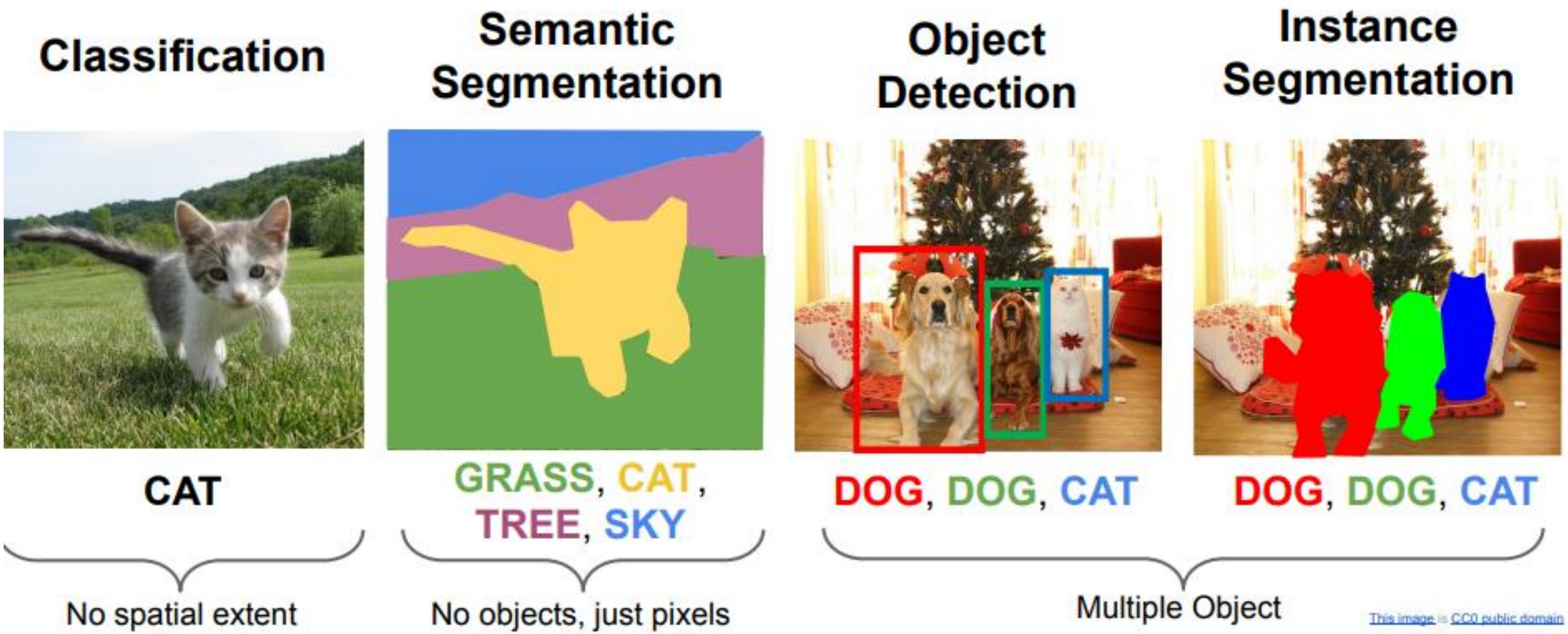
Segmentation and Object Detection using CNN

Outline

- How CNN can be used for semantic segmentation?
- Unet
- Object detection problem
- Sliding Window-based Object Detection
- How CNN can be used for object detection?

Computer Vision Tasks



identifying the class of an entire image

pixel-wise classification but doesn’t distinguish between different instances of the same object

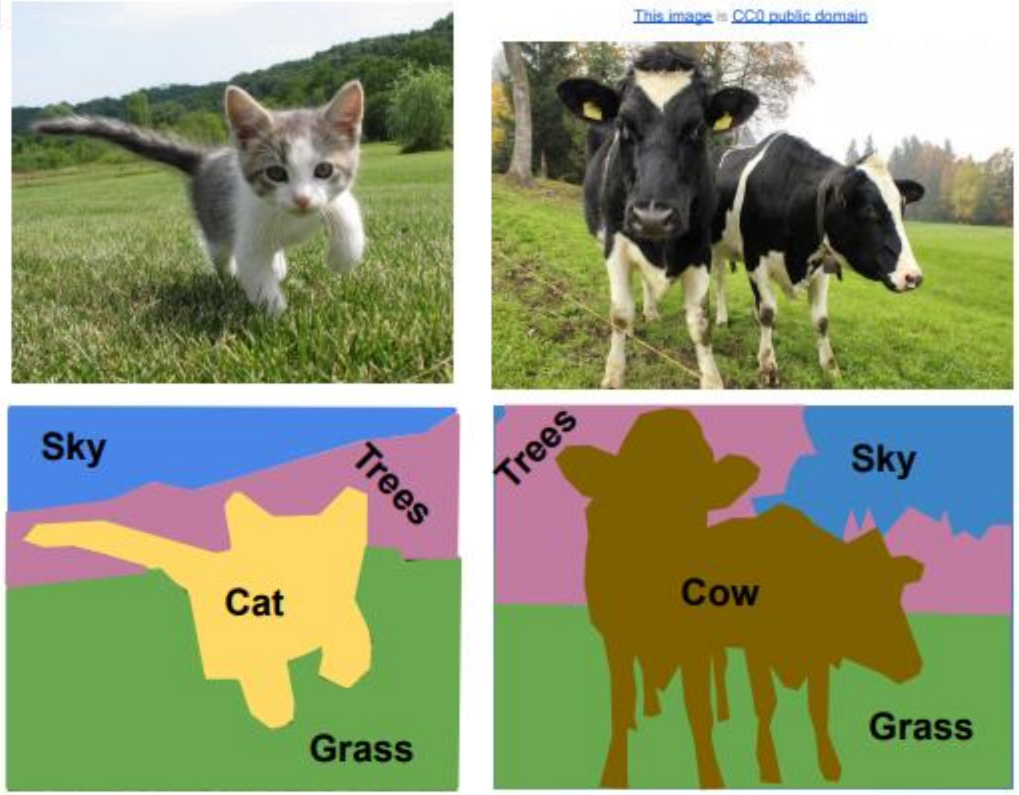
distinguishing objects of the same category

Task	Output	Main Focus	Example
Classification	Single class label for the whole image	Identifying what the image contains	Classifying an image as "cat" or "dog"
Localization	Bounding box coordinates (x, y, width, height)	Identifying the location of an object	Finding the location of a cat in an image
Object Detection	Bounding boxes + class labels + confidence score	Finding and classifying objects in an image	Detecting multiple cars and people in an image
Semantic Segmentation	Pixel-wise label of the entire image	Classifying every pixel into a category	Labeling every pixel in an image as "dog" or "grass"
Instance Segmentation	Pixel-wise mask for each object instance	Classifying each pixel and distinguishing instances of the same object	Differentiating between two dogs in an image with separate masks

Semantic Segmentation

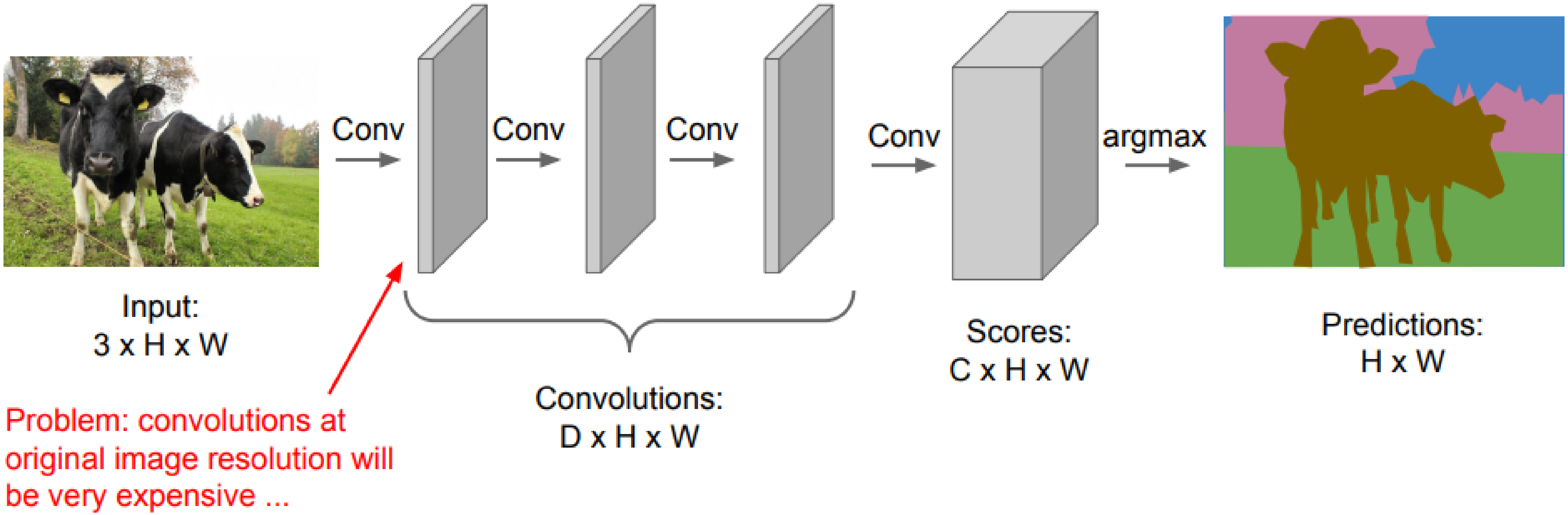
Label each pixel in the image with a category label

Don't differentiate instances, only care about pixels



Semantic Segmentation Idea: Fully Convolutional

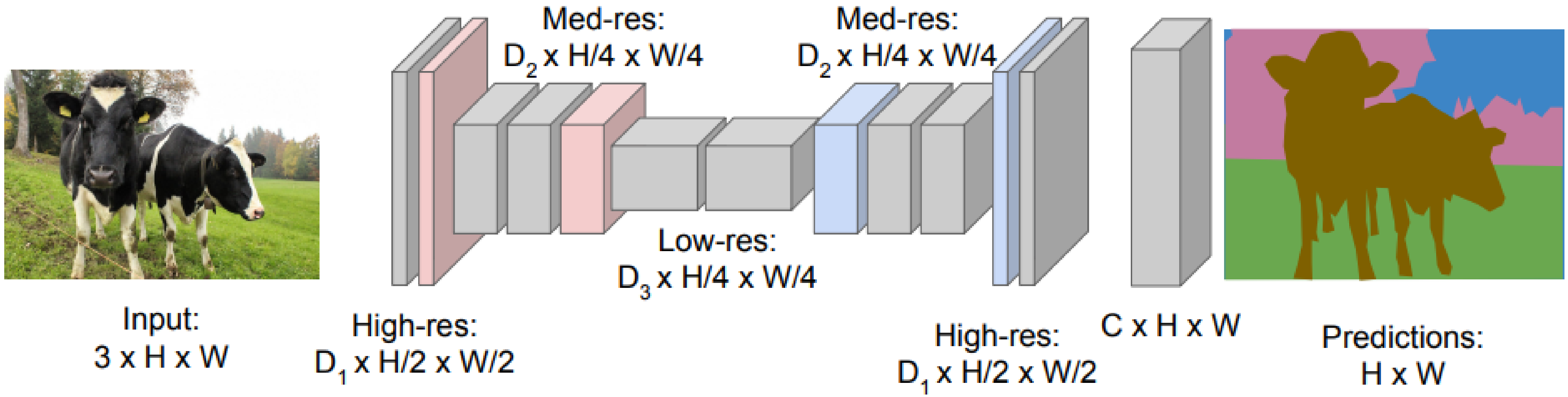
Design a network with only convolutional layers without downsampling operators to make predictions for pixels all at once!



classify each pixel in the image into a category (e.g., cow, grass, sky).

Semantic Segmentation Idea: Fully Convolutional

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

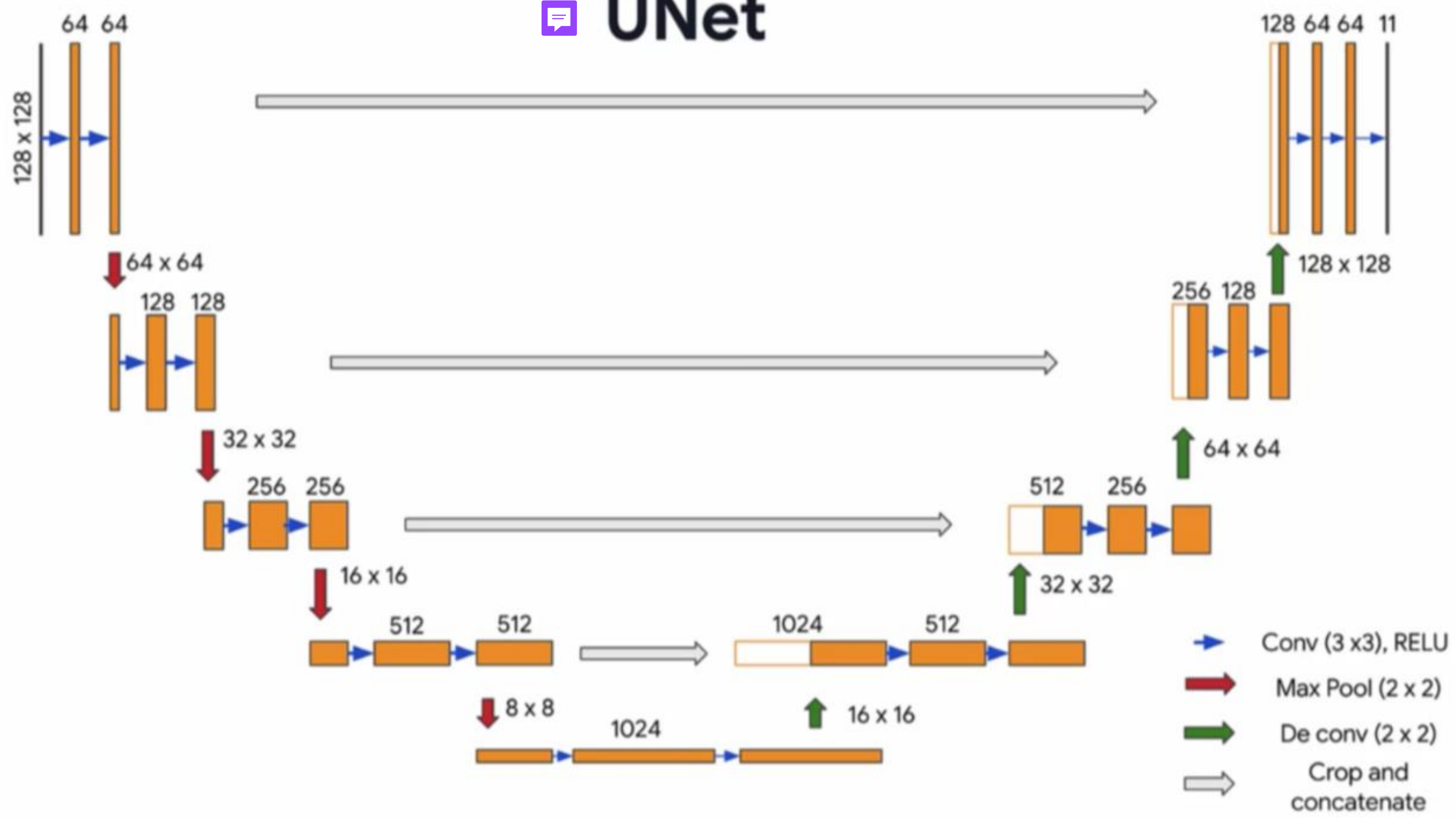


Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

Transposed Convolution

Downsampling reduces the spatial dimensions (height and width) of the input while retaining important features. Upsampling increases the spatial dimensions of feature maps, restoring them to the original input size or higher.

