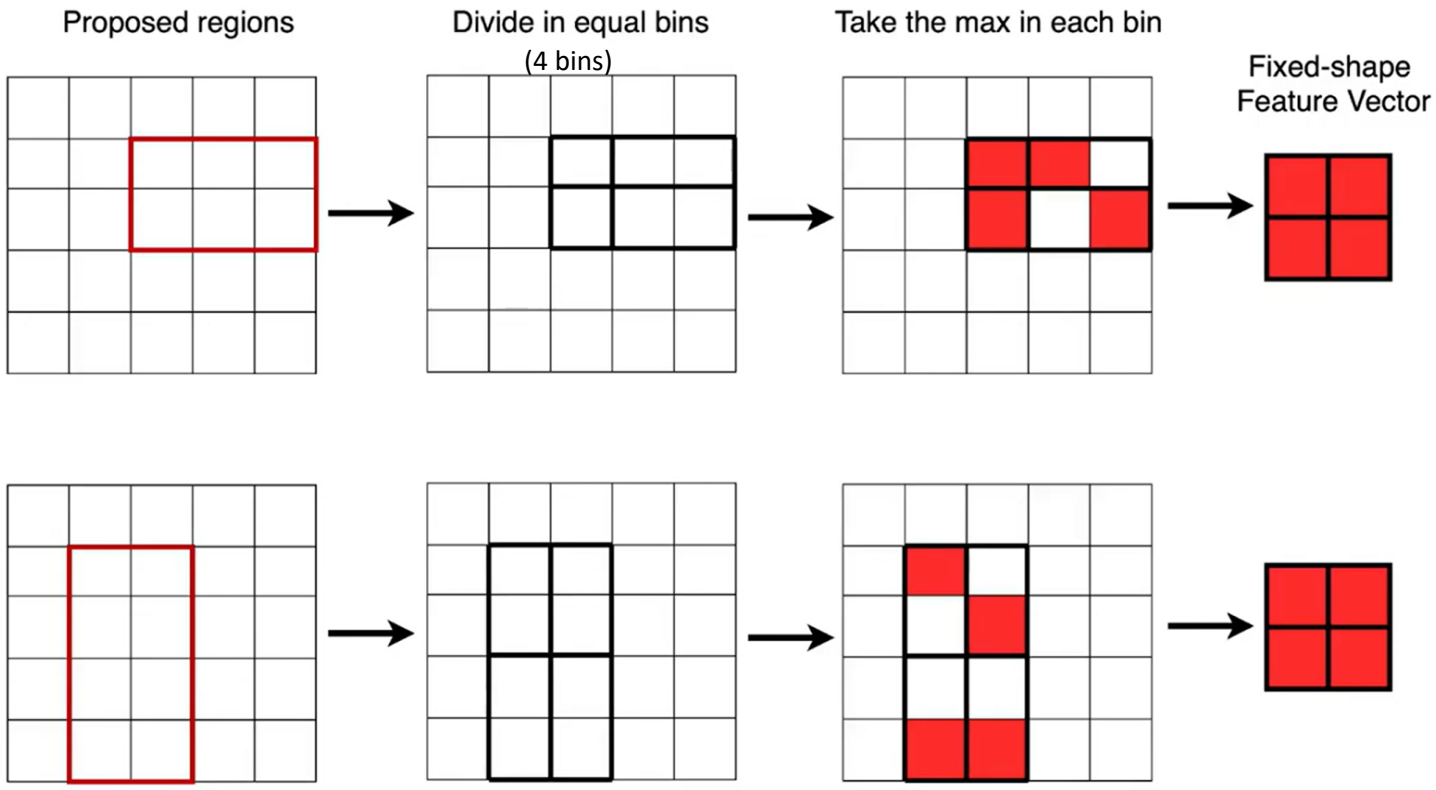
Fast R-CNN ROI pooling

- Region of Interest Pooling



Fast R-CNN: Region of Interest (ROI) Pooling

- Fast R-CNN uses Region of Interest (ROI) Pooling Layer to extract a fixed length feature vector for each region proposal.



- ROI Pooling Layer divides each of Proposed regions into a fixed number of rectangular beams independent of the input shape and outputs a single value by performing Max Pooling with in each bin.

Fast R-CNN Results

Fast R-CNN Results

- VGG-16 CNN on Pascal VOC 2007 dataset

The test times do not include proposal generation!

	R-CNN	Fast R-CNN
Training Time:	84 hours	9.5 hours
(Speedup)	1x	8.8x
Test time per image	47 seconds	0.32 seconds
(Speedup)	1x	146x
mAP (VOC 2007)	66.0	66.9

Faster!

FASTER!

Better!

Fast R-CNN Results

Fast R-CNN Results

With proposals included

- VGG-16 CNN on Pascal VOC 2007 dataset

	R-CNN	Fast R-CNN
Training Time:	84 hours	9.5 hours
(Speedup)	1x	8.8x
Test time per image	50 seconds	2 seconds
(Speedup)	1x	25x
mAP (VOC 2007)	66.0	66.9

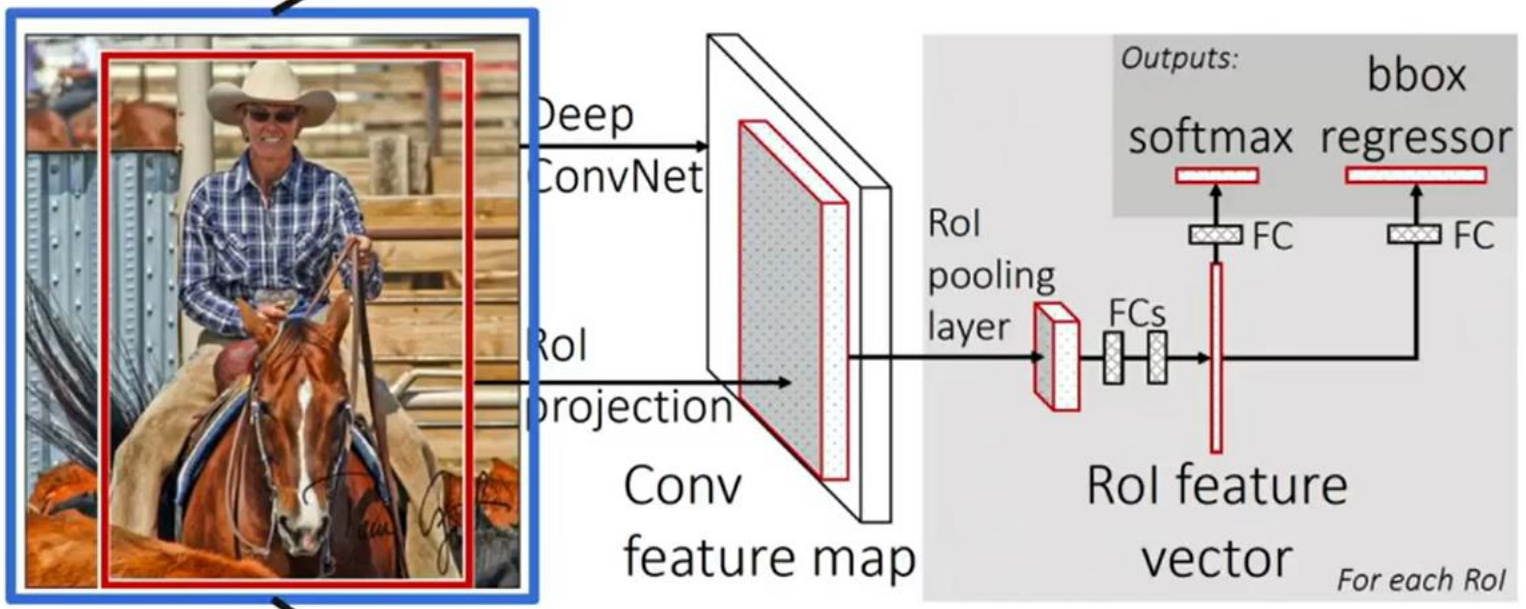
Faster!

FASTER!

Better!

Fast R-CNN Issues

Still quite slow for most of practical applications



The region proposal algorithm is an external algorithm that was not specifically tuned for the object detection task at hand

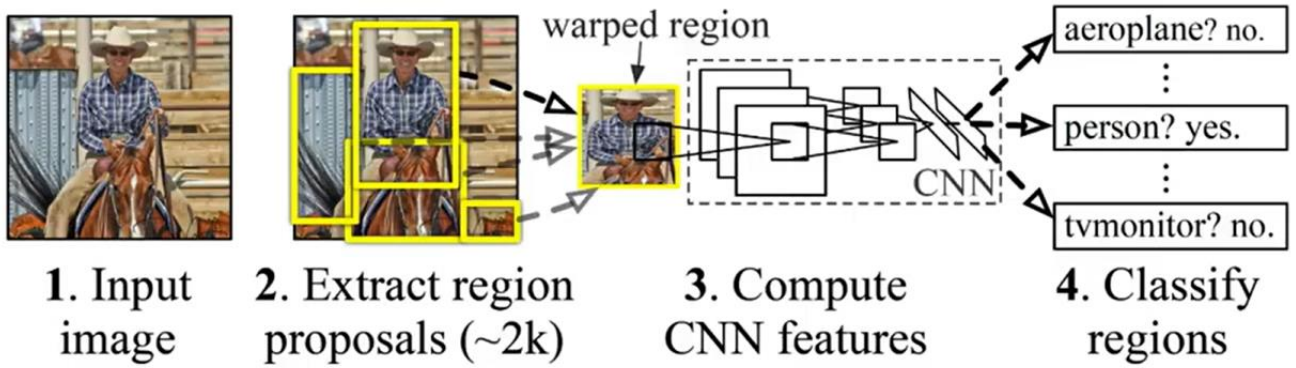
Girshick, Ross. "Fast r-cnn." Proceedings of the IEEE international conference on computer vision. 2015.

Object Detection

Faster R-CNN

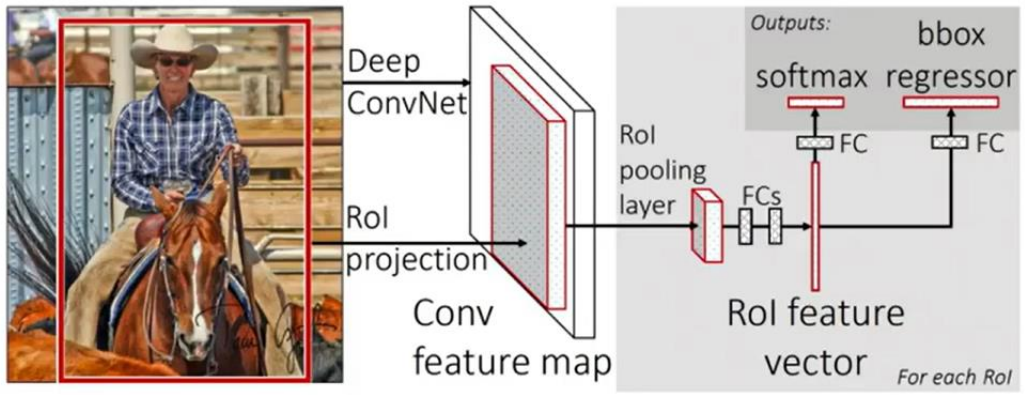
Recall: R-CNN and Fast R-CNN

R-CNN



Uses the CNN network on each proposed region

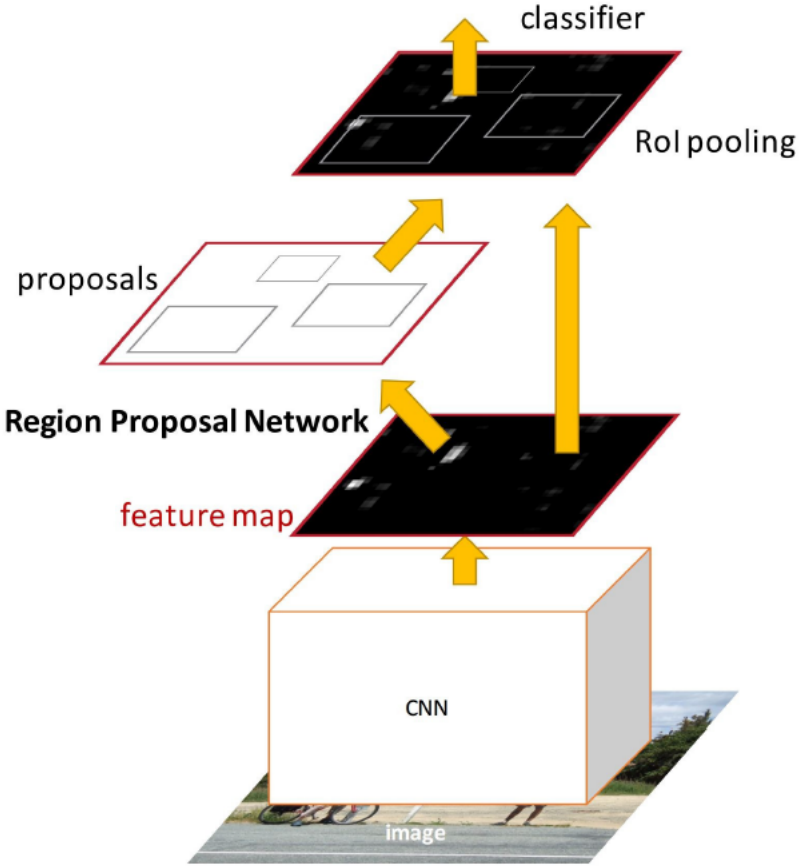
Fast R-CNN



Uses the CNN network once on the image and classifies the region projections in the feature map.

Faster R-CNN idea

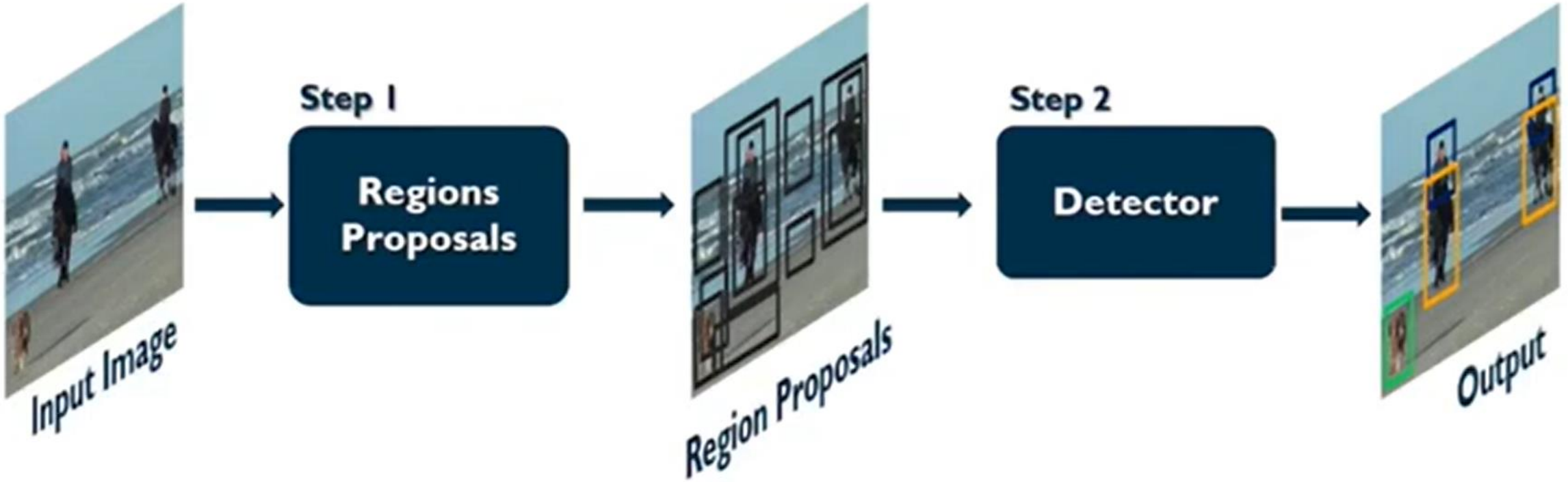
Replaces the region proposal algorithm with a significantly faster neural network that can learn to propose better regions for the task at hand.



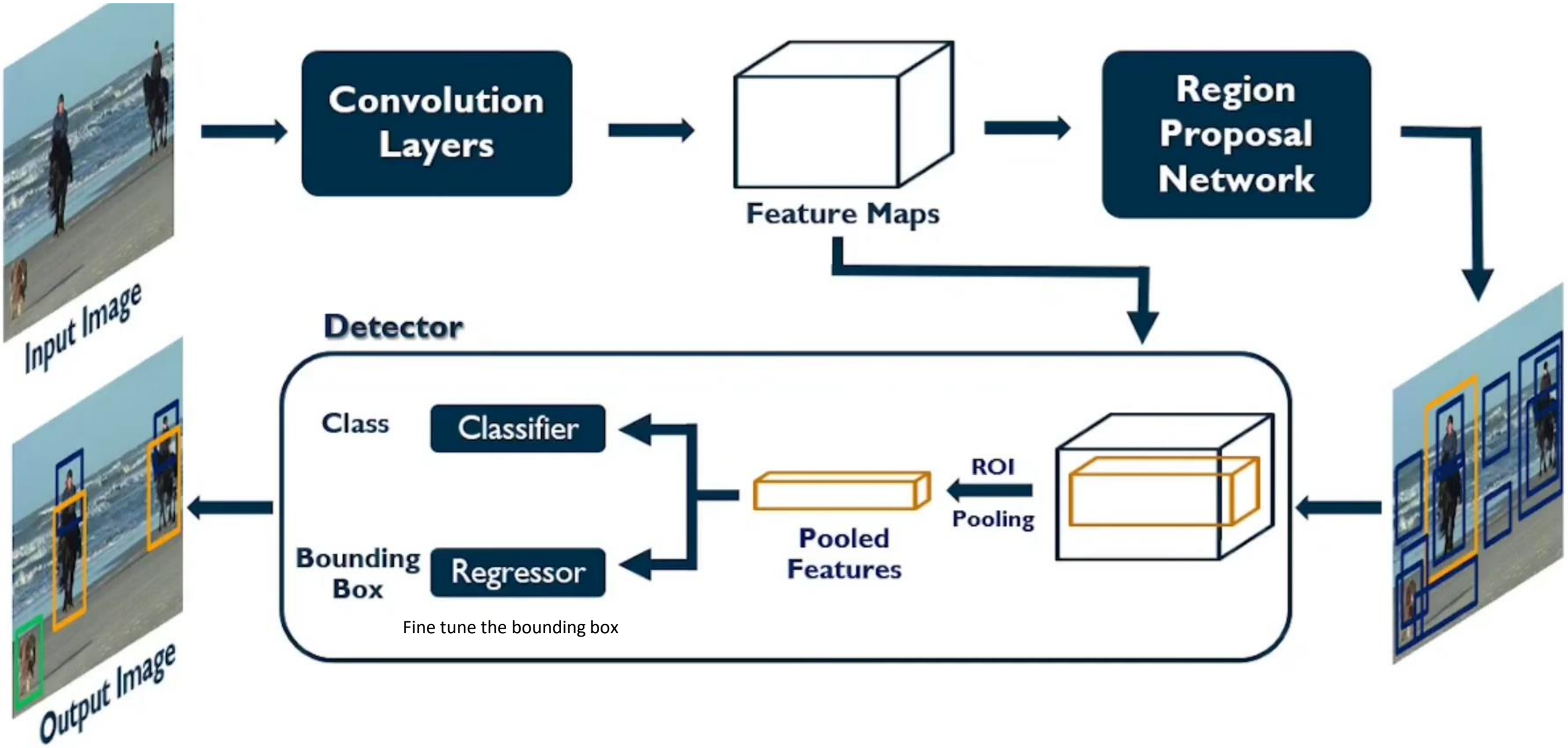
- Solution: Have the proposal generation integrated with the rest of the pipeline
- Region Proposal Network (RPN) trained to produce region proposals directly.
- After RPN, everything is like Fast R-CNN

Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Faster R-CNN Overview



Faster R-CNN Overview



Faster R-CNN Algorithm steps

Faster R-CNN, introduced by Shaoqing Ren et al. in 2015, is an improvement over Fast R-CNN that further reduces the computational cost of object detection. It achieves this by using a single CNN to generate both the ROIs and the features for each ROI, rather than using a separate CNN for each task, as in Fast R-CNN.

