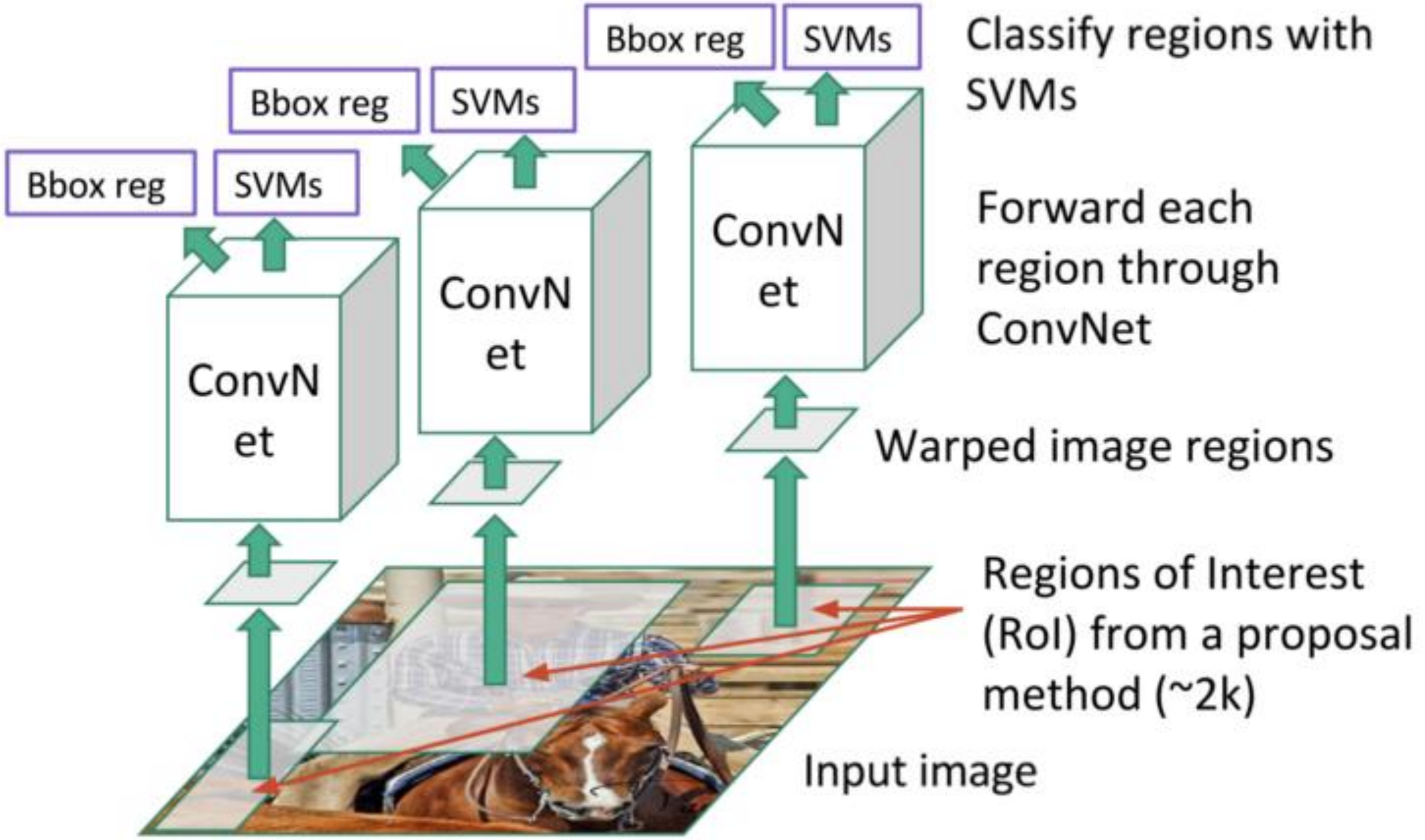
CMPT 743. Practices on Visual Computing II

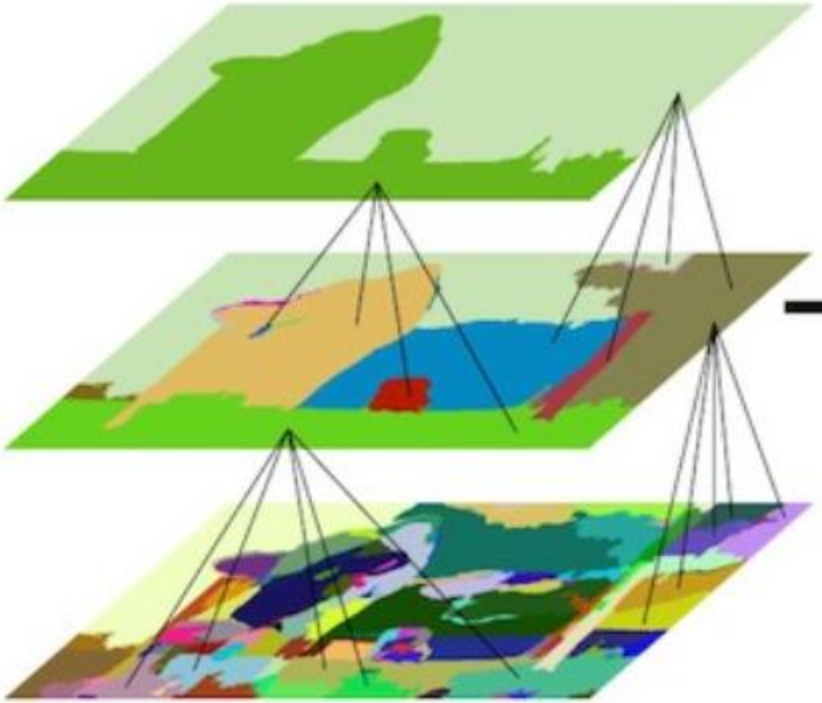
Ali Mahdavi Amiri

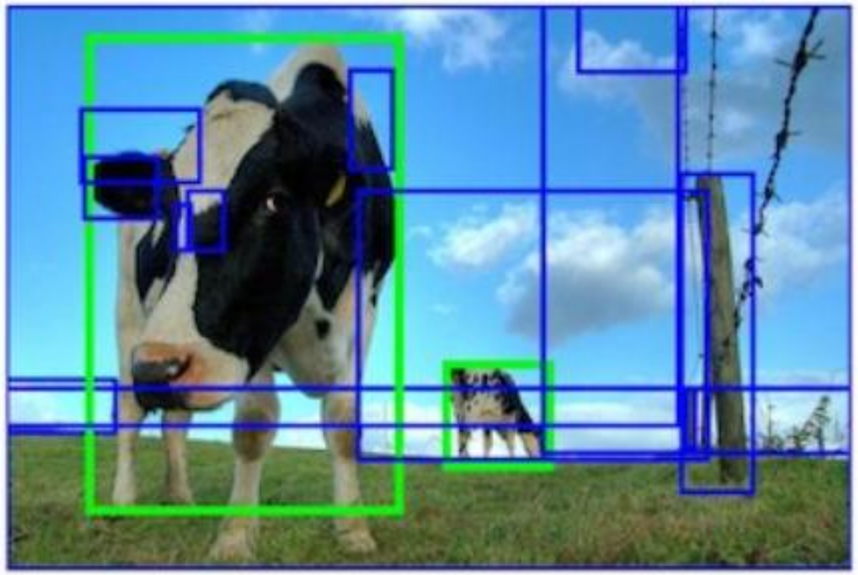
RCNN


Bbox reg
SVMs
Bbox reg
SVMs
Bbox reg
SVMs
Classify regions with SVMs
ConvNet
ConvNet
ConvNet
Forward each region through ConvNet
Warped image regions
Regions of Interest (RoI) from a proposal method (~2k)
Input image

Region Proposal

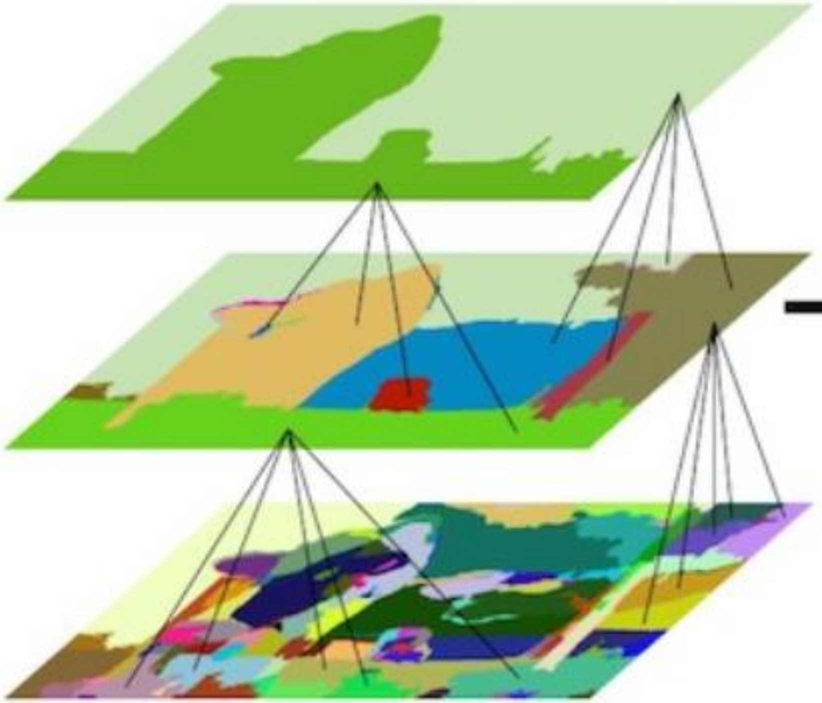
- Selective search
  - It works based on grouping segments

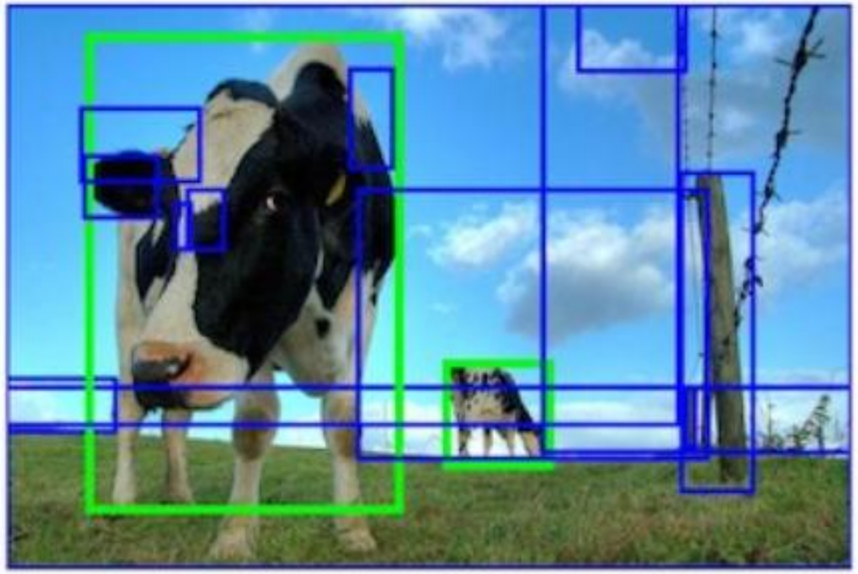




Region Proposal

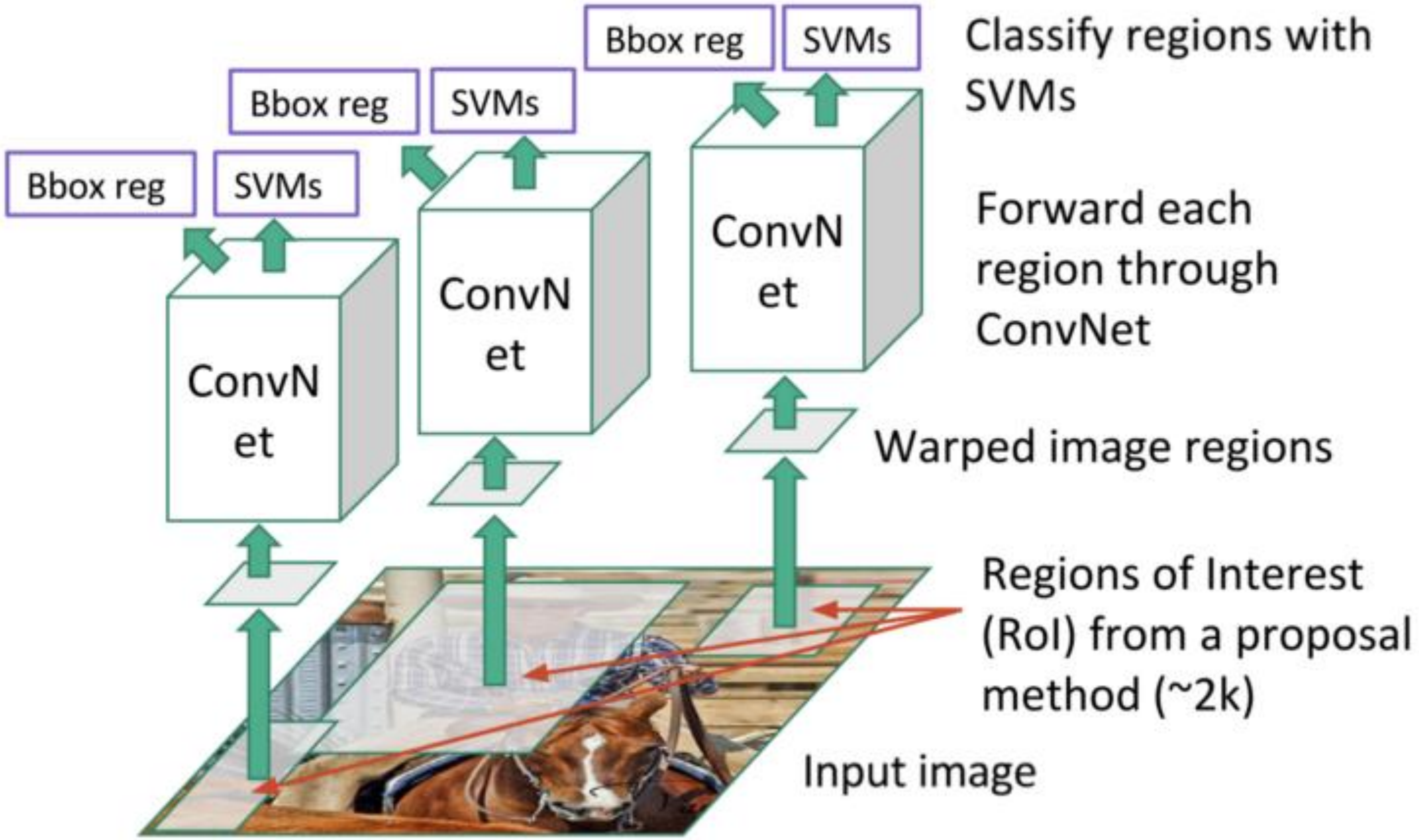
1. Add all bounding boxes corresponding to segmented parts to the list of regional proposals
2. Group adjacent segments based on similarity
3. Go to step 1





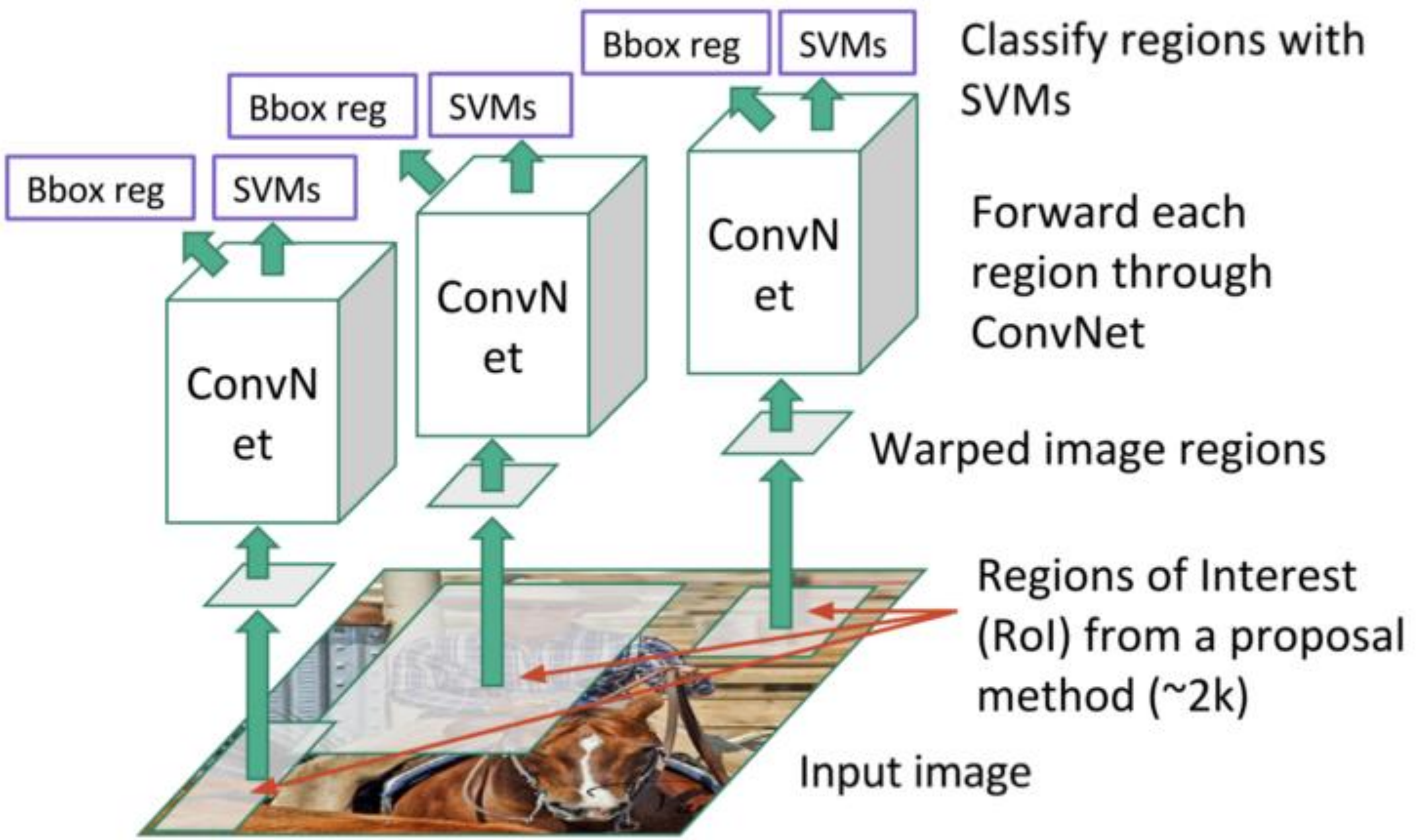
RCNN

- What are the limitations?



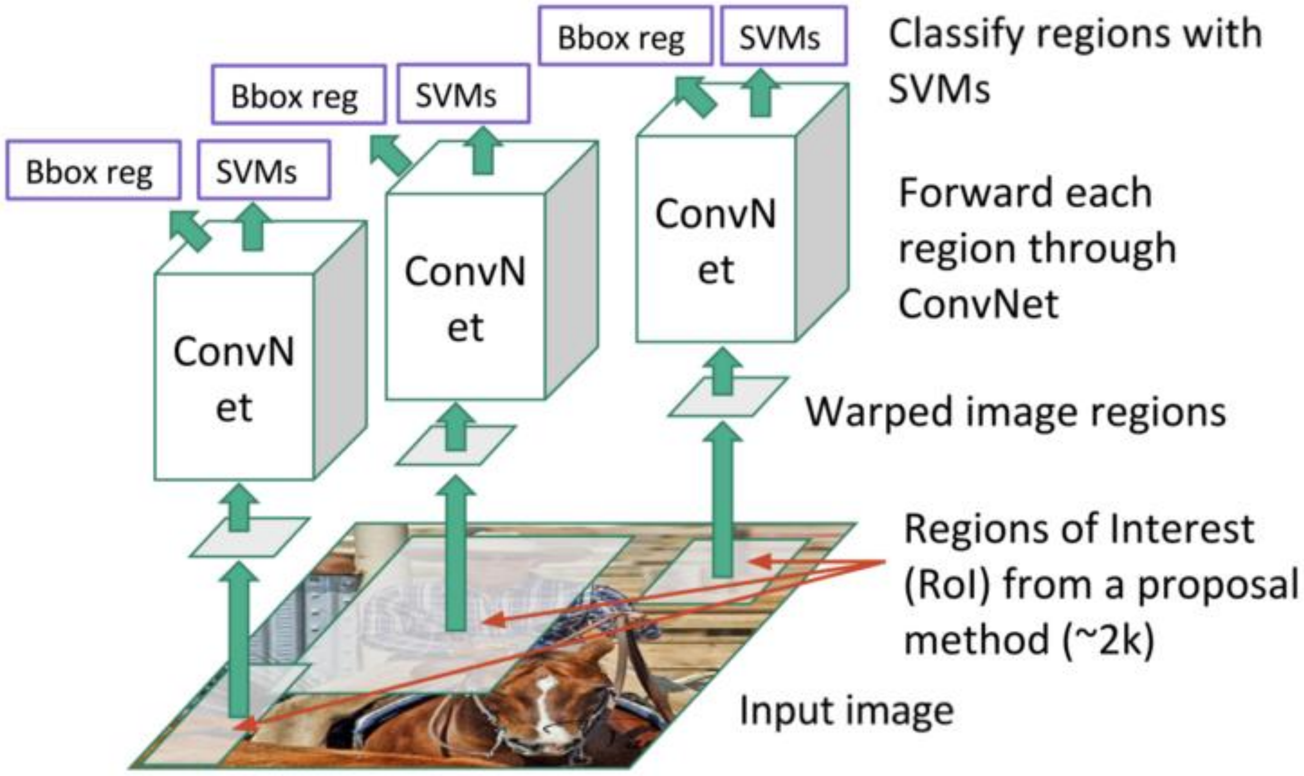
RCNN

- For each ROI, ConvNet must be applied once.



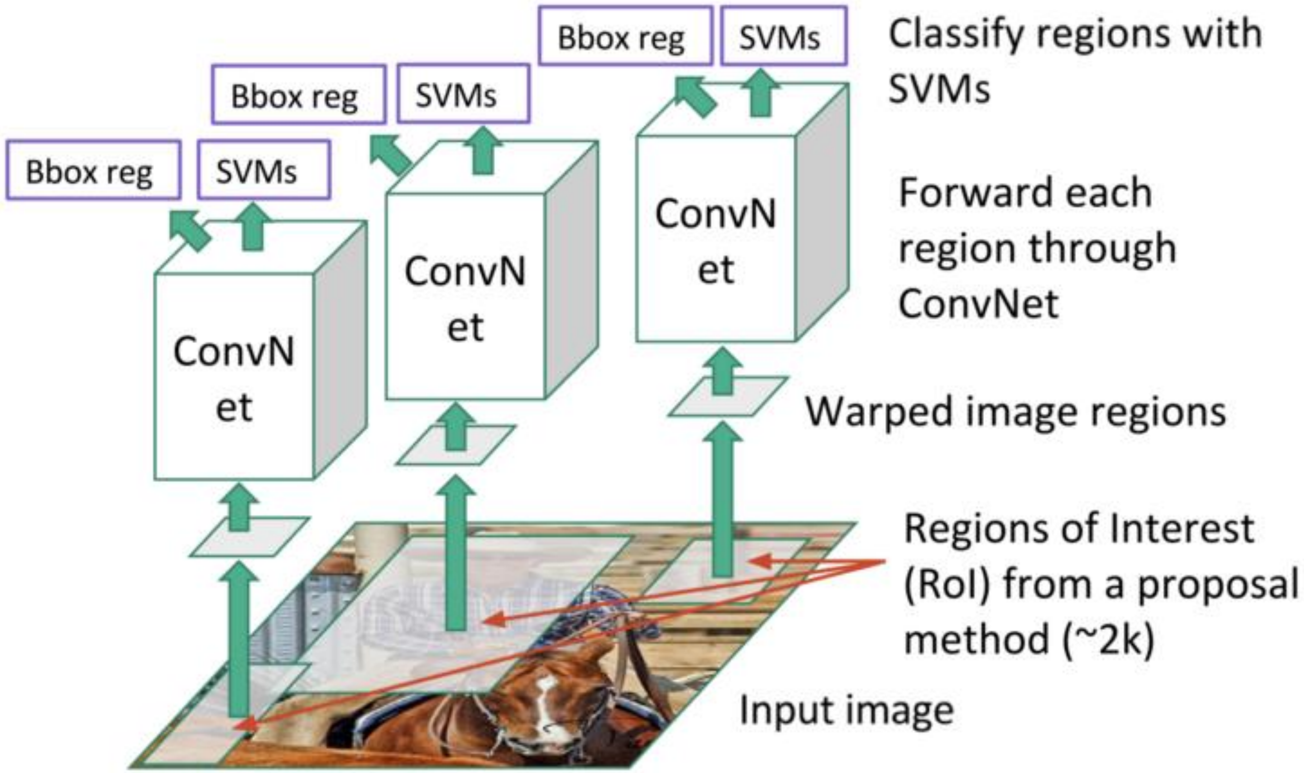
RCNN

- RCNN is slow
  - RCNN is time-consuming, because it repeatedly applies the deep convolutional networks to the raw pixels of thousands of warped regions per image.