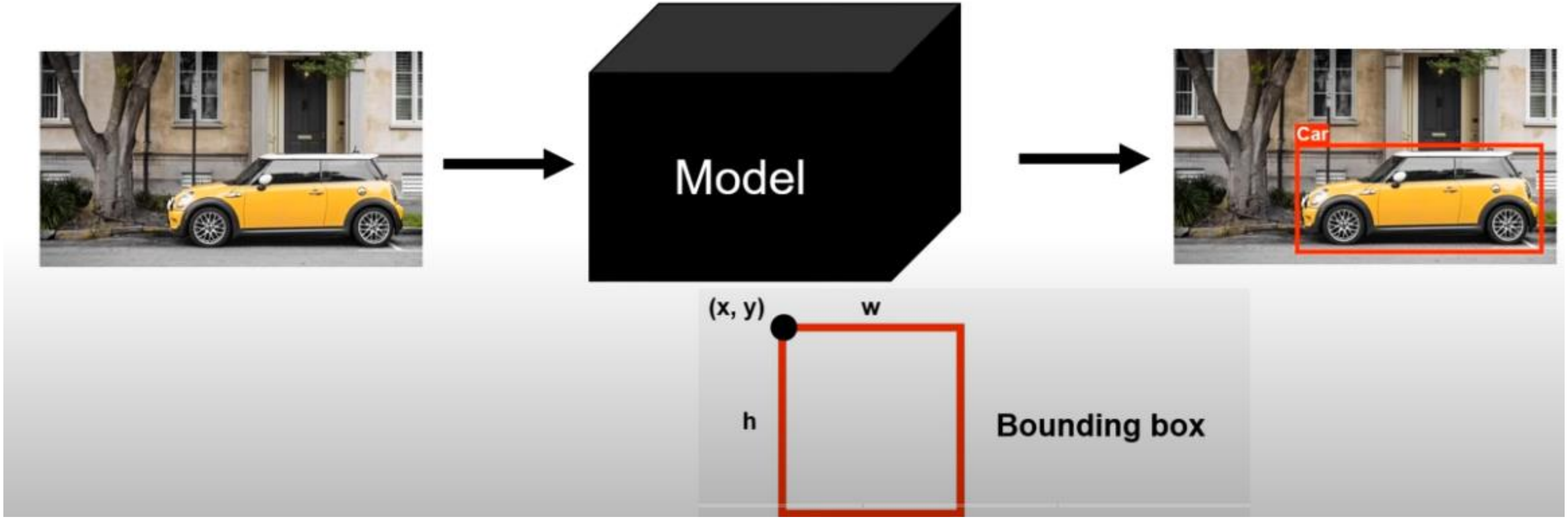
UNet



- encoder extracts hierarchical features by applying convolutional layers followed by max pooling, progressively reducing the spatial resolution while increasing the feature depth.
- bottleneck, the network captures the most abstract features.
- decoder then upsamples the feature maps progressively restoring the spatial resolution.
- **skip connections**, enabling the transfer of high-resolution details lost during downsampling.

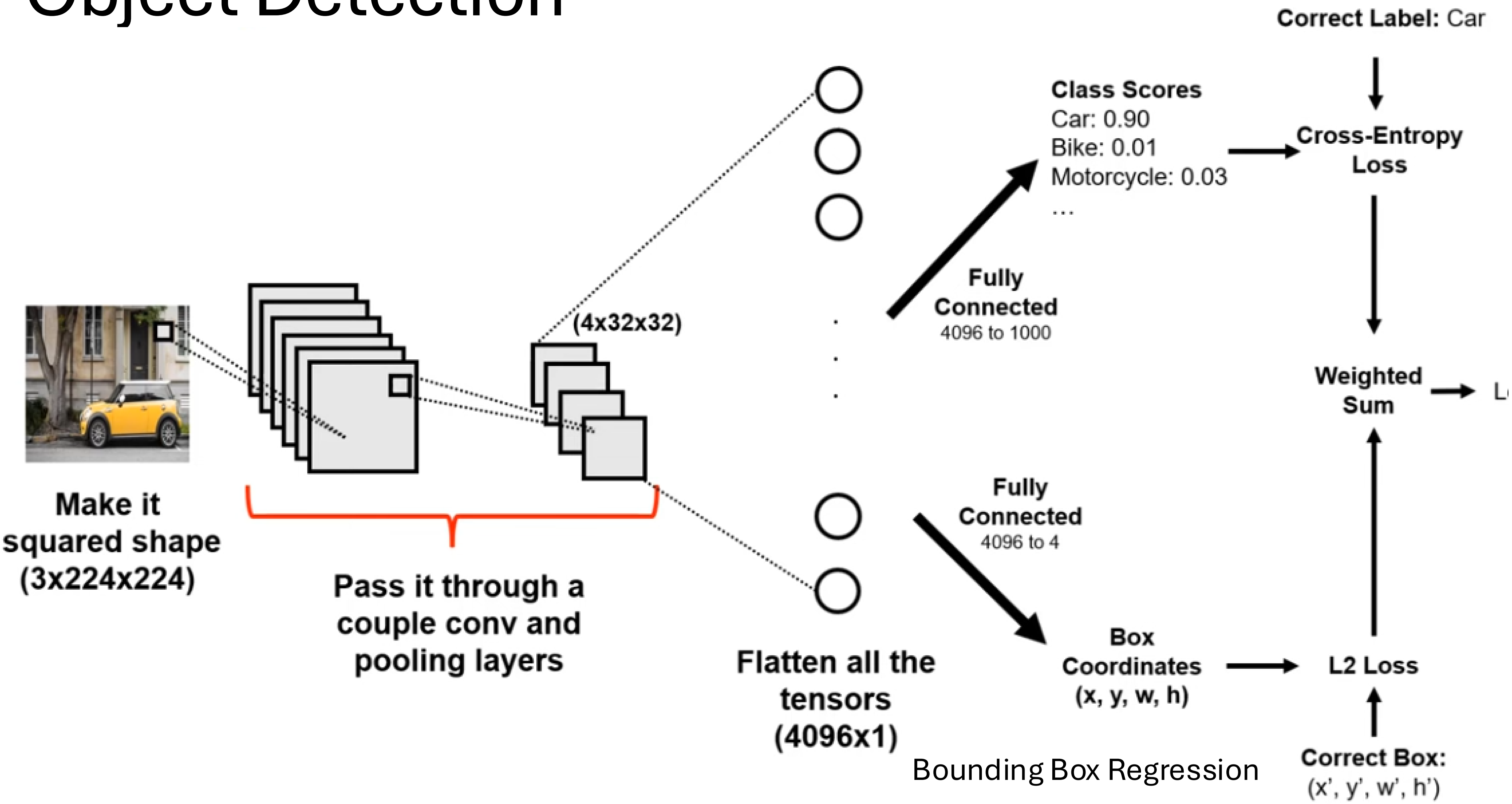
Object Detection

Object Detection



- **Classification**: Identifying what object(s) are present (e.g., cat, car, person).
- **Localization**: Determining where in the image the object(s) are located.
- **Multiple Objects**: Handling images with multiple objects of different classes.

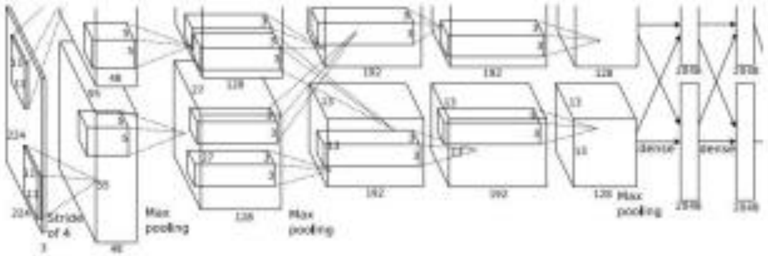
Object Detection



Bounding Box Regression

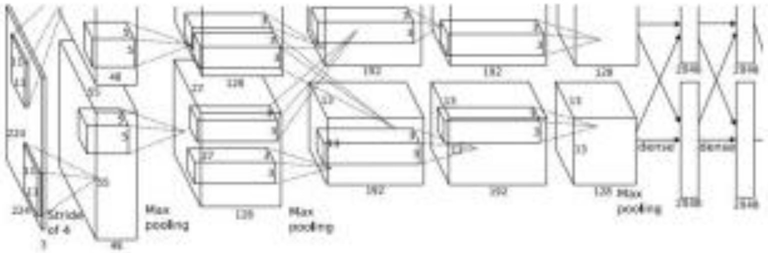
Object Detection: Multiple Objects

Each image needs a different number of outputs!

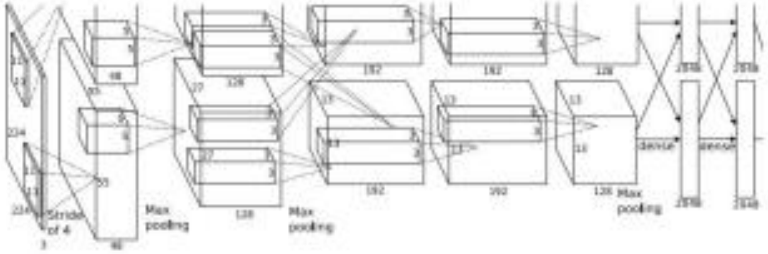
CAT: (x, y, w, h)

4 numbers

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

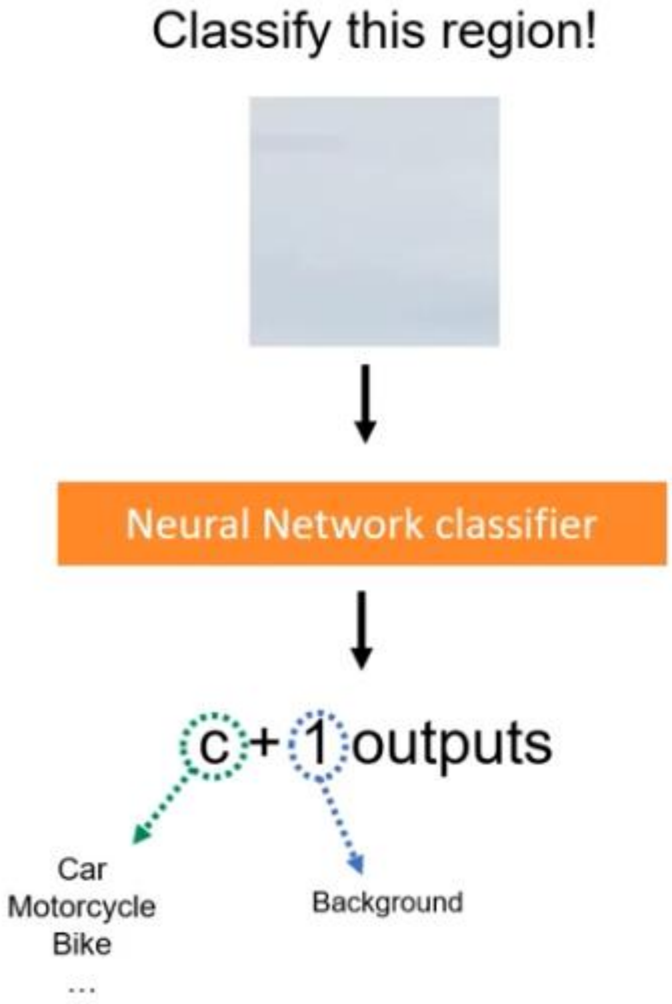
12 numbers

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

Many numbers!

Earliest Approach



Sliding Window-based Object Detection

Earliest Approach

Classify this region!

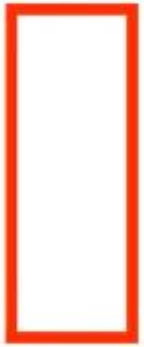
↓

Neural Network classifier

↓

Mountain

Earliest Approach



Possible Positions:

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

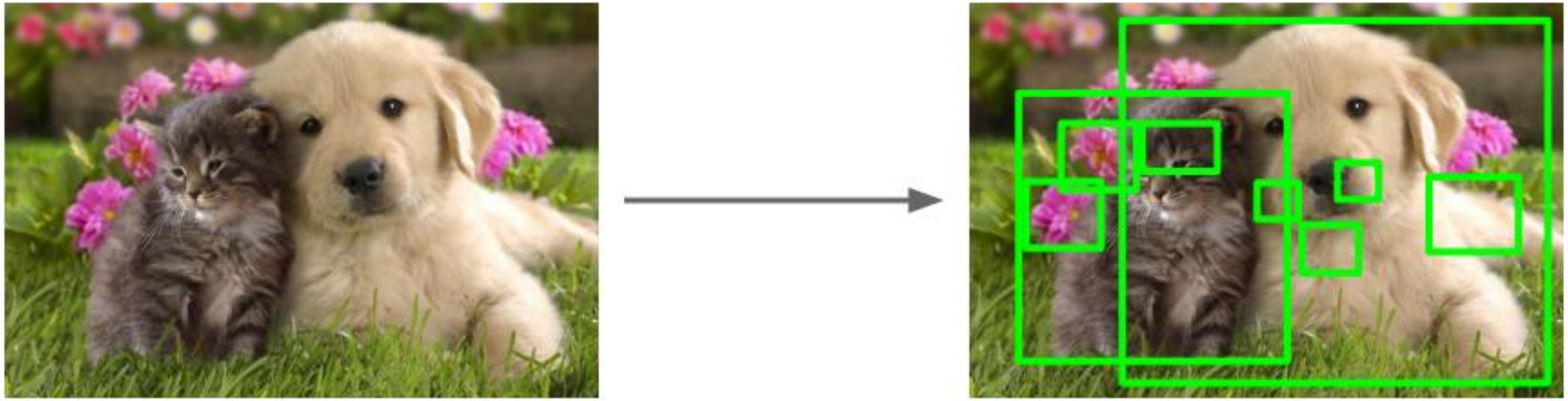
High Computational Cost, Poor Localization for Irregular Objects, Inefficiency for Sparse Scenes, Fixed Window Size, Lack of Context Awareness

Limitations of Traditional Methods

- **High Computational Cost**: Exhaustively evaluates all regions and scales, making it slow and impractical for real-time applications.
- **Fixed Window Size Limitations**: Struggles with objects of varying sizes, shapes, and aspect ratios, requiring multiple window sizes.
- **Redundant Computations**: Overlapping windows lead to wasted calculations, especially in sparse scenes with minimal objects.
- **Lack of Context Awareness**: Treats each region independently, failing to utilize global context for better object detection.