❖ The Faster R-CNN pipeline can be divided into four main steps:

1. **Feature extraction:** A CNN (modified VGG-16) extracts features from the entire image to generate a feature map
2. **Region proposal Network:** A set of Region of Interest(ROI) Proposals are generated for each location in the feature map using a fully convolutional network (FCN) that processes the extracted features. The network is initiated a set of anchor boxes (9 anchors: 3 scales X 3 aspect ratios). Several Regions proposals are obtained for each location each with a descriptor (Object present/absent and corrected anchor box coordinates). A set proposals are shortlisted using the Non-maximum suppression algorithm. (slides 84-93)
3. **ROI pooling:** The extracted features are then used to compute a fixed-length feature vector for each ROI using the same ROI pooling process as in Fast R-CNN. (slide 94)
4. **Faster R-CNN Detector (Classification and bounding box regression):** The fixed-length feature vectors for each ROI are fed into two separate Fully Connected (FC) layers: one for classification and one for bounding box regression. The classification FC layer predicts the object's class within the ROI, while the bounding box regression FC layer predicts the refined bounding box coordinates for the object. (slides 95-99)

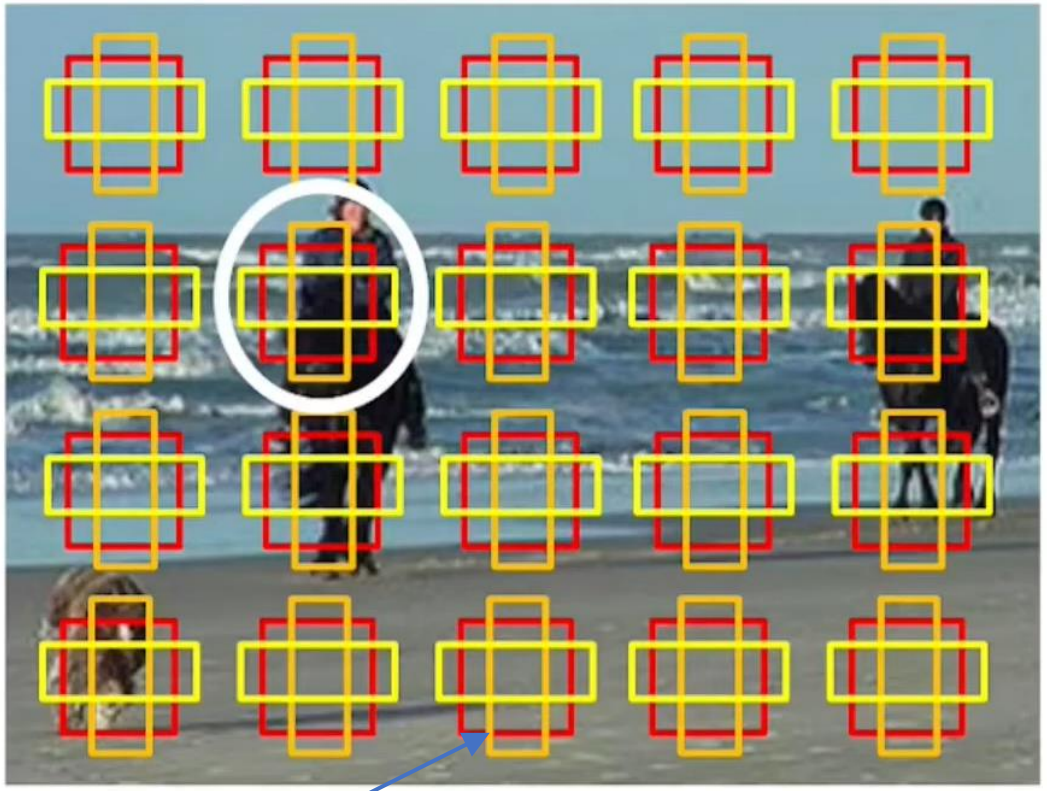
The Faster R-CNN uses VGG-16 (a CNN Network) as a base to extract features from the input image using Transfer learning and the VGG16 Network's FC and SoftMax Layers are modified and trained with K object classes required for the use case. VGG-16 originally, trained with 1000 ImageNet classes.

Cross entropy Loss is used for object classification and Smooth L1 loss is used for doing box regression

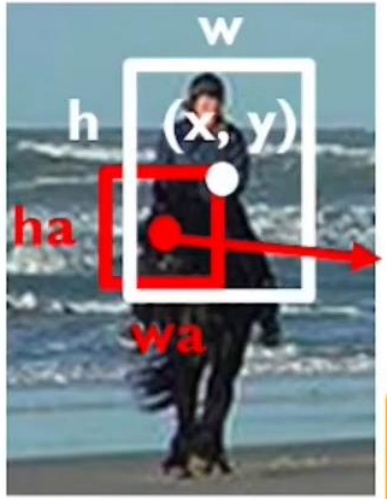
Faster R-CNN combines the feature extraction and region proposal steps of R-CNN and Fast R-CNN into a single CNN, making it more computationally efficient than both methods. It also uses an anchor box mechanism to handle multiple scales and aspect ratios, which can improve the robustness of object detection.

Region Proposal Network - overview

A CNN is used to predict and generate Regional Proposals.
Several Regions proposals are obtained. Shortlist a set proposals using Non-maximum suppression



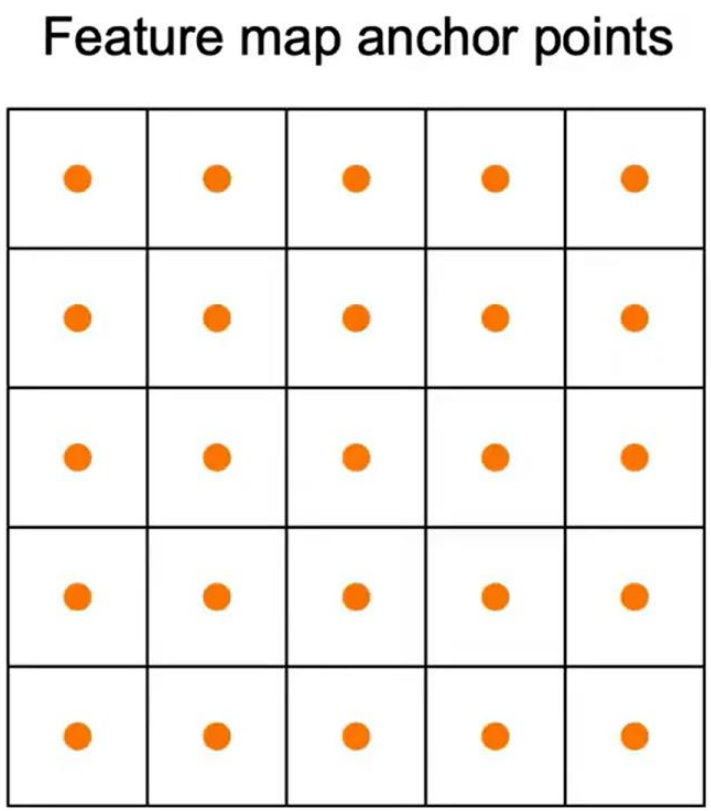
Anchor boxes (of different sizes and aspect ratios) per location
(3 here (9 used in original paper)
Anchor boxes support scale space.

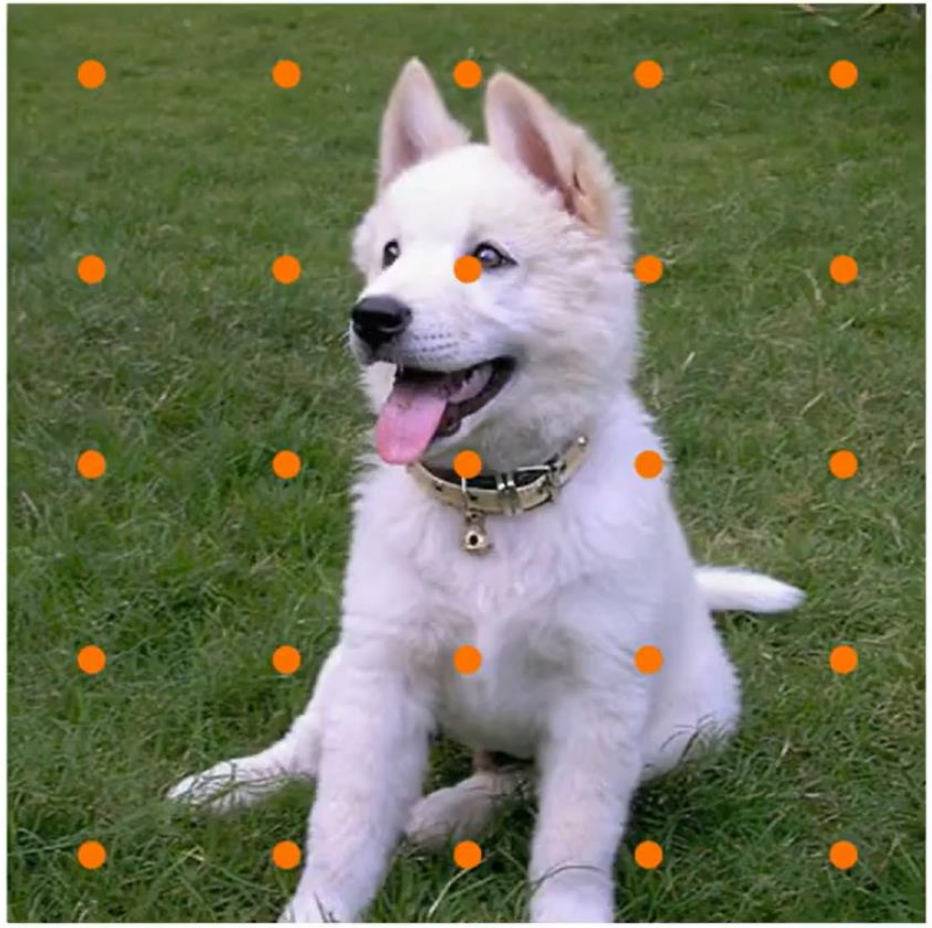


Region Proposal predicts a Objectness score

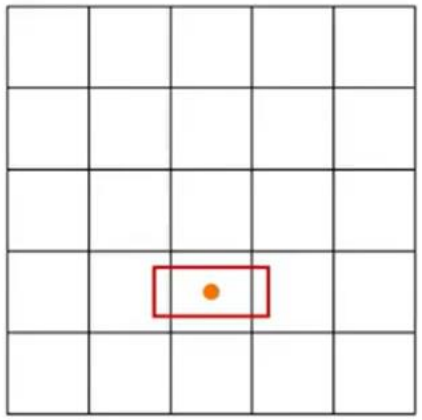
$$
\begin{aligned}
t_x &= (x - x_a)/w_a \\
t_y &= (y - y_a)/h_a \\
t_w &= \log(w/w_a) \\
t_h &= \log(h/h_a)
\end{aligned}
$$