RCNN

- All the ROI must be warped and warping results distortions



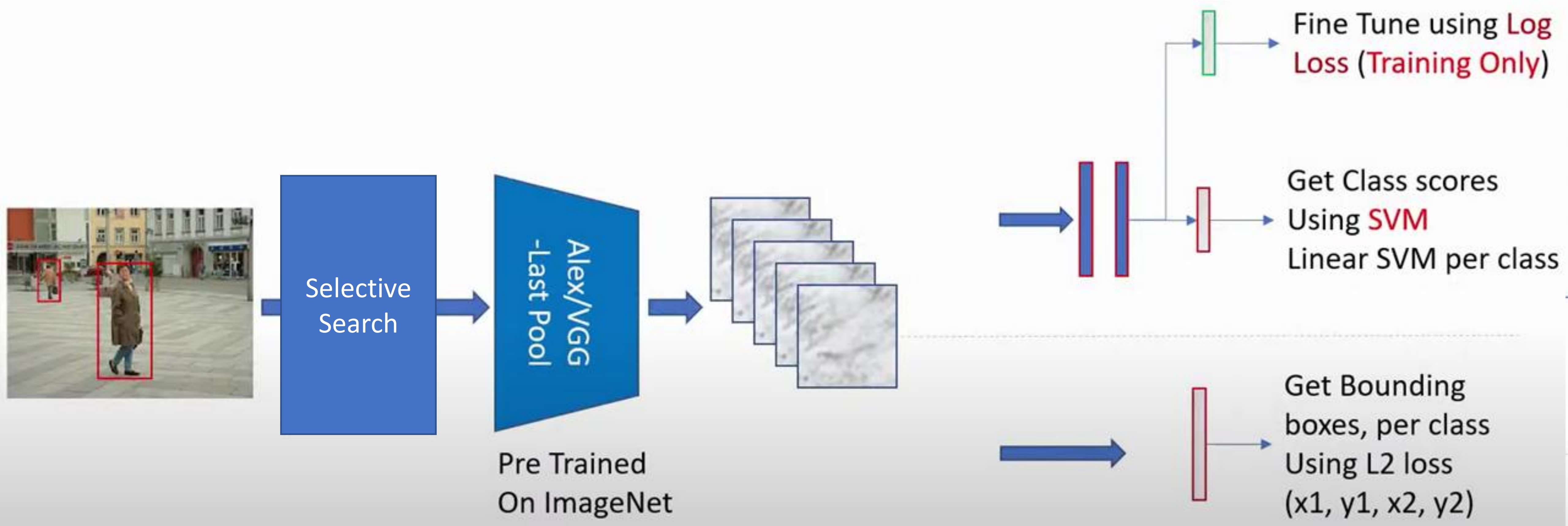
SPPNet

Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition

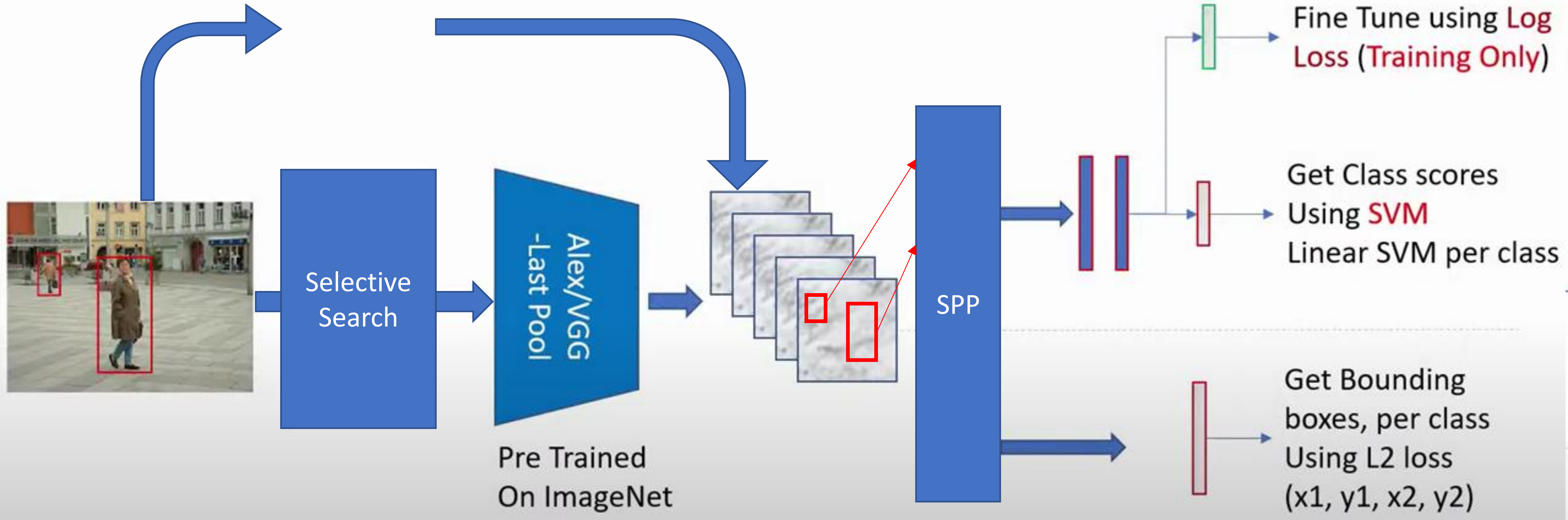
Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun

ECCV 2014

RCNN


Fine Tune using Log Loss (Training Only)
Get Class scores Using SVM
Linear SVM per class
Selective Search
Alex/VGG -Last Pool
Get Bounding boxes, per class Using L2 loss (x1, y1, x2, y2)
Pre Trained On ImageNet

RCNN


Selective
Search
Alex/VGG
-Last Pool
Pre Trained
On ImageNet
SPP
Fine Tune using Log
Loss (Training Only)
Get Class scores
Using SVM
Linear SVM per class
Get Bounding
boxes, per class
Using L2 loss
(x1, y1, x2, y2)

SPPNet

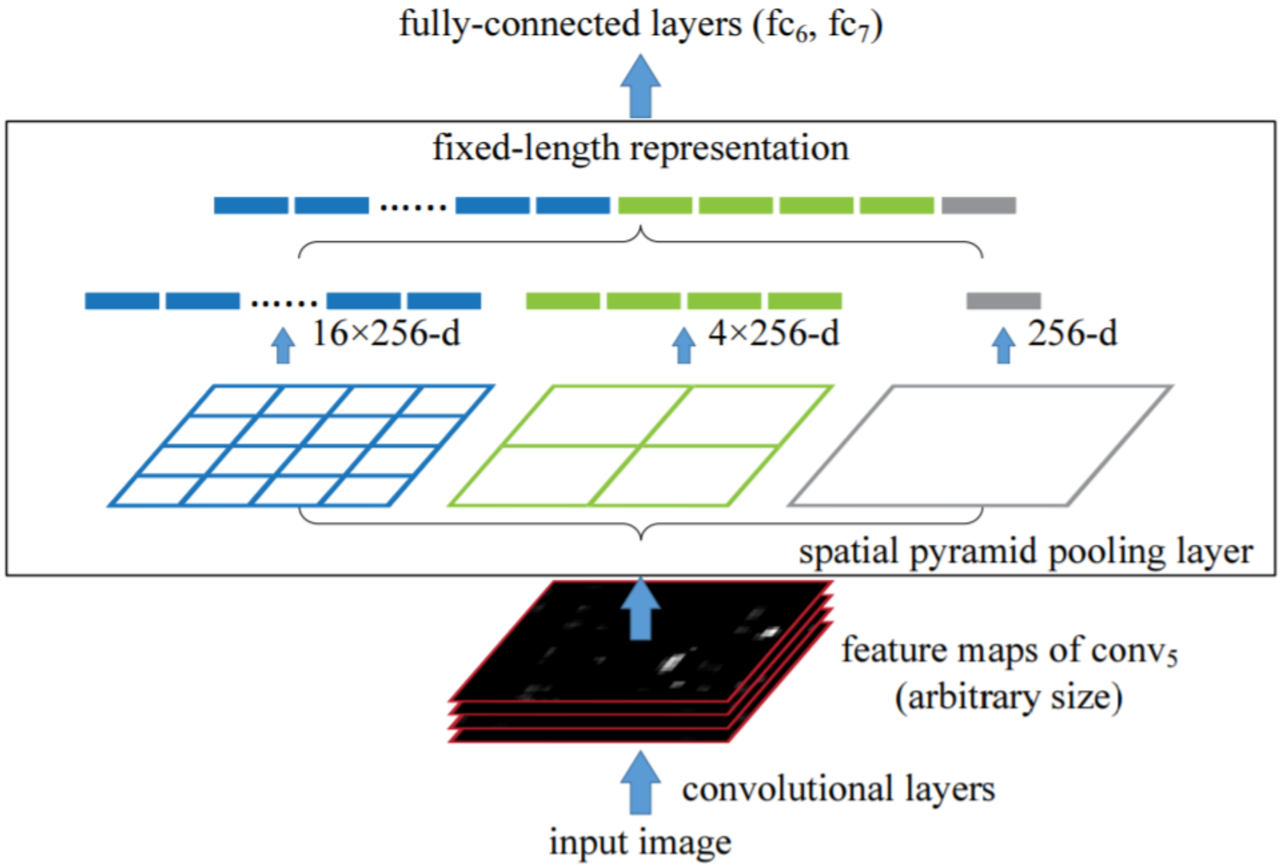
- Spatial Pyramid Pooling
  - To avoid warping => it should work on any image size.

SPPNet

- Spatial Pyramid Pooling
  - To avoid warping => it should work on any image size.
  - Make it a lot faster => it should avoid running ConvNet for several times.

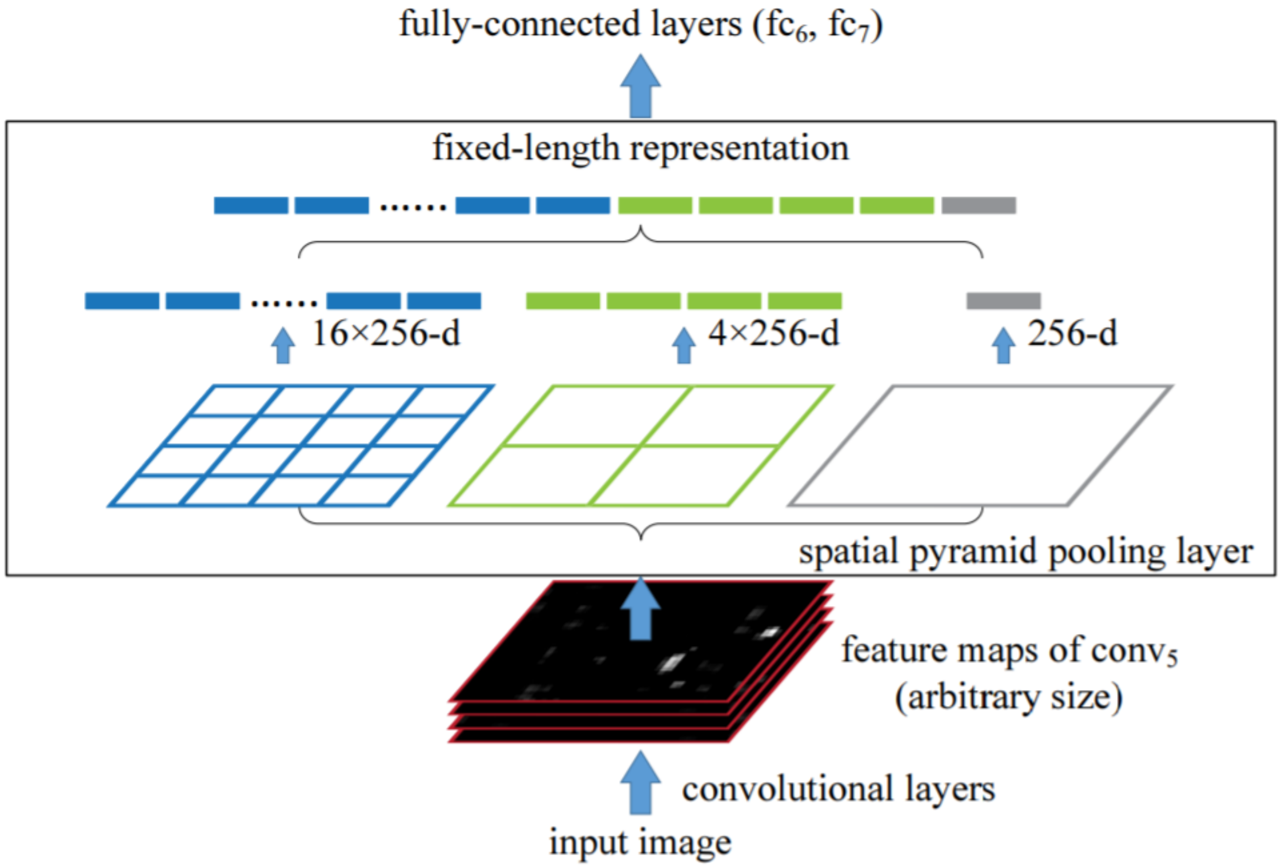
Spatial Pyramid Pooling

- No matter what the input size is, obtain a fixed number of features.



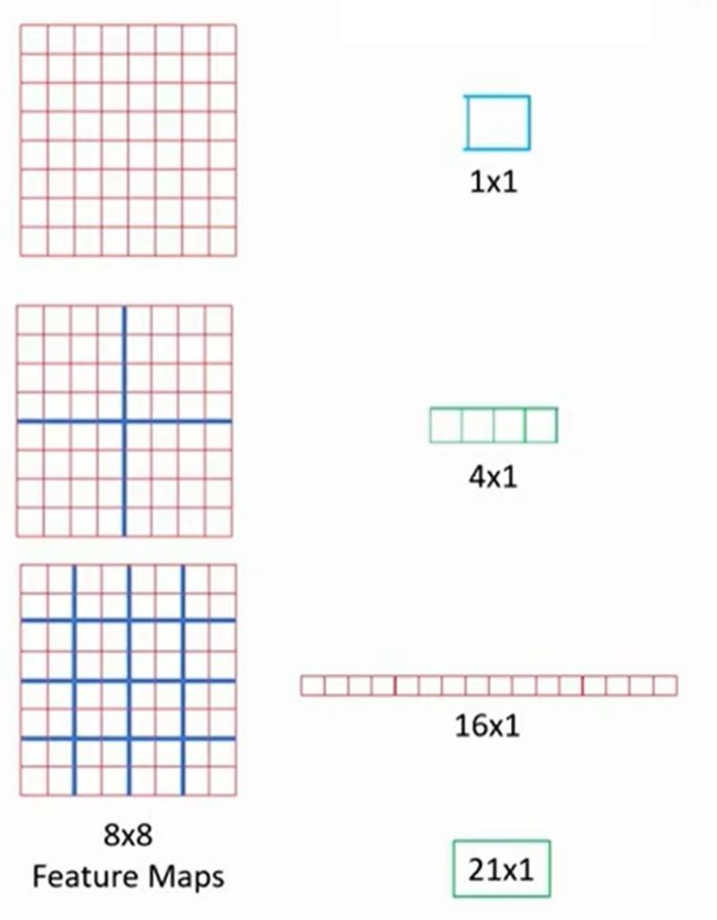
Spatial Pyramid Pooling

- Obtain features with different sizes by max-pooling, concatenate them and give them to fully connected layers.



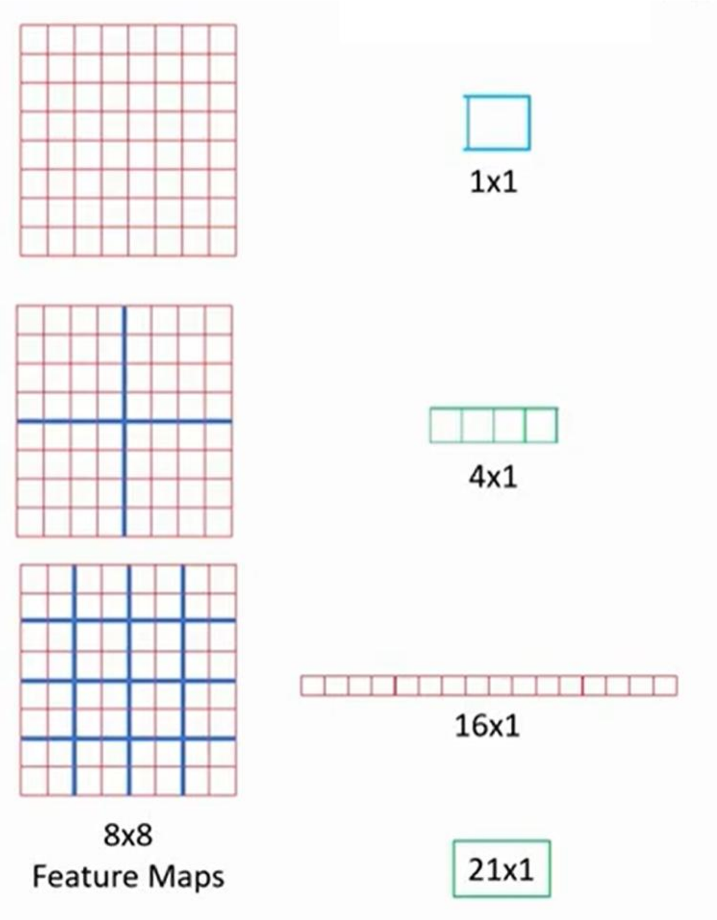
Spatial Pyramid Pooling

- How does it work?
  - You have different sizes of max pooling applied on features with the same size.



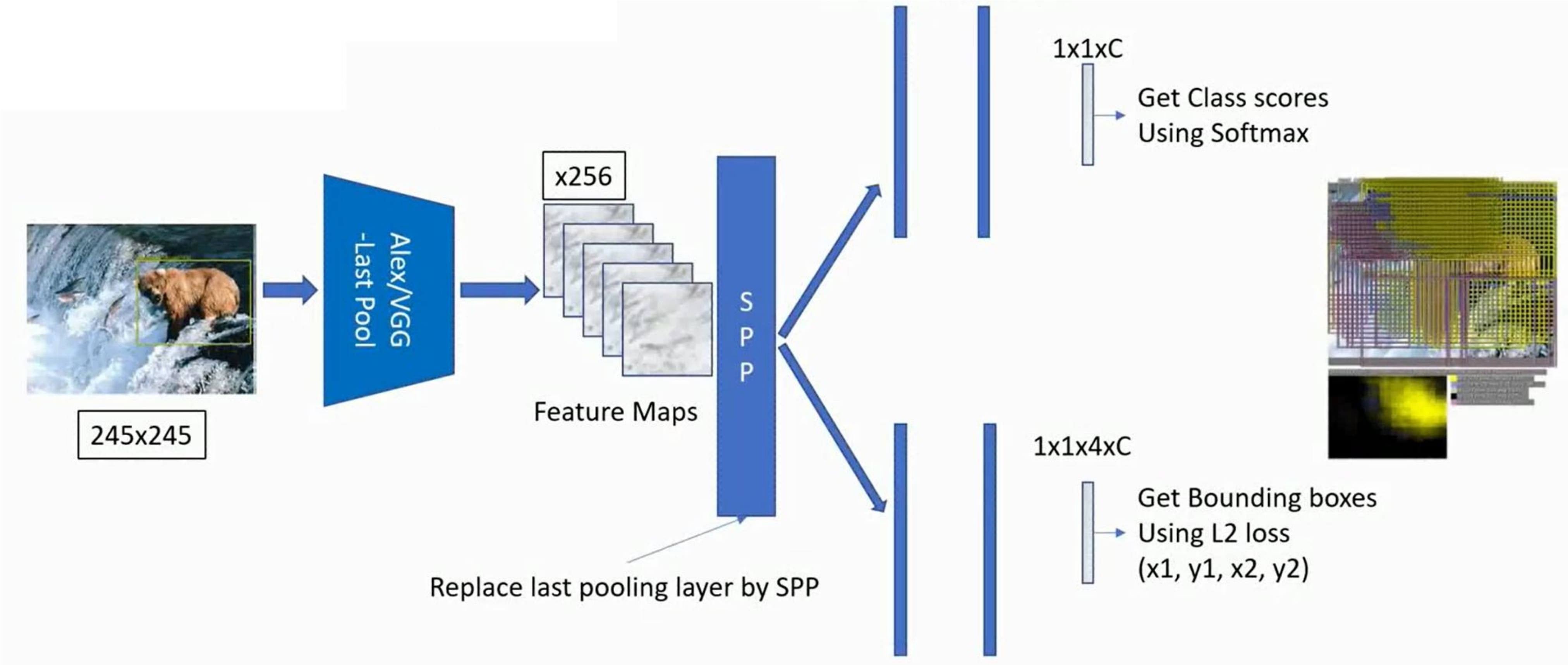
Spatial Pyramid Pooling

- How does it work?
  - You have different sizes of max pooling applied on features with the same size.
  - Note that the feature map has 256 channels so after max pooling, you will have 256*each pooled features