Region Proposals: Selective Search

- Find “blobby” image regions that are likely to contain objects
- Relatively fast to run; e.g. Selective Search gives 2000 region proposals in a few seconds on CPU



lexe et al, “Measuring the objectness of image windows”, TPAMI 2012
ijlings et al, “Selective Search for Object Recognition”, IJCV 2013
heng et al, “BING: Binarized normed gradients for objectness estimation at 300fps”, CVPR 2014
itnick and Dollar, “Edge boxes: Locating object proposals from edges”, ECCV 2014

- **Image Segmentation:**
- **Region Merging:** Small neighboring regions (superpixels) are merged iteratively based on similarity measures
  Color Texture : Size : Shape
- **Regions with similar shapes are combined.**
- **Hierarchical Grouping:** The algorithm builds a hierarchy of regions by merging smaller regions
- **Region Proposal Selection:**

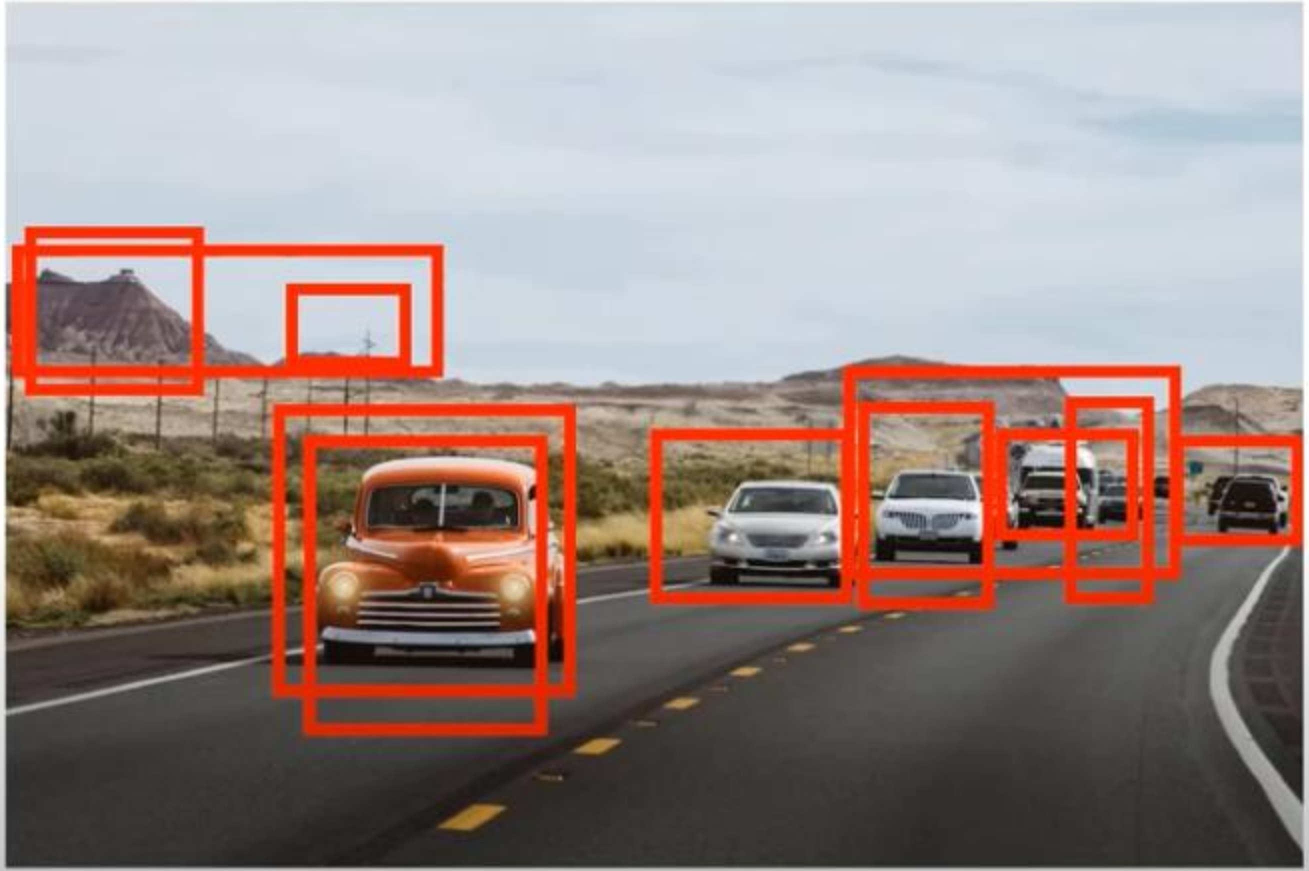
R-CNN

Region-based
Convolutional
Neural Networks

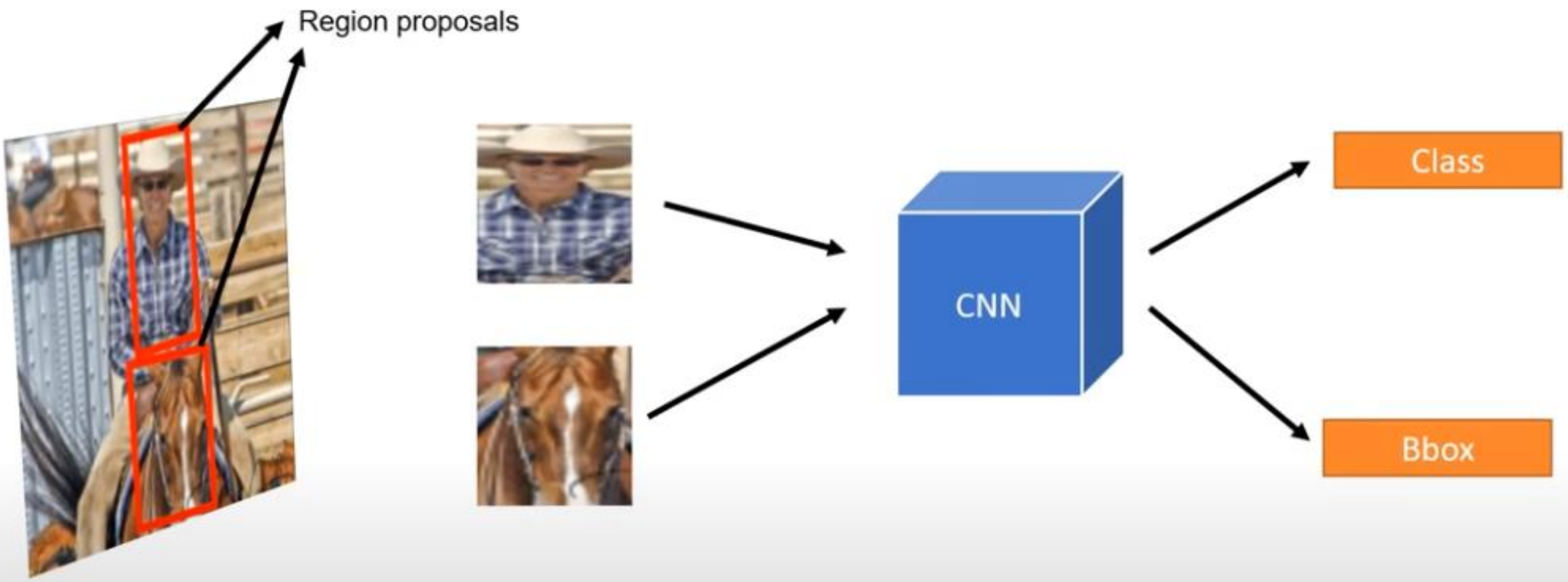
It uses an external algorithm to propose some regions.

Selective Search

It gives us like 2000 region proposals within one or two seconds



R-CNN



Steps in R-CNN

- **Steps in RCNN:**
- 1. **Region Proposal Generation**:
- The first step is to identify regions in the image that might contain objects.
- RCNN uses **Selective Search** to generate approximately **2000 region proposals** (candidate bounding boxes) for each input image.
- 2. **Warping Region Proposals**:
- Each region proposal is resized (warped) to a fixed size (e.g., **224x224 pixels**) to ensure compatibility with the input size of the CNN.
- This is necessary because CNNs require fixed-sized inputs, and the region proposals vary in shape and size.
- 3. **Feature Extraction**:
- Each resized region proposal is passed through a **Convolutional Neural Network (CNN)** (e.g., AlexNet, VGG) to extract features.
- The output of the CNN is a feature vector that represents the characteristics of the region.
- 4. **Classification**:
- The extracted feature vector is fed into a **linear SVM classifier** to predict the class of the object (e.g., car, bike, person, etc.) for each region proposal.
- Each SVM is trained separately for different classes.
- 5. **Bounding Box Regression**:
- To refine the region proposal and better fit the actual object, RCNN applies a **bounding box regression** model.

Bounding box regression is a key step in object detection models like R-CNN, Fast R-CNN, and Faster R-CNN. It is used to refine the coordinates of bounding boxes proposed by the model, ensuring they fit the objects as accurately as possible.
The Need for Bounding Box Regression:
Initial region proposals may not perfectly align with the objects (e.g., slightly shifted, too large, or too small).
Bounding box regression adjusts these proposals to better match the object's shape and position.

R-CNN

Region proposal: (p_x, p_y, p_h, p_w)

Bbox

Transform: (t_x, t_y, t_h, t_w)

Output: (b_x, b_y, b_h, b_w)

Translation:

$b_x = p_x + p_w t_w$
(Horizontal translation)

$b_y = p_y + p_h t_h$
(Vertical translation)

Log-space scale transform:

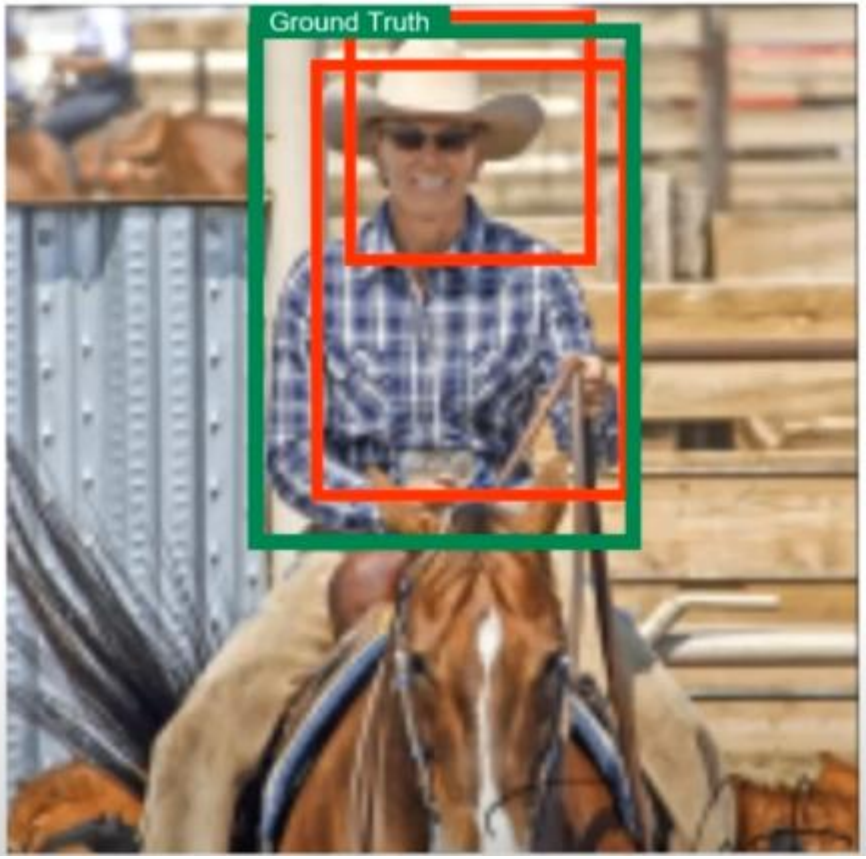
$b_w = p_w \exp(t_w)$
(Horizontal scale)

$b_h = p_h \exp(t_h)$
(Vertical scale)

Region proposal may not perfectly align with the object in the image.

both smaller and larger object scales are appropriately handled

Non-maximum suppression



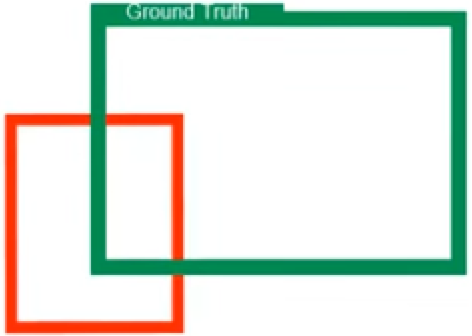
post-processing technique ensuring that the final output contains only the best possible bounding boxes for each detected object

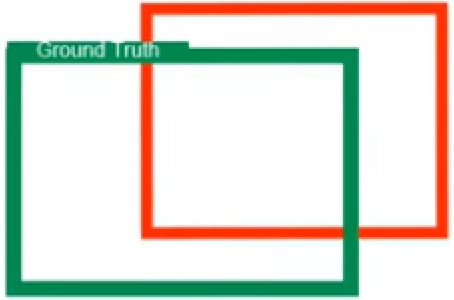
- **Bounding Box Scores**: how confident the model is about the presence of an object in each bounding box. Higher scores indicate higher confidence.
- **Sorting Boxes**: The bounding boxes are then sorted based on their confidence scores in descending order.
- **Overlap Calculation**: For each bounding box, the Intersection over Union (IoU) with other boxes is calculated.
- **Suppression Process**:
  - The algorithm compares each pair of overlapping boxes keeps only the box with the highest confidence score
  - The rest of the overlapping boxes are suppressed or discarded.

What is it? A technique to remove redundant bounding boxes and keep only the most confident one for each detected object.
Why is it needed? Models often predict multiple overlapping boxes for the same object; NMS refines these into a single, accurate box.
How is it done? Sort boxes by confidence score, keep the highest-scoring box, discard overlapping ones based on the IoU formula:
IoU=Area of OverlapArea of UnionIoU= Area of UnionArea of Overlap
Parameters involved: Confidence score (used to rank boxes) and IoU threshold (determines overlap allowed).
Output: A final set of non-overlapping boxes representing detected objects.