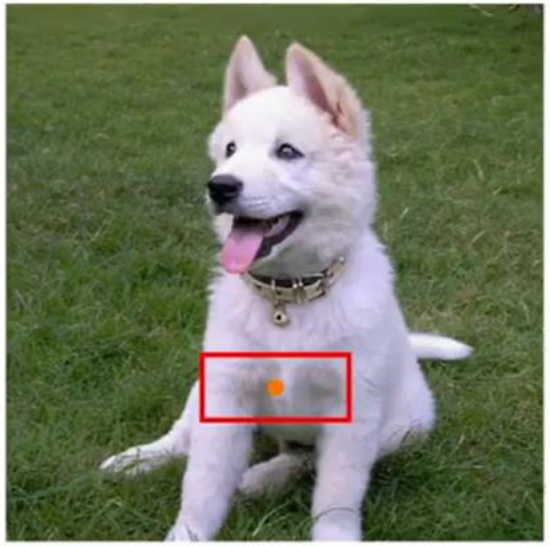
Anchor Points

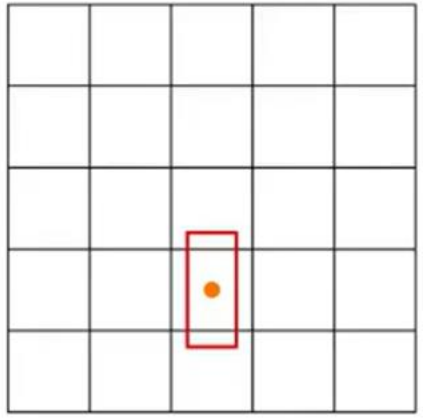


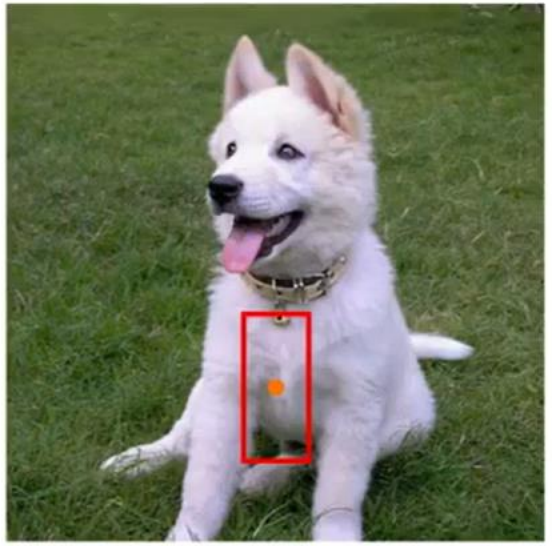


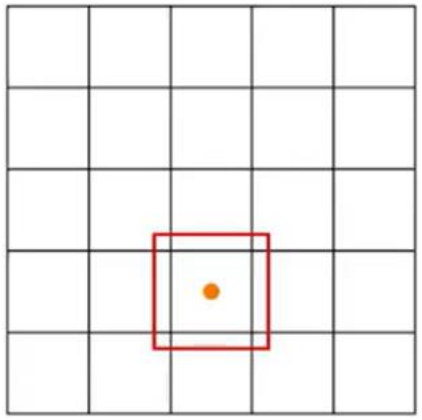
Anchor Boxes

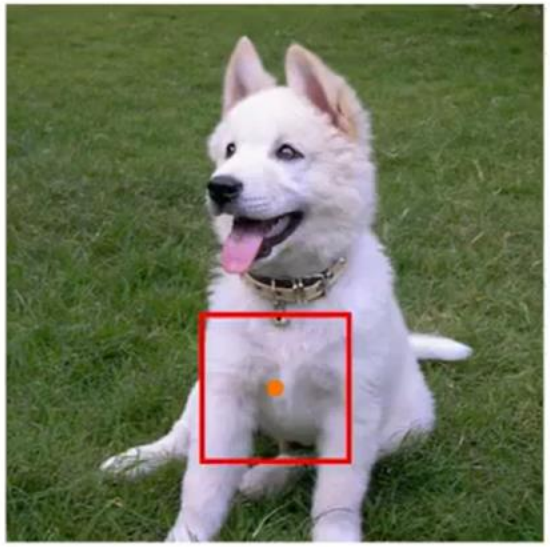


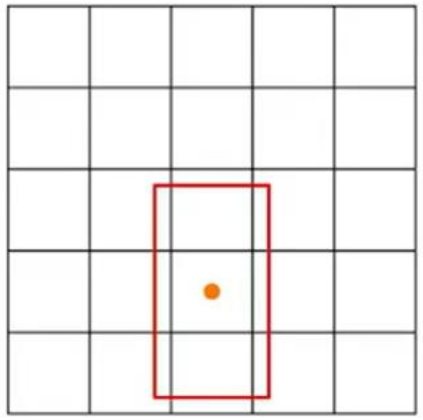


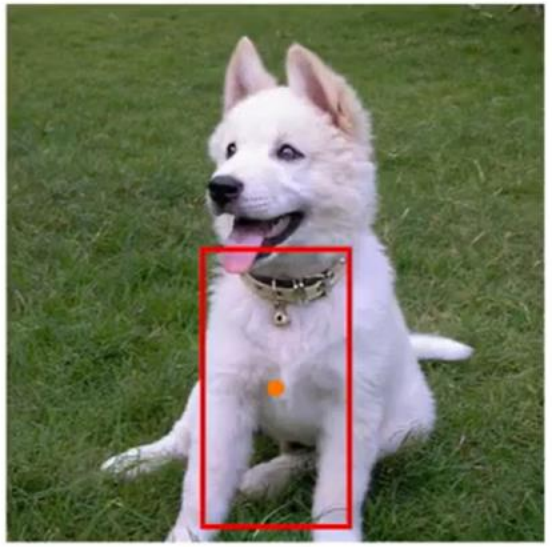






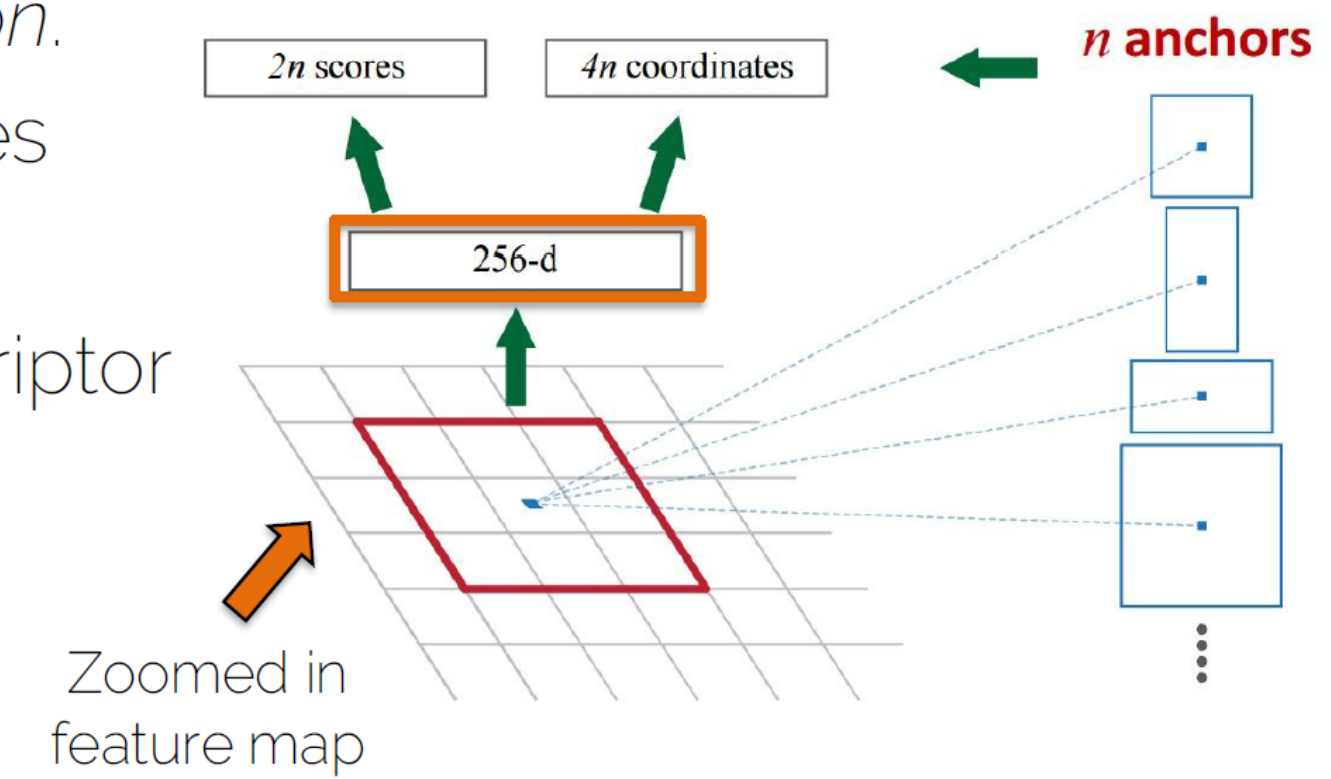






Region Proposal Network

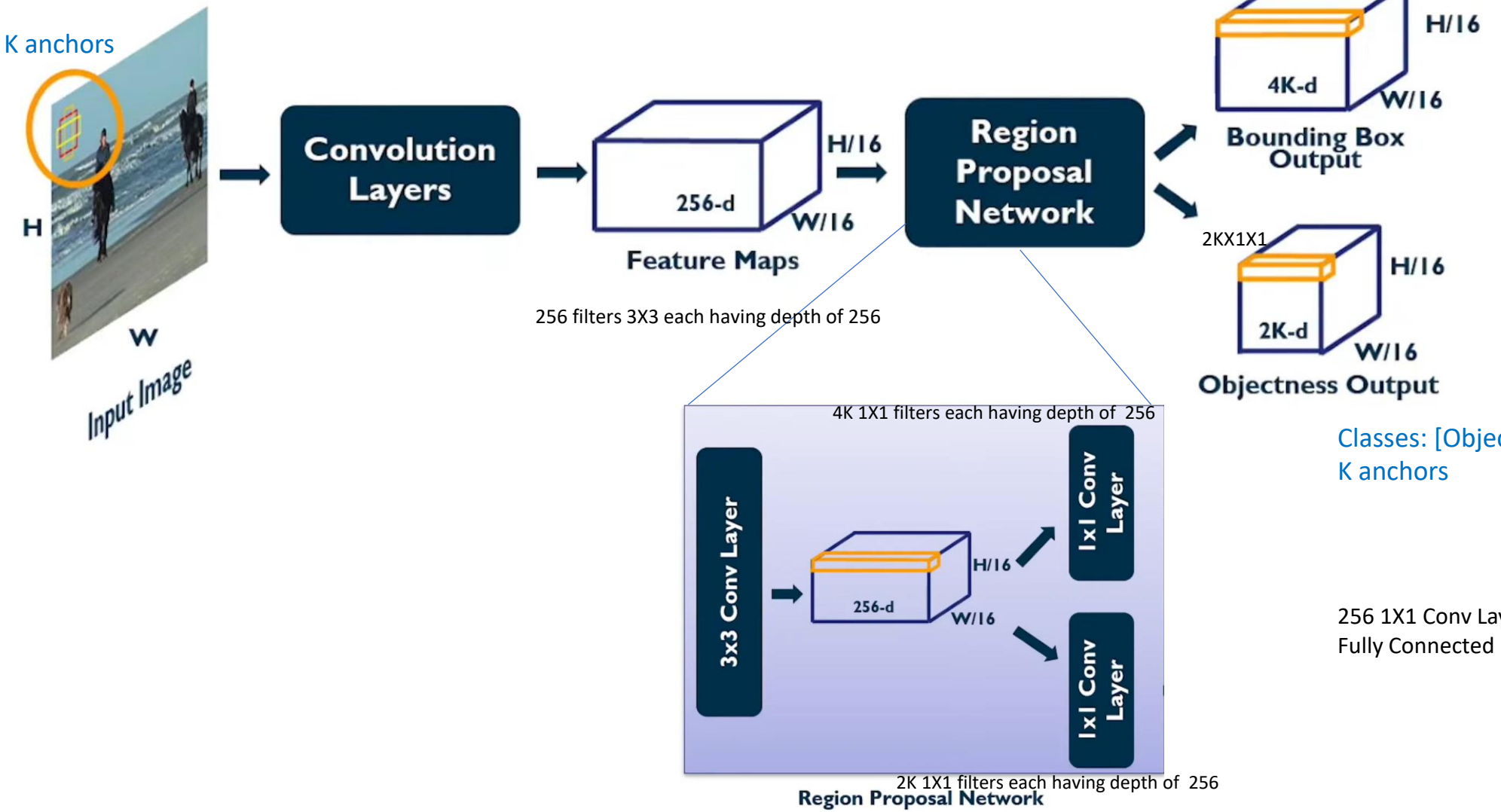
- We fix the number of proposals by using a set of n=9 anchors *per location*.
- 9 anchors = 3 scales and 3 aspect ratios
- We extract a descriptor per *location*



Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Region Proposal Network – Network Architecture

Bounding Box: [x,y, w, h]
K Anchors



Classes: [Object, No Object]
K anchors

256 1X1 Conv Layers can be interpreted as Fully Connected Layer

Region Proposal Network: Ground Truth and Loss functions

We need to assign a binary label (object / no-object) to each anchor.

A positive label is assigned to two kinds of anchors:

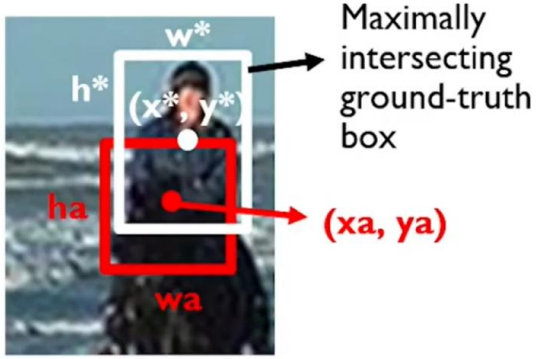
- The anchors with the highest Intersection-over-Union (IoU) overlap with a ground-truth box, or
- An anchor that has an IoU overlap higher than 0.7 with any ground-truth box.

A negative label to a non-positive anchor if its IoU ratio is lower than 0.3 for all ground-truth boxes.

Anchors that are neither positive nor negative do not contribute to the training objective.

Next, to every positive anchor, we assign a unique ground-truth bounding box with which it has max IOU.

$$t_x^* = (x^* - x_a)/w_a$$
$$t_y^* = (y^* - y_a)/h_a$$
$$t_w^* = \log(w^*/w_a)$$
$$t_h^* = \log(h^*/h_a)$$

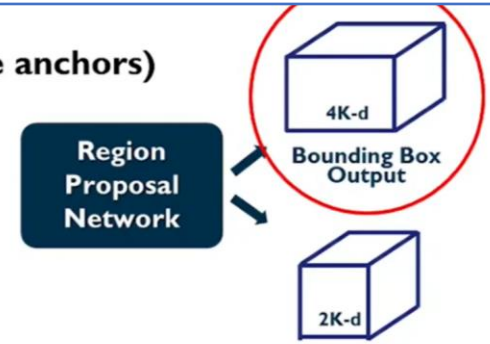


Loss for reg layer (calculated only for +ve anchors)

- $L_{reg}(ti, ti^*) = \text{smooth}L_1\text{loss}\ (ti - ti^*)$

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

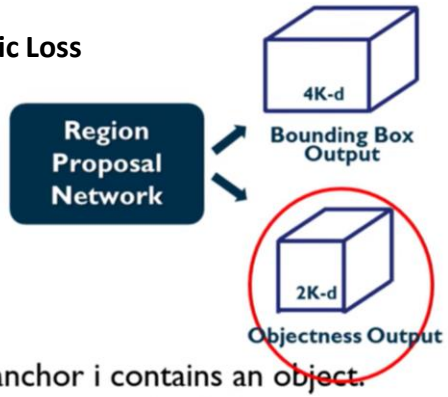
Where x will be the L1 distance between 2 vectors.



- Here, i is the index of an anchor,
- ti is a vector representing the 4 parameterized coordinates of the predicted bounding box,
- ti* is that of the ground-truth box associated with a positive anchor.

Loss for cls layer **Cross-Entropy/Logistic Loss**

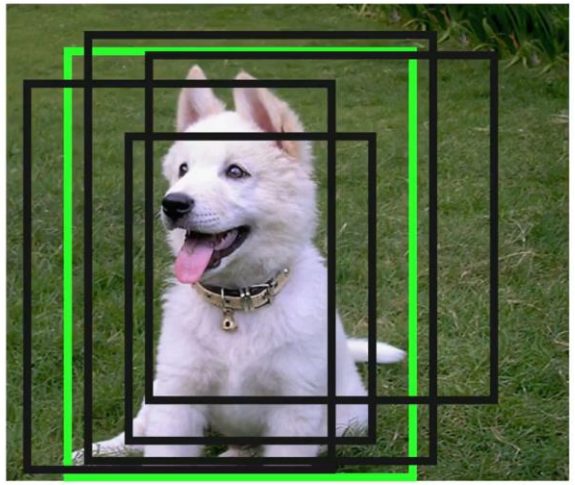
- $L_{cls}(pi, pi^*) = -\ pi^* \times \log(pi)$



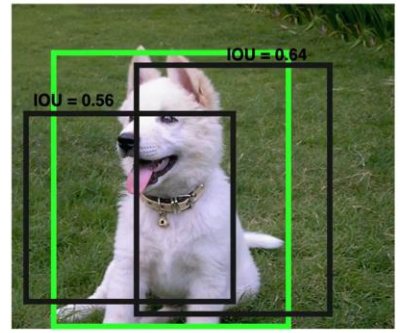
- Here, i is the index of an anchor,
- pi is the predicted probability that an anchor i contains an object.
- The ground-truth pi* is 1 if the anchor is +ve, and is 0 if -ve

https://mohitjain.me/wp-content/uploads/2018/03/smoothl1loss.pdf

Which anchor bounding boxes to use for training?



Which anchor bounding box to use for a specific object during training?



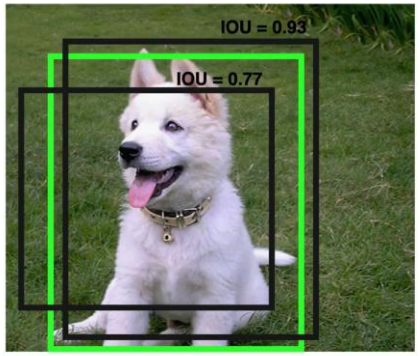
0.5 < IoU ≤ 0.7

+

no IoU > 0.7

⇩

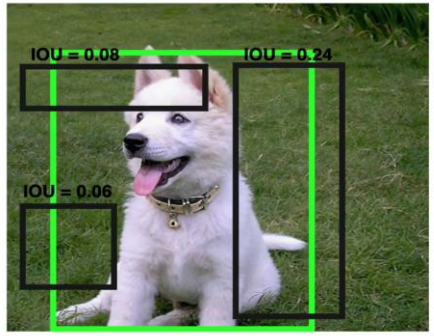
highest one positive



IoU > 0.7

⇩

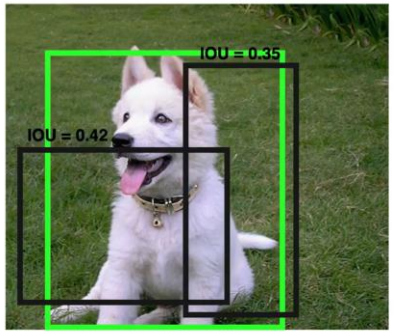
positive



IoU < 0.3

⇩

negative



0.3 ≤ IoU ≤ 0.5

⇩

not positive not negative

Smooth L1 Loss

- The Smooth L1 loss is used for doing box regression on some object detection systems, (SSD, Fast/Faster RCNN) according to those papers this loss is less sensitive to outliers, than other regression loss, like L2 which is used on R-CNN and SPPNet.
- On the Fast RCNN paper, section 2.3 is claimed that the L2 loss need a smaller learning rate to avoid exploding gradients.
- What is L1 Loss?
  - L1 loss is also known as mean absolute loss or mean absolute error. It's simply the summation of the absolute difference between actual and predicted values.

$$\text{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

  - Here, MAE: Mean absolute error , y_i = prediction value, x_i = true value and n = no of values
- What is Smooth L1 Loss?

$$Smooth - L1(x) = \begin{cases} 0.5x^2, if\ |x| < 1 \\ |x| - 0.5, otherwise \end{cases}$$

Where x will be the L1 distance between 2 vectors.

- Example shows L1 and Smooth L1 Loss between two vectors having 3 elements. Bounding box anchors will have 4 elements (x, y, w, h)

Defining L1 or Manhatan distance

Let

$p := [1, 2, 3]$

$[1, 2, 3]$

$q := [1.1, 2, 3.3]$

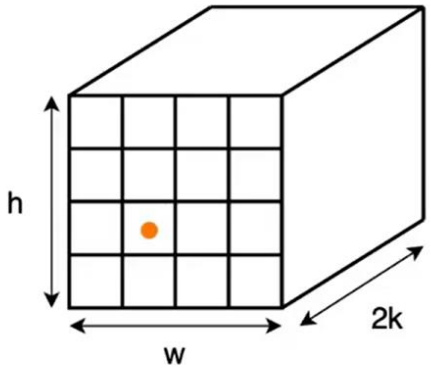
$[1.1, 2, 3.3]$

$$L_1 := (a, b, N) \to \sum_{i=1}^{N} |a[i] - b[i]|$$

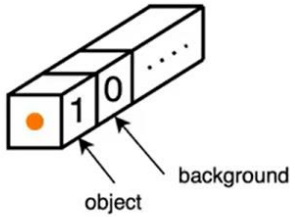
L1 (p, q, 3)=0.4 ➔ x=0.4

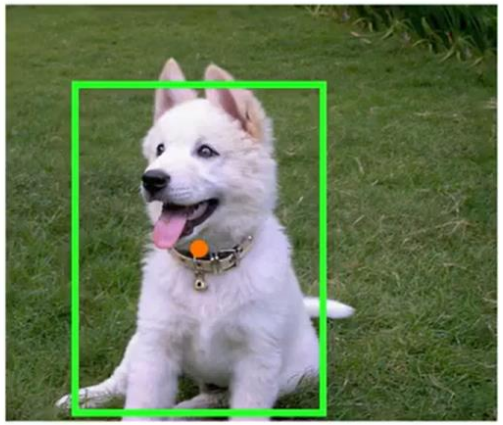
Smooth L1 (p,q, number of elements(p)) = 0.5 * 0.4 *0.4 = 0.080

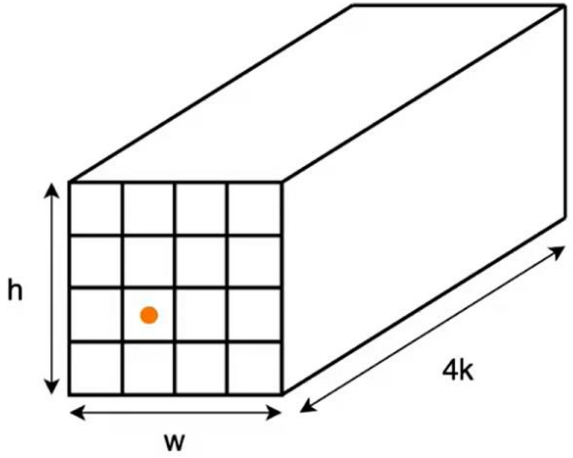
Sample 1: Object Classifier and Bounding Box Regressor