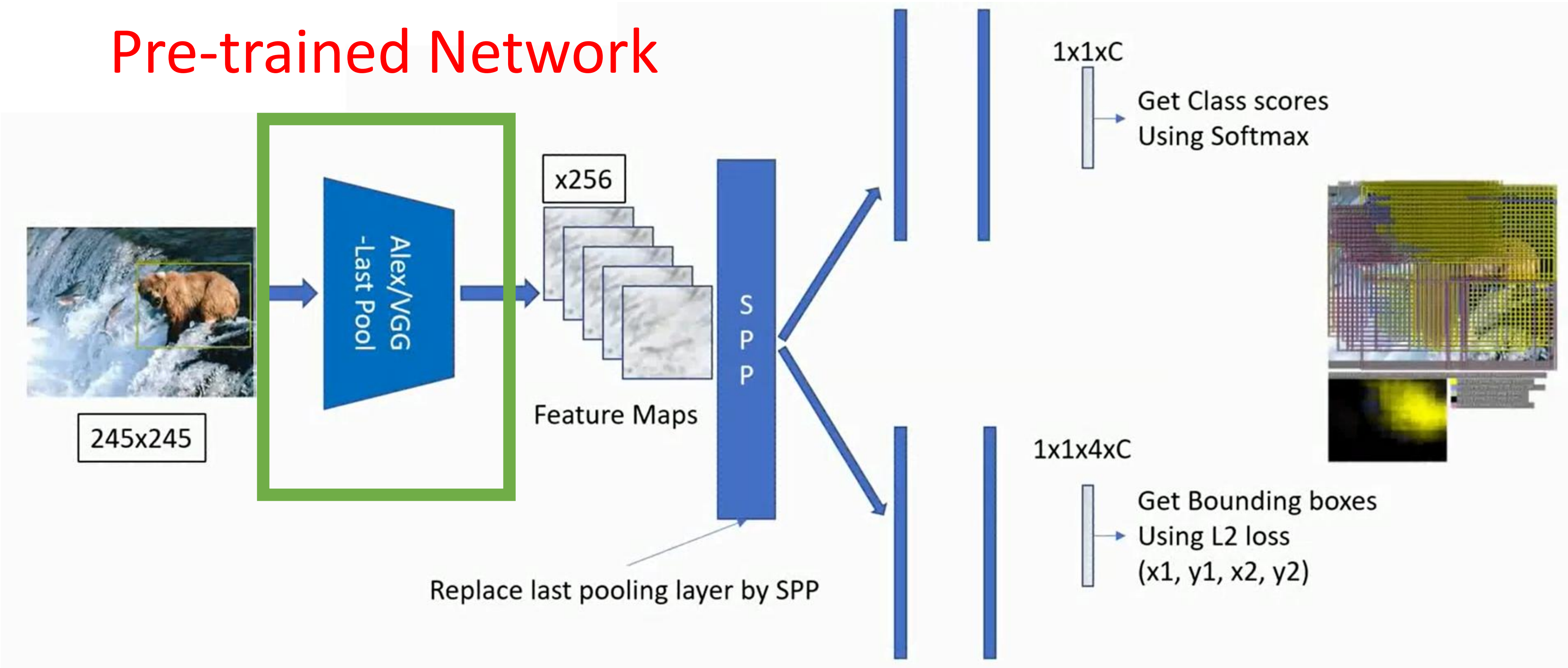
Spatial Pyramid Pooling

- Overall Network

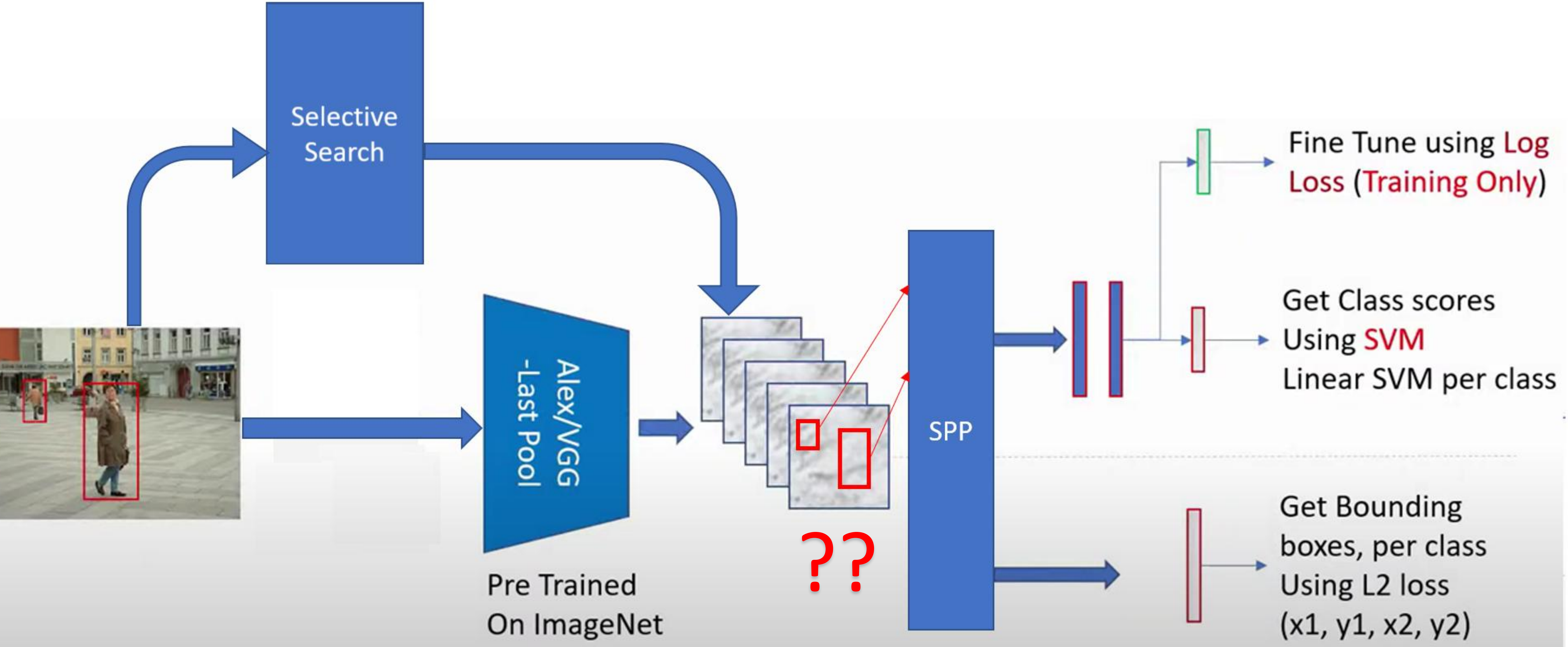


Spatial Pyramid Pooling

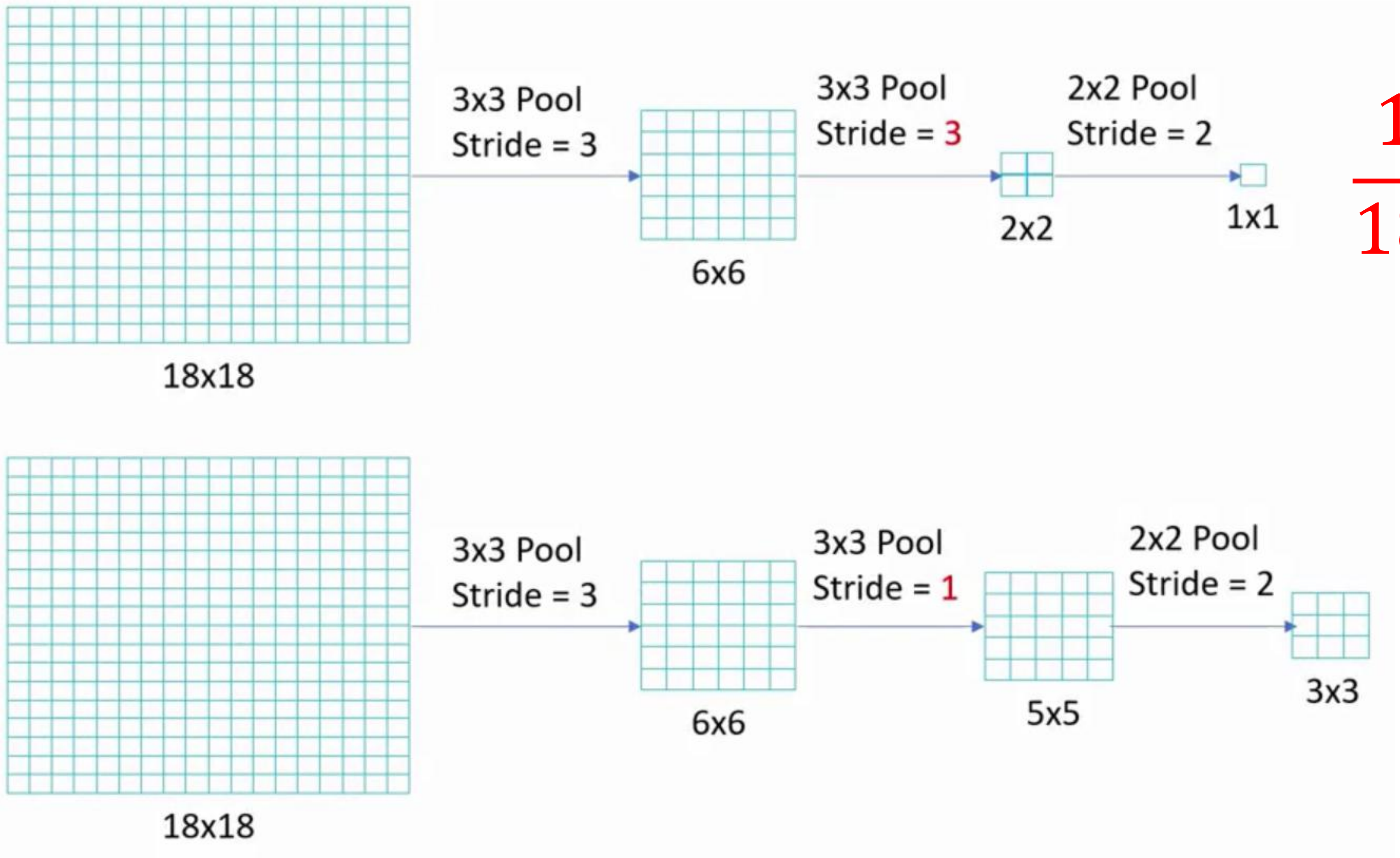
- Overall Network



How to find BB features?



Subsampling Ratio

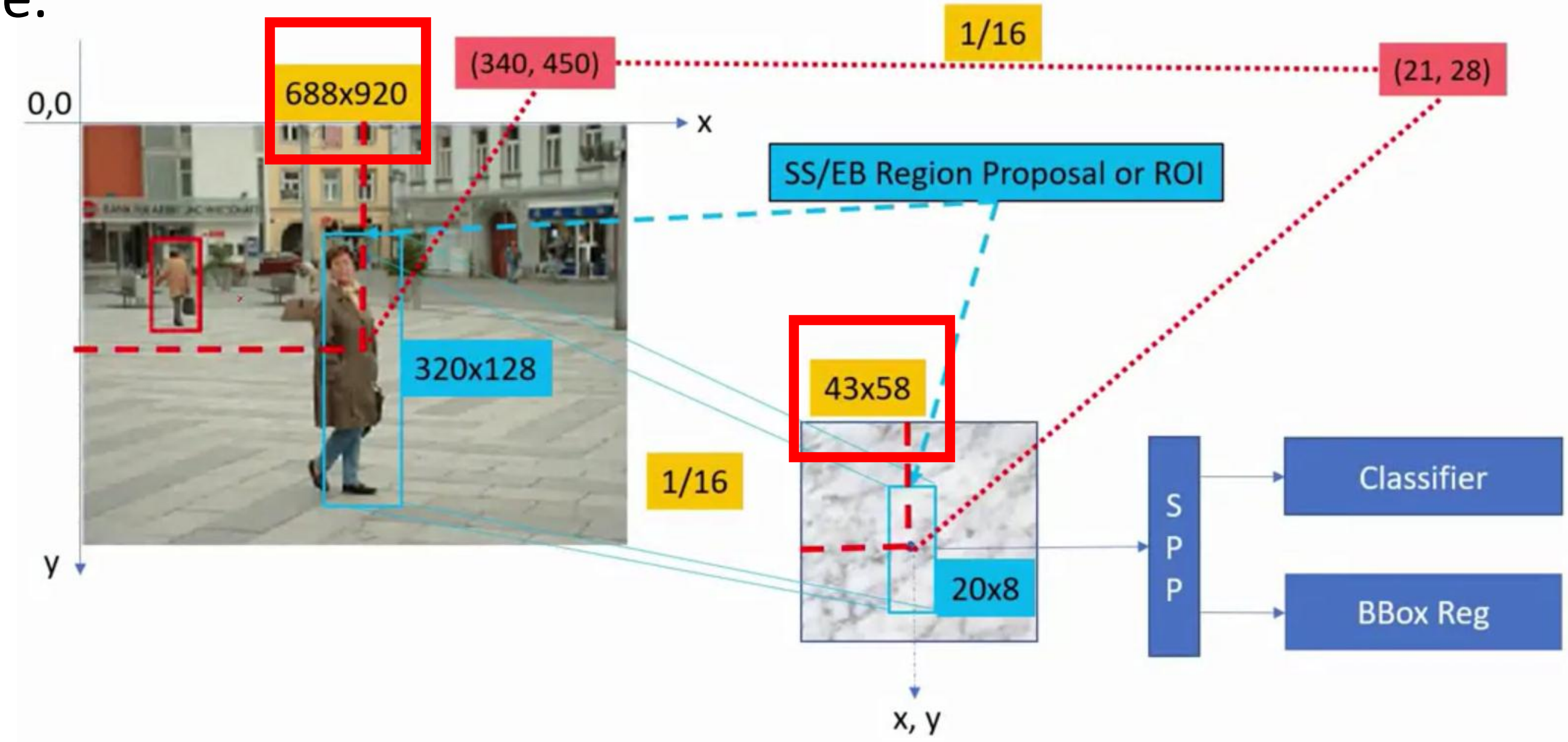


$$\frac{1}{18}$$

$$\frac{1}{6}$$

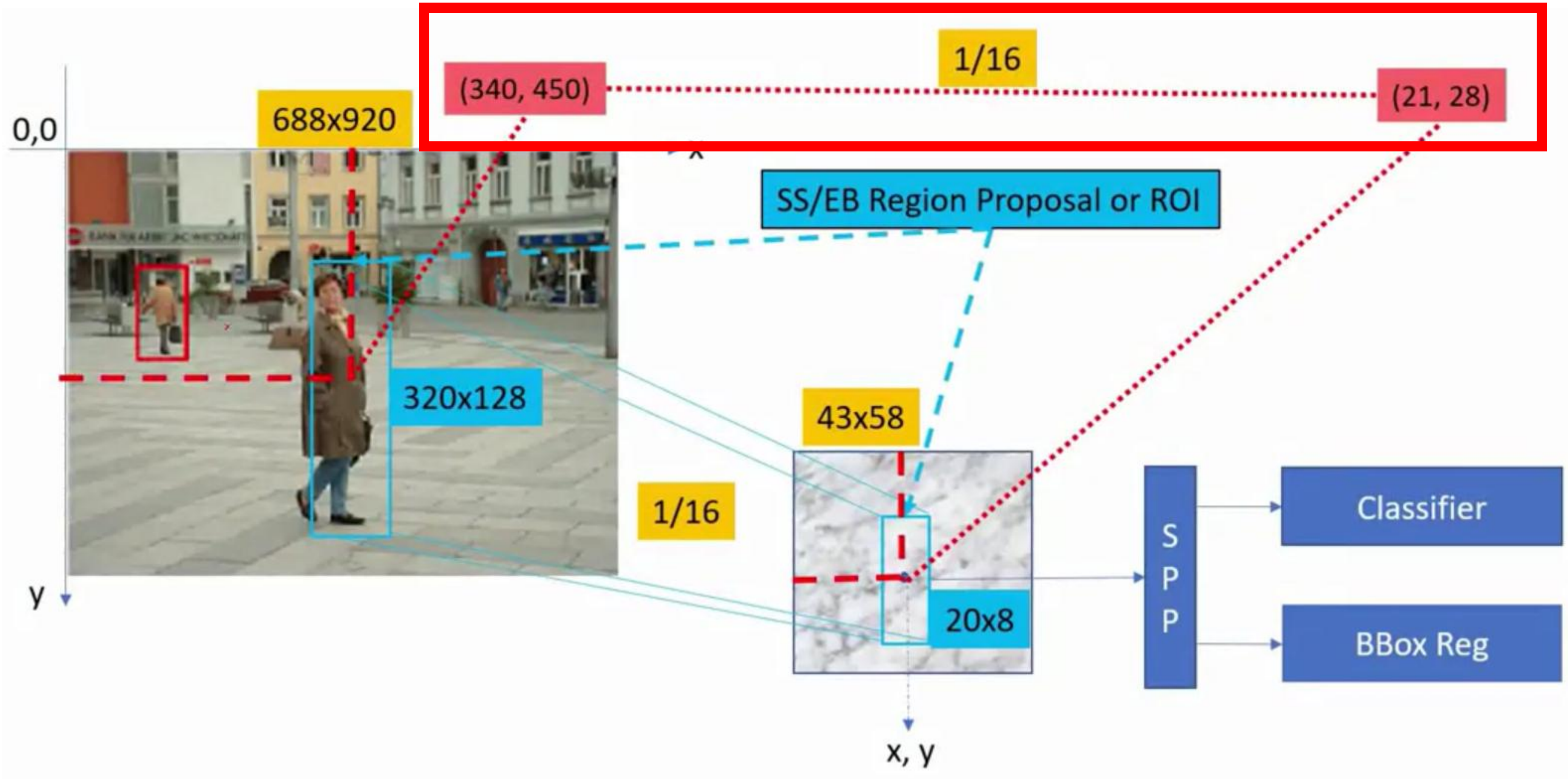
ROI Projection

- $\frac{1}{16}$ is the subsampling ratio since 688×920 image is mapped to a 43×58 feature.



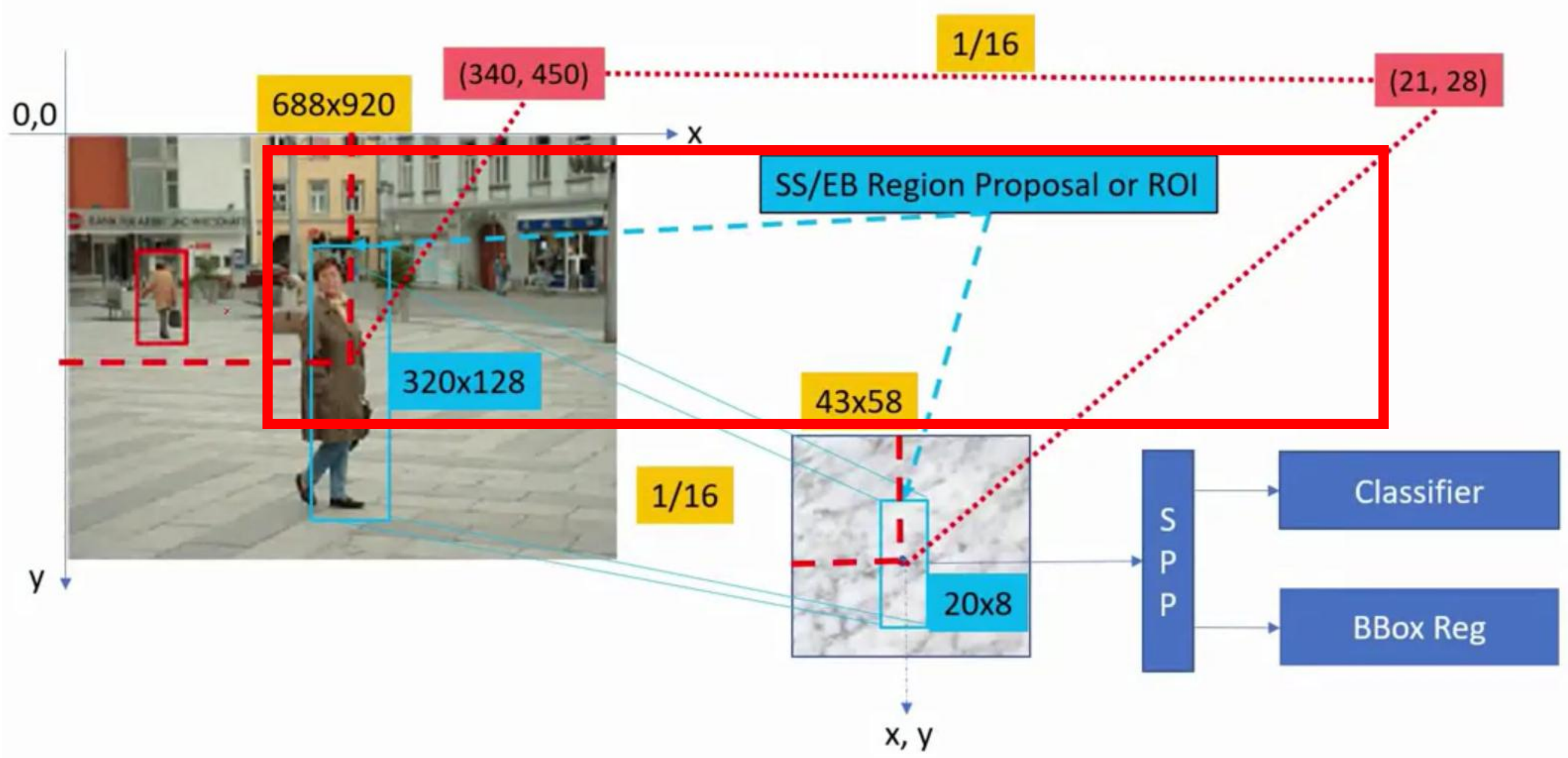
ROI Projection

- Divide the centroid of BB (340,450) by 16 and you get (21,28)



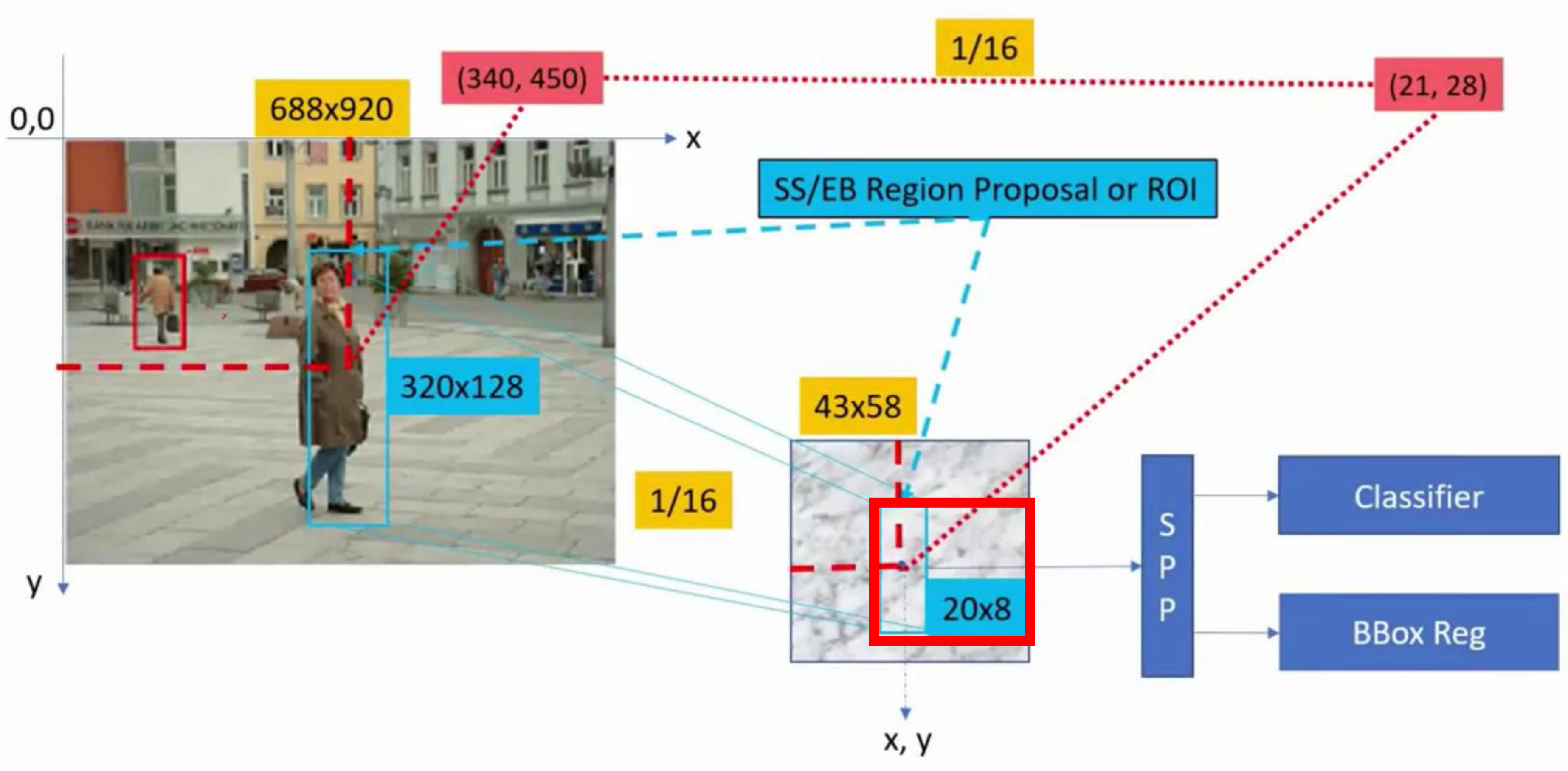
ROI Projection

- Divide the height and width (320,128) by 16 and you get (20,8)

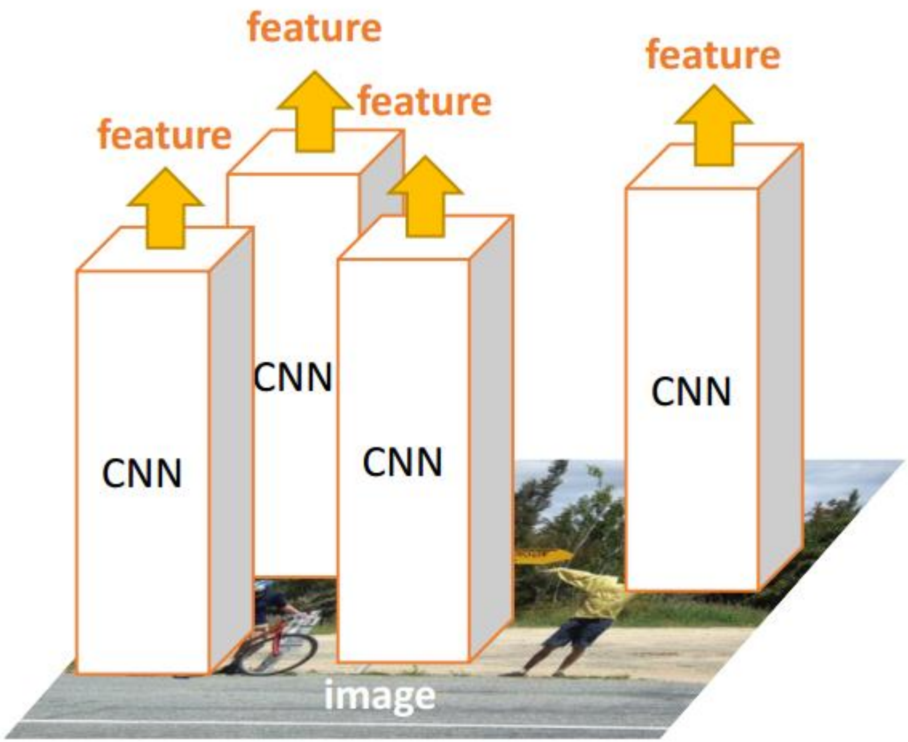


ROI Projection

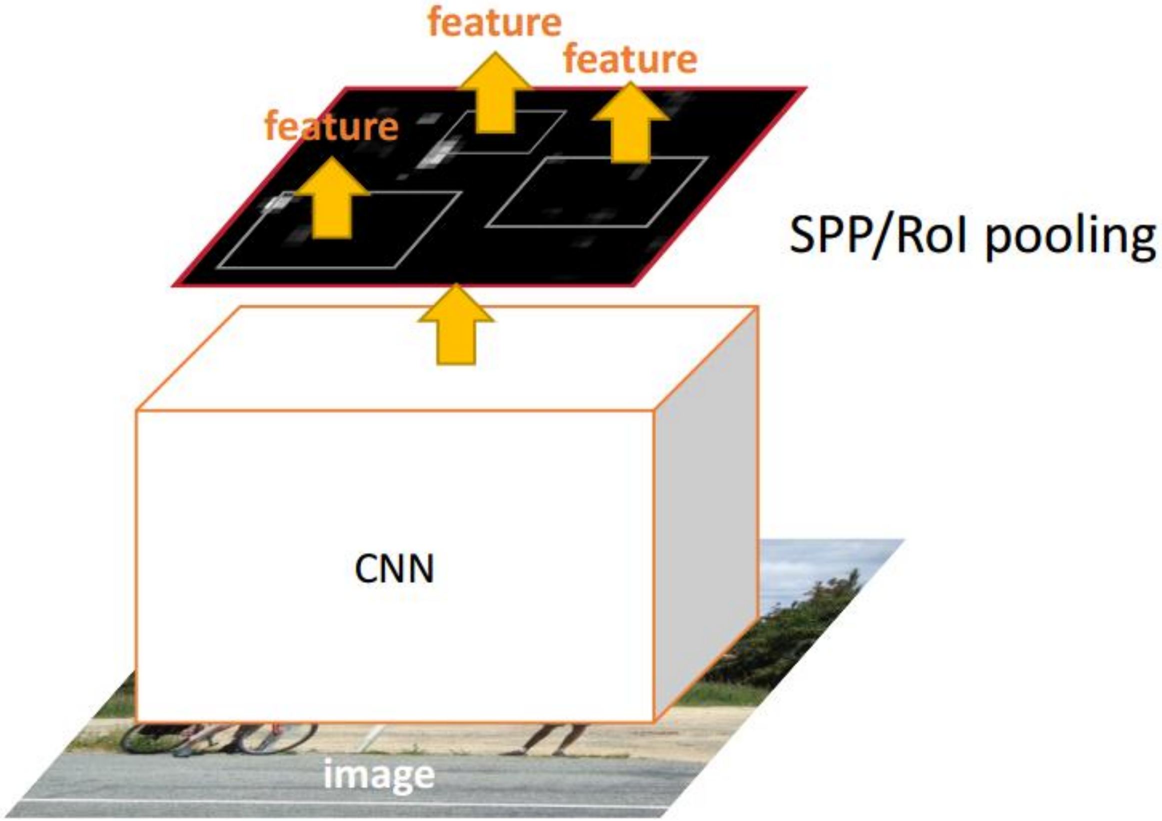
- You can look up the region of interest on the feature and give it to SPP.