Non-maximum suppression

Intersection over Union (IoU)





$$\frac{3}{20} = 0.15$$

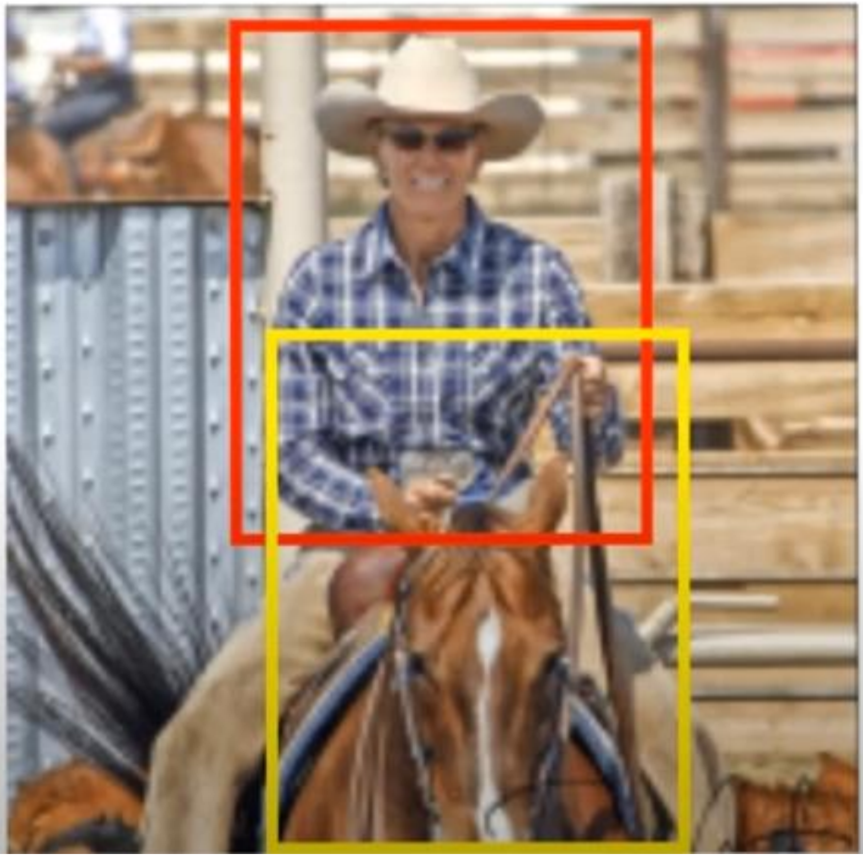
$$\frac{12}{25} = 0.48$$

IoU is a metric that measures the overlap between two bounding boxes. It is calculated as:

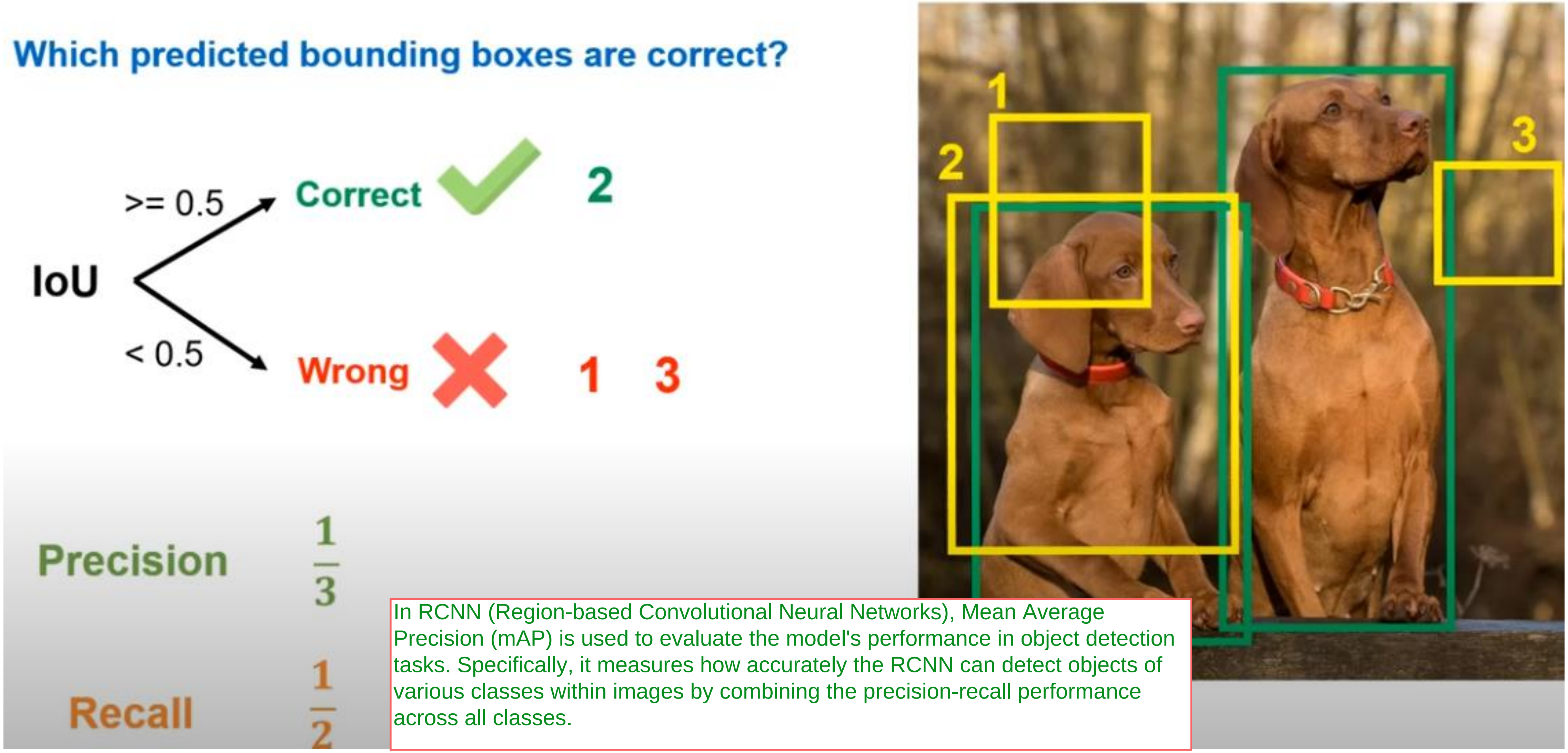
$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

IoU values range from 0 (no overlap) to 1 (perfect overlap).

Note: We use non-max suppression when the object is the same for all of the bounding boxes



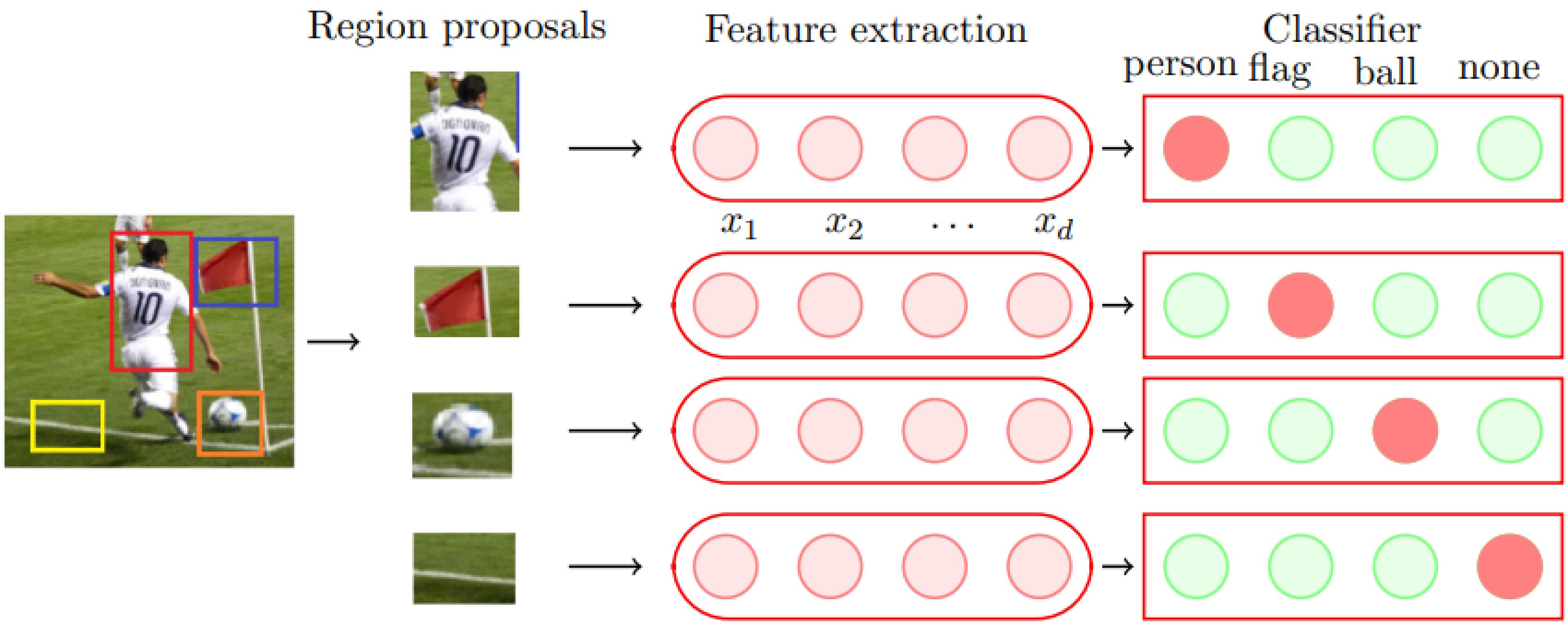
Mean Average Precision (mAP)



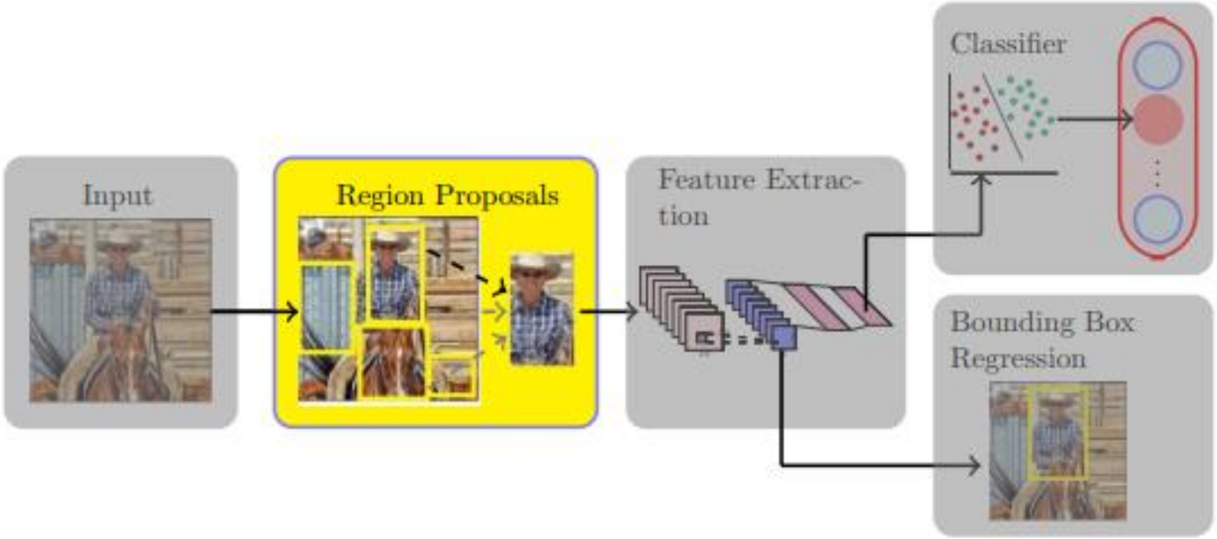
In RCNN (Region-based Convolutional Neural Networks), Mean Average Precision (mAP) is used to evaluate the model's performance in object detection tasks. Specifically, it measures how accurately the RCNN can detect objects of various classes within images by combining the precision-recall performance across all classes.

combines the precision-recall curve and provides a single value representing the overall accuracy of the model.

Pipeline for object detection

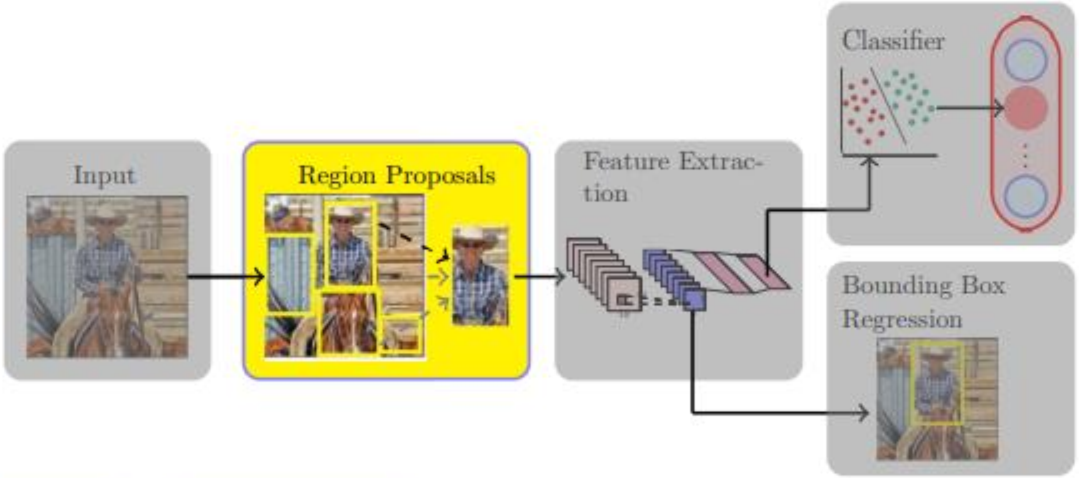


R-CNN



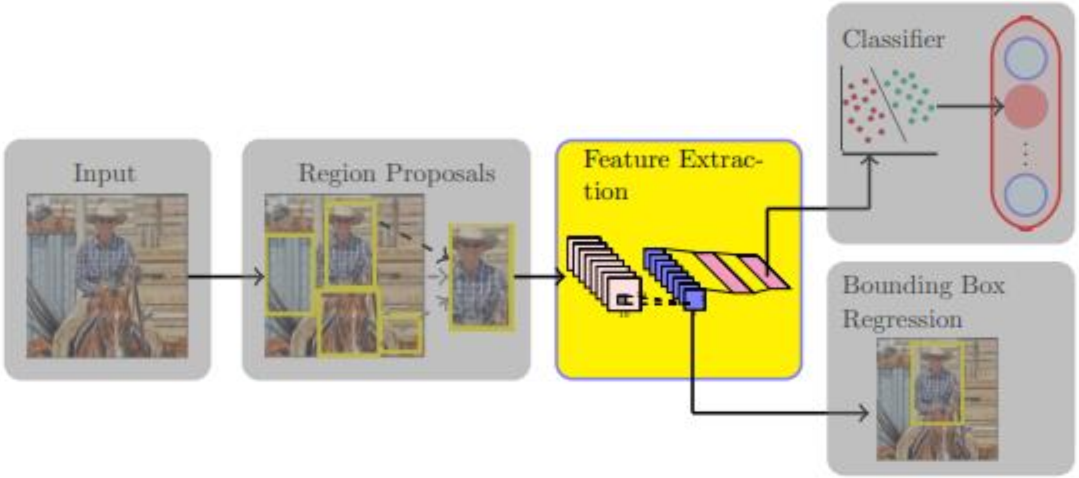
- **Selective Search** for region proposals
- Does hierarchical clustering at different scales
- For example the figures from left to right show clusters of increasing sizes
- Such a hierarchical clustering is important as we may find different objects at different scales

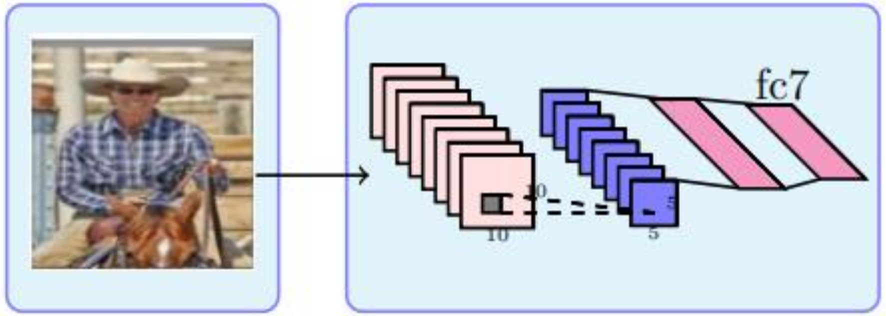
R-CNN



- Proposed regions are cropped to form mini images
- Each mini image is scaled to match the CNN's (feature extractor) input size