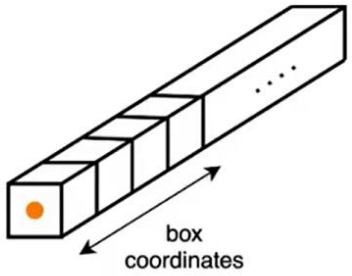
Object Classifier



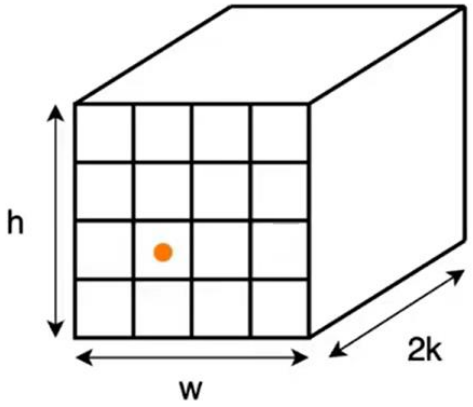




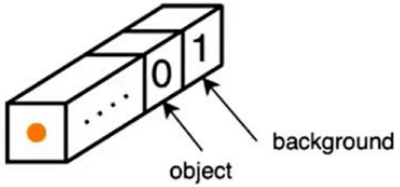
Bounding Box Regressor

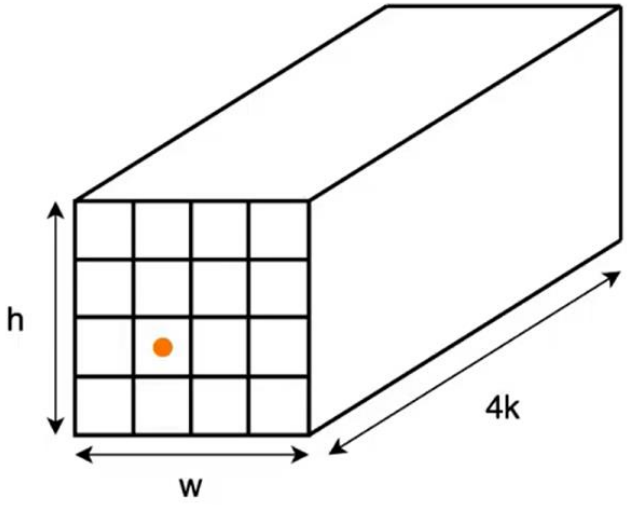


Sample 2: Object Classifier and Bounding Box Regressor

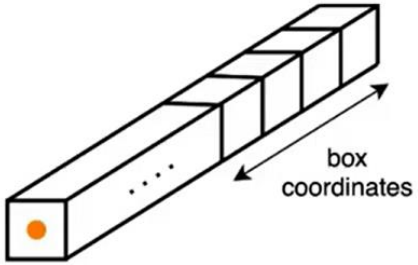


Object Classifier



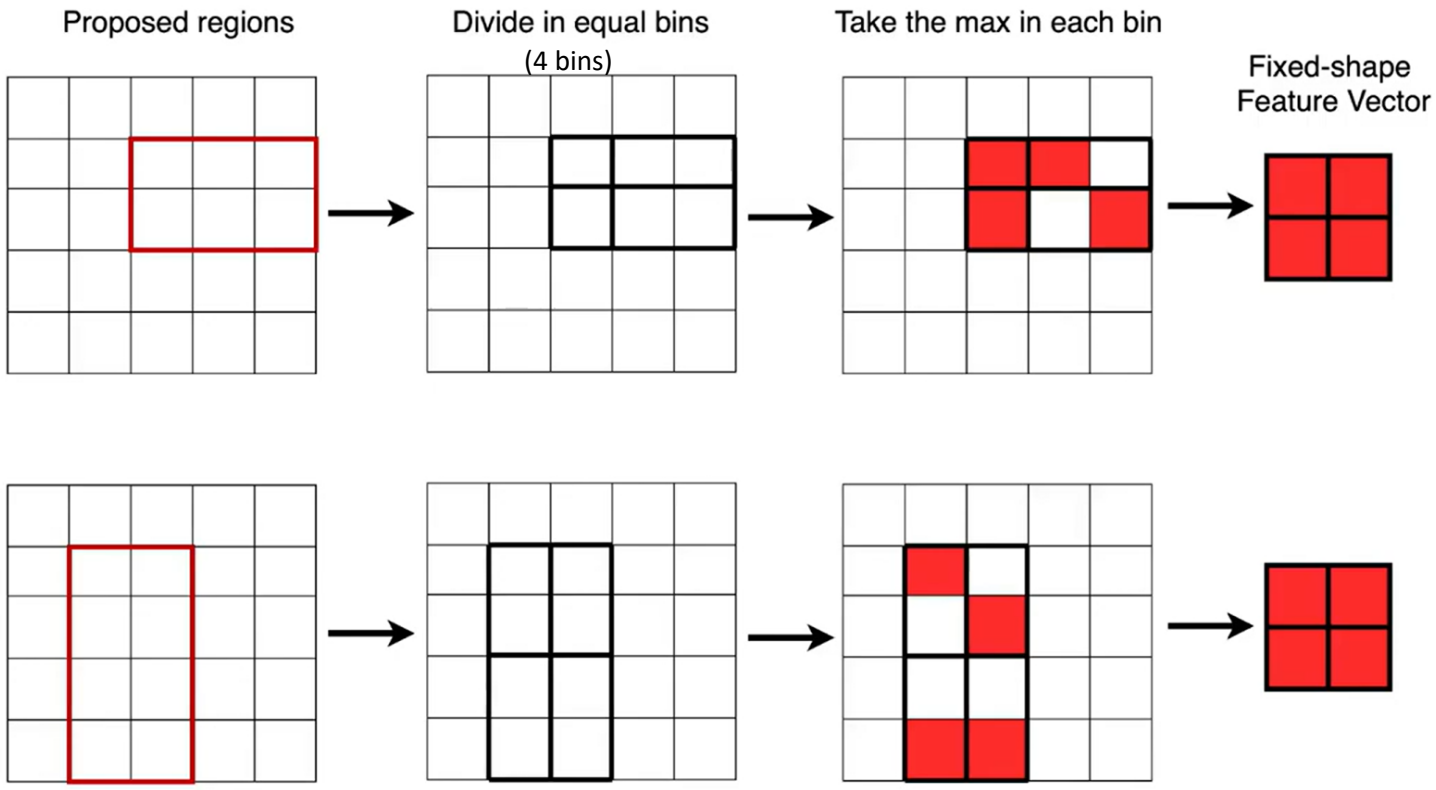


Bounding Box Regressor



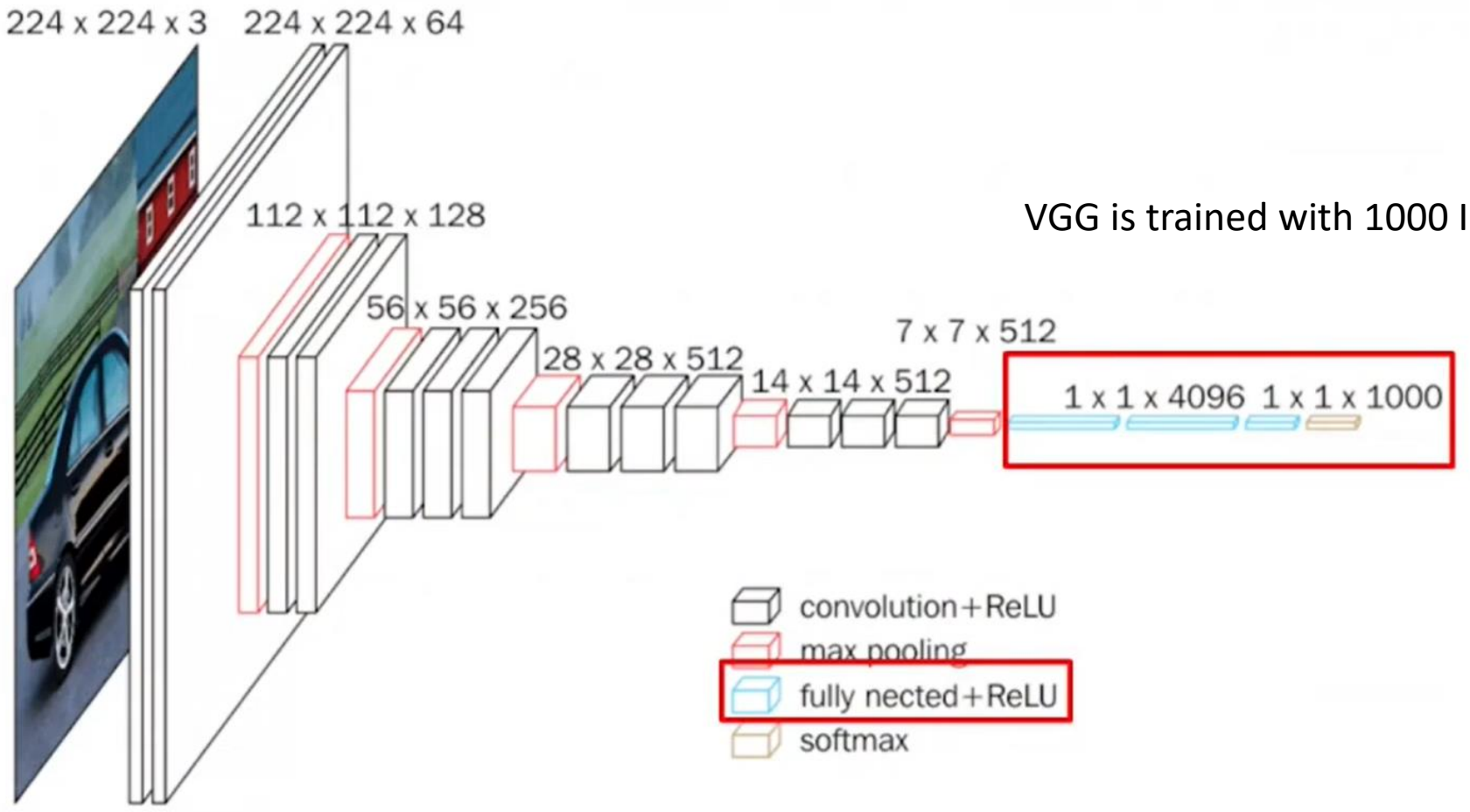
Faster R-CNN: Region of Interest (ROI) Pooling (same as Fast R-CNN)

- Fast R-CNN uses Region of Interest (ROI) Pooling Layer to extract a fixed length feature vector for each region proposal.



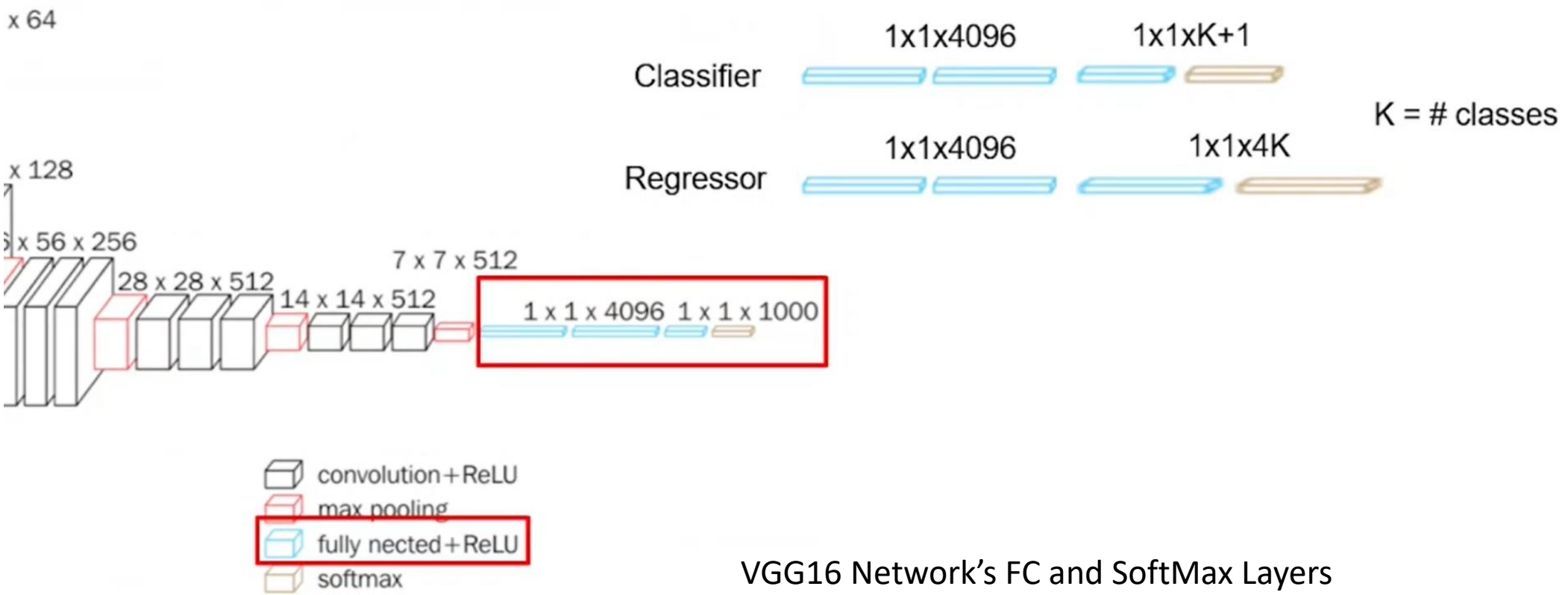
- ROI Pooling Layer divides each of Proposed regions into a fixed number of rectangular beams independent of the input shape and outputs a single value by performing Max Pooling with in each bin.

Faster R-CNN Detector: VGG-16 is used as a base



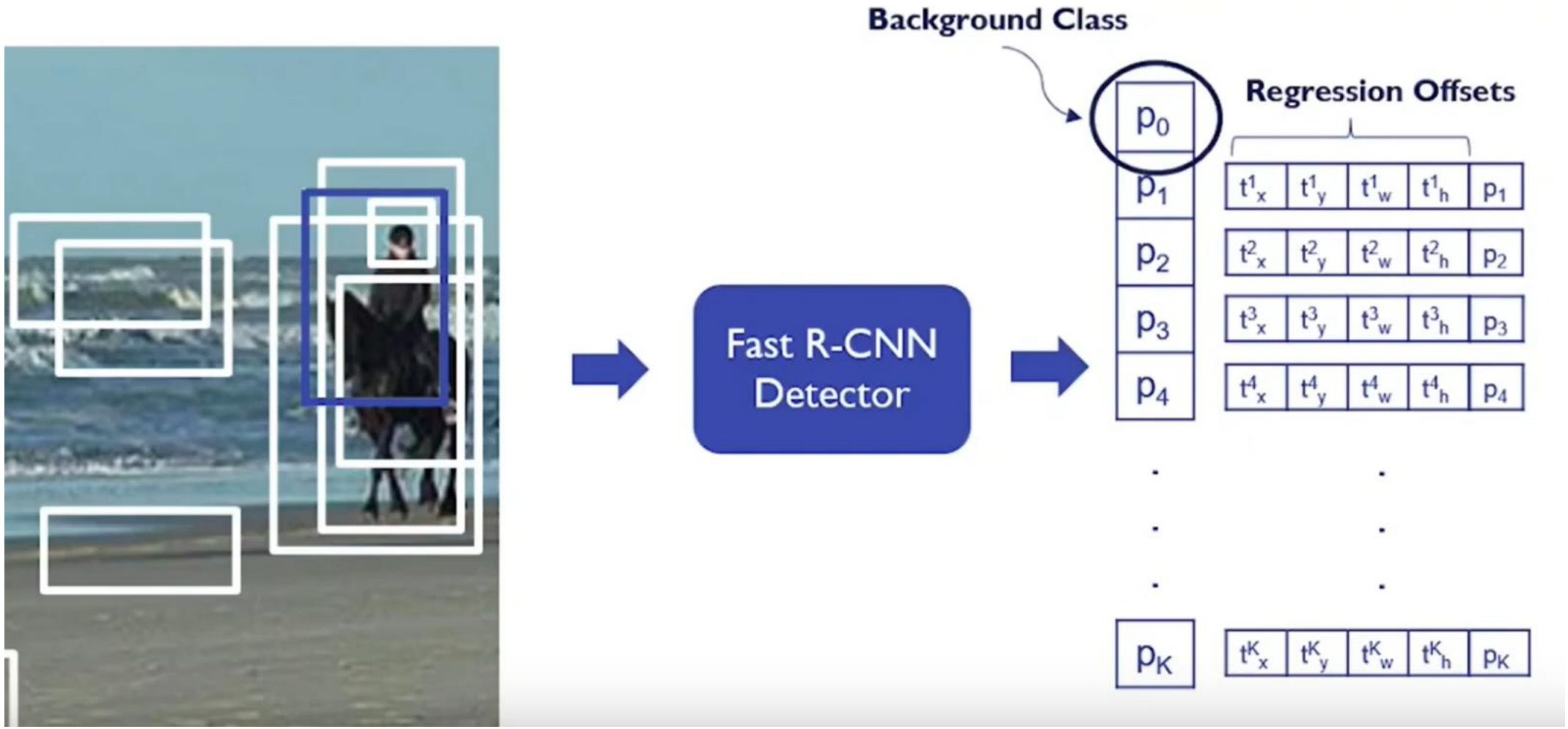
VGG is trained with 1000 ImageNet classes

Modified VGG16 used as Faster R-CNN Detector



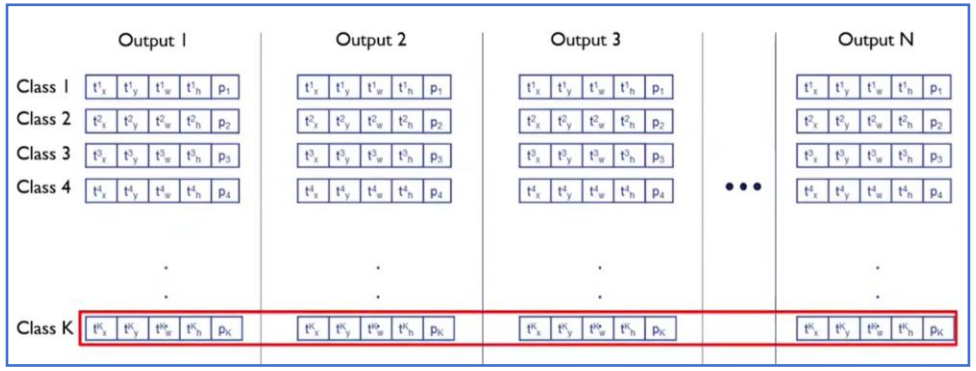
VGG16 Network's FC and SoftMax Layers modified to train it with K object classes

Testing

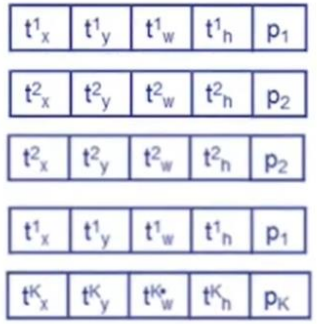

Background Class
Regression Offsets
Fast R-CNN Detector
p_0
p_1
p_2
p_3
p_4
p_K
t^1_x t^1_y t^1_w t^1_h p_1
t^2_x t^2_y t^2_w t^2_h p_2
t^3_x t^3_y t^3_w t^3_h p_3
t^4_x t^4_y t^4_w t^4_h p_4
t^K_x t^K_y t^K_w t^K_h p_K