Comparison



R-CNN



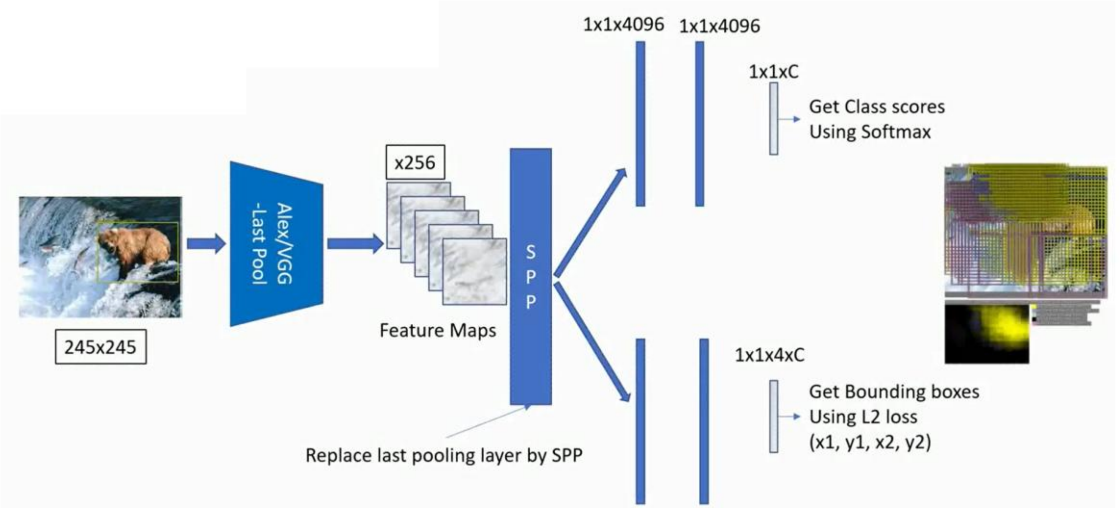
SPP-net

Comparison

	VOC2007	Speed
R-CNN (ZFNet)	59.2%	14.5 s/im
R-CNN (VGGNet)	66.0%	47.0 s/im
SPP (ZFNet)	59.2%	0.38 s/im
SPP (VGGNet)	63.1%	2.3 s/im

Limitations

- What are the limitations of SPPNet?
  - It is multi-stage pipe-line including
    - extracting features
    - fine-tuning a network with log loss
    - training SVMs
    - bounding-box regression

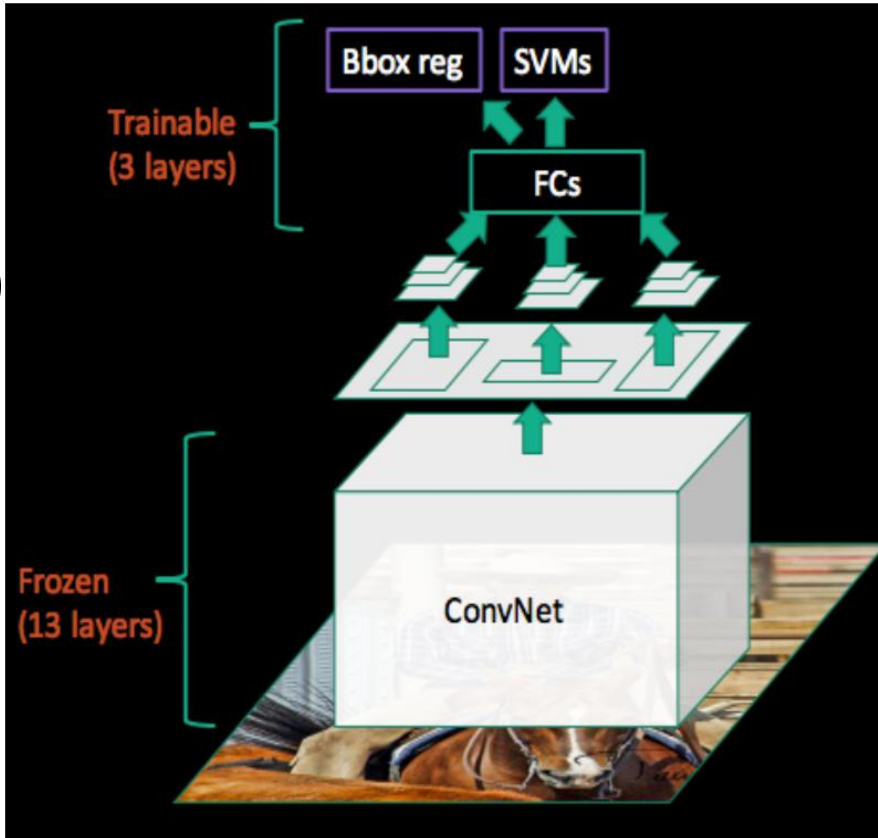


Limitations

- What are the limitations of SPPNet?
  - It is multi-stage pipe-line including
    - extracting features
    - fine-tuning a network with log loss
    - training SVMs
    - bounding-box regression
  - Features are written on a disk (Pre-trained network)

Limitations

- What are the limitations of SPPNet?
  - It is multi-stage pipe-line including
    - extracting features
    - fine-tuning a network with log loss
    - training SVMs
    - bounding-box regression
  - Features are written on a disk (Pre-trained network)
  - CNN is not updated during training.



Fast RCNN

Fast R-CNN

Ross Girshick
Microsoft Research
rbg@microsoft.com

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

Fast RCNN

- Main objectives:
  - Training is single-stage.

Fast RCNN

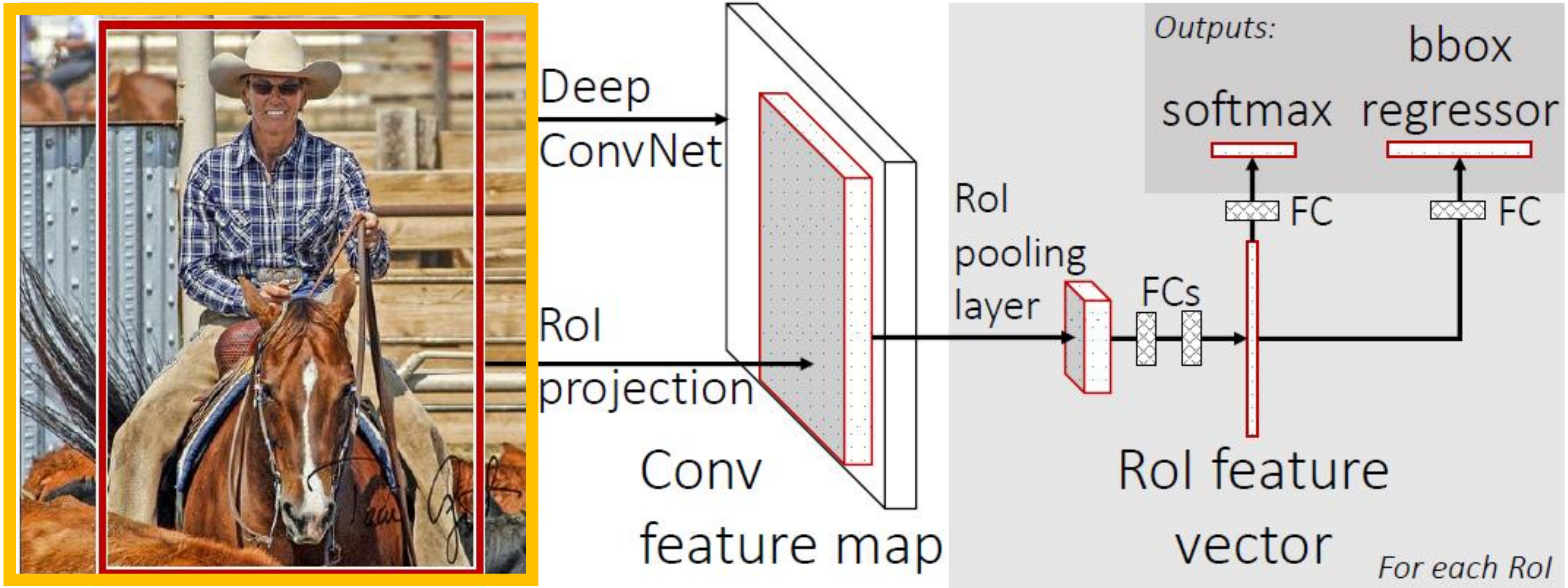
- Main objectives:
  - Training is single-stage.
  - The entire network is trainable.

Fast RCNN

- Main objectives:
  - Training is single-stage.
  - The entire network is trainable.
  - No disk storage is required for feature caching.

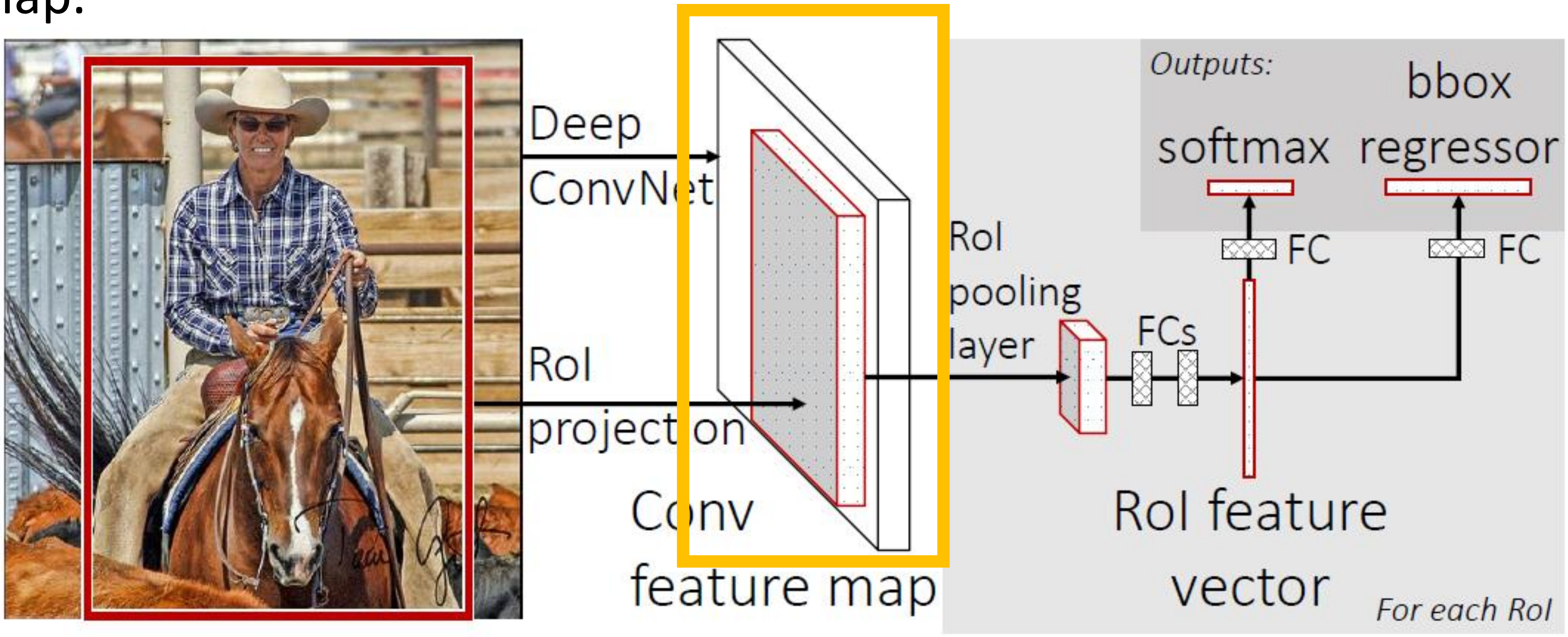
Fast RCNN

- Input: an entire image and a set of object proposals



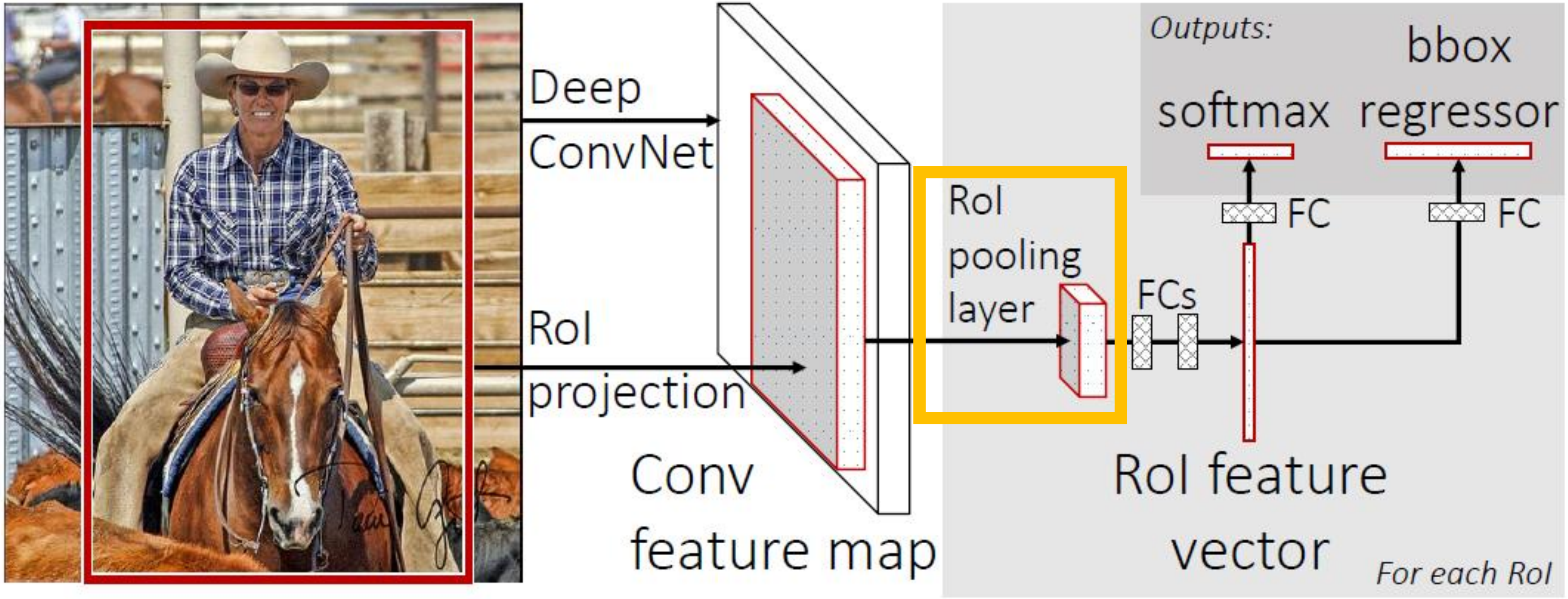
Fast RCNN

- The network first processes the whole image with several convolutional (conv) and max pooling layers to produce a conv feature map.



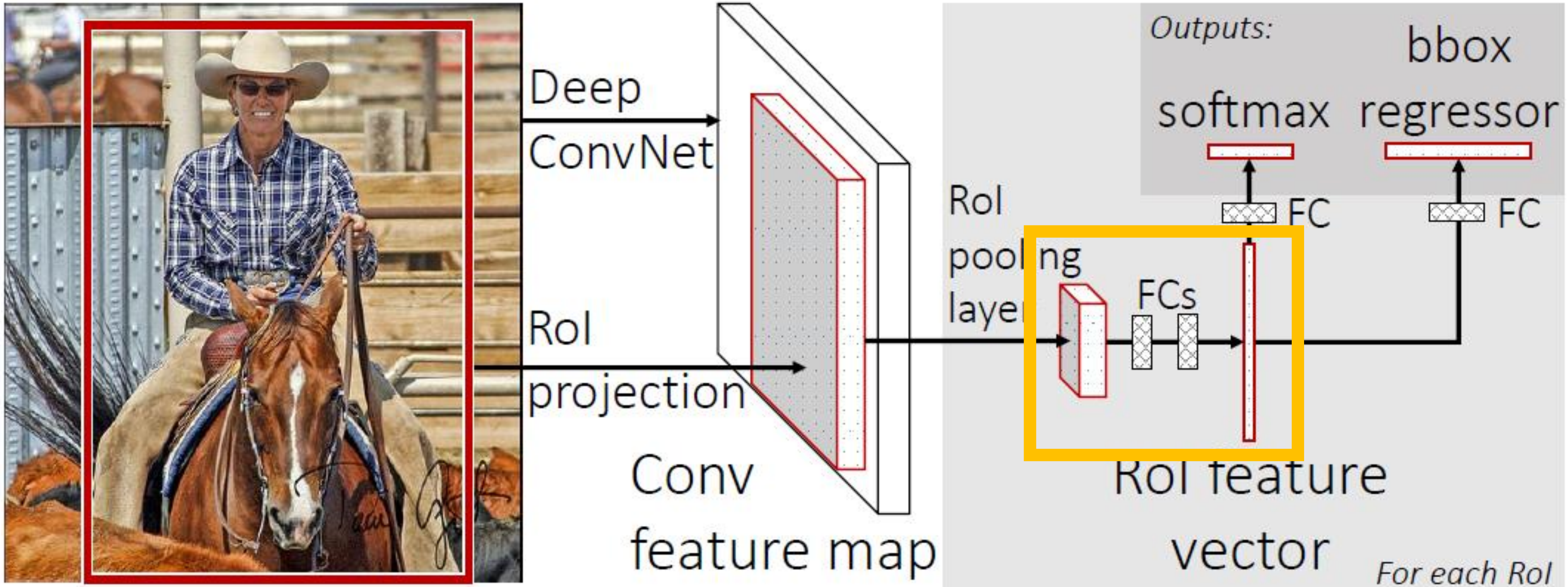
Fast RCNN

- Apply Pooling on the feature map.



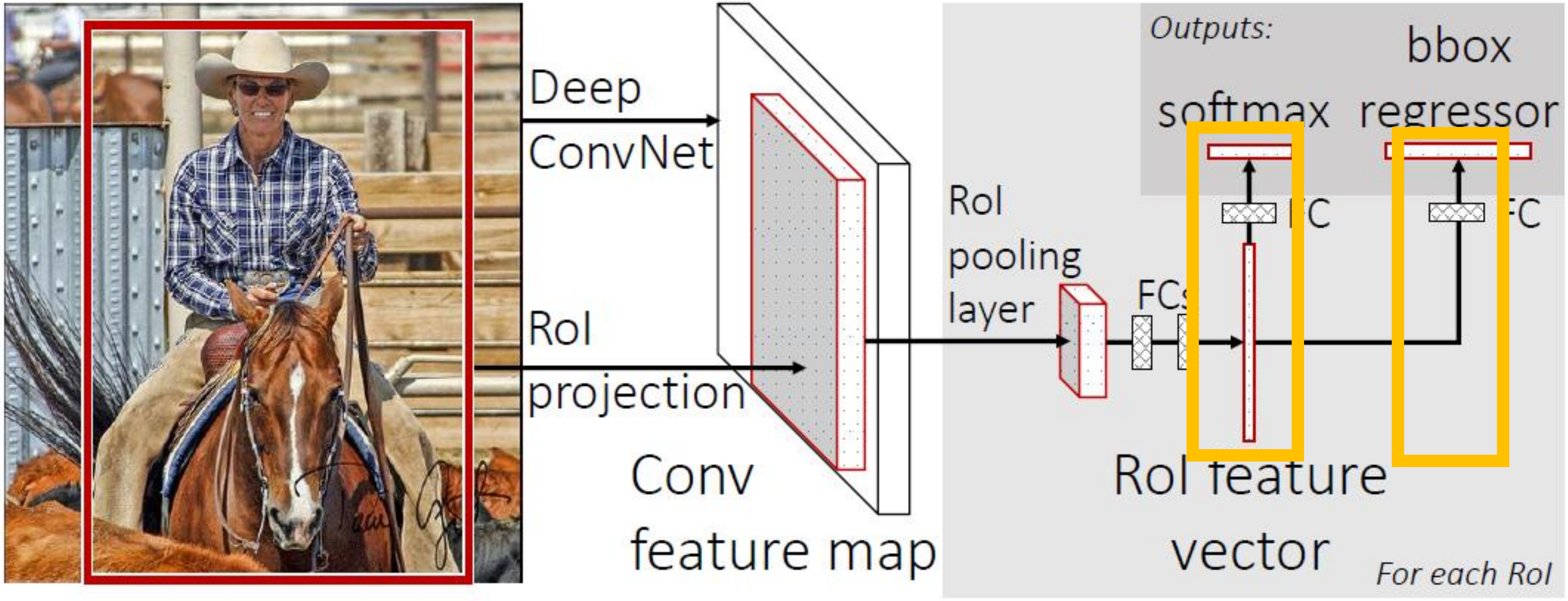
Fast RCNN

- Extract a fixed-length feature vector by feeding the ROI feature to fully connected layers.



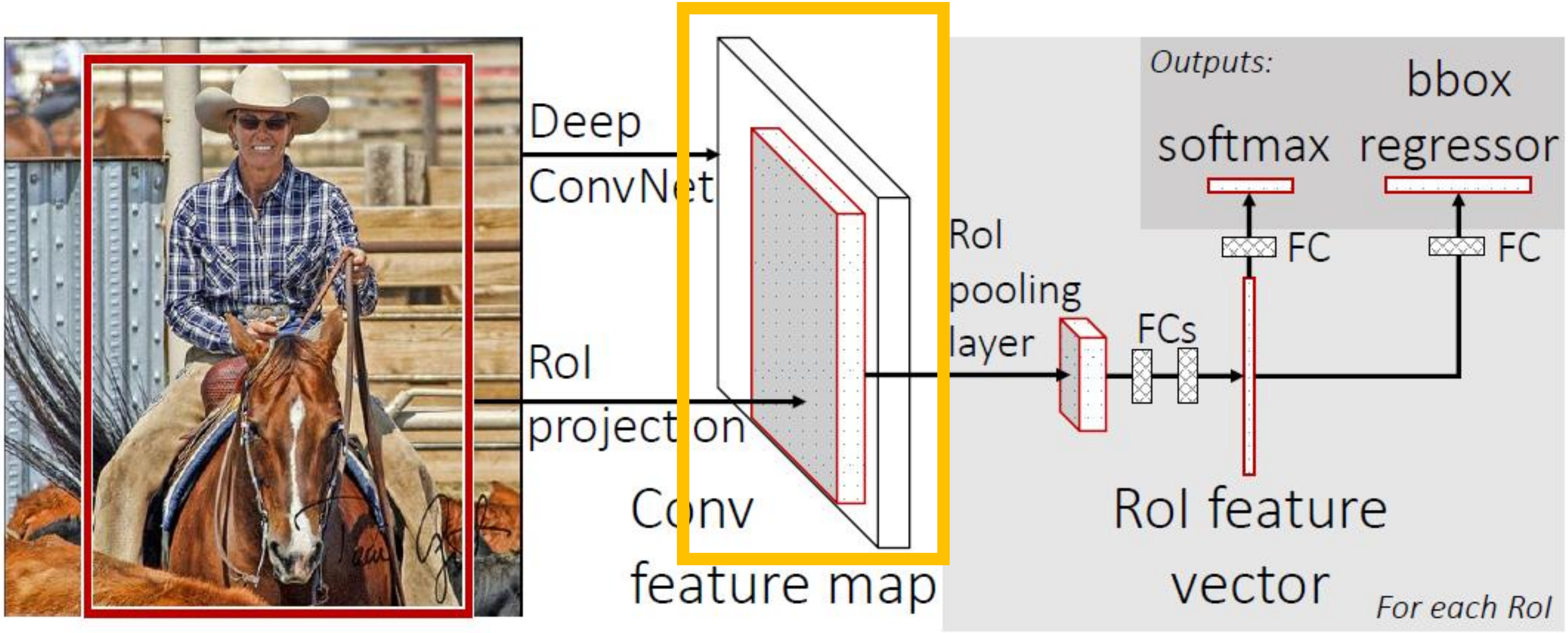
Fast RCNN

- Give the fixed feature vector to fully connected layers to find the class of object and also regress its bounding box.



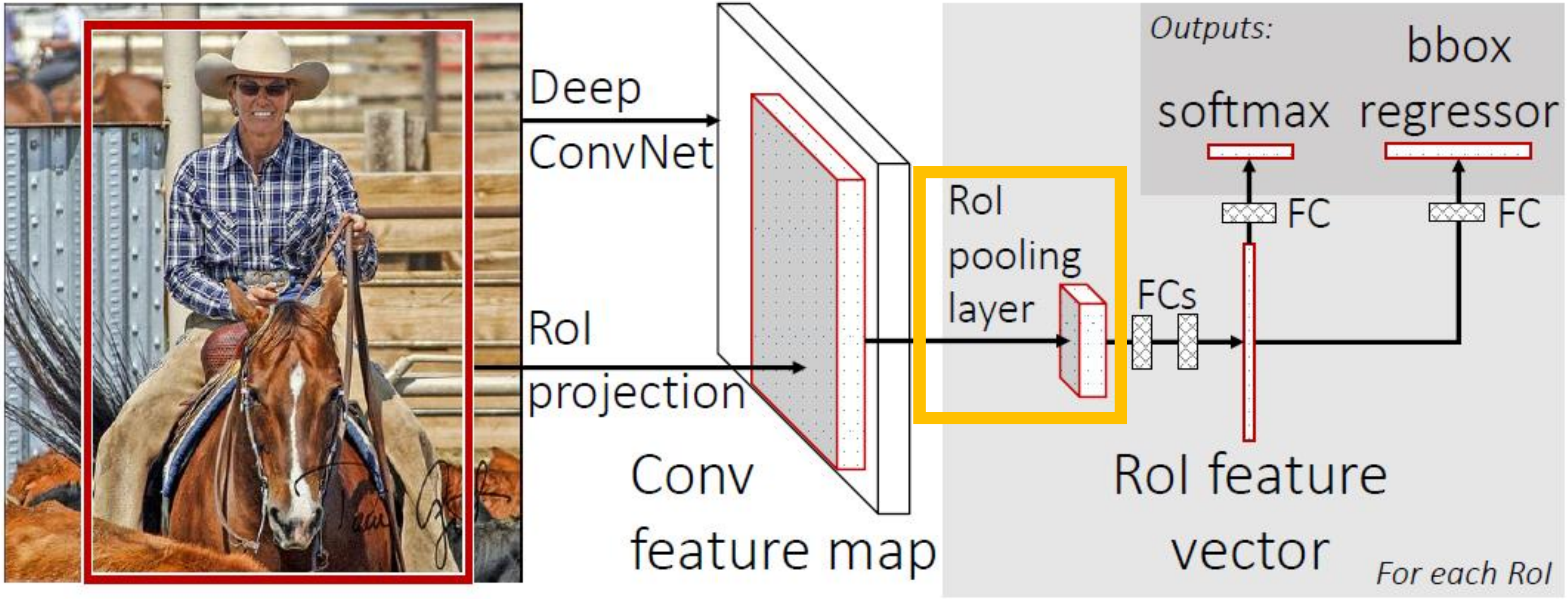
Fast RCNN

- Network is initialized by a pre-trained image classification network (e.g., AlexNet).