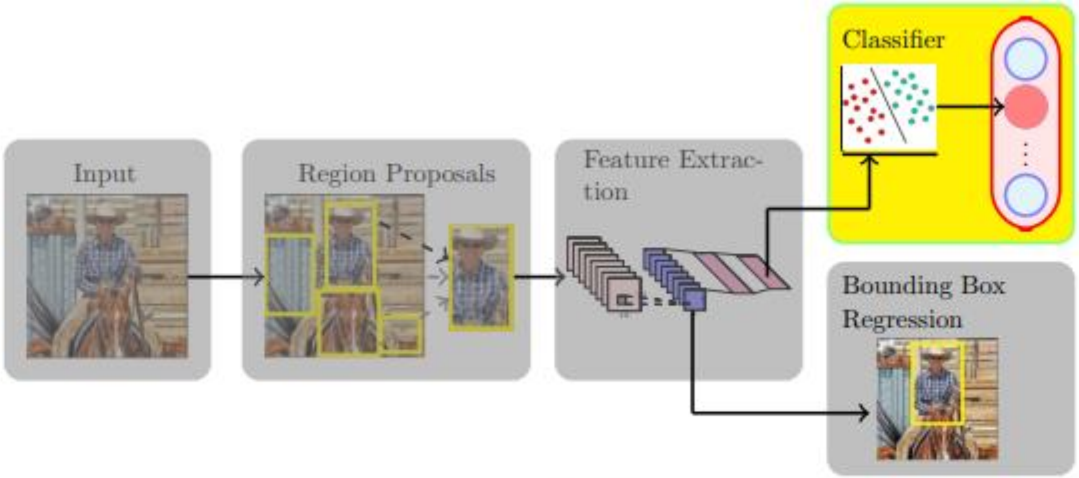
R-CNN

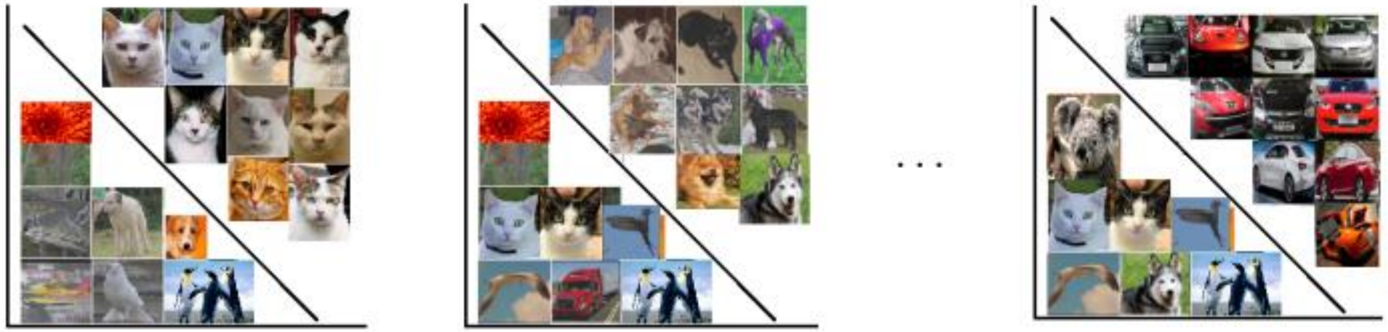




- For feature extraction any CNN trained for Image Classification can be used (AlexNet/ VGGNet etc.)
- Outputs from fc7 layer are taken as features
- CNN is fine tuned using ground truth (cropped) object images

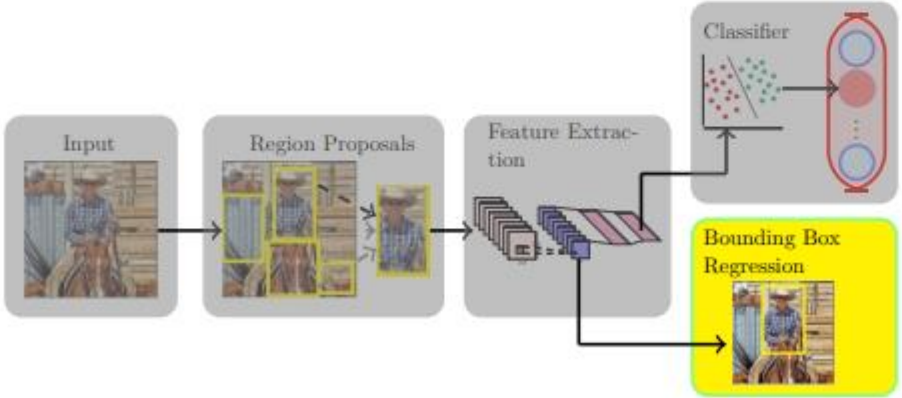
R-CNN





- Linear models (SVMs) are used for classification (1 model per class)

R-CNN



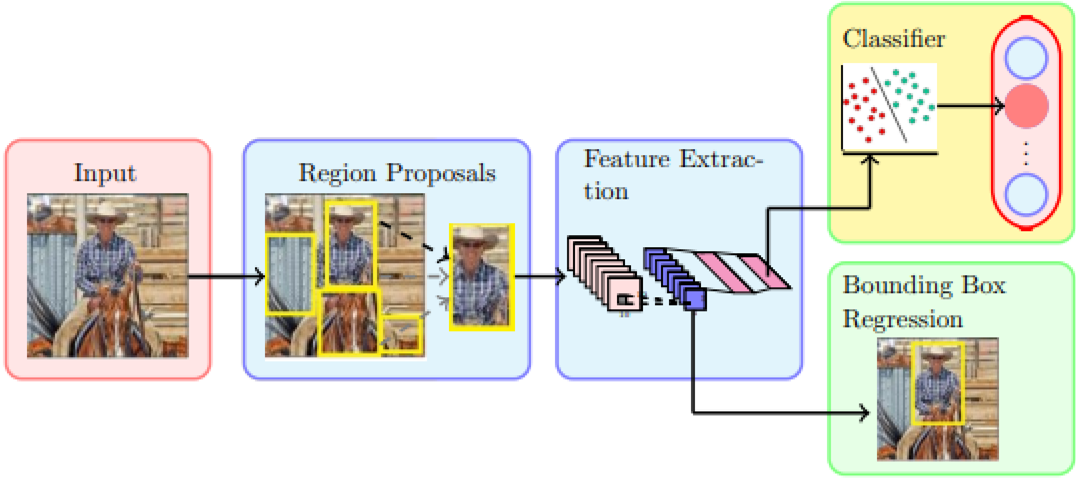
Proposed Box

True Box

z : features from pool5 layer of the network

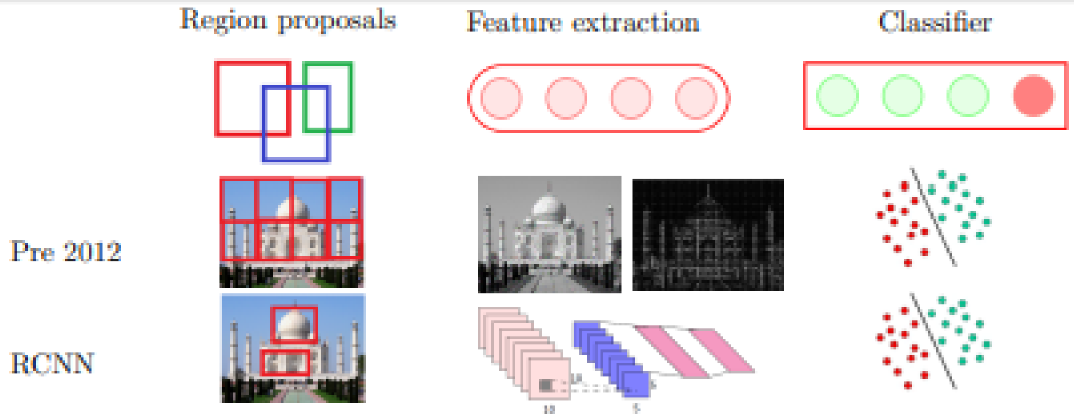
- The proposed regions may not be perfect
- We want to learn four regression models which will learn to predict x^*, y^*, w^*, h^*
- We will see their respective objective functions

R-CNN



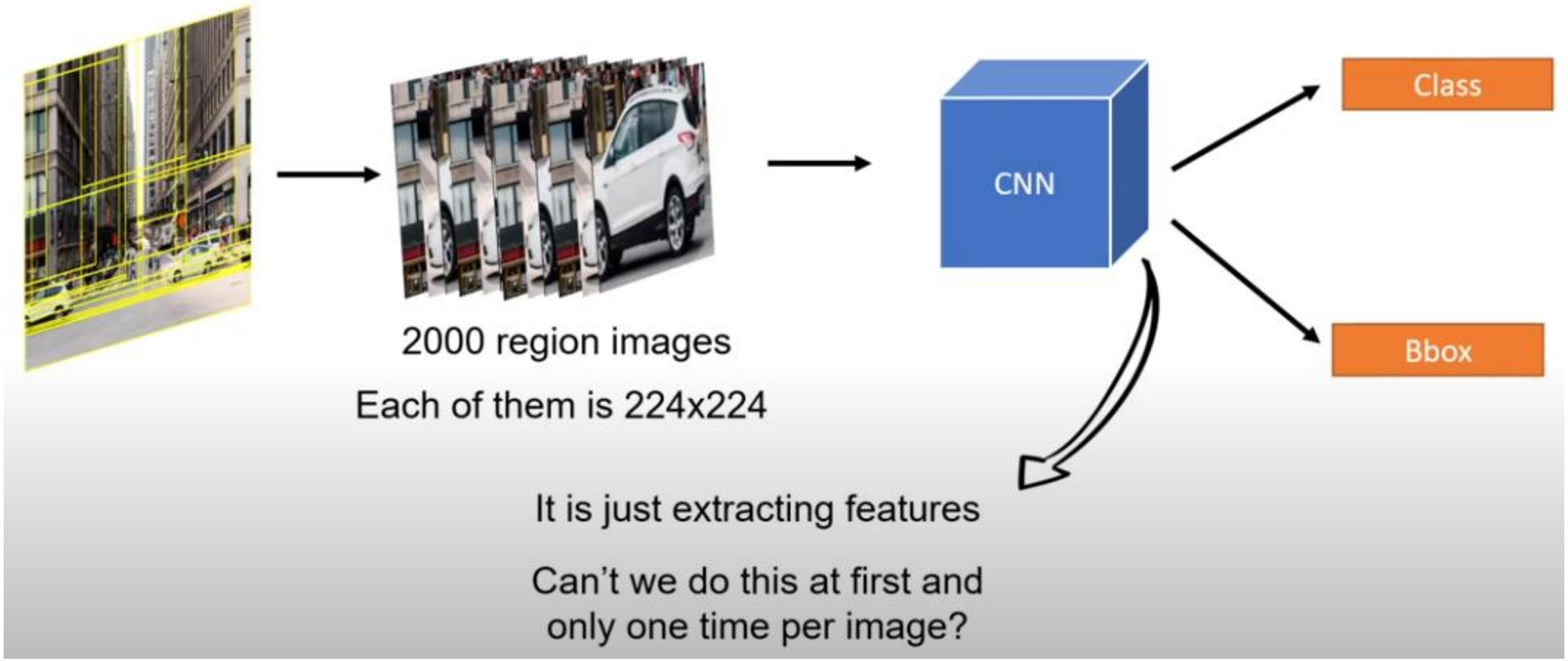
- What is the computational cost for processing one image at test time?
- Inference Time = Proposal Time + # Proposals × Convolution Time + # Proposals × classification + # Proposals × regression

Comparison



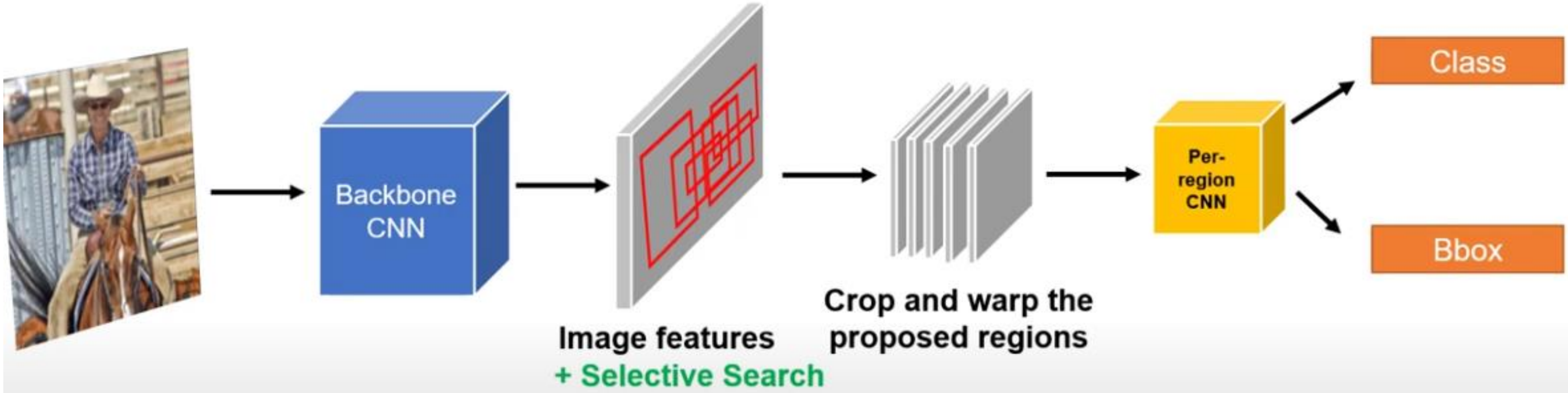
- **Region Proposals:** Selective Search
- **Feature Extraction:** CNNs
- **Classifier:** Linear

R-CNN

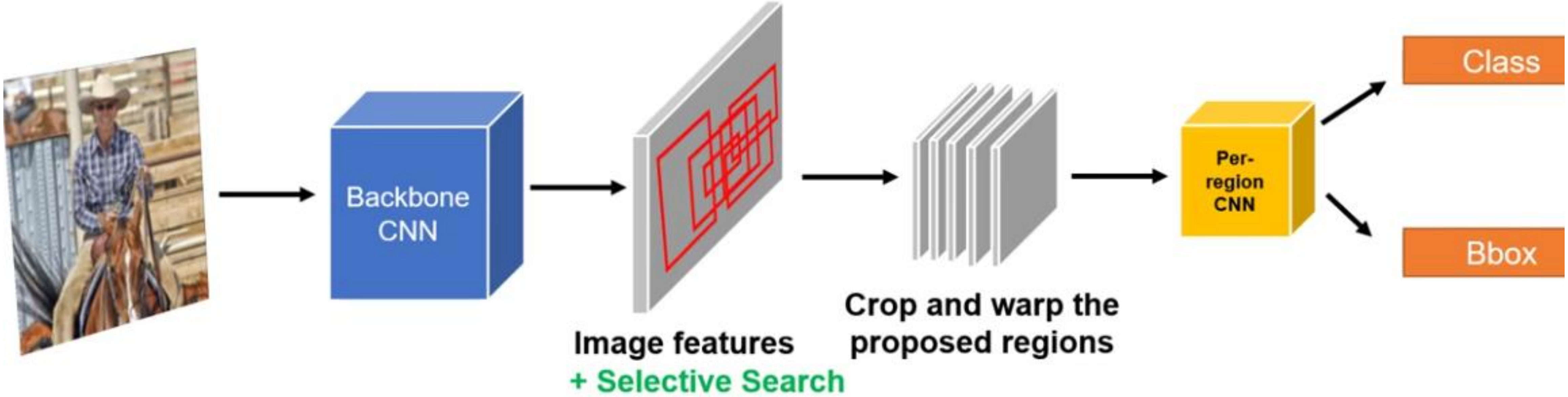


Once region proposals are generated, each region (bounding box) is cropped from the image and resized to a fixed size