Candidates + Non-Maximum Suppression ➔ Final output

a) Candidate bounding boxes predicted for N proposals

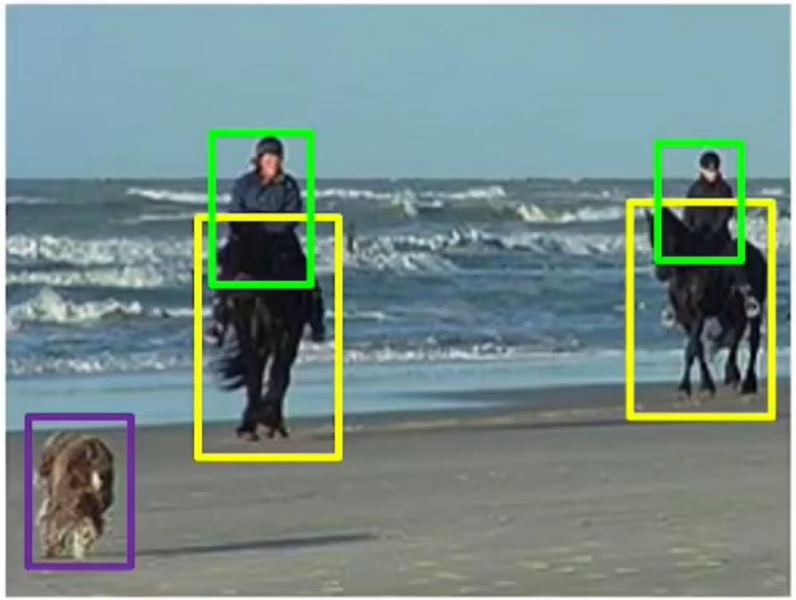


b) Suppress redundant bounding boxes using Non-maximum suppression



c) Displaying final output (using the formula given below)

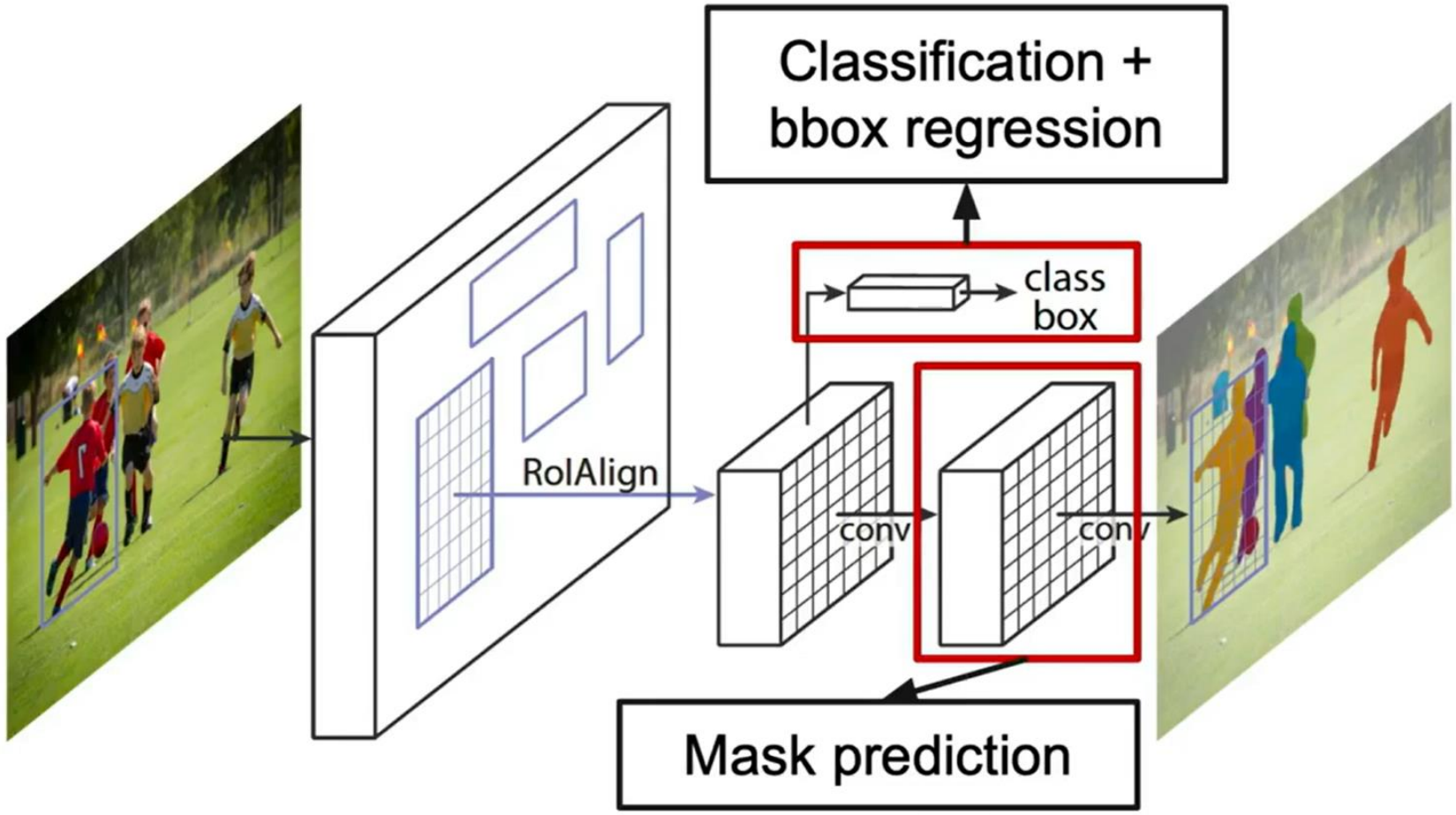
$$t^u_x = (x^o - x)/w$$
$$t^u_y = (y^o - y)/h$$
$$t^u_w = log(w^o/w)$$
$$t^u_h = log(h^o/h)$$



Object Detection

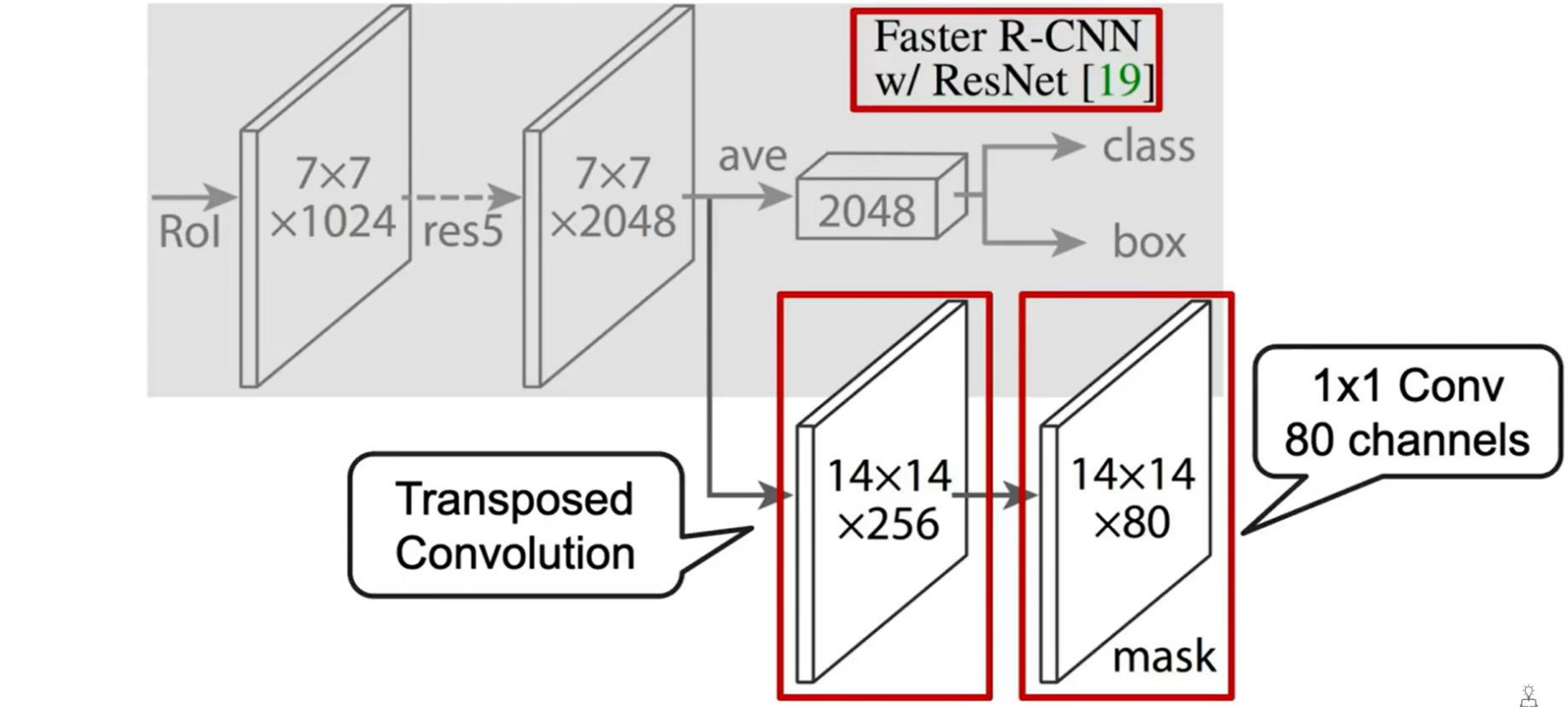
Mask R-CNN

Mask R-CNN



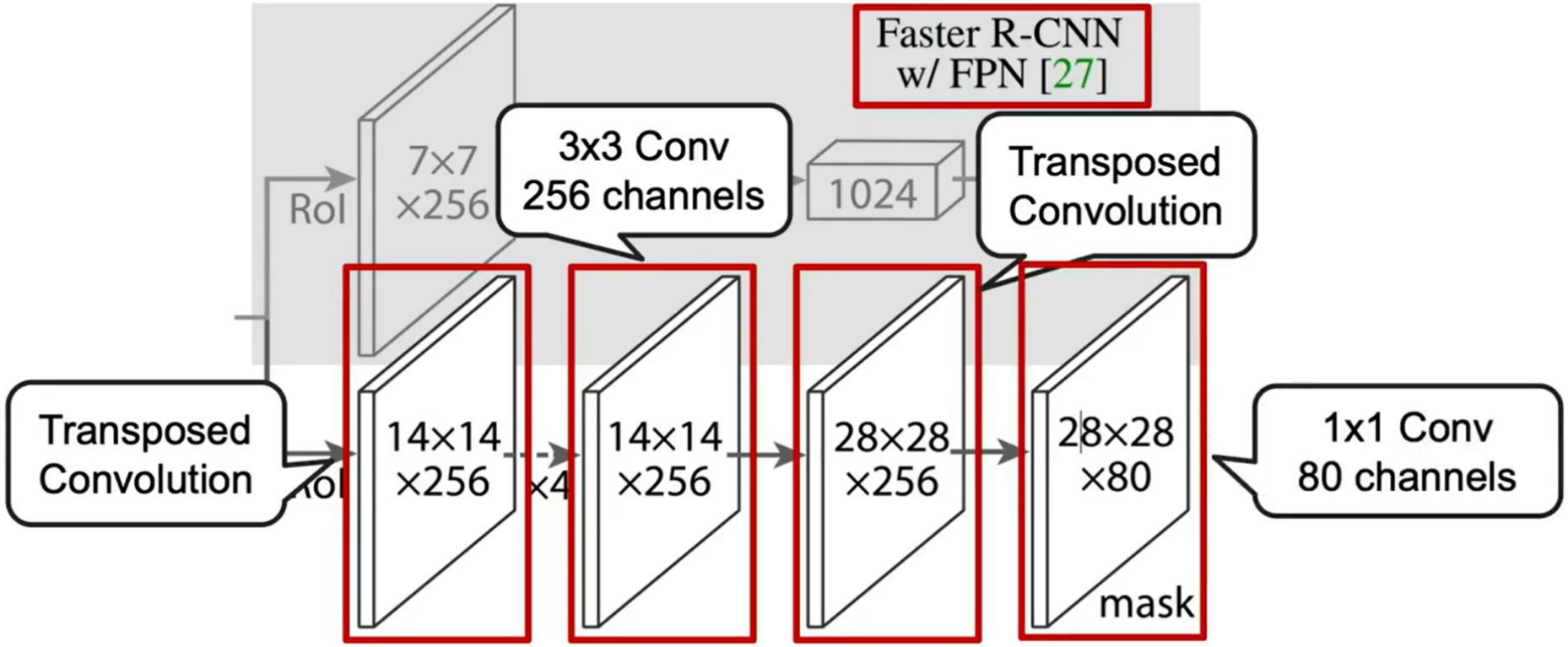
He, Kaiming, et al. "Mask r-cnn." Proceedings of the IEEE international conference on computer vision. 2017.

Mask Branch - ResNet



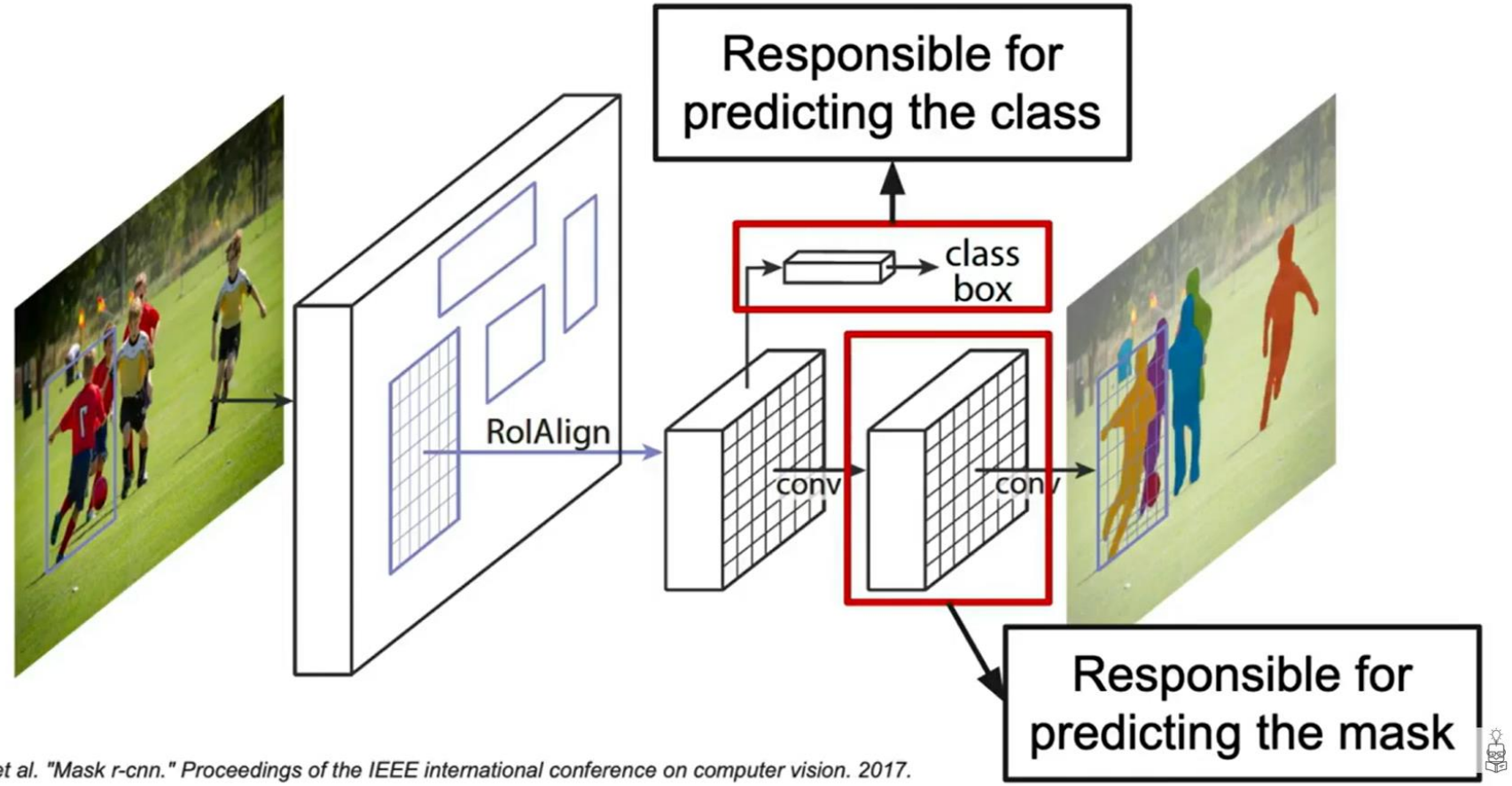
He, Kaiming, et al. "Mask r-cnn." Proceedings of the IEEE international conference on computer vision. 2017.