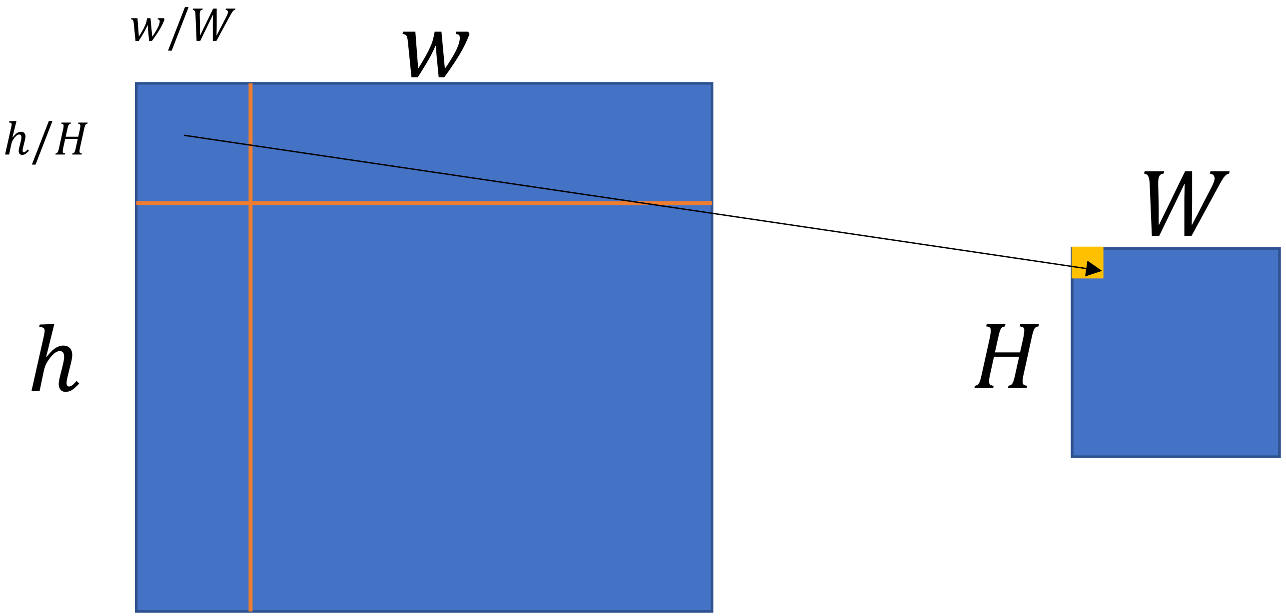
Fast RCNN

- ROI Pooling layer
  - Input a feature map with size $h \times w$
  - Output feature map with size $\mathrm{H} \times W$ (e.g., 7×7)



Fast RCNN

- ROI Pooling layer
  - Input a feature map with size $h \times w$
  - Output feature map with size $\mathrm{H} \times W$



Fast RCNN

- Loss

$$L(p, u, t^u, v) = L_{cls}(p, u) + \lambda[u \geq 1]L_{loc}(t^u, v)$$

Fast RCNN

- Loss

$$L(p, u, t^u, v) = L_{cls}(p, u) + \lambda[u \geq 1]L_{loc}(t^u, v)$$

Discrete probability distribution per RoI, $p = (p_0, \ldots, p_k)$

Fast RCNN

- Loss

$$L(p, u, t^u, v) = L_{cls}(p, u) + \lambda[u \geq 1]L_{loc}(t^u, v)$$

Output of the second sibling layer: bounding box for each of the K object classes $t^k = (t_x^k, t_y^k, t_w^k, t_h^k)$

Fast RCNN

- Loss

$$L(p, u, t^u, v) = L_{cls}(p, u) + \lambda[u \geq 1]L_{loc}(t^u, v)$$

Each training RoI is labeled with a ground truth class u

Fast RCNN

- Loss

$$L(p, u, t^u, v) = L_{cls}(p, u) + \lambda[u \geq 1]L_{loc}(t^u, v)$$

a ground truth bounding box target v.

Fast RCNN

- Loss

$$L(p, u, t^u, v) = L_{cls}(p, u) + \lambda[u \geq 1] L_{loc}(t^u, v)$$

$L_{cls}(p, u) = -\log p_u$ is log loss for true class u

Fast RCNN

- Loss

$$L(p, u, t^u, v) = L_{cls}(p, u) + \lambda[u \geq 1]L_{loc}(t^u, v)$$

$$L_{loc}(t^u, v) = \sum_{i \in \{x,y,w,h\}} smooth_{L1}(t_i^u - v_i)$$

Fast RCNN

- Loss

$$L(p,u,t^u,v) = L_{cls}(p,u) + \lambda[u \geq 1]L_{loc}(t^u,v)$$

$$L_{loc}(t^u,v) = \sum_{i\in\{x,y,w,h\}} smooth_{L1}(t_i^u - v_i)$$

Bounding box regression

Fast RCNN

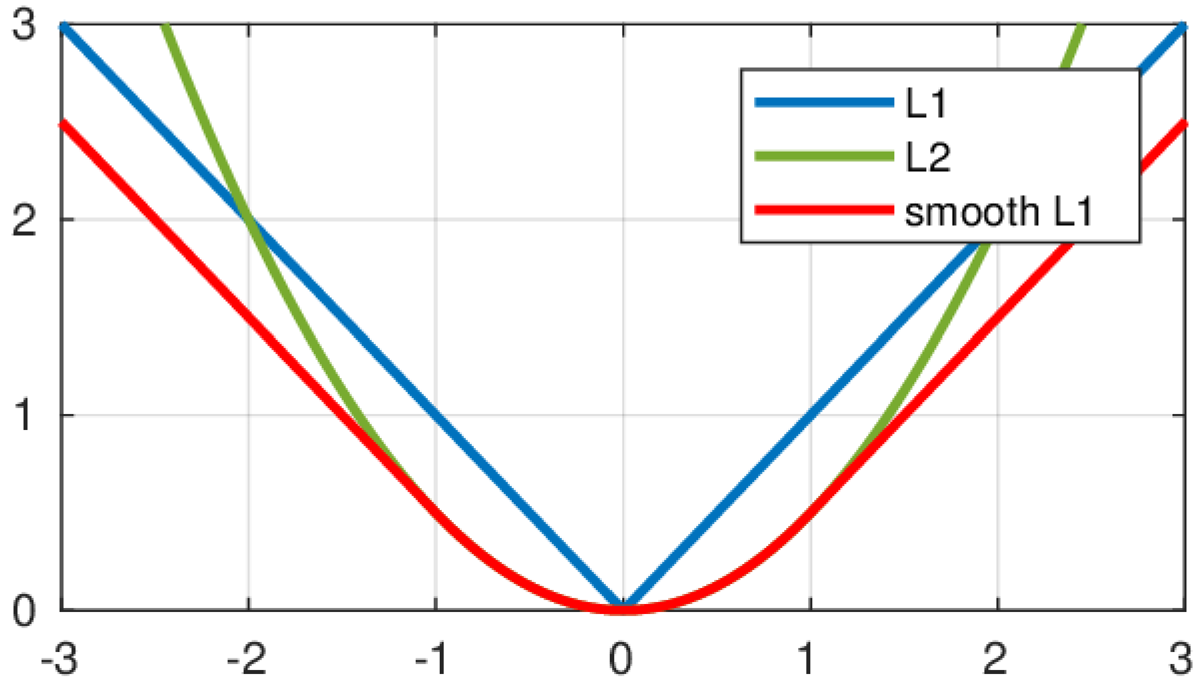
- L1 Loss

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

Fast RCNN

- L1 Loss

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



Fast RCNN

- L1 Loss is less sensitive to outliers rather than L2 used in RCNN and SPPNet.

Fast RCNN

- L1 Loss is less sensitive to outliers rather than L2 used in RCNN and SPPNet.

- When the regression targets are unbounded, training with L2 loss can require careful tuning of learning rates in order to prevent exploding gradients.

why?

Training

- Mini-batches of size R = 128, sampling 64 RoIs from each image are used.

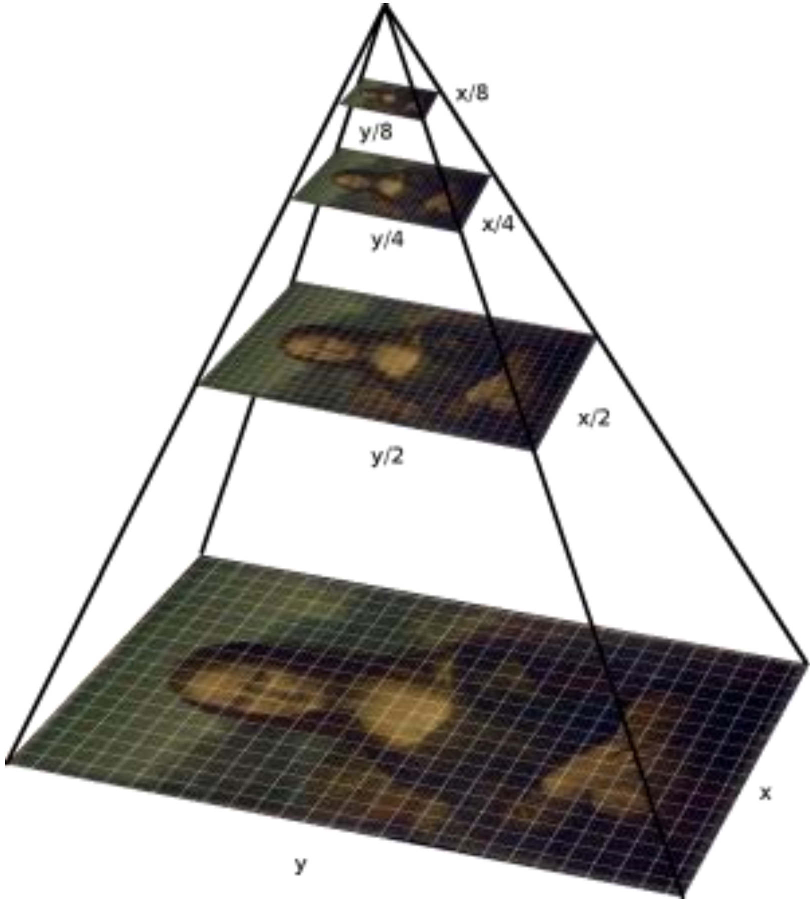
Training

- Mini-batches of size R = 128, sampling 64 RoIs from each image are used.

- 25% of the RoIs from object proposals that have intersection over union (IoU) overlap with a ground-truth bounding box of at least 0.5 are used.

Fast RCNN

- Scale Invariance
  - To make the network scale invariant, images with different scales are used from a pyramid made on the image.



Fast RCNN Detection

- At test time, R is around 2000.

Fast RCNN Detection

- At test time, R is around 2000.

- For each test RoI r, a confidence score p_k is defined.

Fast RCNN Detection

- At test time, R is around 2000.

- For each test RoI r, a confidence score p_k is defined.

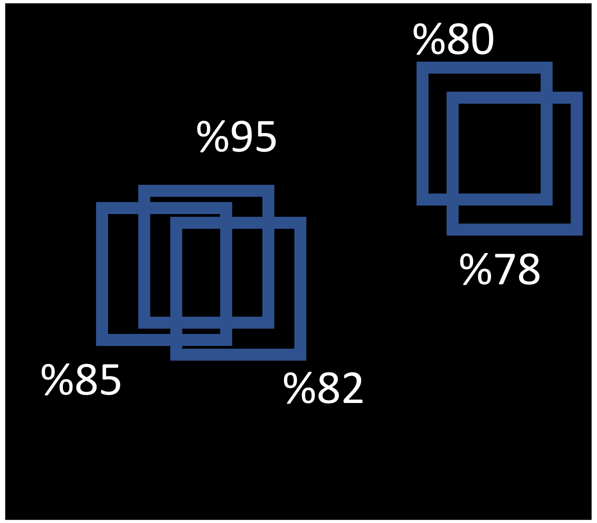
- Non-maximum Suppression is then used for each class to determine the bounding box.

Fast RCNN Detection

- Non-Max Suppression
  - It prunes the bounding boxes.

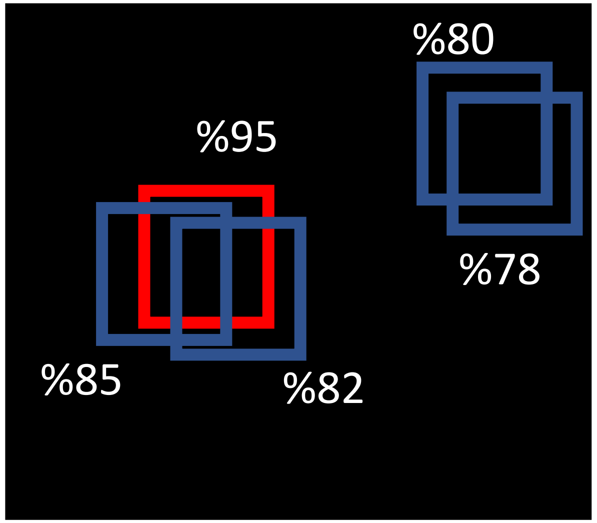
Non-Max Suppression

- Having a set of bounding boxes with their confidence score in set $\mathcal{B}$.



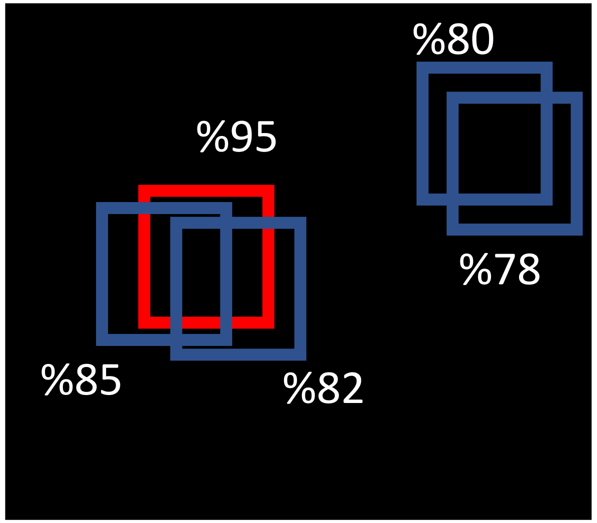
Non-Max Suppression

- Select the box with the highest score, remove it from list $\mathcal{B}$ and add it to list D.



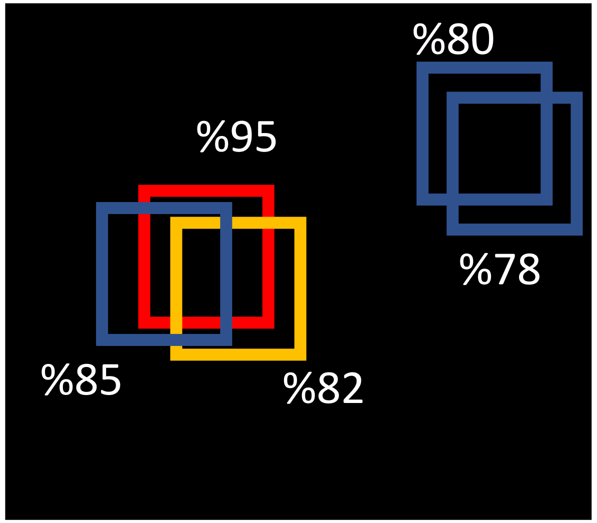
Non-Max Suppression

- Select the box with the highest score, remove it from list $\mathcal{B}$ and add it to list D.
- Compare this with all the boxes and if the IoU (intersection over union) of them is bigger than a threshold, remove them from $\mathcal{B}$.



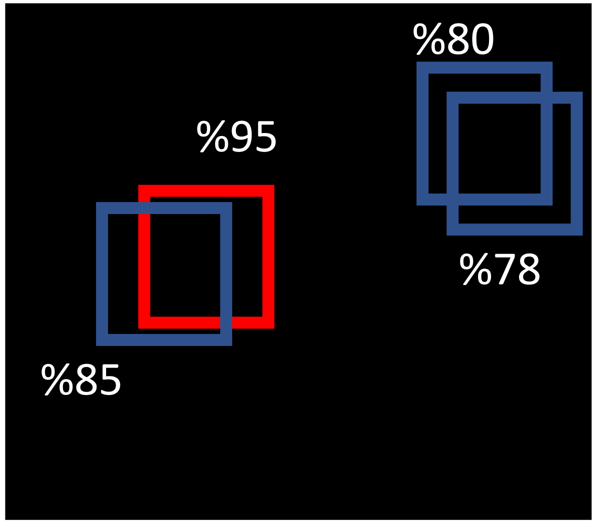
Non-Max Suppression

- Select the box with the highest score, remove it from list $\mathcal{B}$ and add it to list D.
- Compare this with all the boxes and if the IoU (intersection over union) of them is bigger than a threshold, remove them from $\mathcal{B}$.



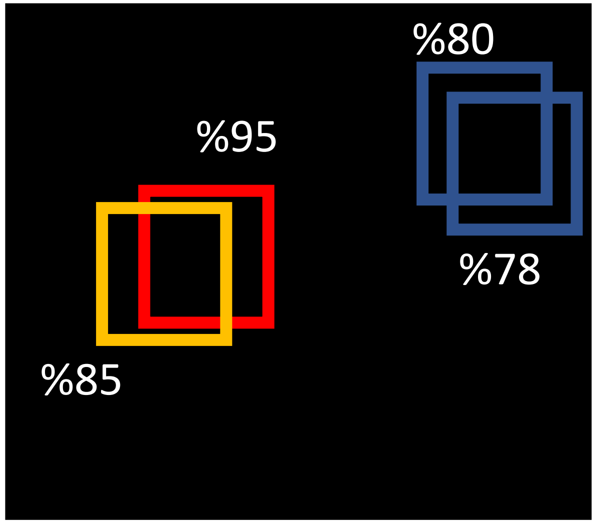
Non-Max Suppression

- Select the box with the highest score, remove it from list $\mathcal{B}$ and add it to list D.
- Compare this with all the boxes and if the IoU (intersection over union) of them is bigger than a threshold, remove them from $\mathcal{B}$.



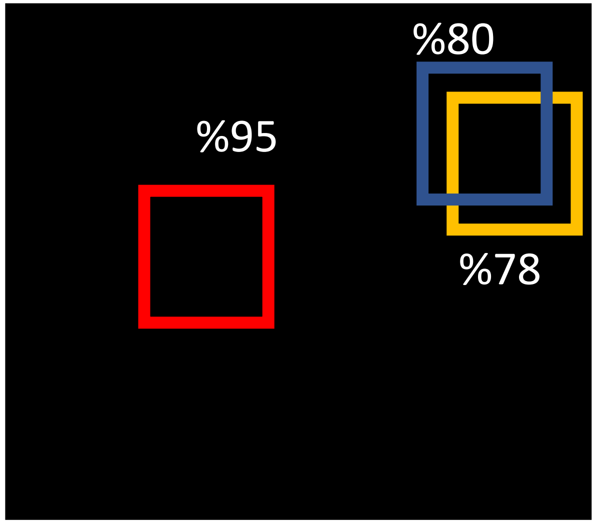
Non-Max Suppression

- Select the box with the highest score, remove it from list $\mathcal{B}$ and add it to list D.
- Compare this with all the boxes and if the IoU (intersection over union) of them is bigger than a threshold, remove them from $\mathcal{B}$.



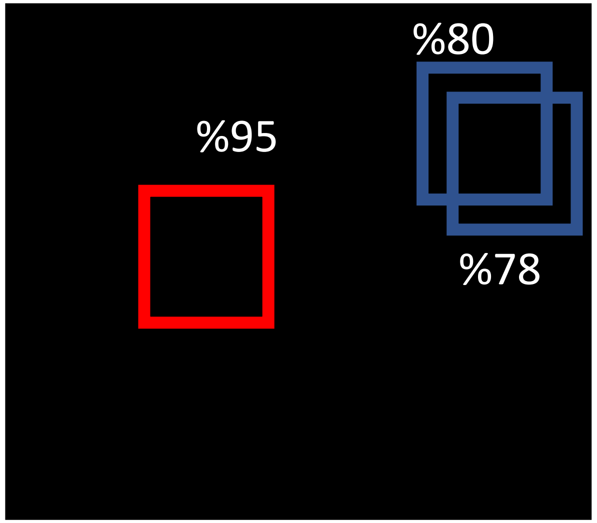
Non-Max Suppression

- Select the box with the highest score, remove it from list $\mathcal{B}$ and add it to list D.
- Compare this with all the boxes and if the IoU (intersection over union) of them is bigger than a threshold, remove them from $\mathcal{B}$.



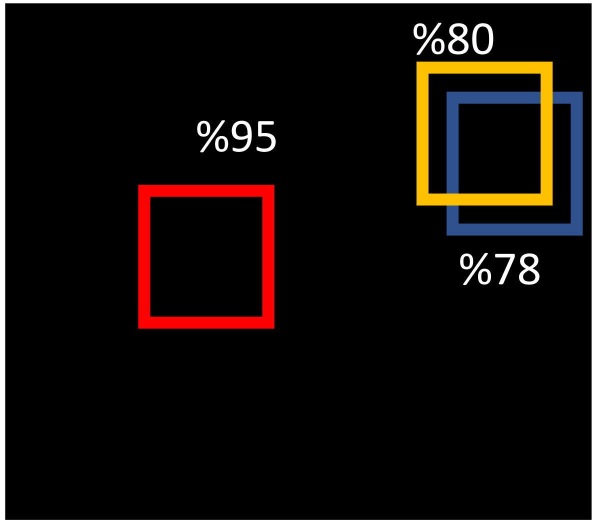
Non-Max Suppression

- Select the box with the highest score, remove it from list $\mathcal{B}$ and add it to list D.
- Compare this with all the boxes and if the IoU (intersection over union) of them is bigger than a threshold, remove them from $\mathcal{B}$.



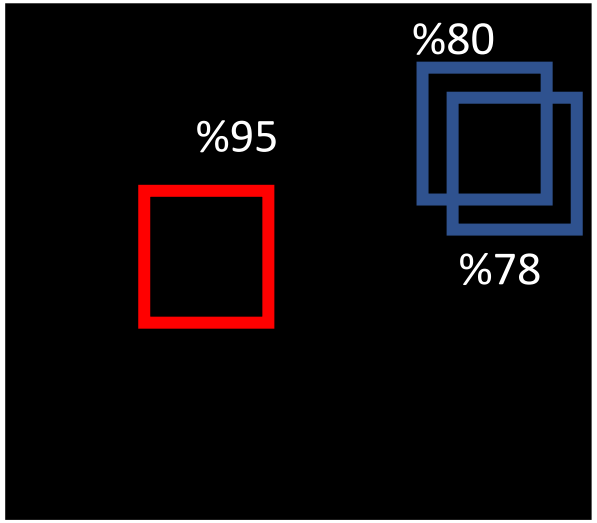
Non-Max Suppression

- Select the box with the highest score, remove it from list $\mathcal{B}$ and add it to list D.
- Compare this with all the boxes and if the IoU (intersection over union) of them is bigger than a threshold, remove them from $\mathcal{B}$.



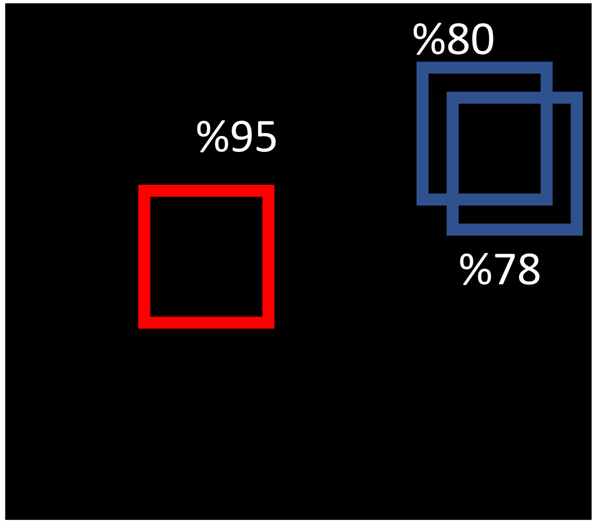
Non-Max Suppression

- Select the box with the highest score, remove it from list $\mathcal{B}$ and add it to list D.
- Compare this with all the boxes and if the IoU (intersection over union) of them is bigger than a threshold, remove them from $\mathcal{B}$.



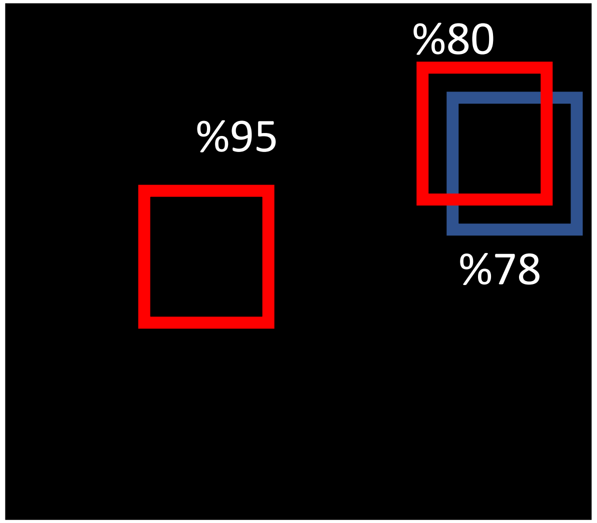
Non-Max Suppression

- Select the box with the highest score, remove it from list $\mathcal{B}$ and add it to list D.
- Compare this with all the boxes and if the IoU (intersection over union) of them is bigger than a threshold, remove them from $\mathcal{B}$.
- Repeat on $\mathcal{B}$.