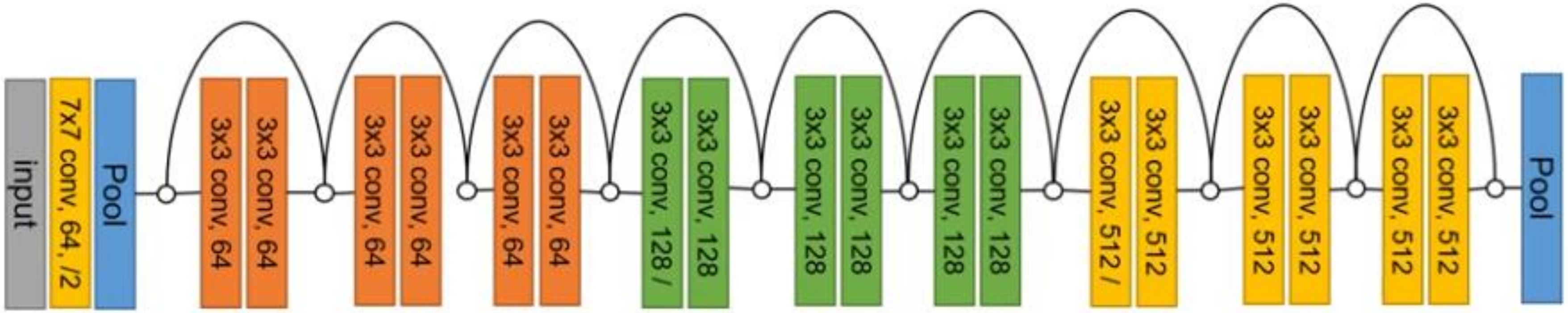
Fast R-CNN


Backbone CNN
Per-region CNN
Class
Bbox
Image features
+ Selective Search
Crop and warp the proposed regions

Fast R-CNN

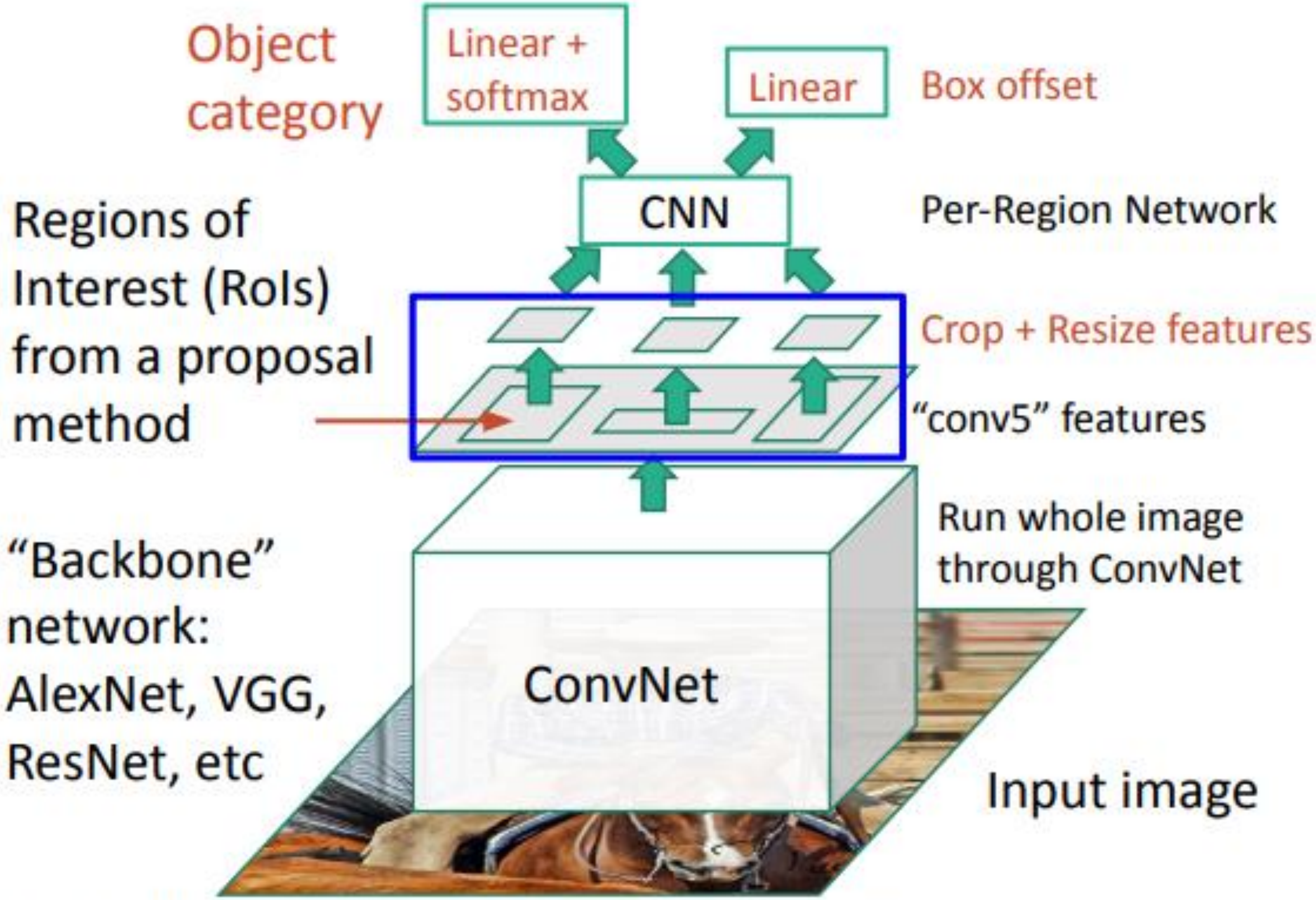


Backbone CNN
Image features
+ Selective Search
Crop and warp the proposed regions
Per-region CNN
Class
Bbox

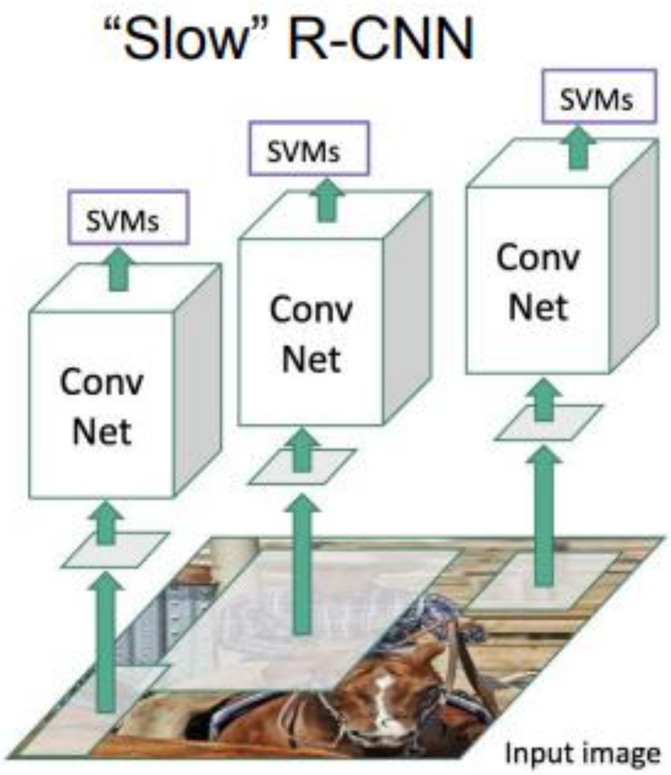


input
7x7 conv, 64, /2
Pool
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 128 /
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 512 /
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool

Fast R-CNN

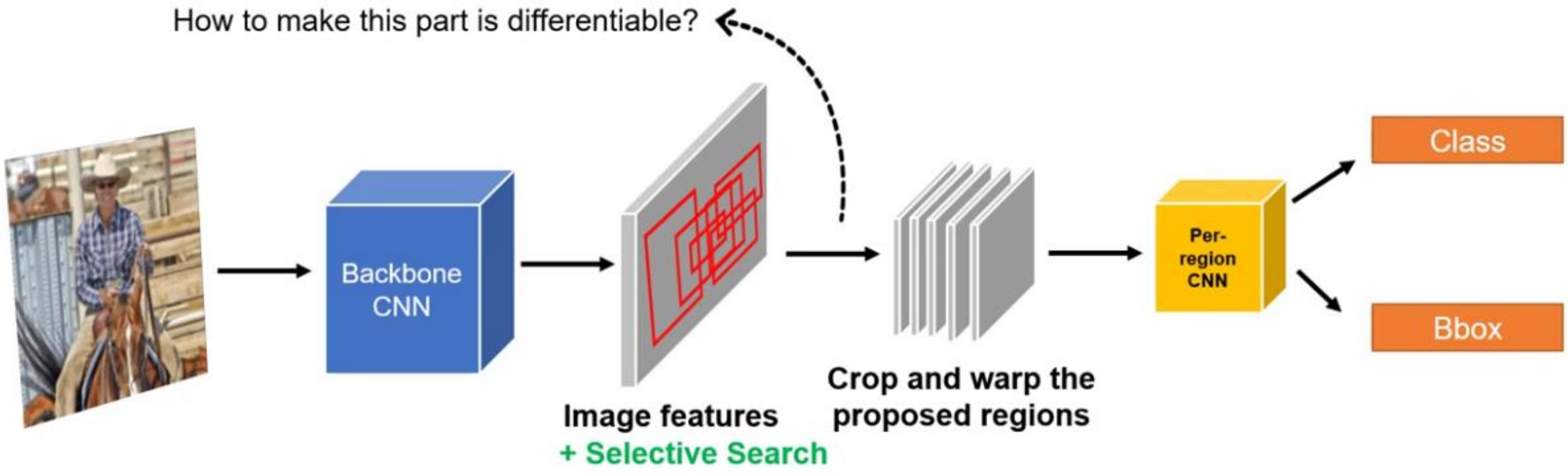


Object category
Linear + softmax
Linear
Box offset
CNN
Per-Region Network
Regions of Interest (RoIs) from a proposal method
Crop + Resize features
"conv5" features
Run whole image through ConvNet
"Backbone" network: AlexNet, VGG, ResNet, etc
ConvNet
Input image

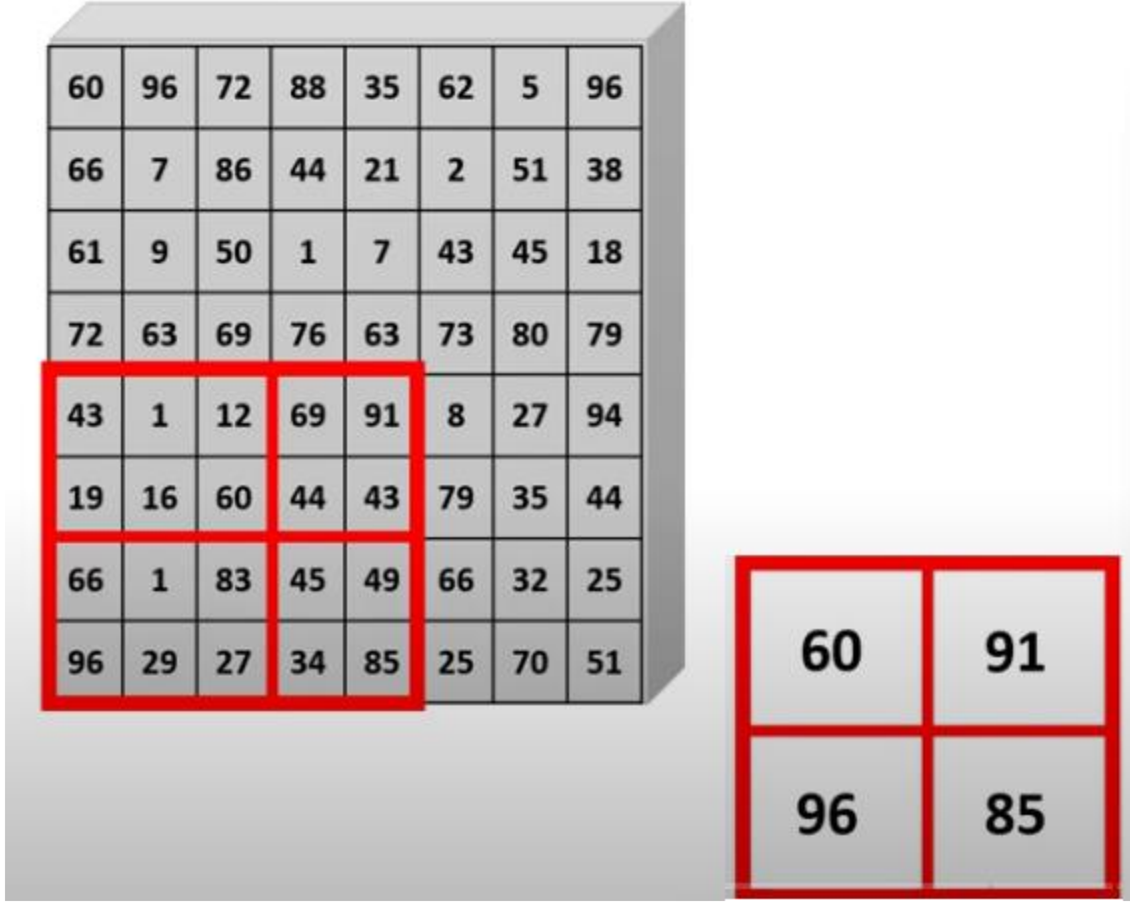


"Slow" R-CNN
SVMs
SVMs
SVMs
Conv Net
Conv Net
Conv Net
Input image

Fast R-CNN


How to make this part is differentiable?
Backbone CNN
Image features
+ Selective Search
Crop and warp the proposed regions
Per-region CNN
Class
Bbox