Mask Branch - FPN



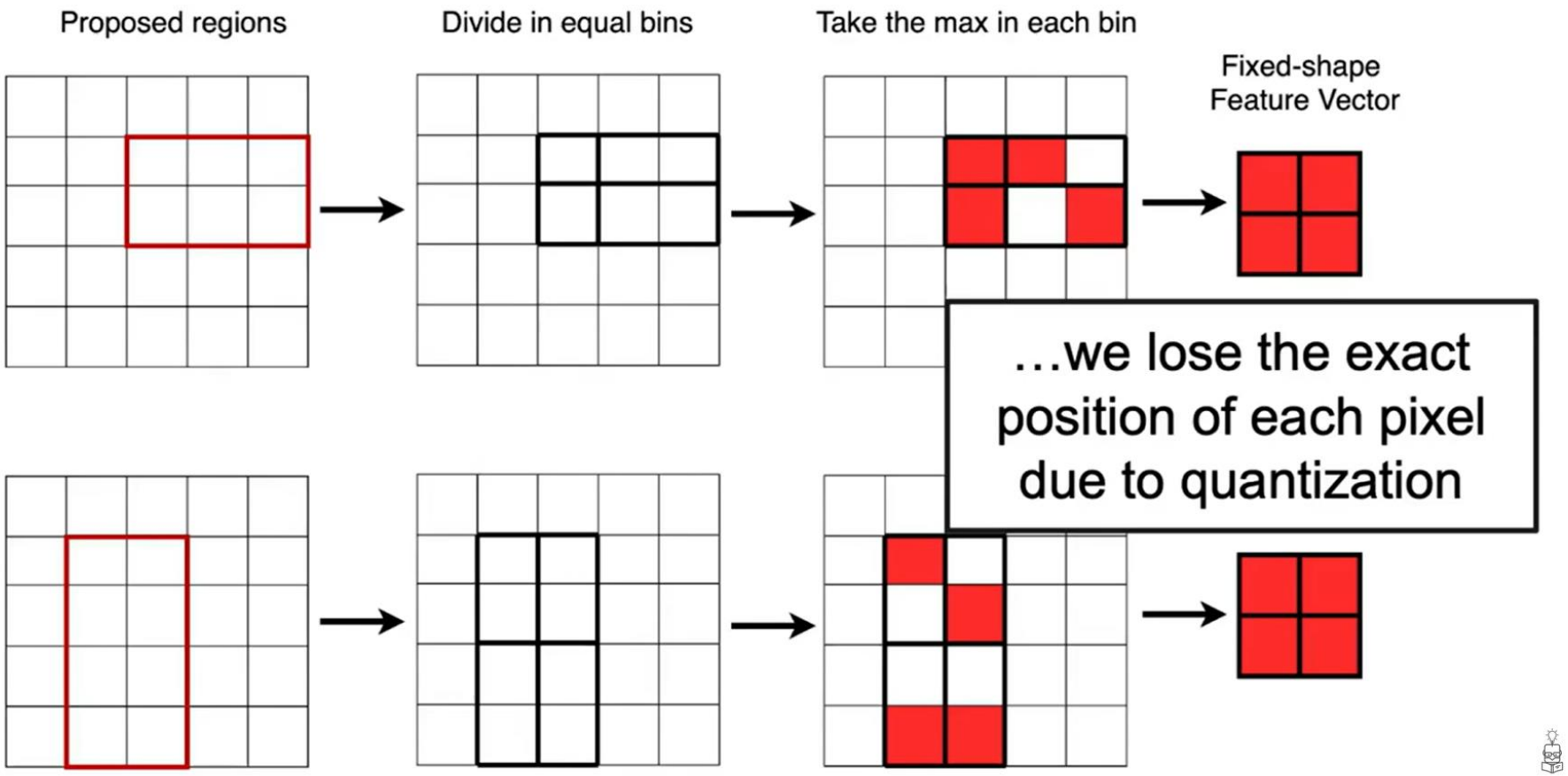
He, Kaiming, et al. "Mask r-cnn." Proceedings of the IEEE international conference on computer vision. 2017.

Mask R-CNN Decoupling

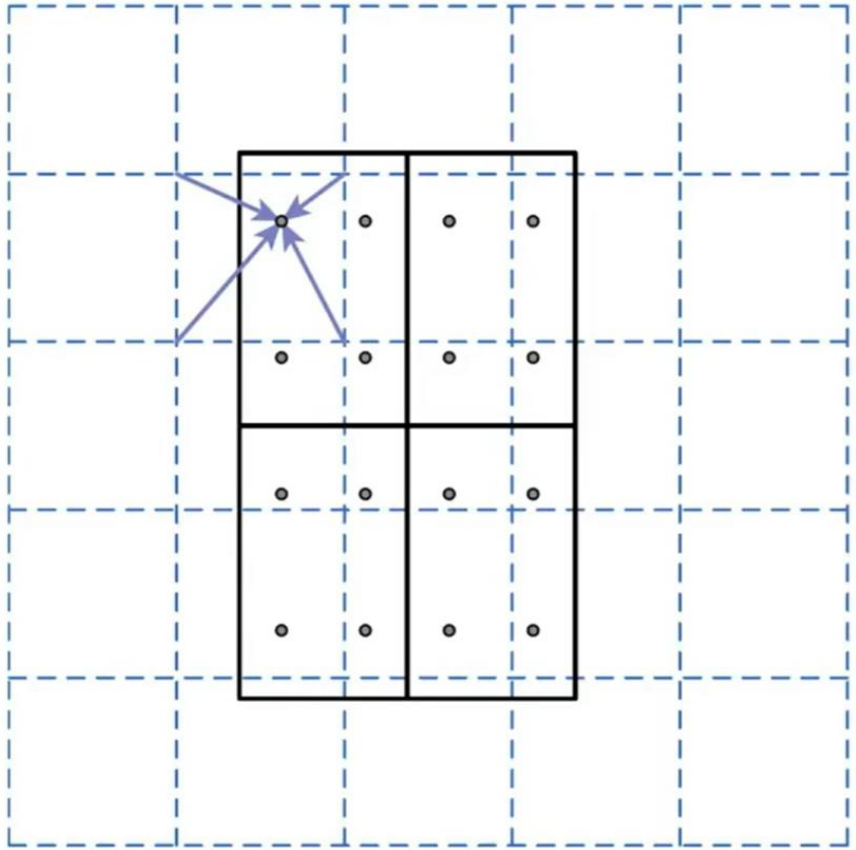


He, Kaiming, et al. "Mask r-cnn." Proceedings of the IEEE international conference on computer vision. 2017.

ROI Pooling


Proposed regions
Divide in equal bins
Take the max in each bin
Fixed-shape
Feature Vector
…we lose the exact position of each pixel due to quantization

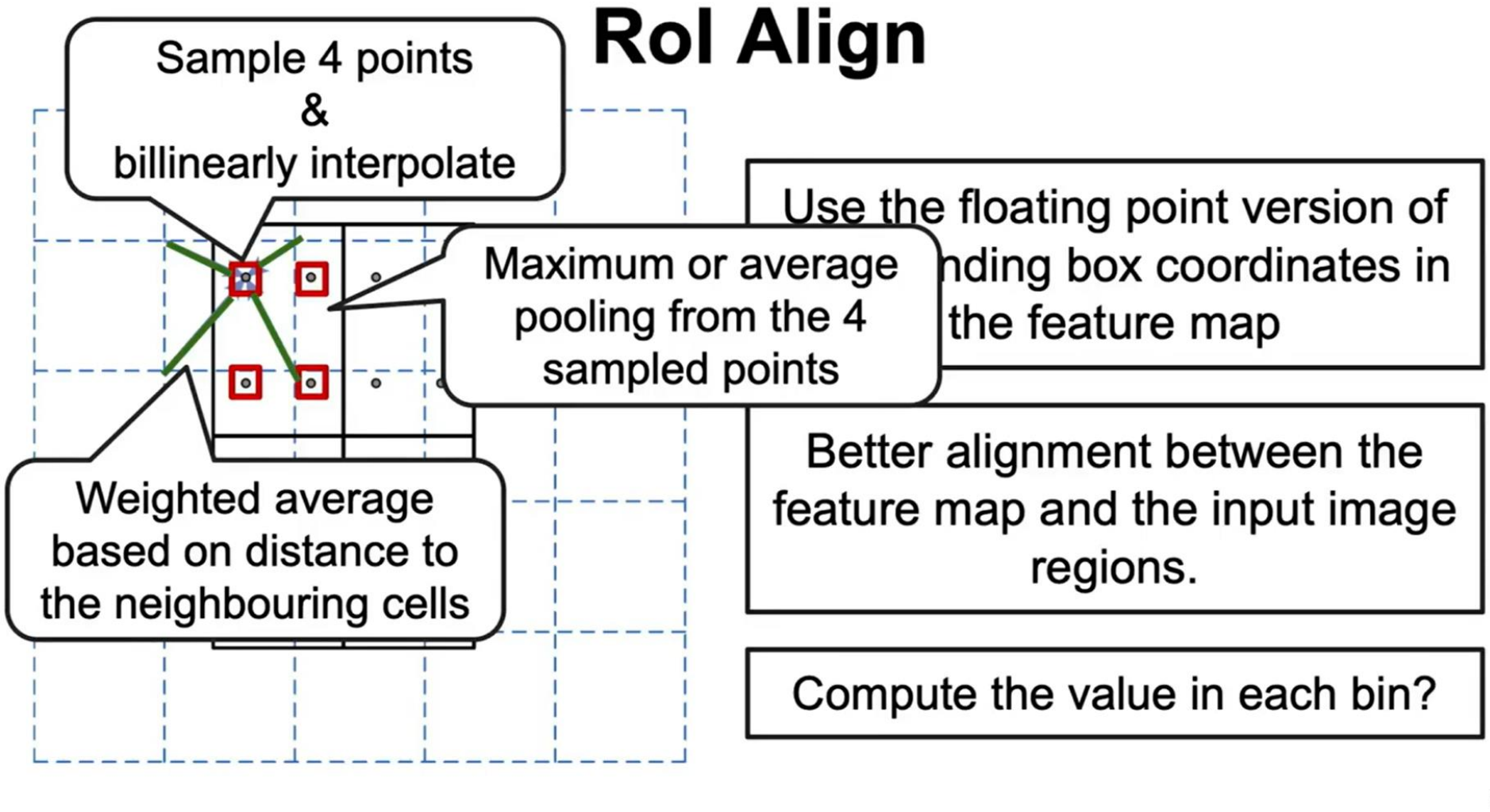
ROI Align



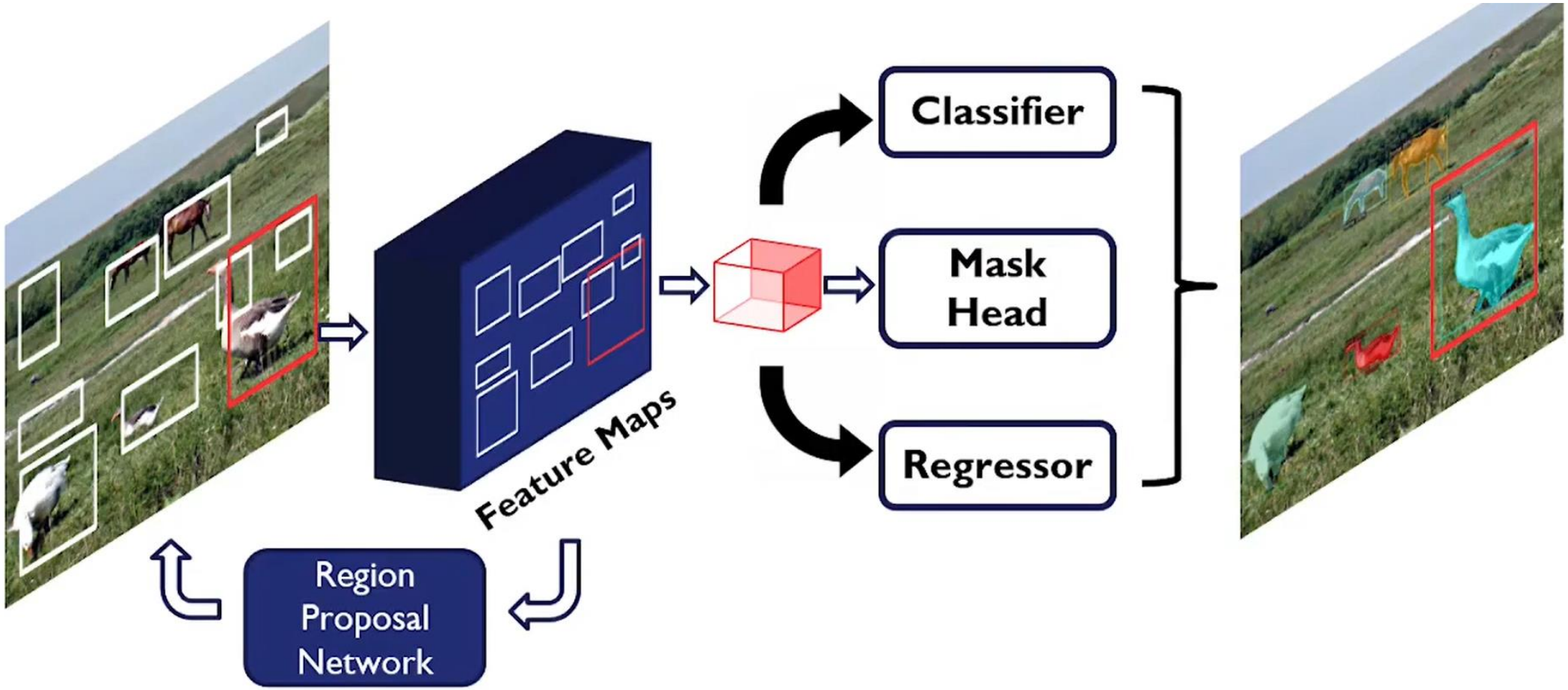
Use the floating point version of the bounding box coordinates in the feature map

Better alignment between the feature map and the input image regions.

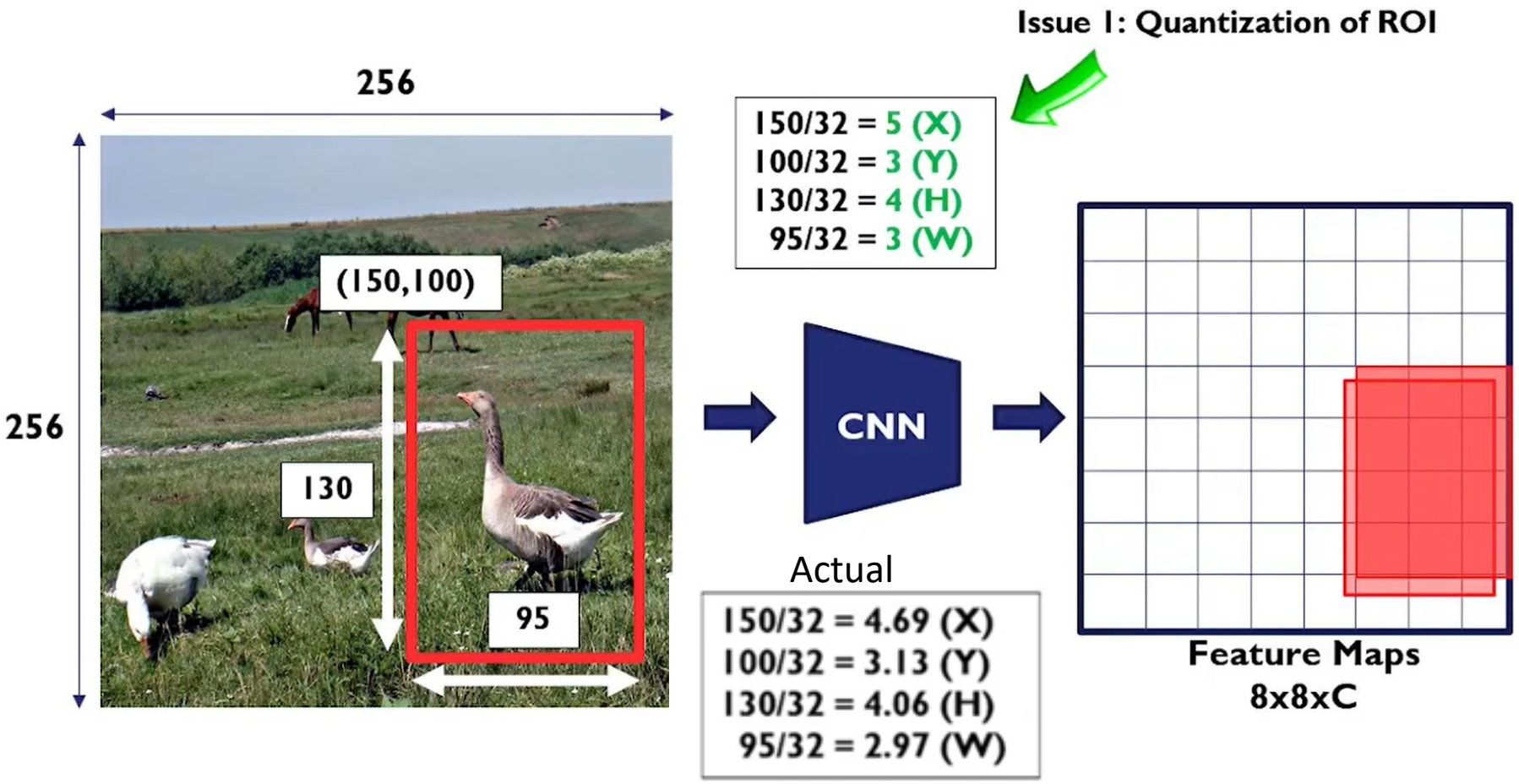
ROI Align


RoI Align
Sample 4 points
&
billinearly interpolate
Maximum or average
pooling from the 4
sampled points
Weighted average
based on distance to
the neighbouring cells
Use the floating point version of
nding box coordinates in
the feature map
Better alignment between the
feature map and the input image
regions.
Compute the value in each bin?