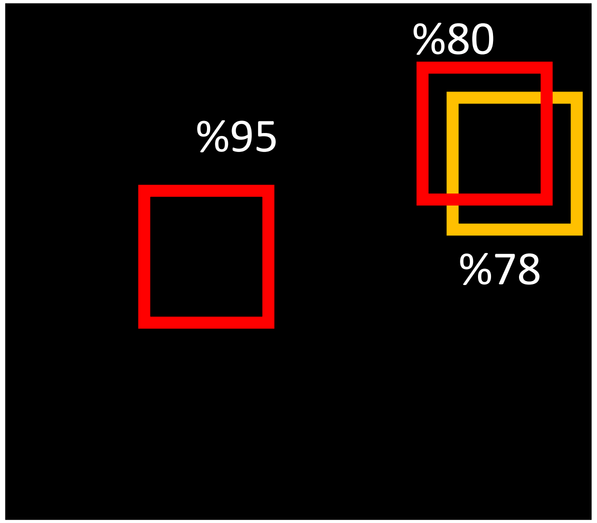
Non-Max Suppression

- Select the box with the highest score, remove it from list $\mathcal{B}$ and add it to list D.
- Compare this with all the boxes and if the IoU (intersection over union) of them is bigger than a threshold, remove them from $\mathcal{B}$.
- Repeat on $\mathcal{B}$.



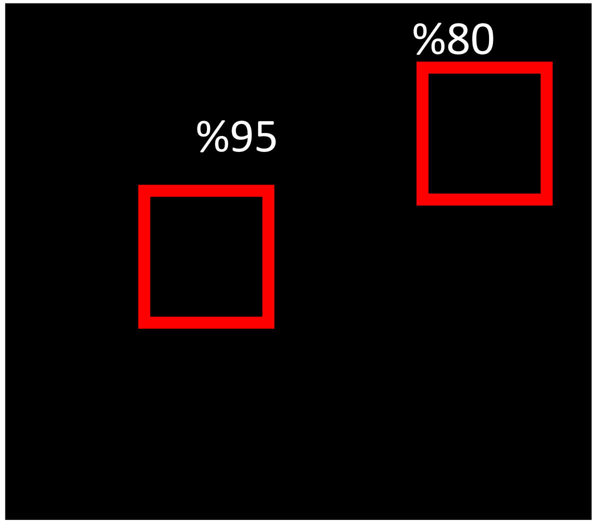
Non-Max Suppression

- Select the box with the highest score, remove it from list $\mathcal{B}$ and add it to list D.
- Compare this with all the boxes and if the IoU (intersection over union) of them is bigger than a threshold, remove them from $\mathcal{B}$.
- Repeat on $\mathcal{B}$.



Non-Max Suppression

- Select the box with the highest score, remove it from list $\mathcal{B}$ and add it to list D.
- Compare this with all the boxes and if the IoU (intersection over union) of them is bigger than a threshold, remove them from $\mathcal{B}$.
- Repeat on $\mathcal{B}$.



Fast RCNN Detection

- Truncated SVD for faster detection by making an approximation of a layer.

Fast RCNN Detection

- Truncated SVD for faster detection by making an approximation of a layer.

- Fully connected layers are more time consuming in the detection than fully convolution layers.

Fast RCNN Detection

- Truncated SVD for faster detection by making an approximation of a layer.

- Fully connected layers are more time consuming in the detection than fully convolution layers.

- We can use SVD to approximate fully connected layers.

Fast RCNN Detection

- Without SVD:

$$W_{u \times v} X = Y$$

Fast RCNN Detection

- Without SVD:

$$W_{u\times v}X = Y$$

Fully connected layer transformation

Fast RCNN Detection

- Without SVD:

$$W_{u \times v} X = Y$$

Input to the fully connected layer

Fast RCNN Detection

- Without SVD:

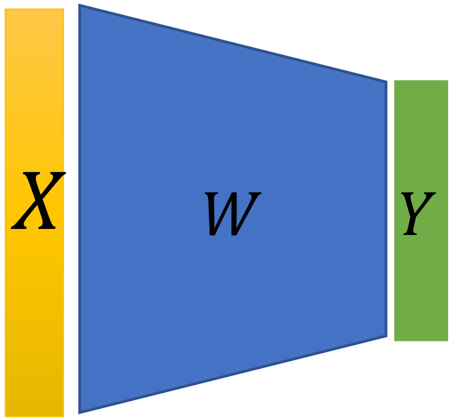
$$W_{u \times v} X = Y$$

output

Fast RCNN Detection

- Without SVD:

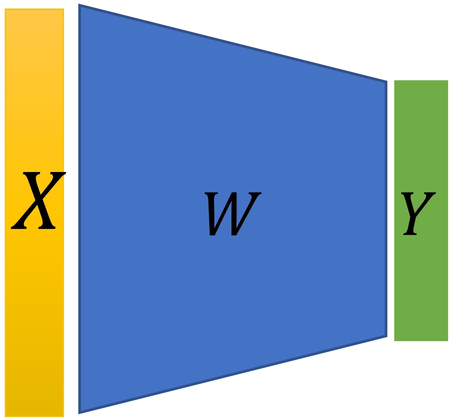
$$W_{u \times v} X = Y$$



Fast RCNN Detection

- Use SVD:

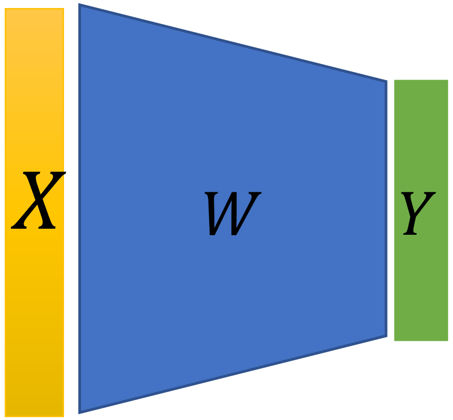
$$W_{u\times v} = U\sum V^T$$

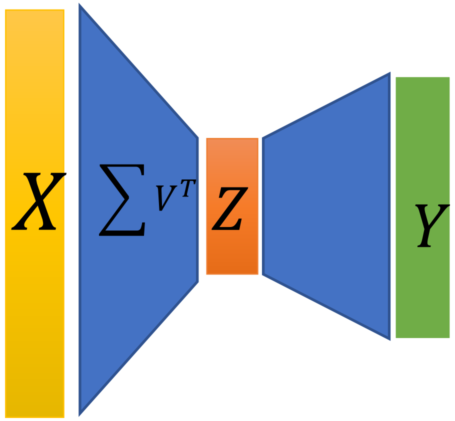


Fast RCNN Detection

- Drop singular values with low values.
- Change the layers by connecting fully connected layers with no non-linearity.

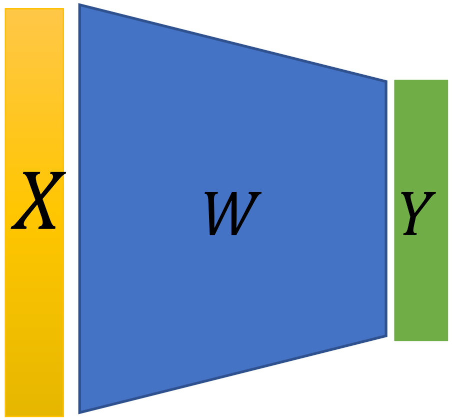
$$W_{u\times v} = U\sum V^T$$

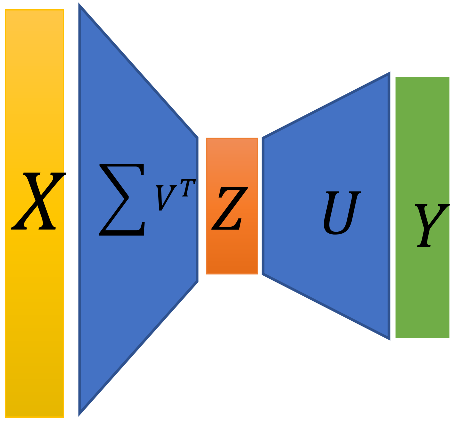




Fast RCNN Detection

$$W_{u \times v} = U \sum V^T$$

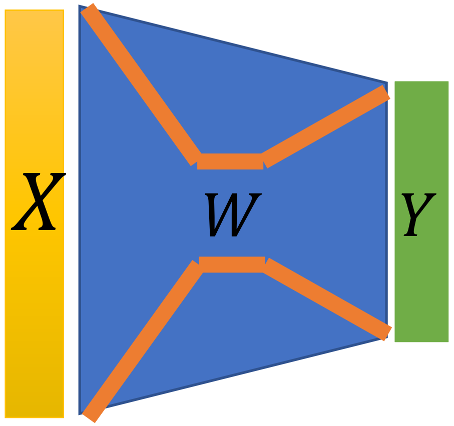


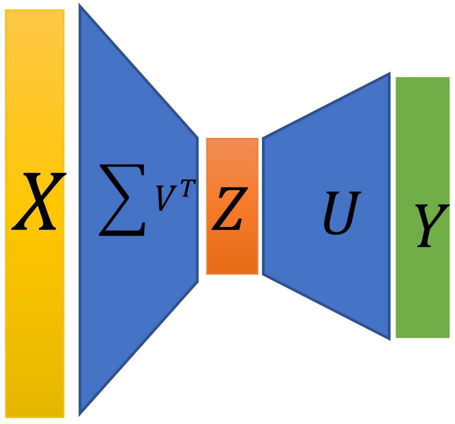


Fast RCNN Detection

- Change the layers by connecting fully connected layers with no non-linearity.

$$W_{u\times v} = U \sum V^T$$





Fast RCNN Results

	VOC2007
SPPNet BB	63.1%
R-CNN BB	66.0%
Fast RCNN	66.9%
Fast RCNN (07+12)	70.0%

Fast RCNN Results

- Fast RCNN is 9 times faster than RCNN in training, 213 times faster at test time.
- Fast RCNN is 3 times faster than SPPNet in training, 10 times faster at test time.

Fast RCNN Limitation

- What is the biggest limitation of Fast RCNN?

Fast RCNN Limitation

- What is the biggest limitation of Fast RCNN?
  - We still need to have RoI proposals with some algorithms.

Fast RCNN Limitation

- What is the biggest limitation of fast RCNN?
  - We still need to have RoI proposals with some algorithms.
  - In RCNN, the region proposals are generated in the pixel level by a selective search (SS) while in Fast RCNN, it happens in the feature level.

Fast RCNN Limitation

- What is the biggest limitation of RCNN?
  - We still need to have RoI proposals with some algorithms.
  - In RCNN, the region proposals are generated in the pixel level by a selective search (SS) while in Fast RCNN, it happens in the feature level.

Faster RCNN

- The motivation is to have an end to end network by also providing some meaningful RoI proposals.

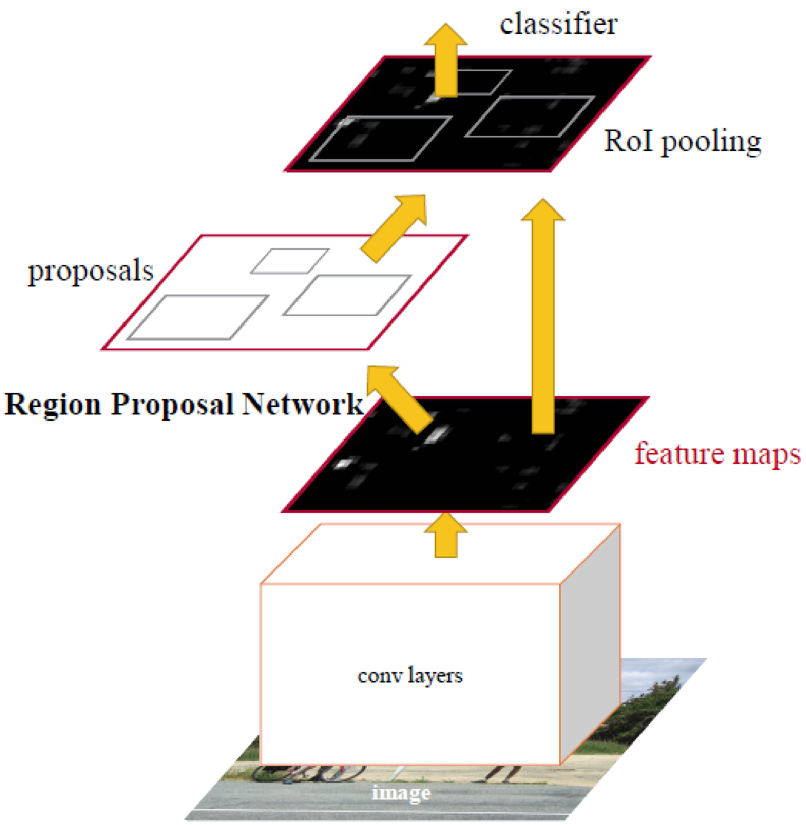
Faster RCNN

- The motivation is to have an end to end network by also providing some meaningful RoI proposals.

How?

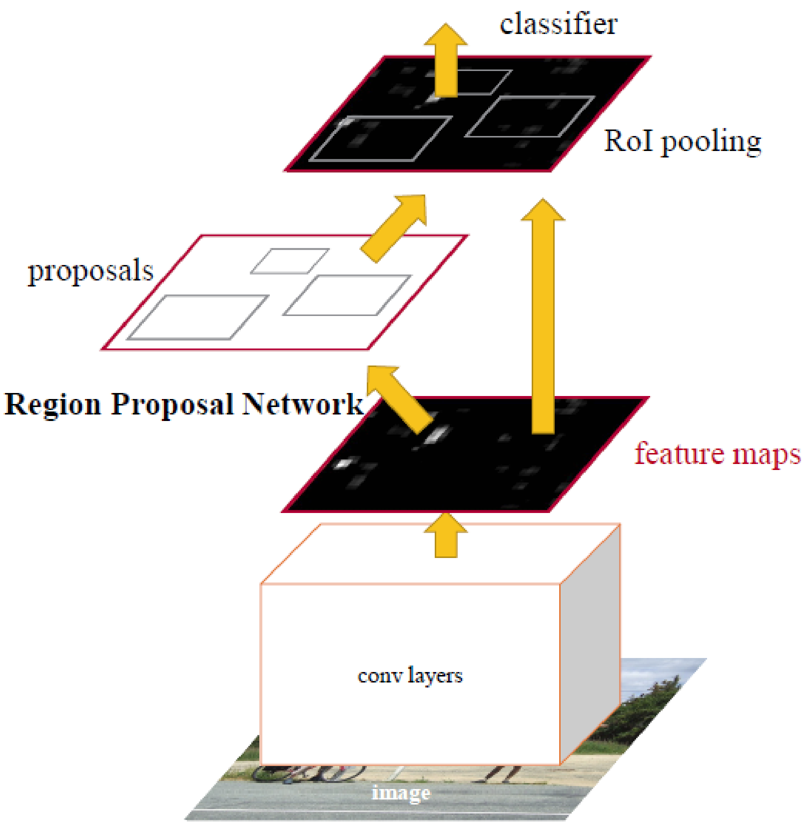
Faster RCNN

- The basic idea is to offer a Region Proposal Network (RPN) that shares some features with the detection network.



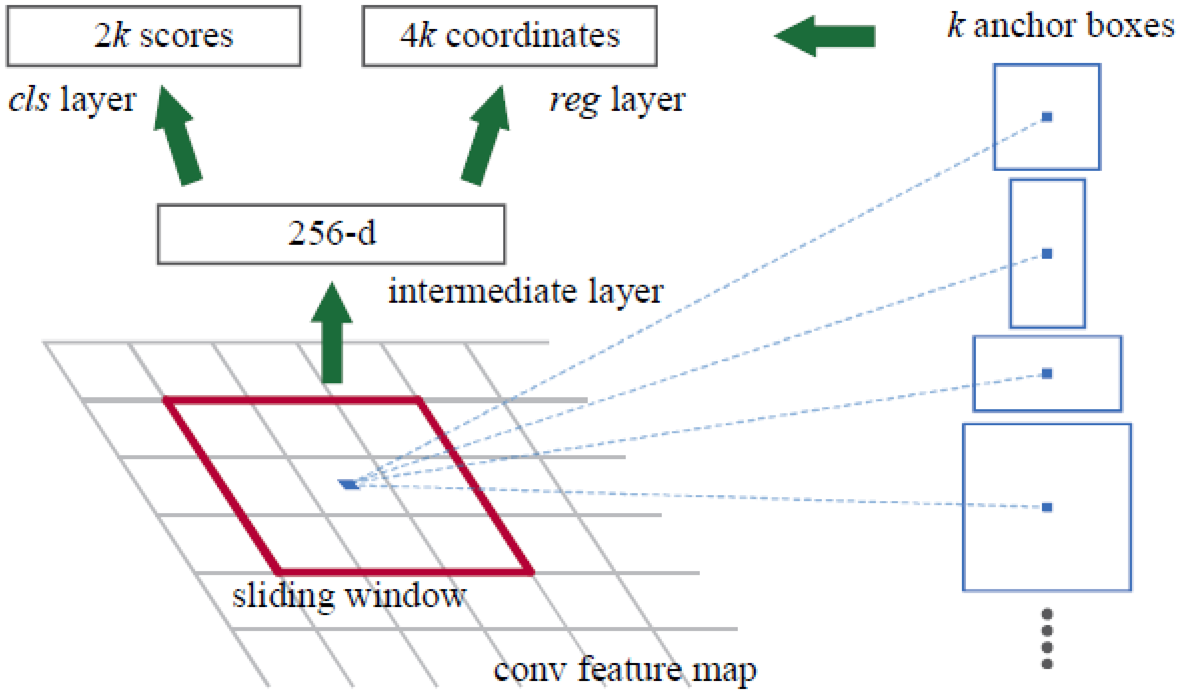
RPN

- An image is first sent to a convolution network to make a feature map.



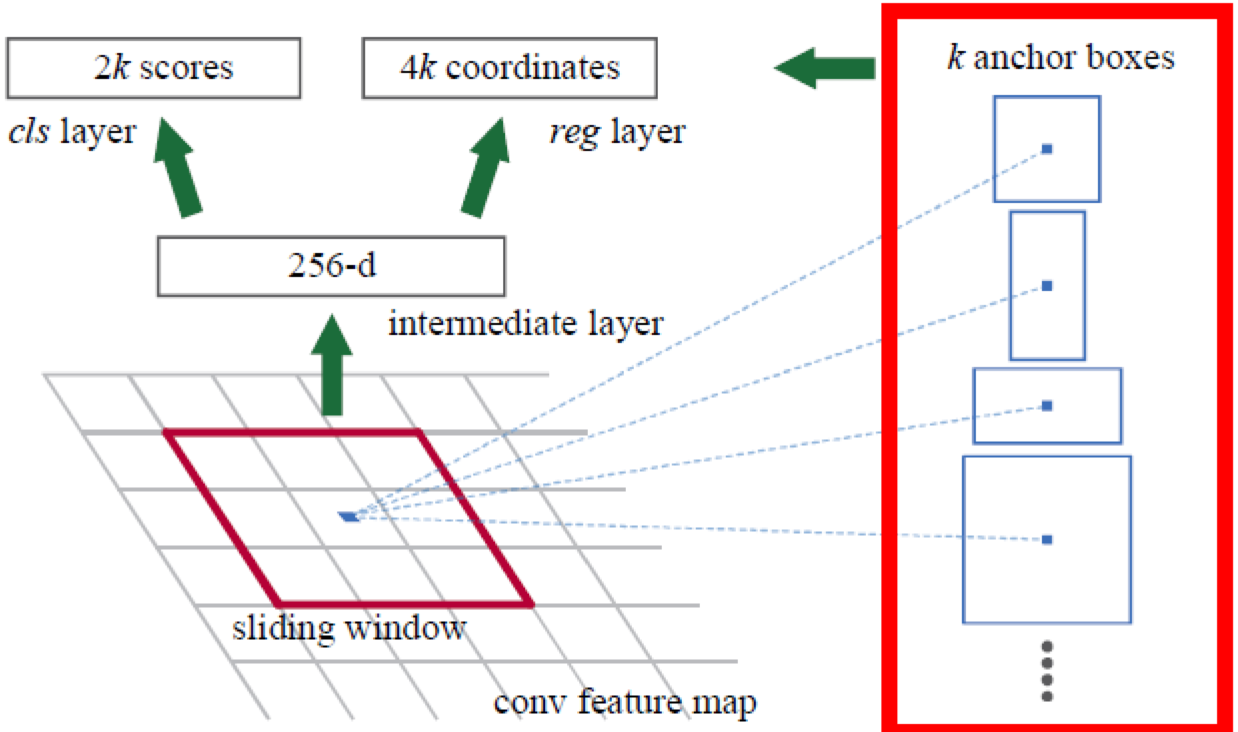
RPN

- Then, a sliding window is used over each feature map.



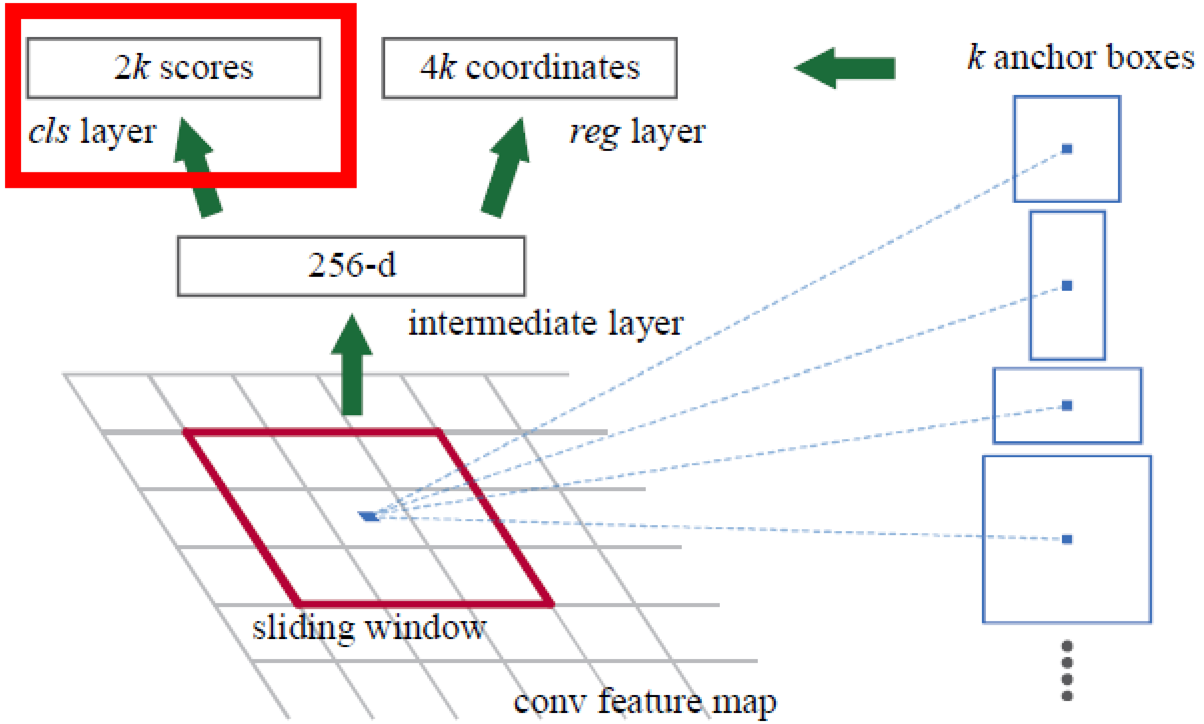
RPN

- For each location, k anchor boxes are used (3 scales of 128, 256 and 512, and 3 aspect ratios of 1:1, 1:2, 2:1) for generating region proposals.



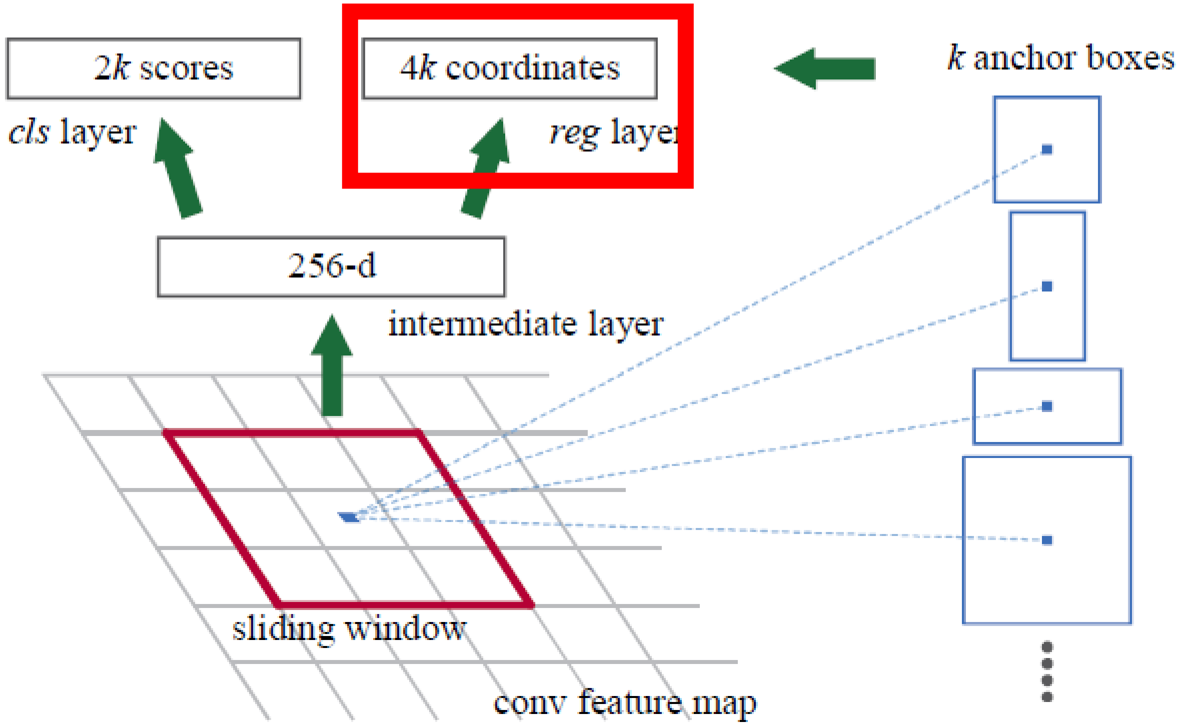
RPN

- A *cls* (classification) layer outputs *2k* scores whether there is object or not for *k* boxes.