RoI Pooling

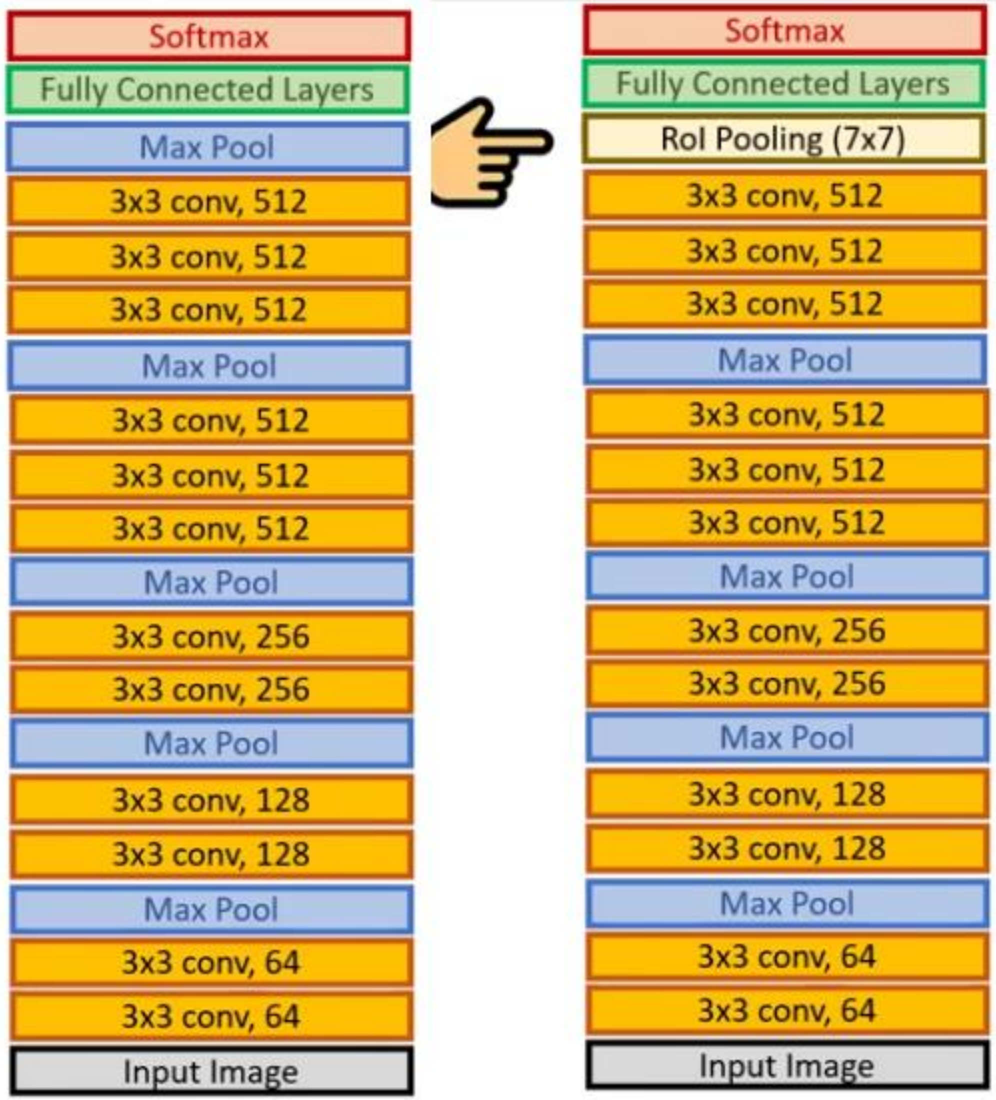


RoI max pooling works by dividing the $h \times w$ RoI window into an $H \times W$ grid of sub-windows of approximate size $h/H \times w/W$ and then max-pooling the values in each sub-window into the corresponding output grid cell. Pooling is applied independently to each feature map channel, as in standard max pooling. The RoI layer is simply the special-case of the spatial pyramid pooling layer used in SPPnets [11] in which there is only one pyramid level. We use the pooling sub-window calculation given in [11].

ROI Pooling is a technique in Fast R-CNN that converts regions of interest (RoIs) of varying sizes into fixed-size feature maps. It works by dividing each RoI into a grid and applying max pooling to each grid cell. This ensures that all RoIs, regardless of their original size, produce consistent outputs. The resulting fixed-size feature maps are then passed through fully connected layers for classification and bounding box regression. ROI Pooling enables the model to handle multiple regions in an image efficiently.

Initializing from pre-trained networks

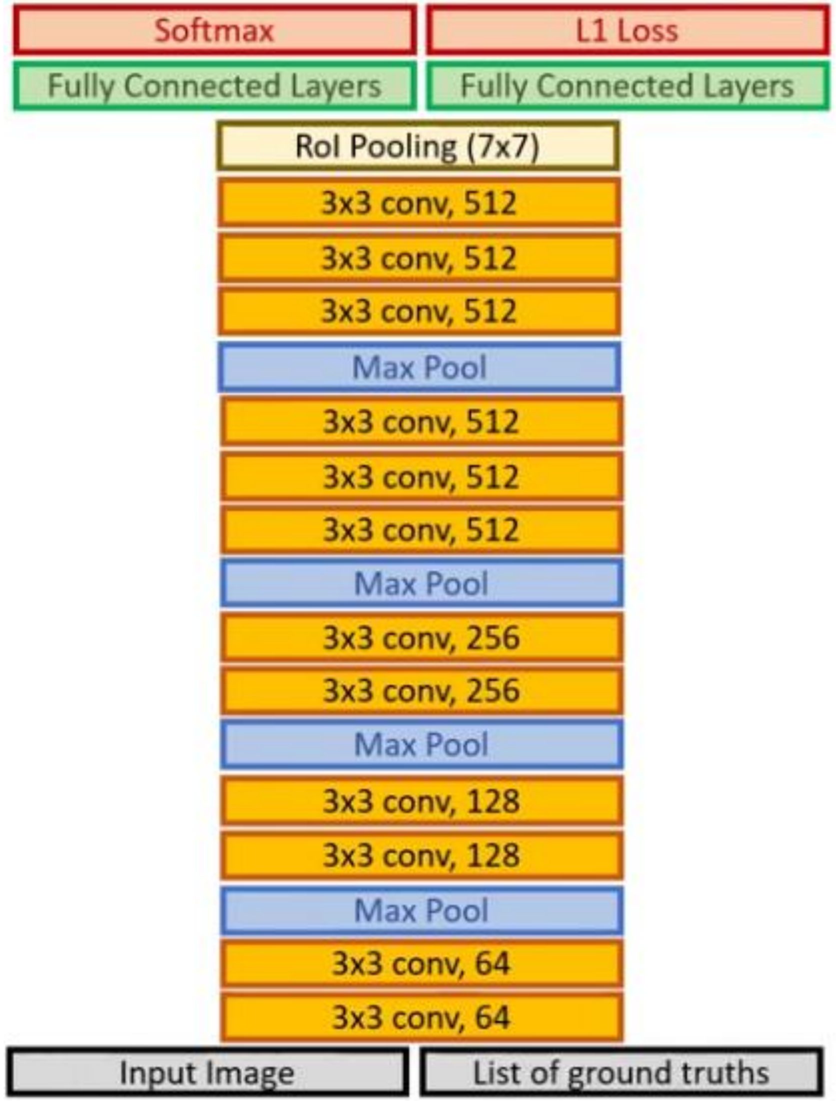
it undergoes three transformations.



First, the last max pooling layer is replaced by a RoI pooling layer that is configured by setting H and W to be compatible with the net's first fully connected layer (*e.g.* $H = W = 7$ for VGG16).

Initializing from pre-trained networks

it undergoes three transformations.

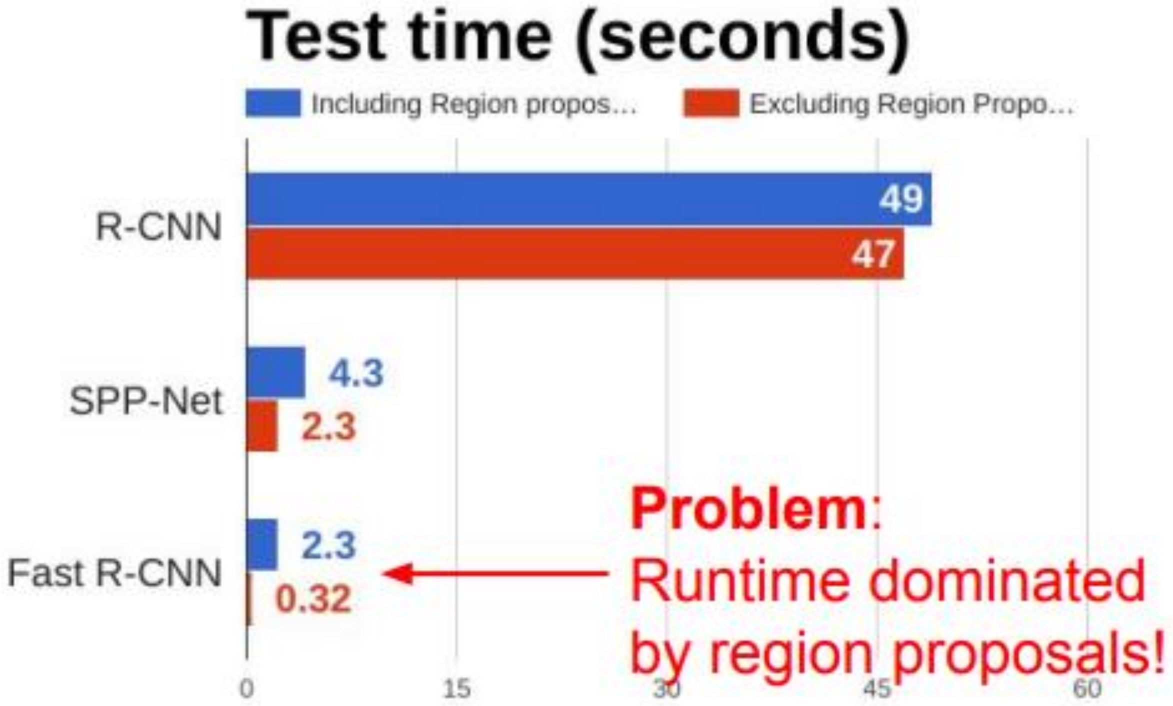


First, the last max pooling layer is replaced by a RoI pooling layer that is configured by setting H and W to be compatible with the net's first fully connected layer (*e.g.*, $H = W = 7$ for VGG16).

Second, the network's last fully connected layer and softmax (which were trained for 1000-way ImageNet classification) are replaced with the two sibling layers described earlier (a fully connected layer and softmax over $K + 1$ categories and category-specific bounding-box regressors).

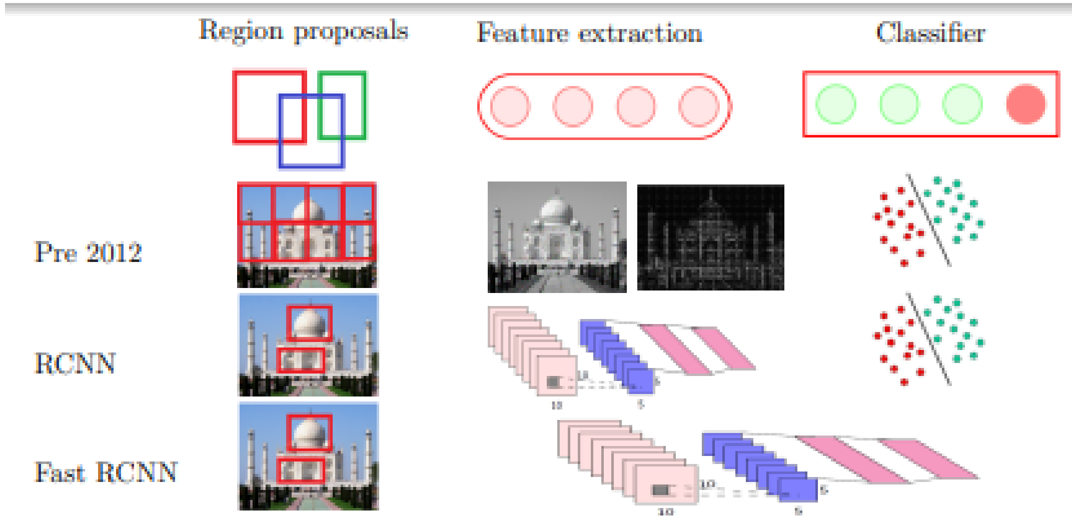
Third, the network is modified to take two data inputs: a list of images and a list of RoIs in those images.

R-CNN vs Fast R-CNN



Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
He et al, "Spatial pyramid pooling in deep convolutional networks for visual recognition", ECCV 2014
Girshick, "Fast R-CNN", ICCV 2015

Comparison



- **Region Proposals:** Selective Search
- **Feature Extraction:** CNN
- **Classifier:** CNN

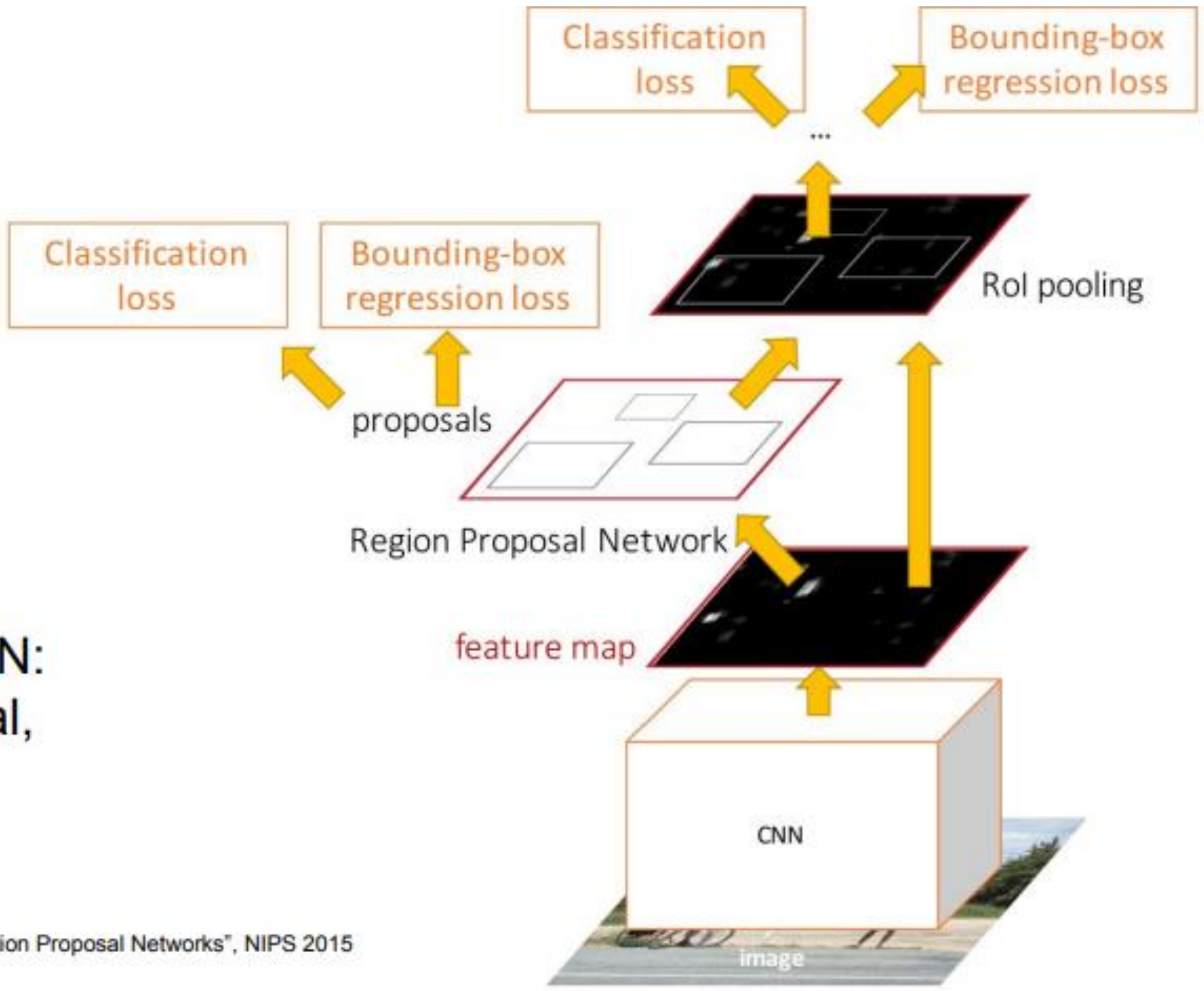
Can we use a CNN for making region proposals also?