Mask R-CNN Overview

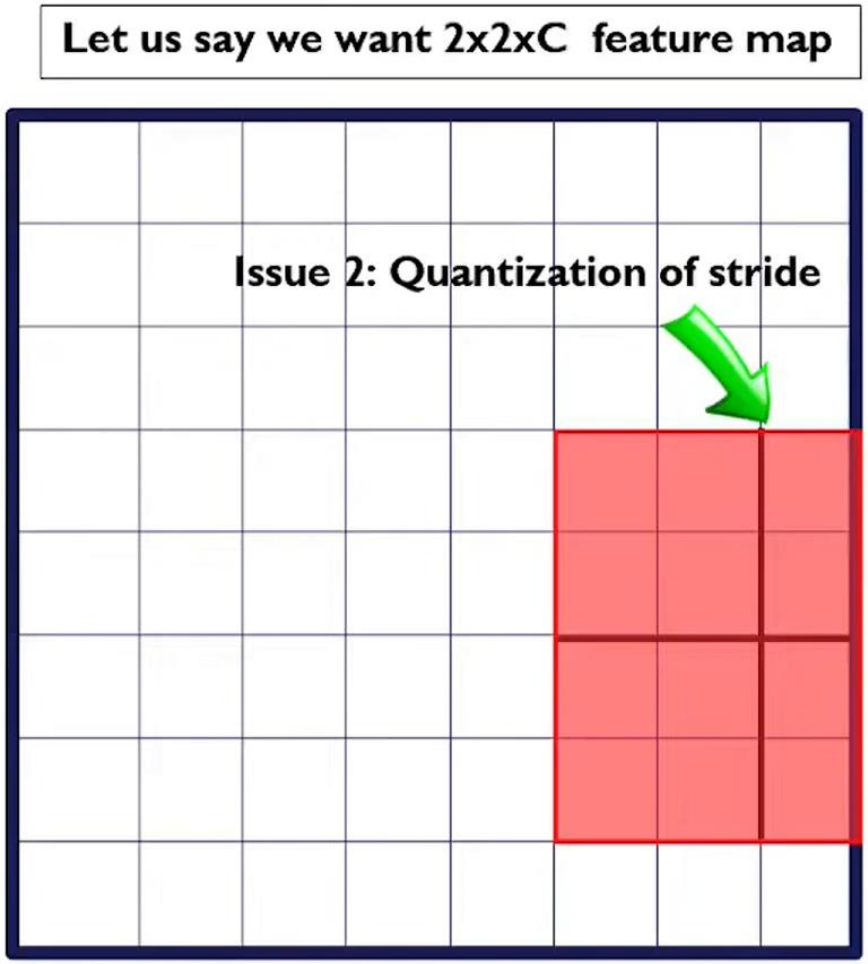


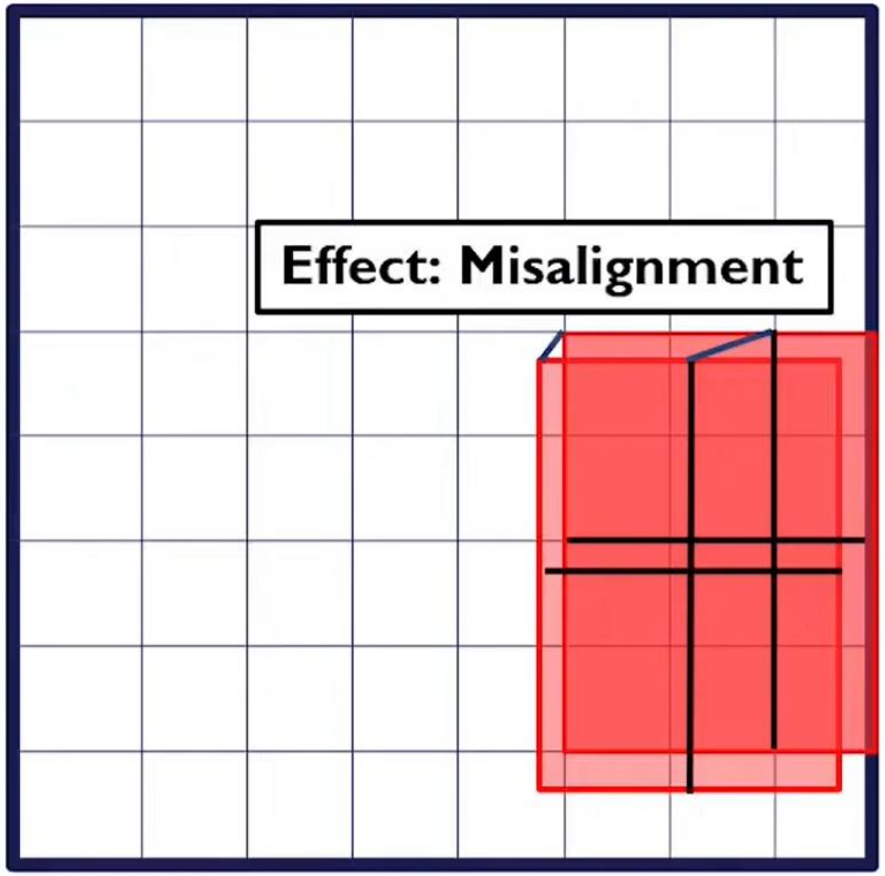
Mask R-CNN Overview



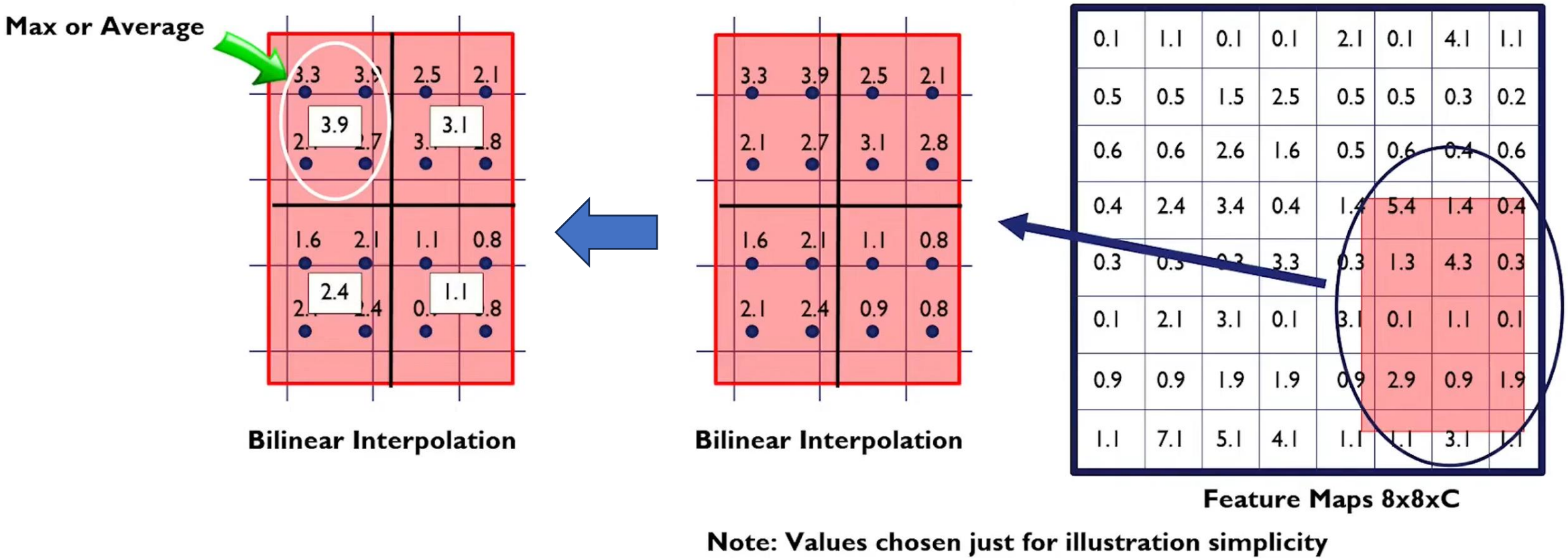
Note: Values chosen just for illustration simplicity

ROI Alignment: Quantization of stride and misalignment



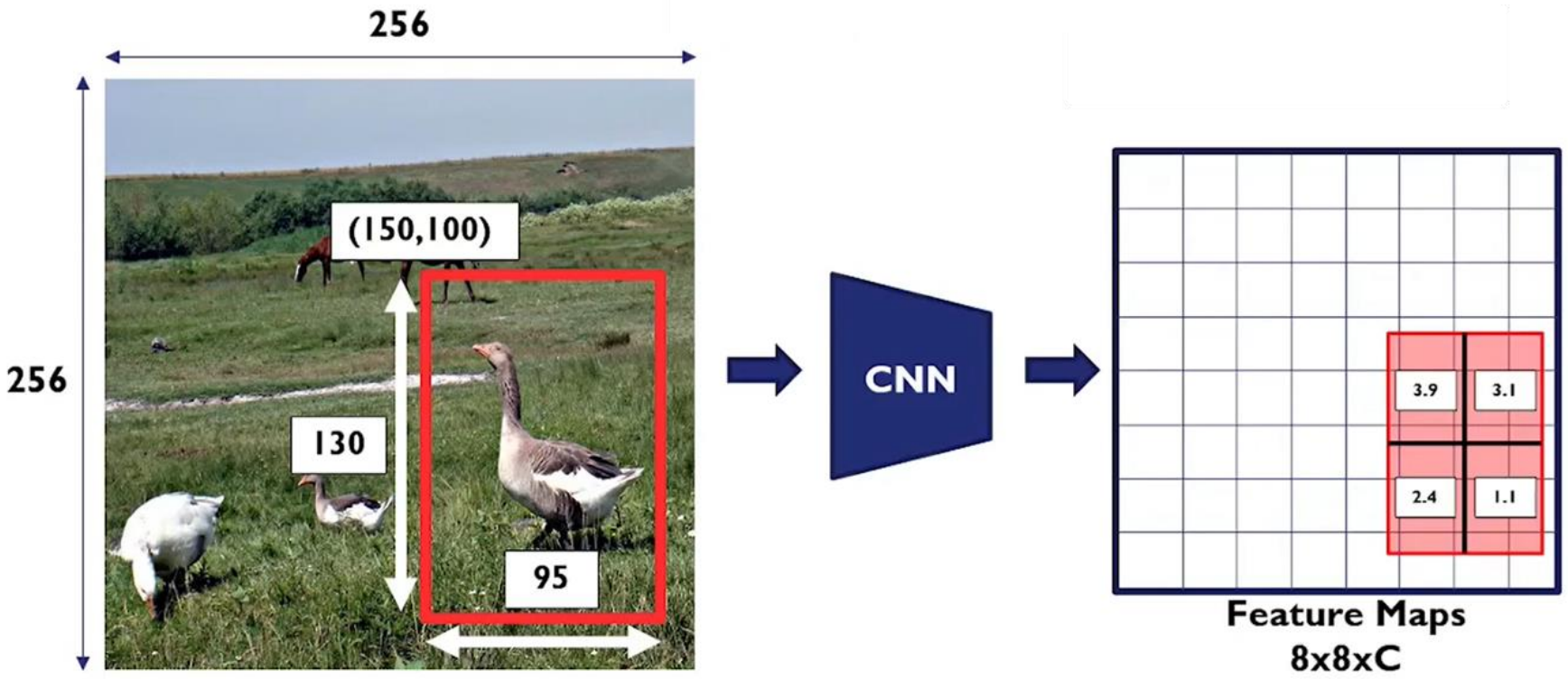


ROI Alignment: Bilinear Interpolation and Pooling

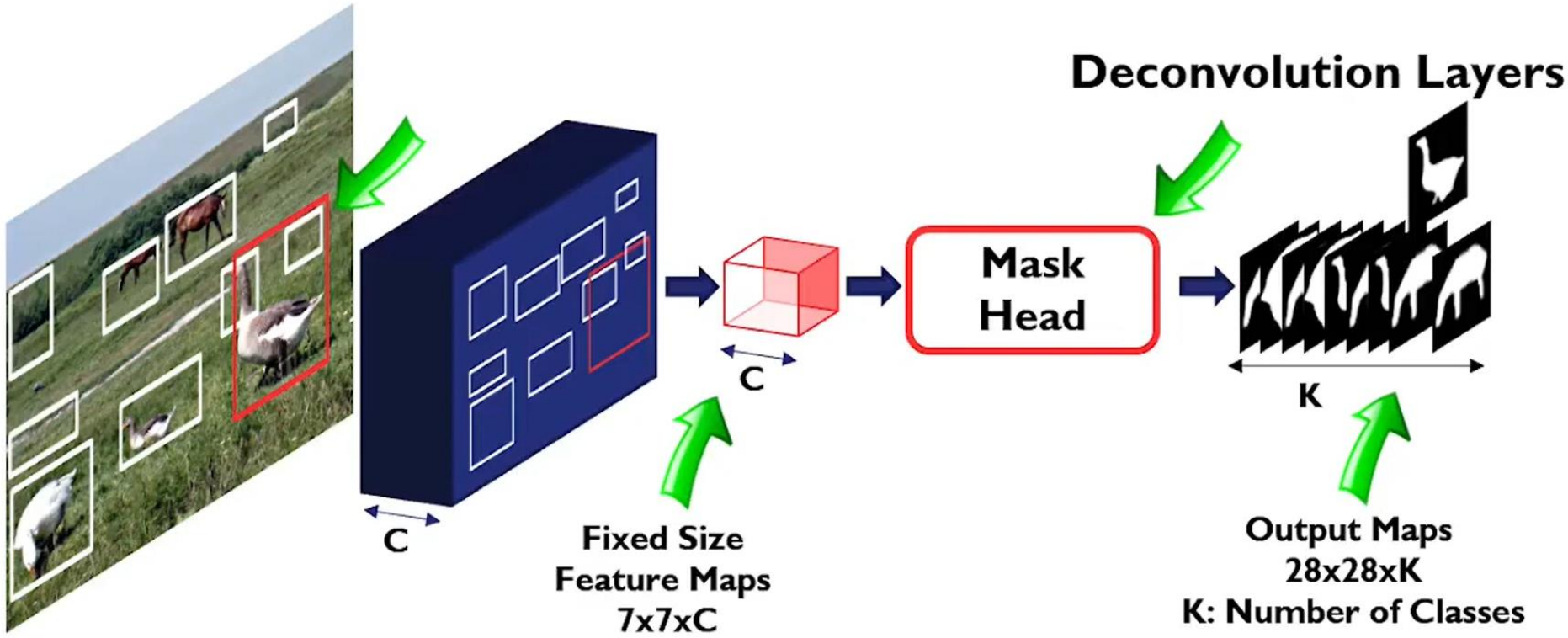


Note: Values chosen just for illustration simplicity

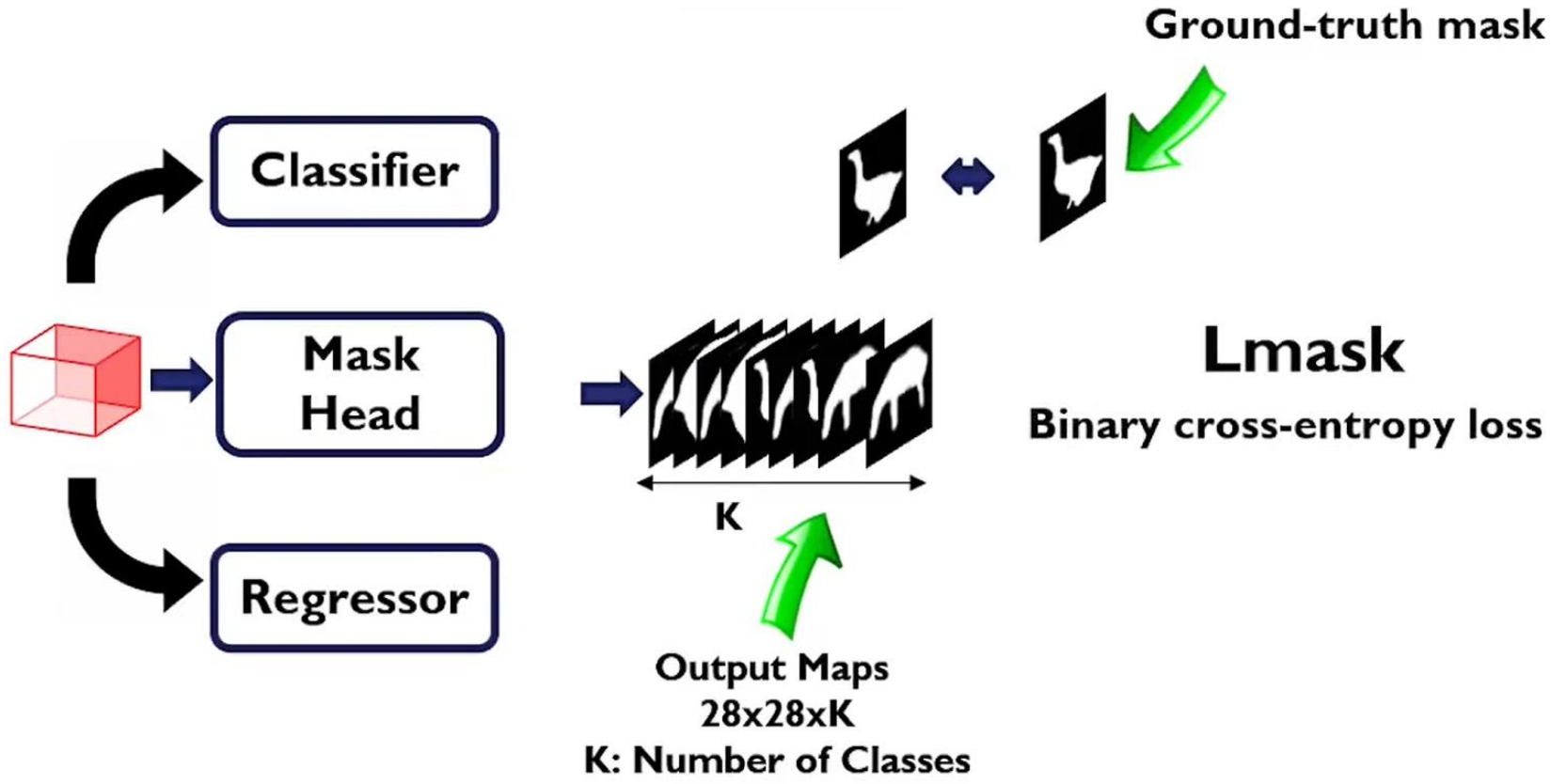
ROI Alignment: Bilinear Interpolation and Pooling



Mask R-CNN: Mask Head



Mask R-CNN: Mask Head – Loss functions



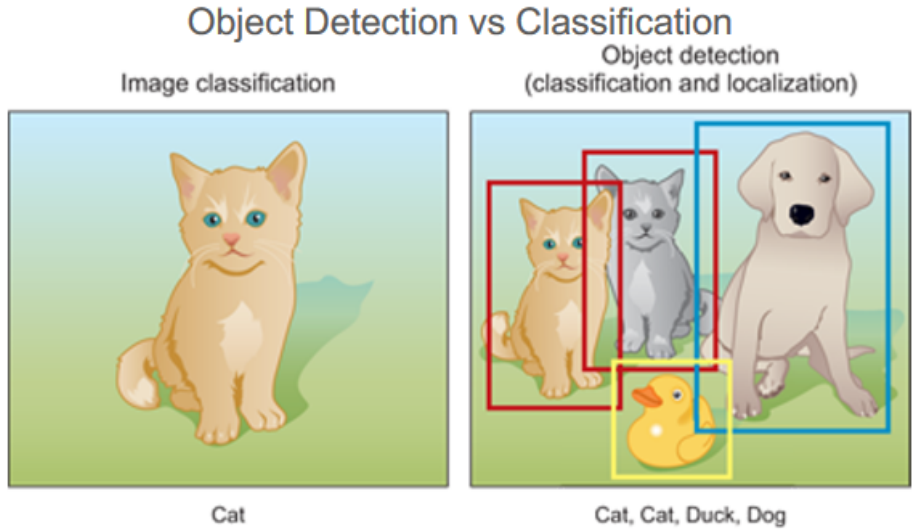
Lmask

Object Detection

YOLO

YOLO

- Object detection is the problem of both locating **AND** classifying objects
- Goal of YOLO algorithm is to do object detection both fast **AND** with high accuracy