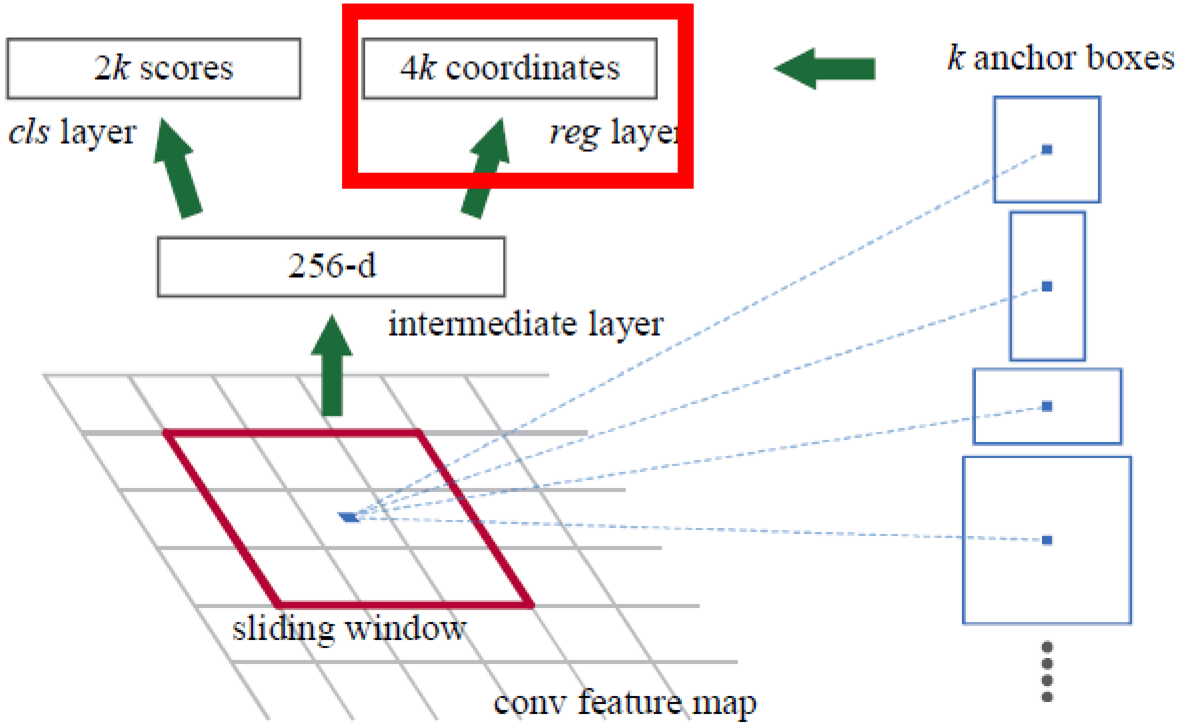
RPN

- A *reg* (regularization) layer outputs *4k* for the coordinates (box center coordinates, width and height) of *k* boxes.



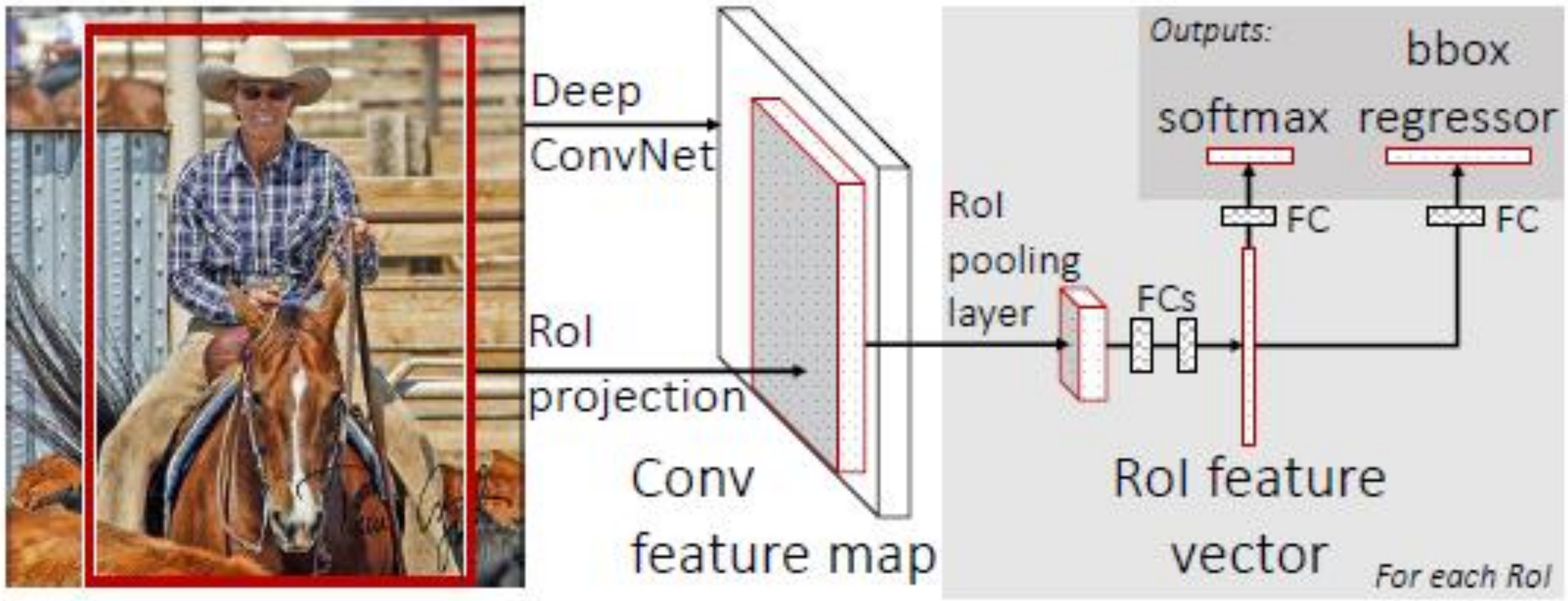
RPN

- This is how you make the region proposals.



Faster RCNN

- Detection network is the same as Fast RCNN.



Results

- Fewer and better proposals not only bring speedup, but also detection performance boost.

method	# proposals	data	mAP (%)	time (ms)
SS	2k	07	66.9	1830
SS	2k	07+12	70.0	1830
RPN+VGG, unshared	300	07	68.5	342
RPN+VGG, shared	300	07	69.9	**196**
RPN+VGG, shared	300	07+12	**73.2**	**196**

YOLO

- You Only Look Once
  - It spits out everything (box, class, probability) all at once as the output

YOLO

- You Only Look Once
- It uses a single convolutional neural network to simultaneously predict multiple bounding boxes and class probabilities for objects in an image.

YOLO

- You Only Look Once
- It uses a single convolutional neural network to simultaneously predict multiple bounding boxes and class probabilities for objects in an image.
- The algorithm frames detection as a single regression problem, going directly from image pixels to bounding box coordinates and class probabilities.
  - All the losses are L2

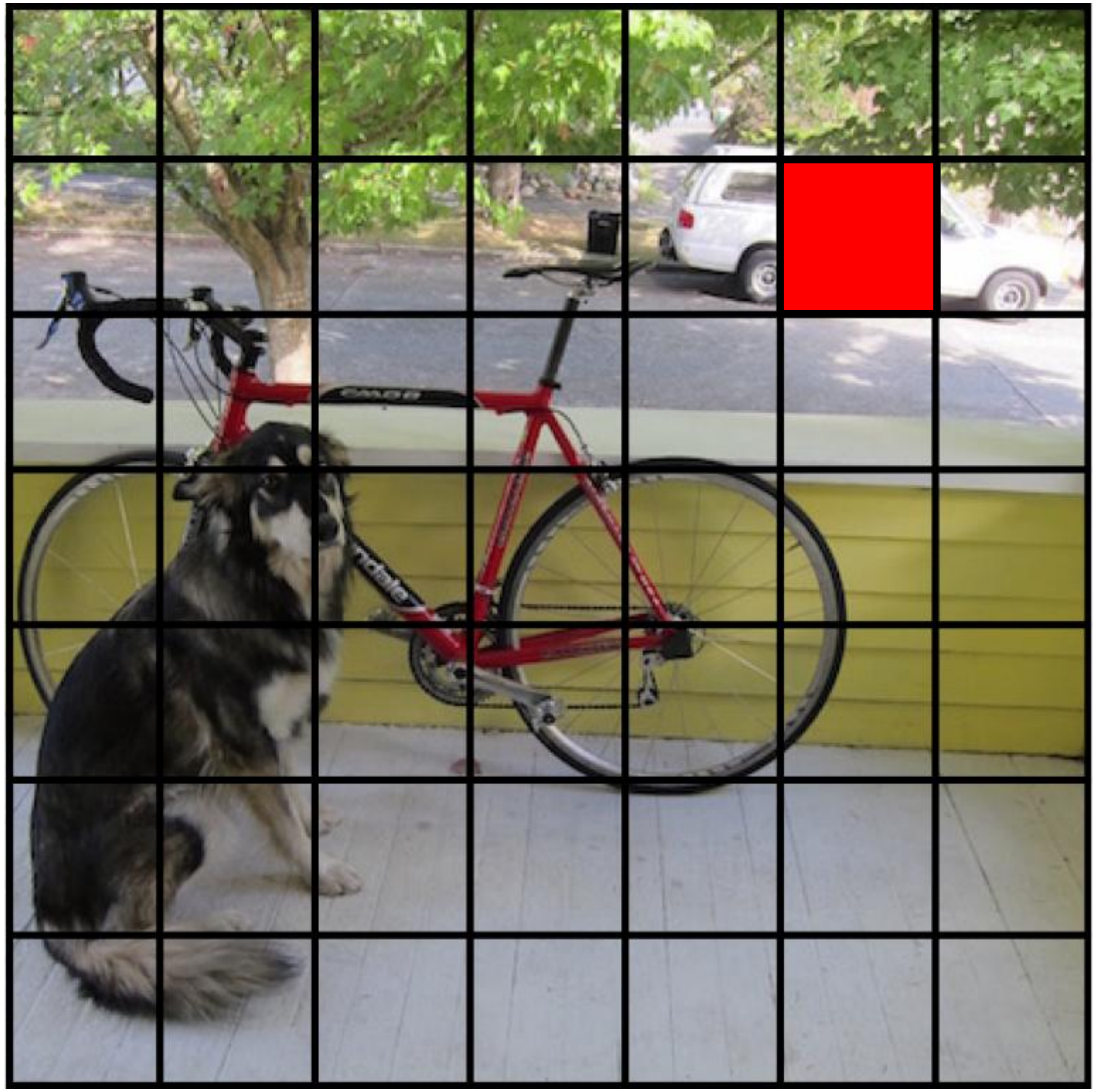
YOLO

- You Only Look Once
- It uses a single convolutional neural network to simultaneously predict multiple bounding boxes and class probabilities for objects in an image.
- The algorithm frames detection as a single regression problem, going directly from image pixels to bounding box coordinates and class probabilities.
- This makes YOLO fast, able to process streaming video in real-time with less than 25 milliseconds of latency.

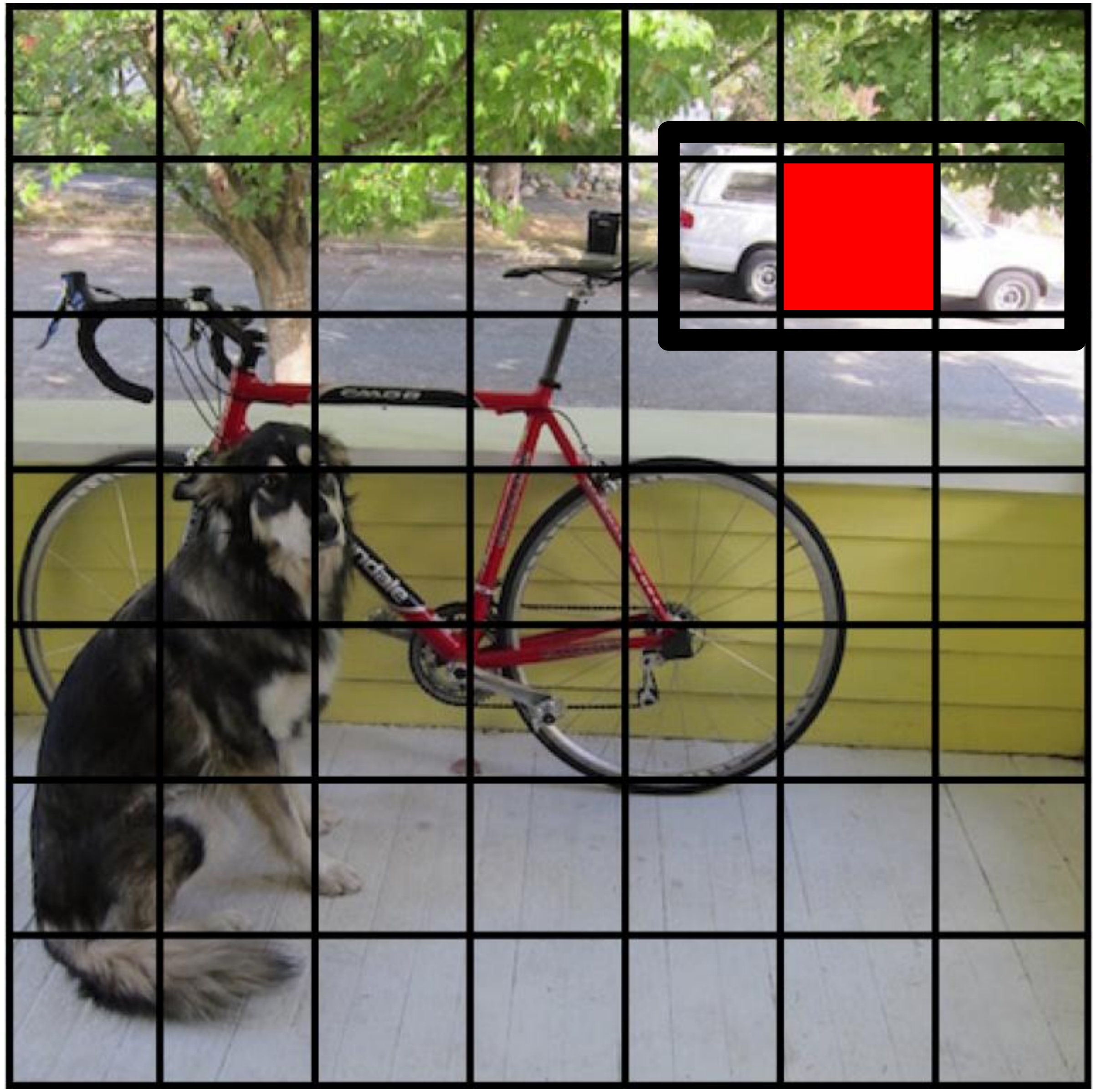
Say you have an image...

Split it into a grid

For each cell predicts P(obj)