Key Insights

Previous Approaches

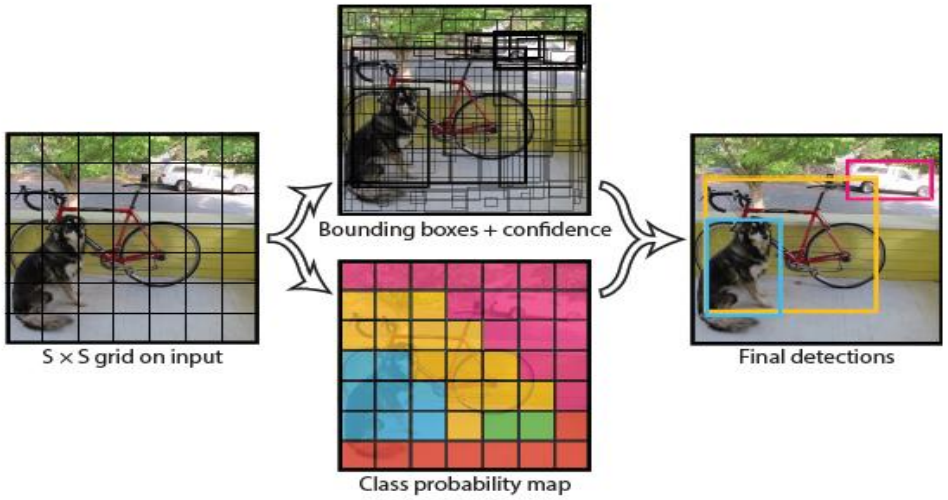
- A separate model for generating bounding boxes and for classification (more complicated model pipeline)
- Need to run classification many times (expensive computation)
- Looks at limited part of the image (lacks contextual information for detection)

YOLO algorithm

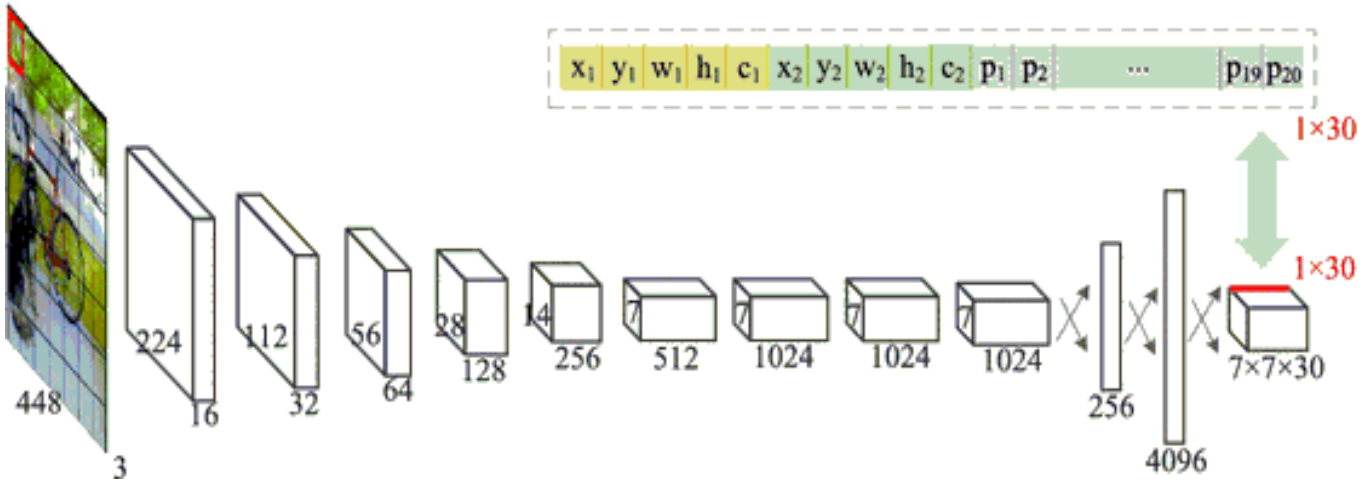
- A single neural network for localization and for classification (less complicated pipeline)
- Need to inference only once (efficient computation)
- Looks at the entire image each time leading to less false positives (has contextual information for detection)

YOLO Overview

- You Only Look Once (YOLO) is a CNN based detector shows promising results on PASCAL VOC 2007 and 2012 with throughput of 45 frames per second.
- YOLO re-frame object detection through a sing pass of the image, unlike traditional sliding window based object detectors.
- YOLO divides the input image into SxS small grids and for each grid cell, there will be B bounding boxes.
- Each grid cell can predict only one object
- CNN predicts the position and the class probabilities for those bounding boxes.



Working of YOLO



YOLO CNN Architecture

YOLO Algorithm Steps: How Does it Work?

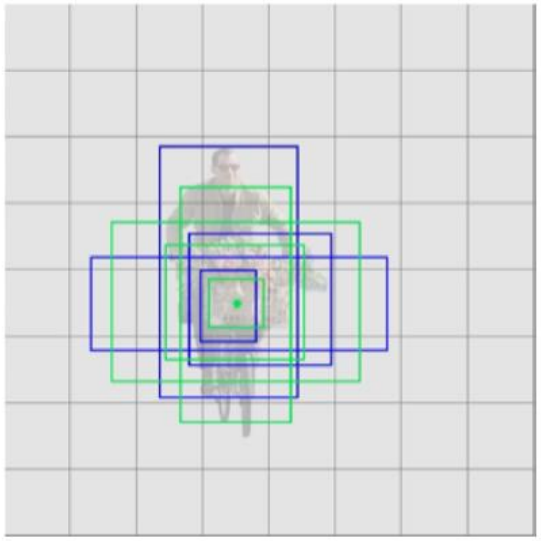
The basic idea behind YOLO is to divide the input image into a grid of cells and, for each cell, predict the probability of the presence of an object and the bounding box coordinates of the object. One of the key advantages of YOLO is that it processes the entire image in one pass, making it faster and more efficient than two-stage object detectors such as R-CNN and its variants.

The process of YOLO can be broken down into several steps:

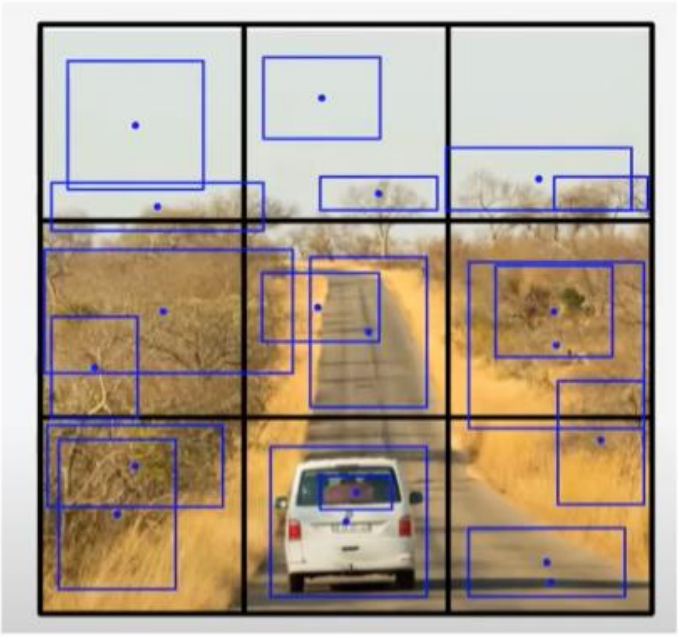
- 1. Input image is passed through a CNN to extract features from the image.
- 2. The features are then passed through a series of fully connected layers, which predict class probabilities and bounding box coordinates.
- 3. The image is divided into a grid of cells, and each cell is responsible for predicting a set of class probabilities and bounding boxes.
- 4. The output of the network is a set of bounding boxes and class probabilities for each cell.
- 5. The bounding boxes are then filtered using non-maximum suppression algorithm to remove overlapping boxes and choose the box with the highest probability.
- 6. The final output is a set of predicted bounding boxes and class labels for each object in the image.

YOLO Overview

- First, image is split into a SxS grid
- For each grid square, generate B bounding boxes
- For each bounding box, there are 5 predictions: x, y, w, h, confidence



Green = anchors
Blue = predictions



S = 3, B = 2

YOLO Training

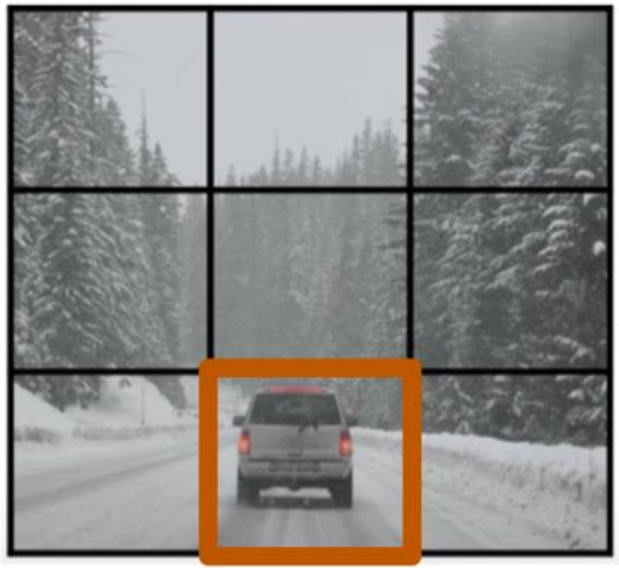
YOLO Training

- 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

For each grid block, we have a vector like this. For this example B is 2 and C is 2

(2 bounding boxes, 2 object classes)

p_1
b_{x_1}
b_{y_1}
b_{h_1}
b_{w_1}
p_2
b_{x_2}
b_{y_2}
b_{h_2}
b_{w_2}
c_1
c_2

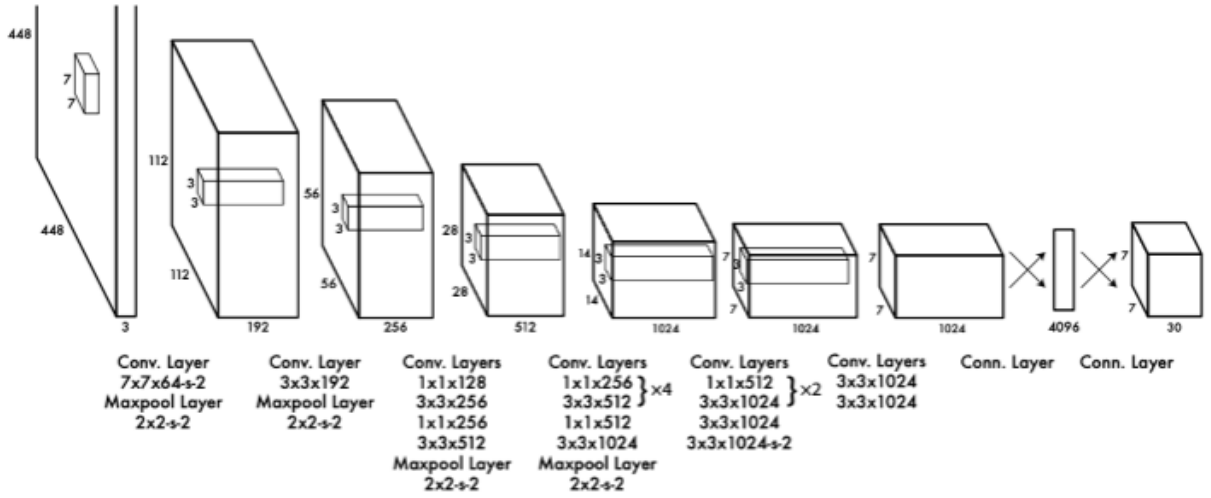


GT label example:

1
b_{x_1}
b_{y_1}
b_{h_1}
b_{w_1}
0
?
?
?
?
$c_1 = 1$
$c_2 = 0$

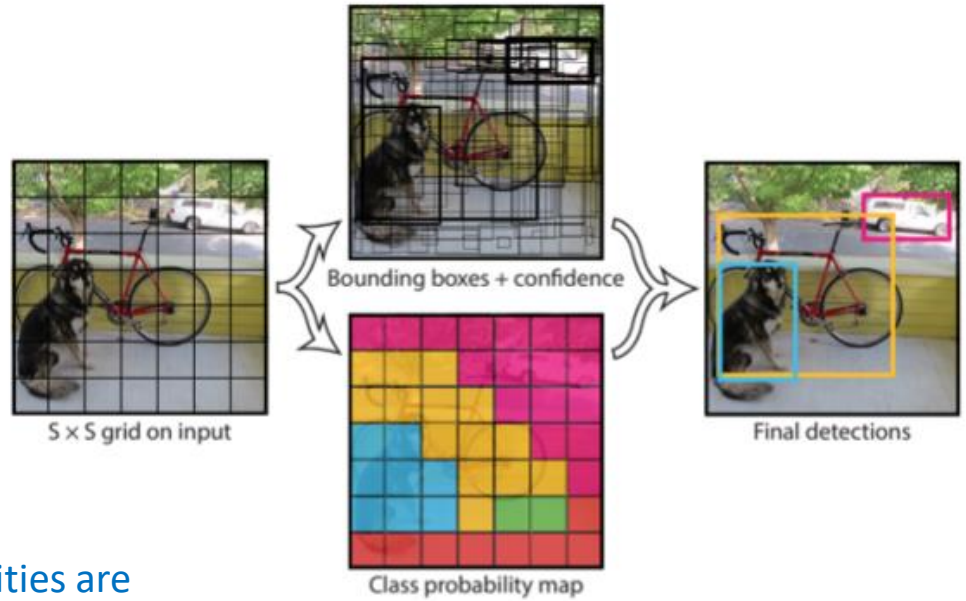
YOLO Architecture

- Now that we know the input and output, we can discuss the model
- We are given 448 by 448 by 3 as our input.
- Implementation uses 7 convolution layers
- Paper parameters: S = 7, B = 2, C = 20
- Output is S*S*(5B+C) = 7*7*(5*2+20) = 7*7*30 parameters



YOLO Prediction

- We then use the output to make final detections
- Use a threshold to filter out bounding boxes with low P(Object)
- In order to know the class for the bounding box compute score take argmax over the distribution Pr(Class|Object) for the grid the bounding box's center is in

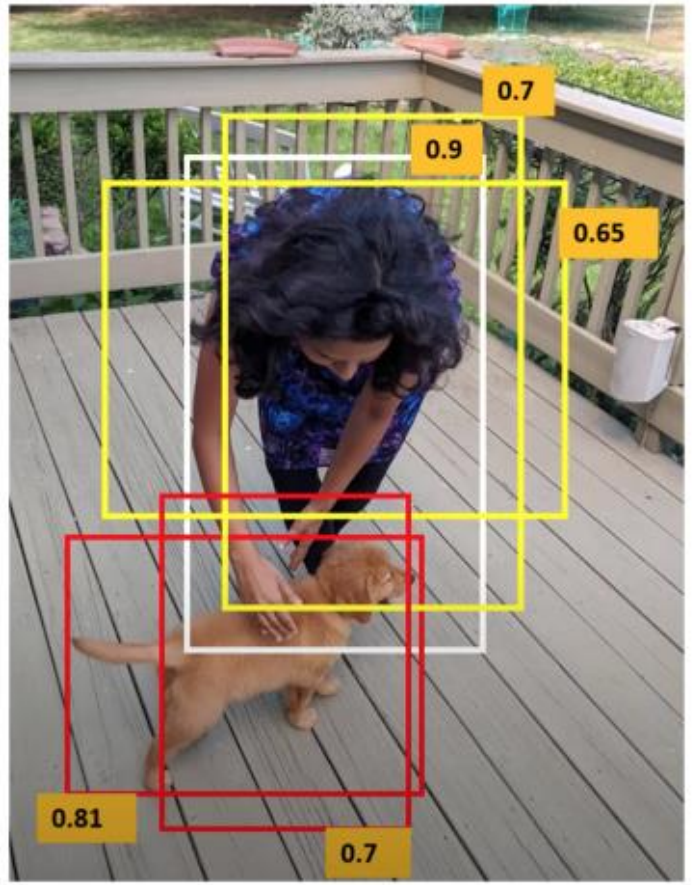


Only one class object is predicted for each cell. Each class object probabilities are Shown with different colours.

$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

Non-Maximum Suppression

- Most of the time objects fall in one grid, however it is still possible to get redundant boxes (rare case as object must be close to multiple grid cells for this to happen)
- Discard bounding box with high overlap (keeping the bounding box with highest confidence)
- Adds 2-3% on final mAP score



YOLO Loss / 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

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

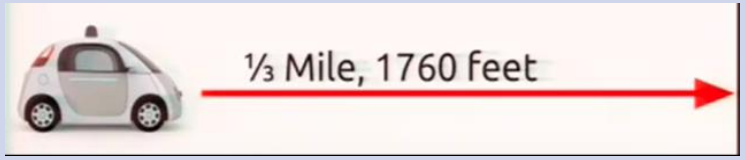
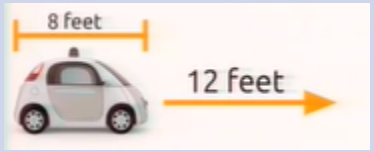
Class loss, minimize loss between true class of object in grid box

YOLO Results

- Baseline YOLO outperform real time detectors by large amount
- Do better than most less than real time as well

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

YOLO comparison(with other object detection algorithms)

Algorithm	Speed	Pascal 2007 mAP	Distance covered by car (60mph) within object detection time
R-CNN	0.05 FPS (20 sec/image)	66.0	
Fast R-CNN	0.5 FPS (2 sec/image)	70.0	
Faster R-CNN	7 FPS (140 msec/image)	73.2	
YOLO	45 FPS (22 msec /image)	63.4	

Reference: CVPR 2016 - You Only Look Once: Unified, Real-Time Object Detection

YOLO: Pros and Cons

- Pro: YOLO is a lot faster than the other algorithms for image detection
- Pro: YOLO's use of global information rather than only local information allows it to understand contextual information when doing object detection
  - Does better in domains such as artwork due to this
- Con: YOLO lagged behind the SOTA models in object detection
  - This is attributed to making many localization errors and unable to detect small object

SOTA models: State of the Art models (what it means is the current best models available)

References

- CV3DST - computer Vision 3: Detection, Segmentation, and Tracking - Technical University of Munich - Prof. Leal-Taixé (WS21/22)
- Object Detection by DataMListic (Part1 – Part 6)
- CAP5415 Computer Vision- Fall 2023: Lecture 13 - Object Detection (13.1 to 13.14)
- https://www.youtube.com/@sagarGS/videos (All Faster R-CNN and Masked R-CNN videos)
- TED How computers learn to recognize objects instantly | Joseph Redmon YOLO creator
- https://www.cs.utexas.edu/~yukez/cs391r_fall2021/slides/pre_09-02_Shivang.pdf
- Instance Segmentation using Mask RCNN | Paper Explanation and Intuition
- https://www.digitalocean.com/community/tutorials/faster-r-cnn-explained-object-detection
- https://www.shiksha.com/online-courses/articles/object-detection-using-rcnn/
- https://kili-technology.com/data-labeling/machine-learning/yolo-algorithm-real-time-object-detection-from-a-to-z

Model Metrics

- Mean Average Precision (mAP) gives you a great view of the way your model is performing against other models on the same test dataset.