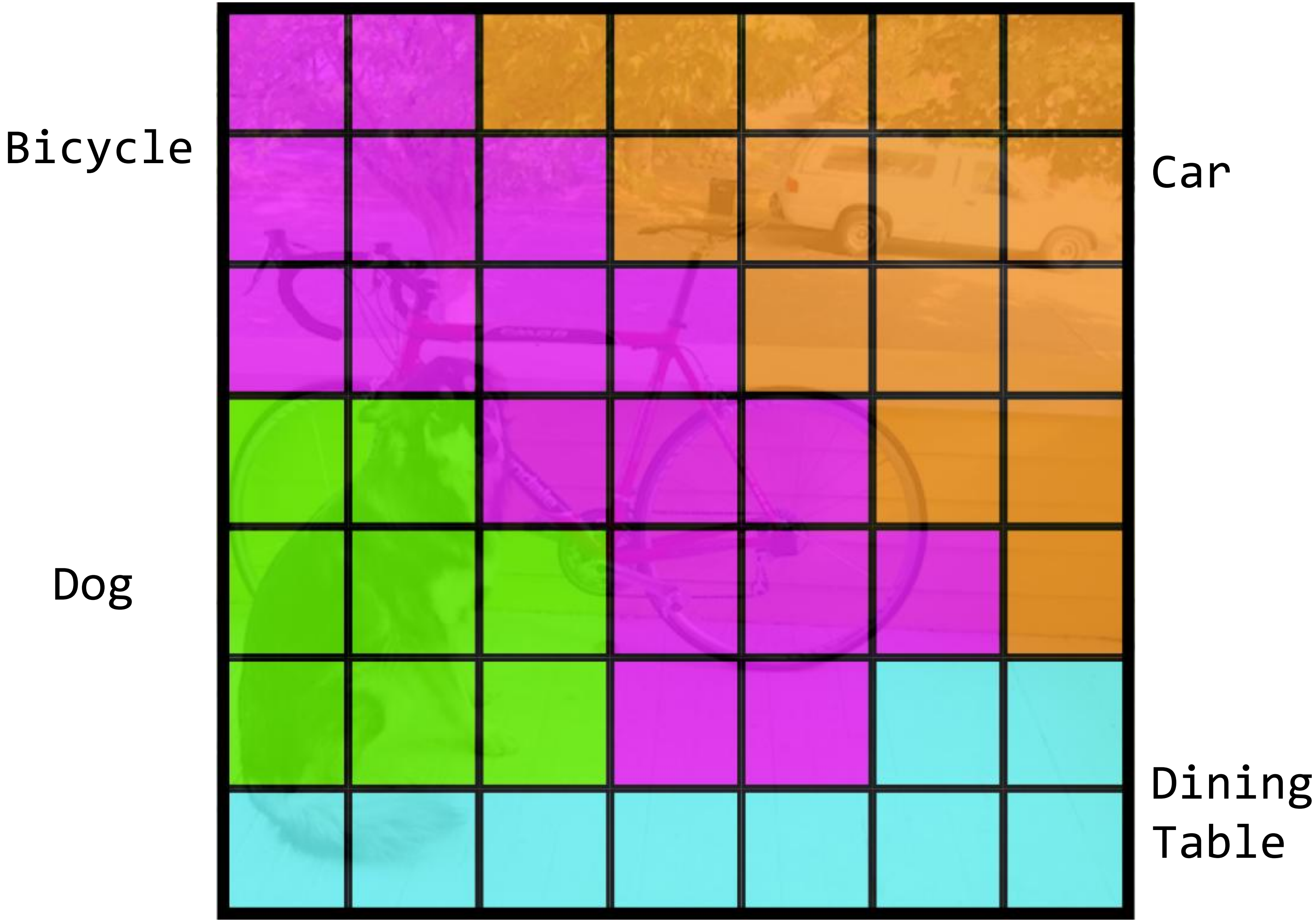


Also predict a bounding box

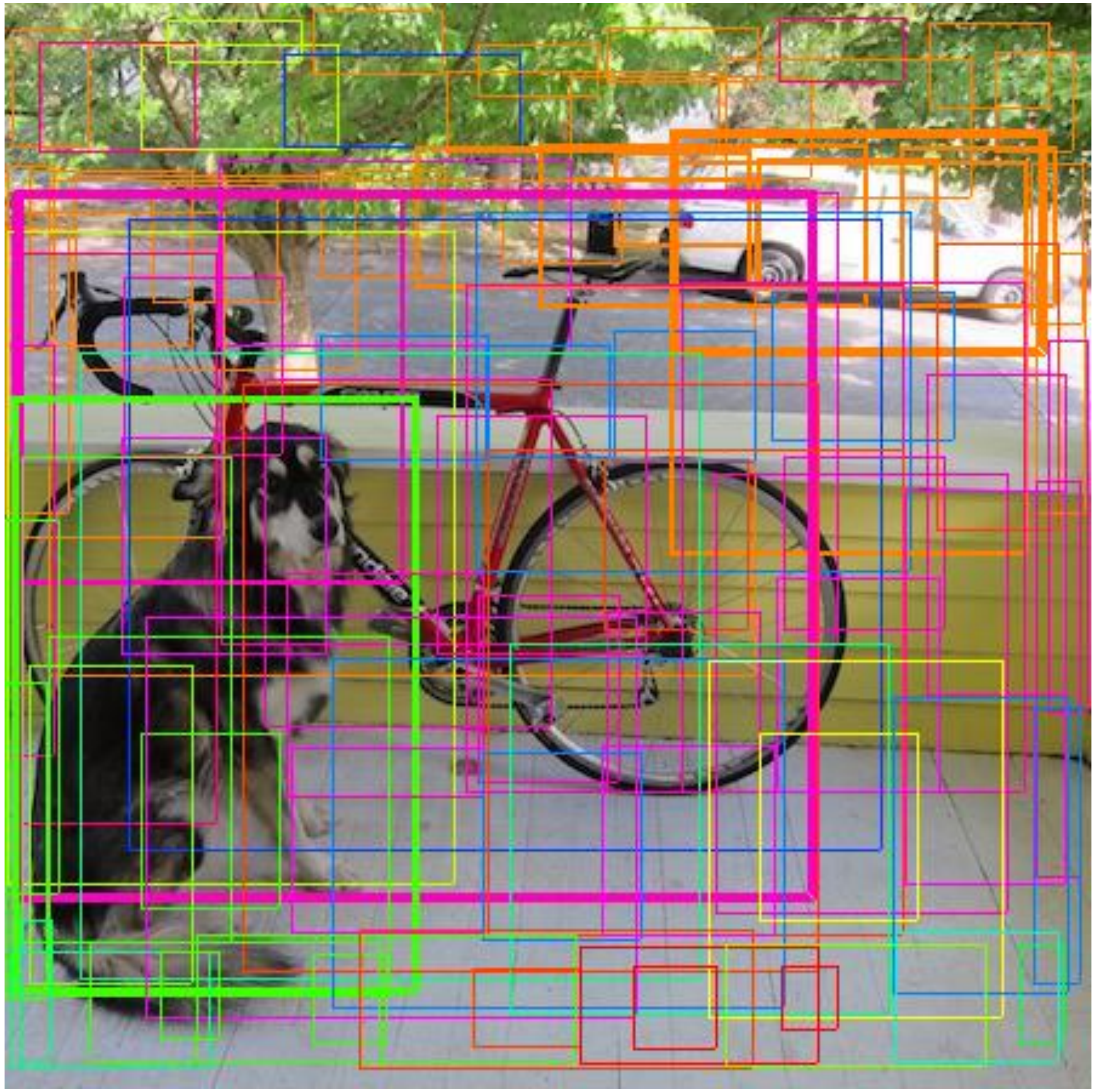


Also predict a bounding box

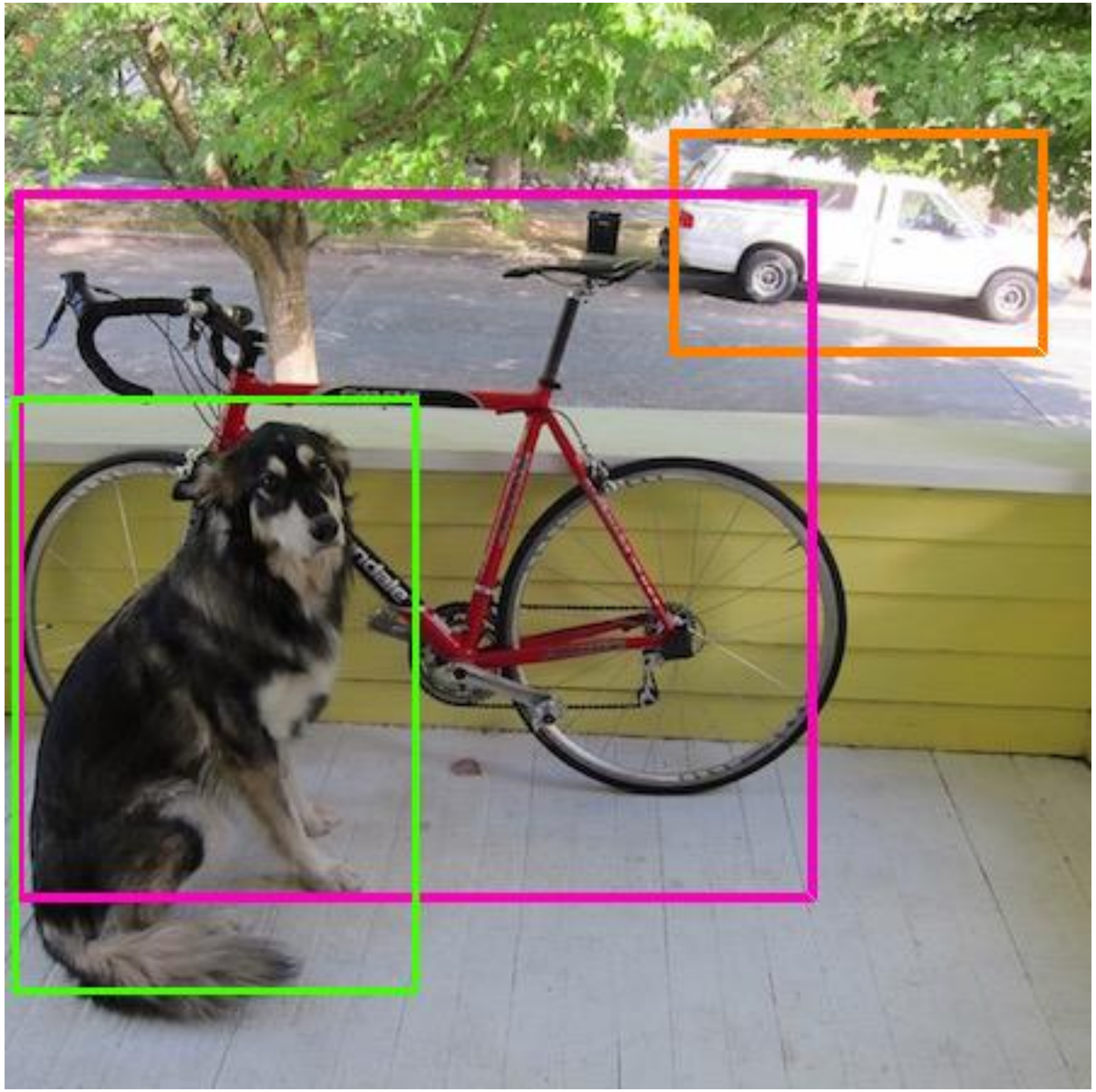
Also class probabilities



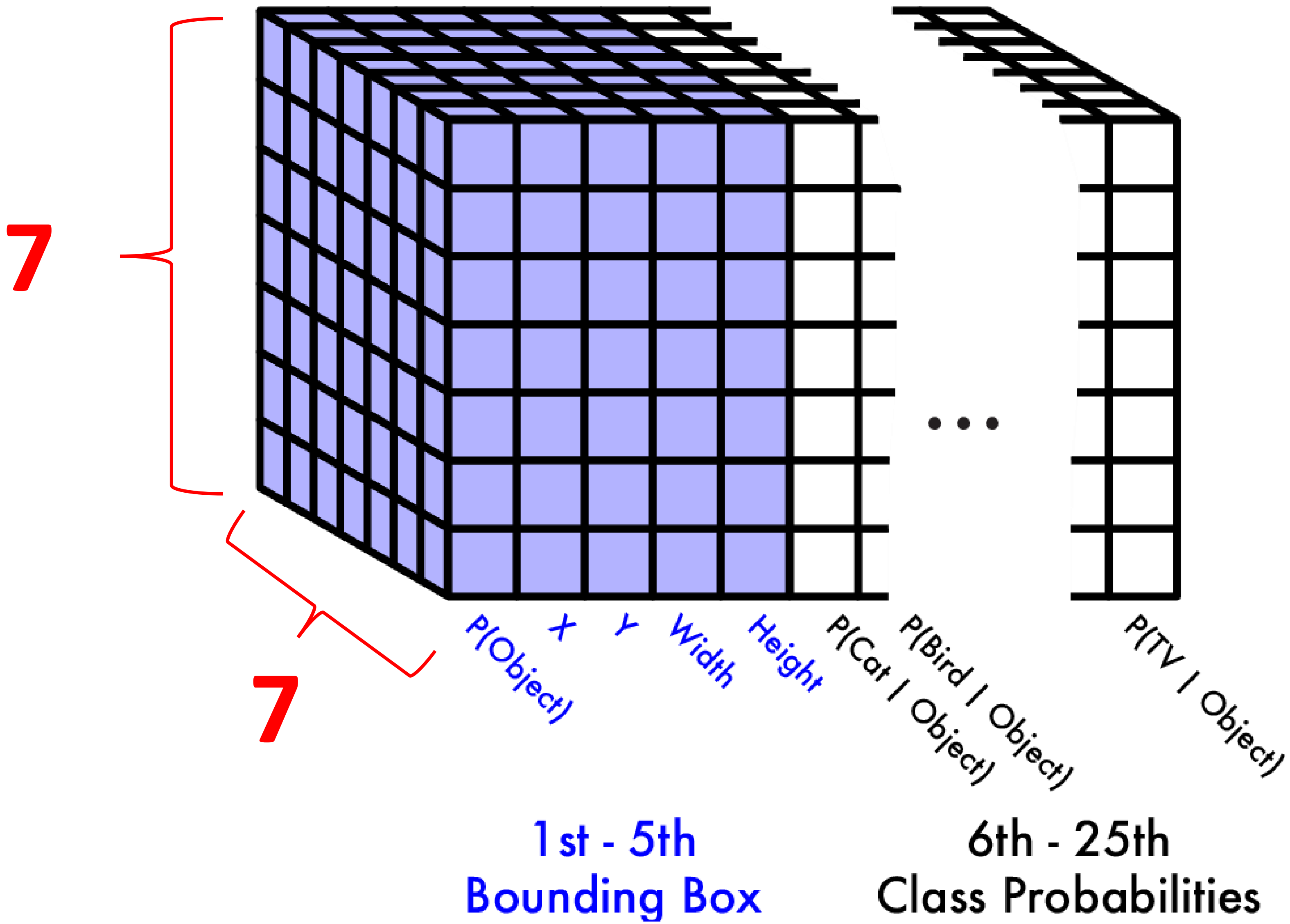
Also class probabilities



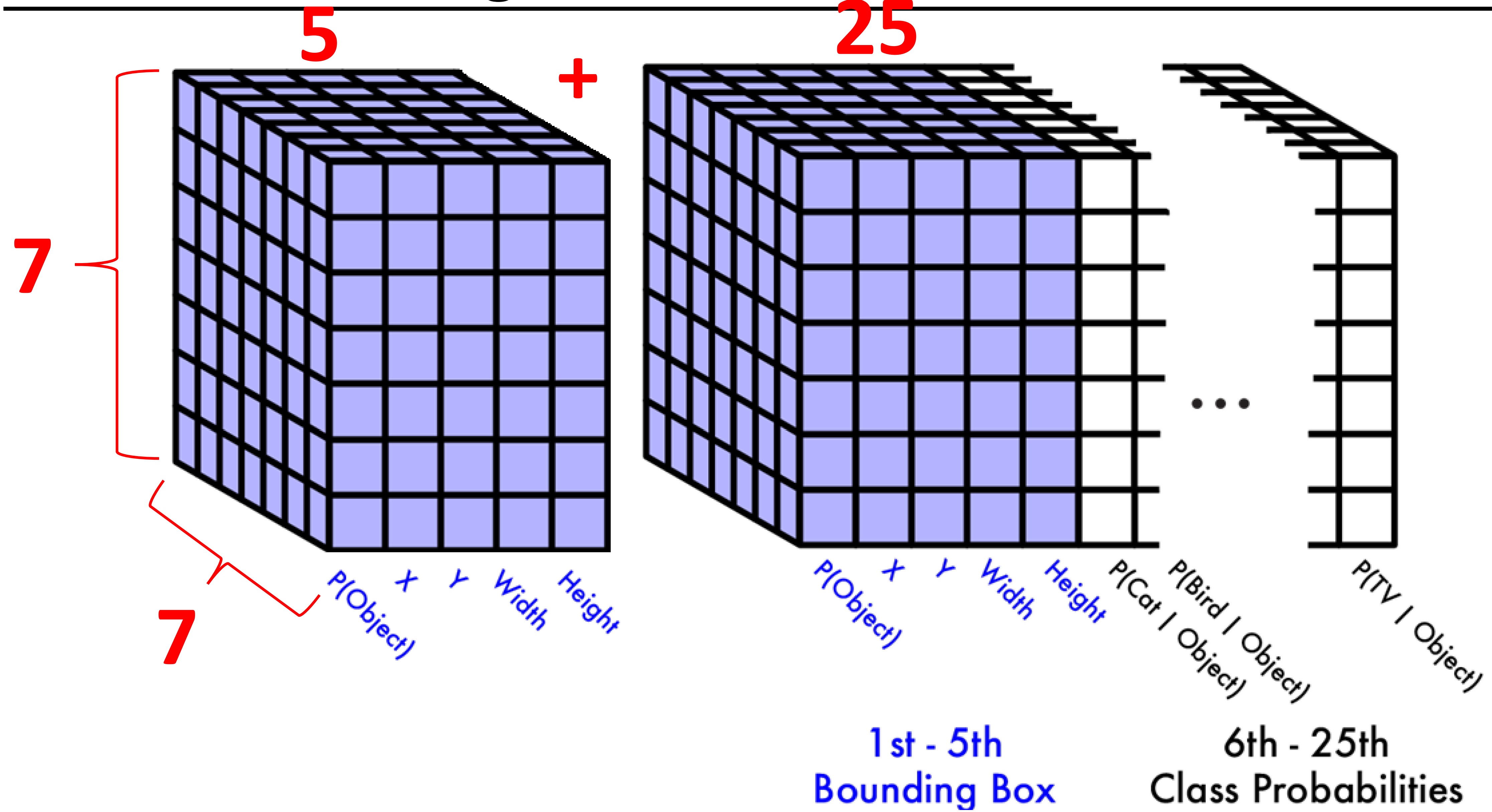
Threshold and non-max suppression



Tensor encoding detection (output)

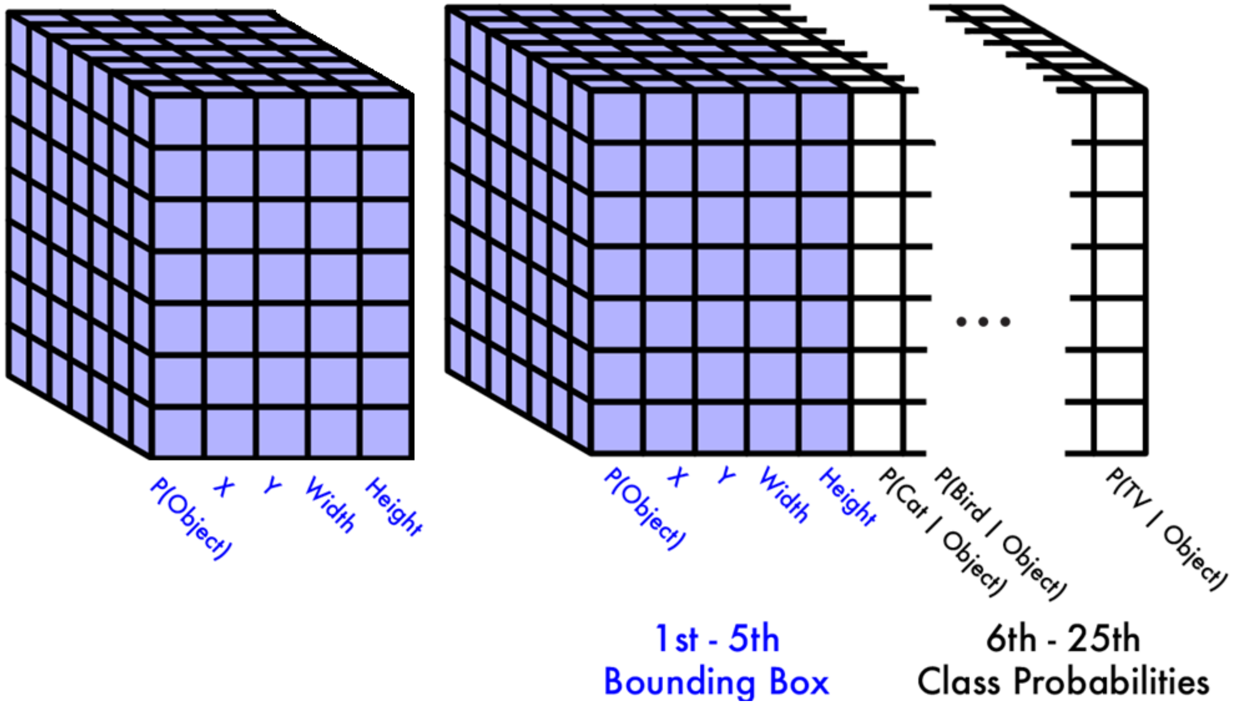


Tensor encoding detection



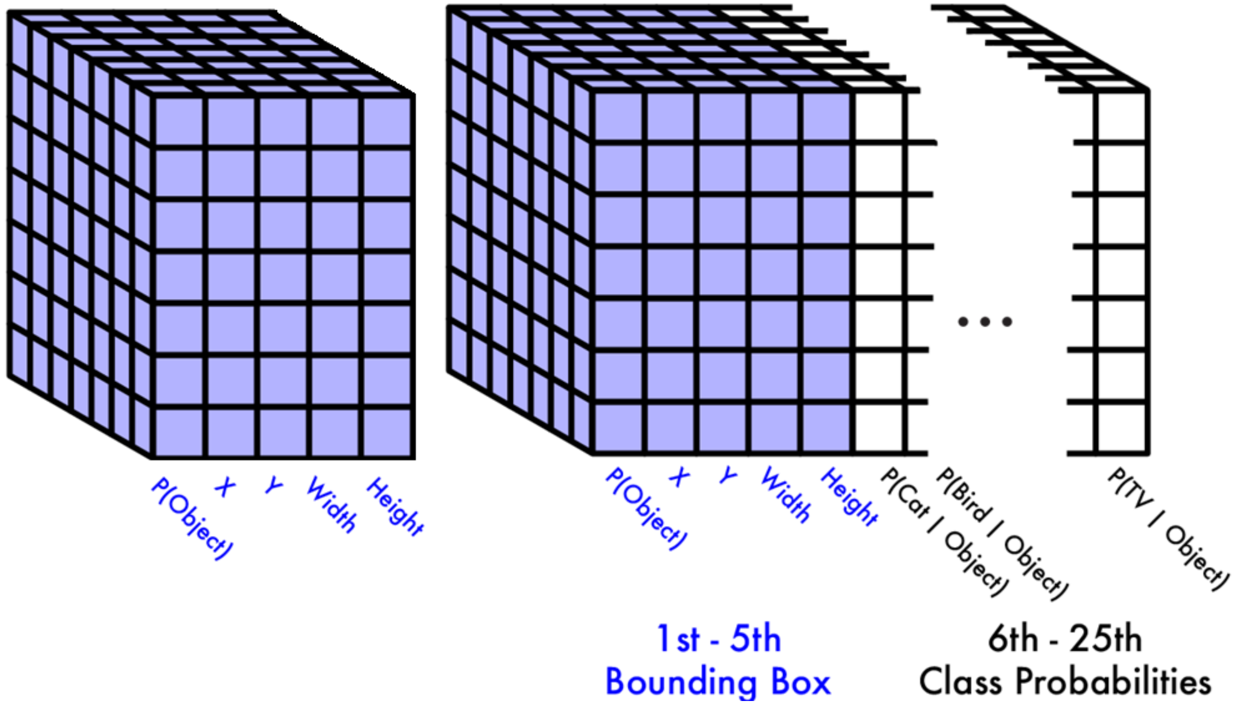
Tensor encoding detection

- P(Object) or objectness score gives you the probability of having an object in a box.



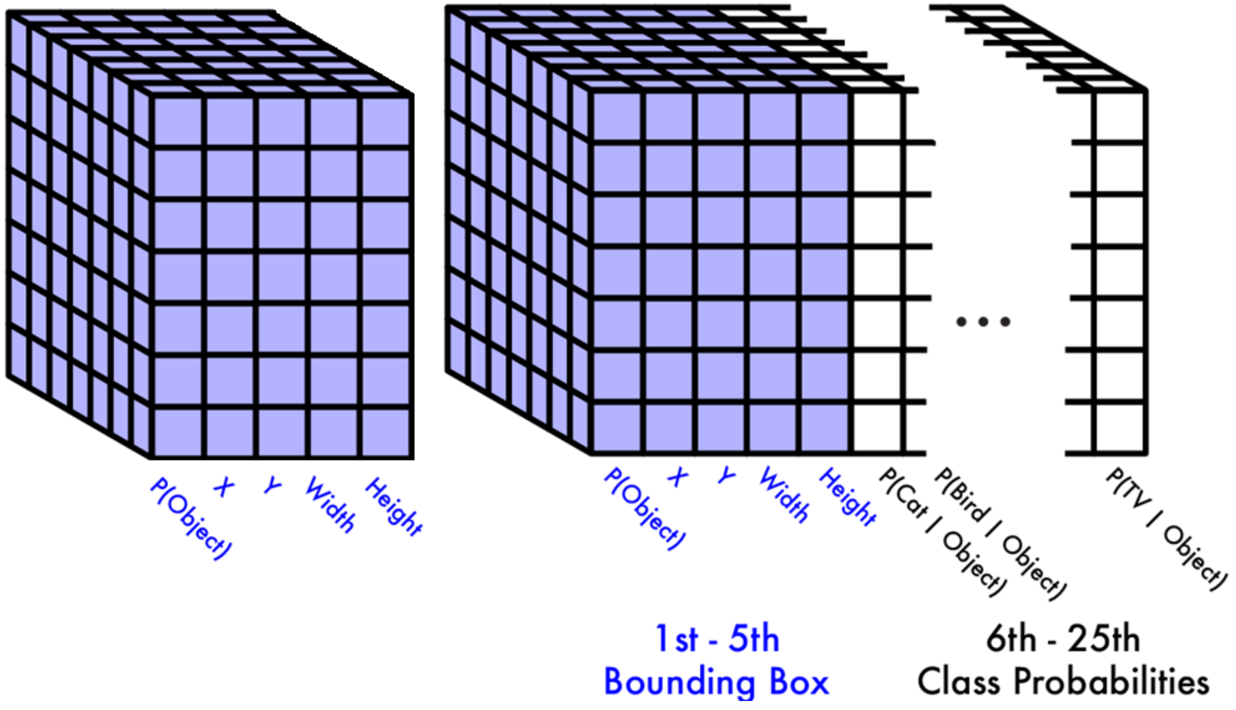
Tensor encoding detection

- P(Object) or objectness score gives you the probability of having an object in a box.
- X, Y, Width, Height are the sizes of each predicted box



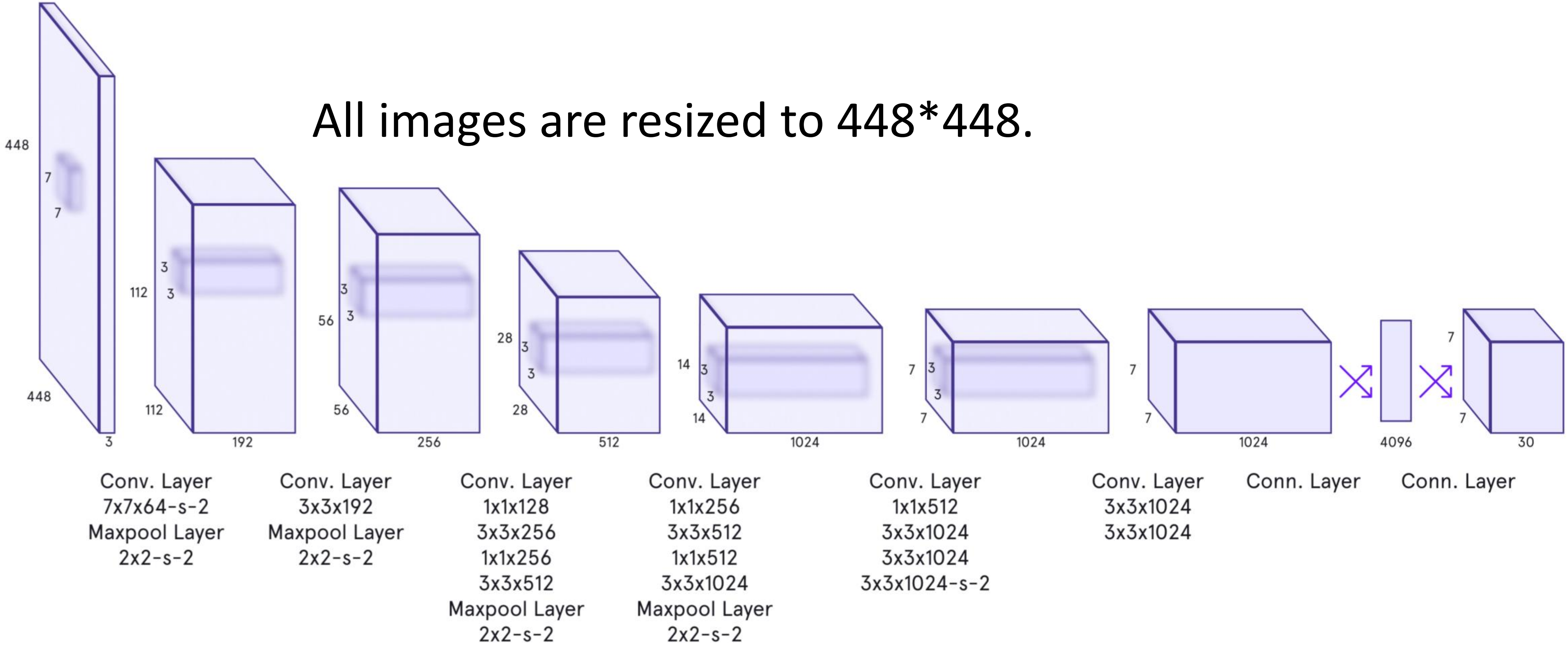
Tensor encoding detection

- P(Object) or objectness score gives you the probability of having an object in a box.
- X, Y, Width, Height are the sizes of each predicted box
- P(class) is the probability of the object belonging to each class.



Network architecture

All images are resized to 448*448.



The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1x1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224x224 input image) and then double the resolution for detection.

Training

- YOLO predicts multiple bounding boxes per grid cell.

Training

- YOLO predicts multiple bounding boxes per grid cell.
- At training time we only want one bounding box predictor to be responsible for each object.

Training

- YOLO predicts multiple bounding boxes per grid cell.
- At training time we only want one bounding box predictor to be responsible for each object.
- We assign one predictor to be “responsible” for predicting an object based on which prediction has the highest current IOU with the ground truth also containing the centroid of the box.

Loss function

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

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

Bounding box coordinates

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

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

Confidence

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

Classification

Loss function

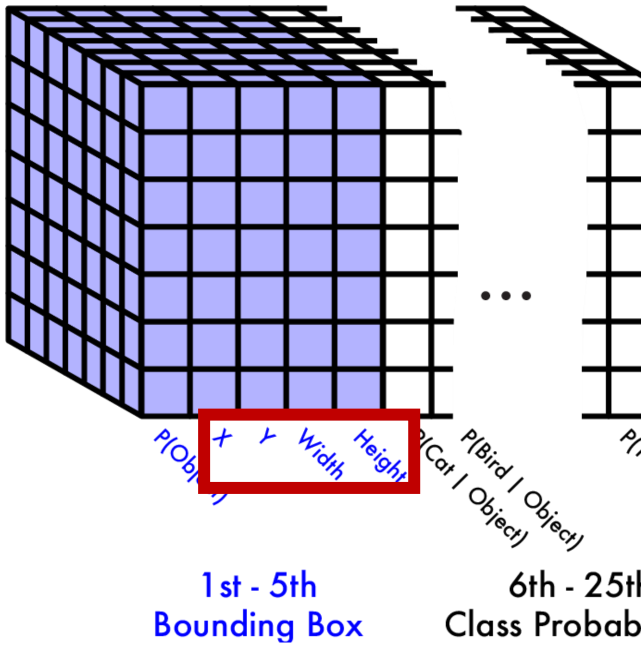
$$\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

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

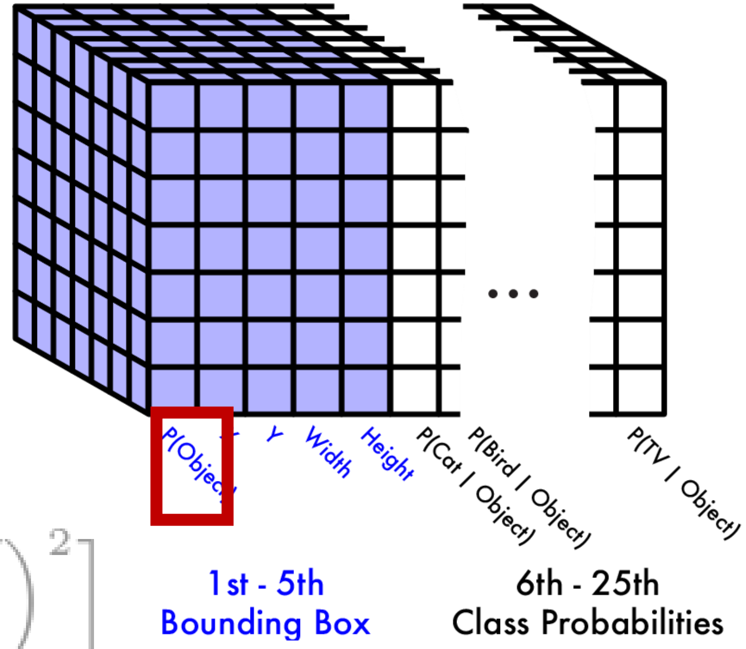
Bounding box coordinates

1: If bounding box j in cell i is responsible for predicting the ground truth box $(\hat{x}_i, \hat{y}_i, \hat{w}_i, \hat{h}_i)$.

0: Otherwise



Loss function



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

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

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

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

Confidence

1: If bounding box *j* in cell has no object.
0: Otherwise

Loss function

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

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

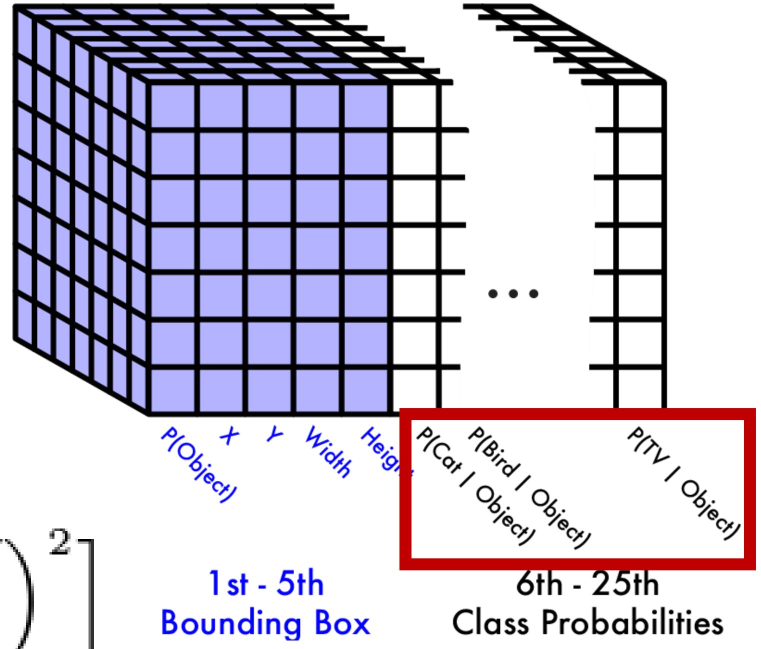
$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

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

Confidence

Confidence is defined as P(Object) $*$ IOU(GT).

Loss function



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

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

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

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

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

Classification

1: If object appears in cell i.

0: Otherwise