Faster R-CNN:

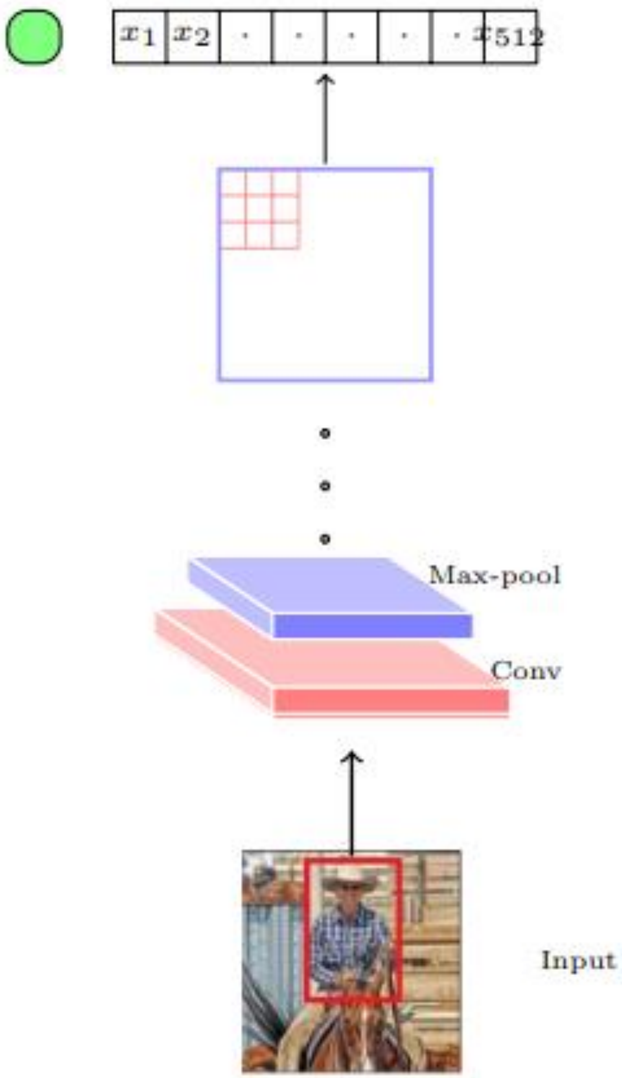
Make CNN do proposals!

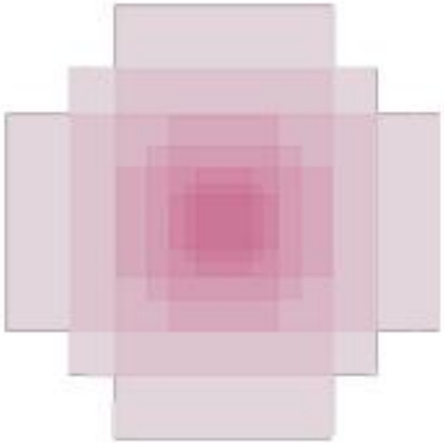
Insert **Region Proposal Network (RPN)** to predict proposals from features

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



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

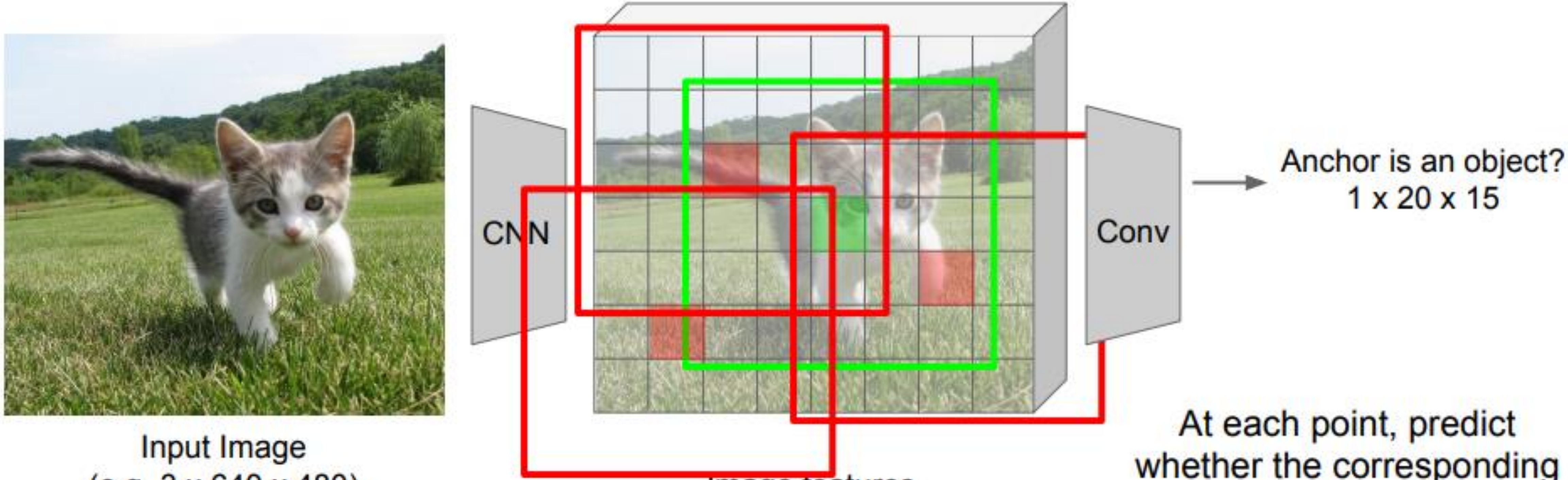




- We now consider k bounding boxes (called anchor boxes) of different sizes & aspect ratio
- We are interested in the following two questions:
- Given the $512d$ representation of a position, what is the probability that a given anchor box centered at this position contains an object? (Classification)
- How do you predict the true bounding box from this anchor box? (Regression)

Region Proposal Network

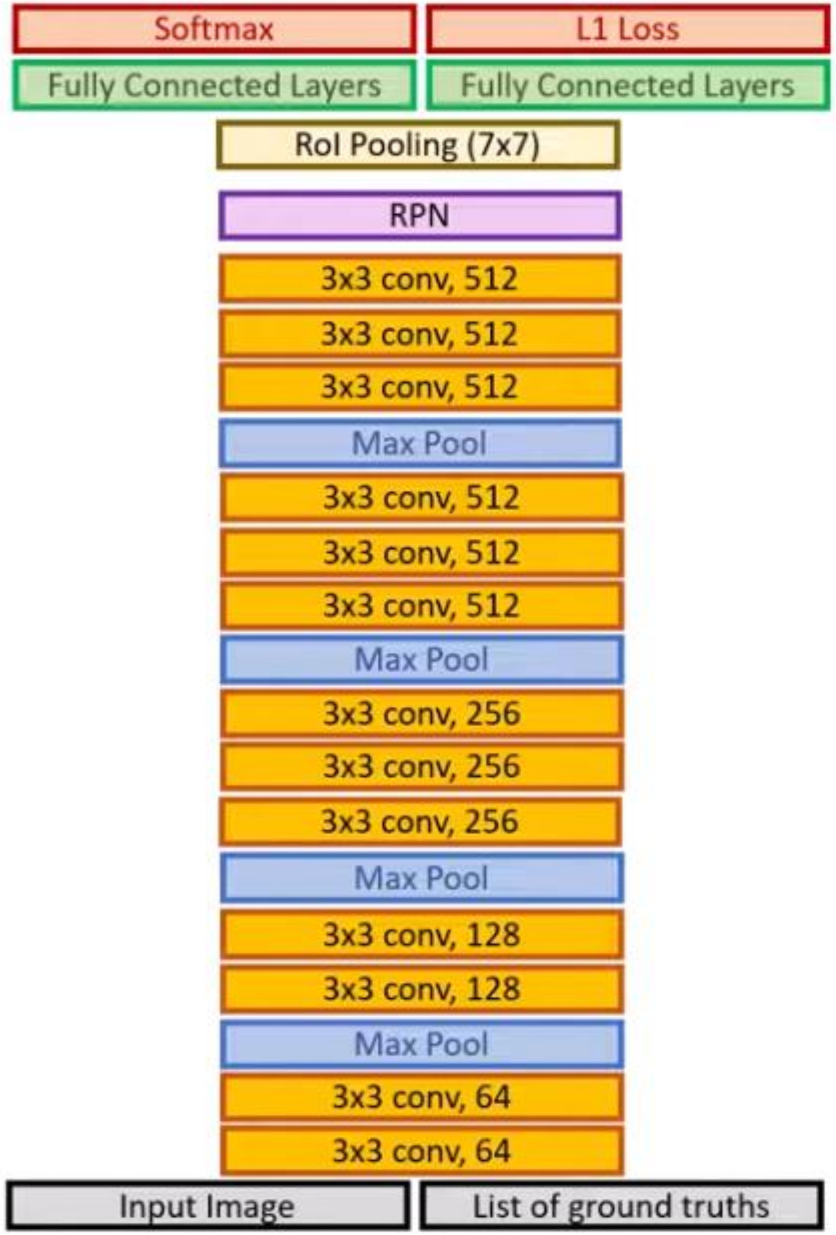
Imagine an **anchor box** of fixed size at each point in the feature map



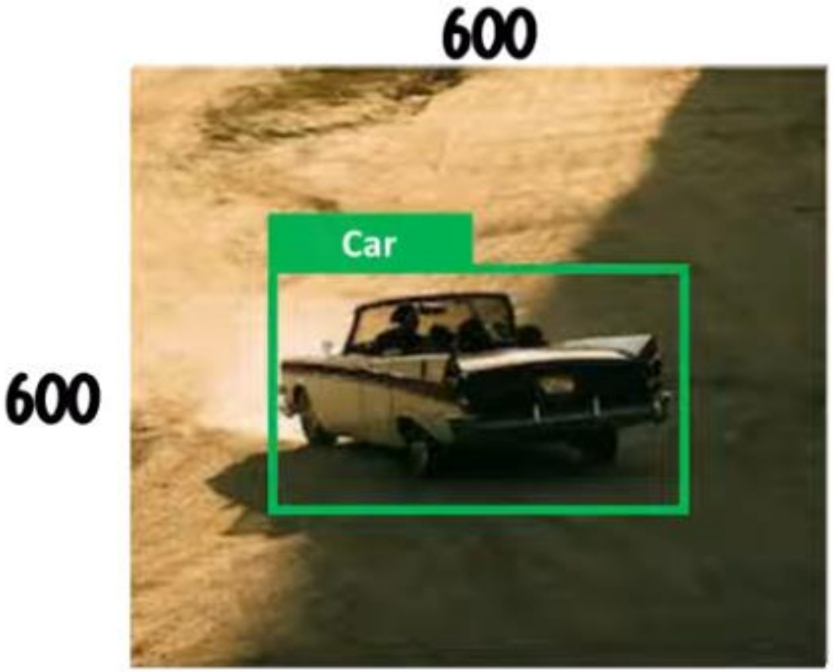
Input Image
(e.g. 3 x 640 x 480)

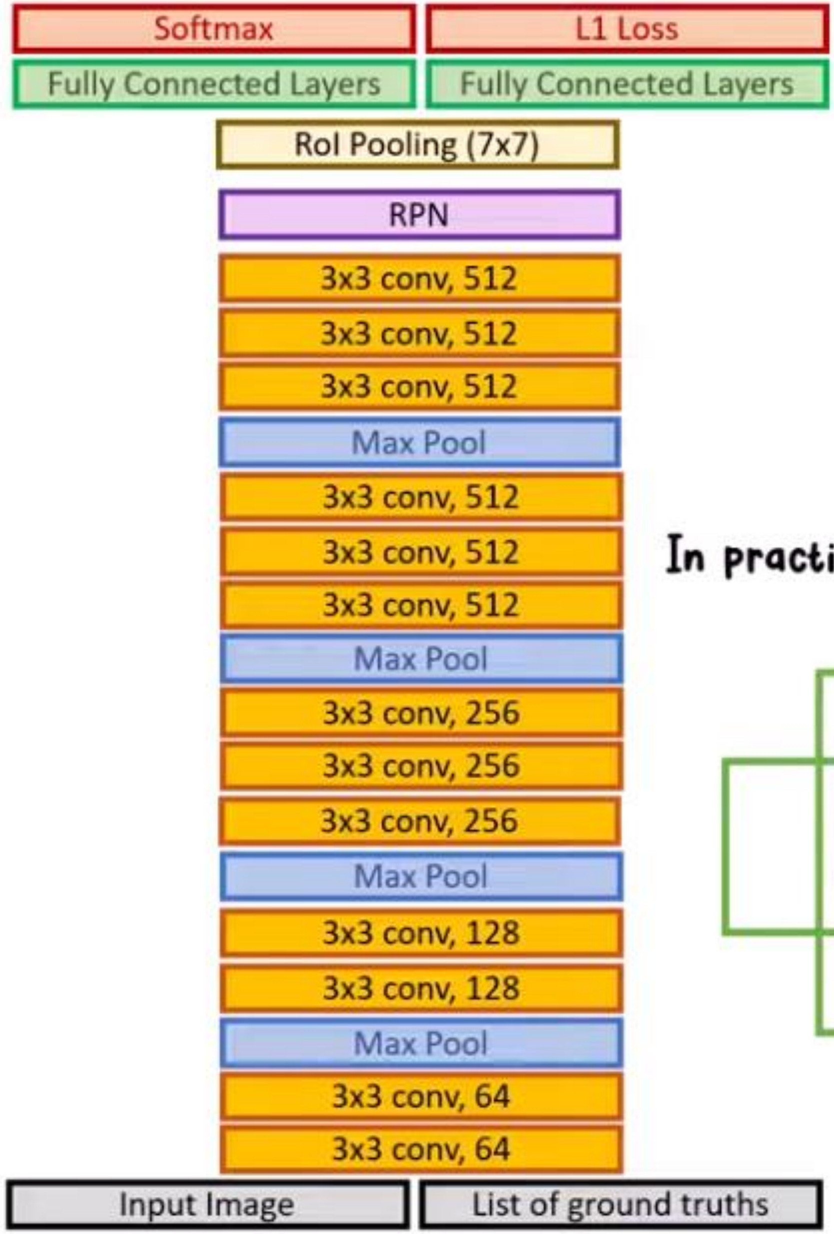
Image features
(e.g. 512 x 20 x 15)

At each point, predict whether the corresponding anchor contains an object (binary classification)


Softmax
L1 Loss
Fully Connected Layers
Fully Connected Layers
RoI Pooling (7x7)
RPN
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Max Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Max Pool
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
Max Pool
3x3 conv, 128
3x3 conv, 128
Max Pool
3x3 conv, 64
3x3 conv, 64
Input Image
List of ground truths

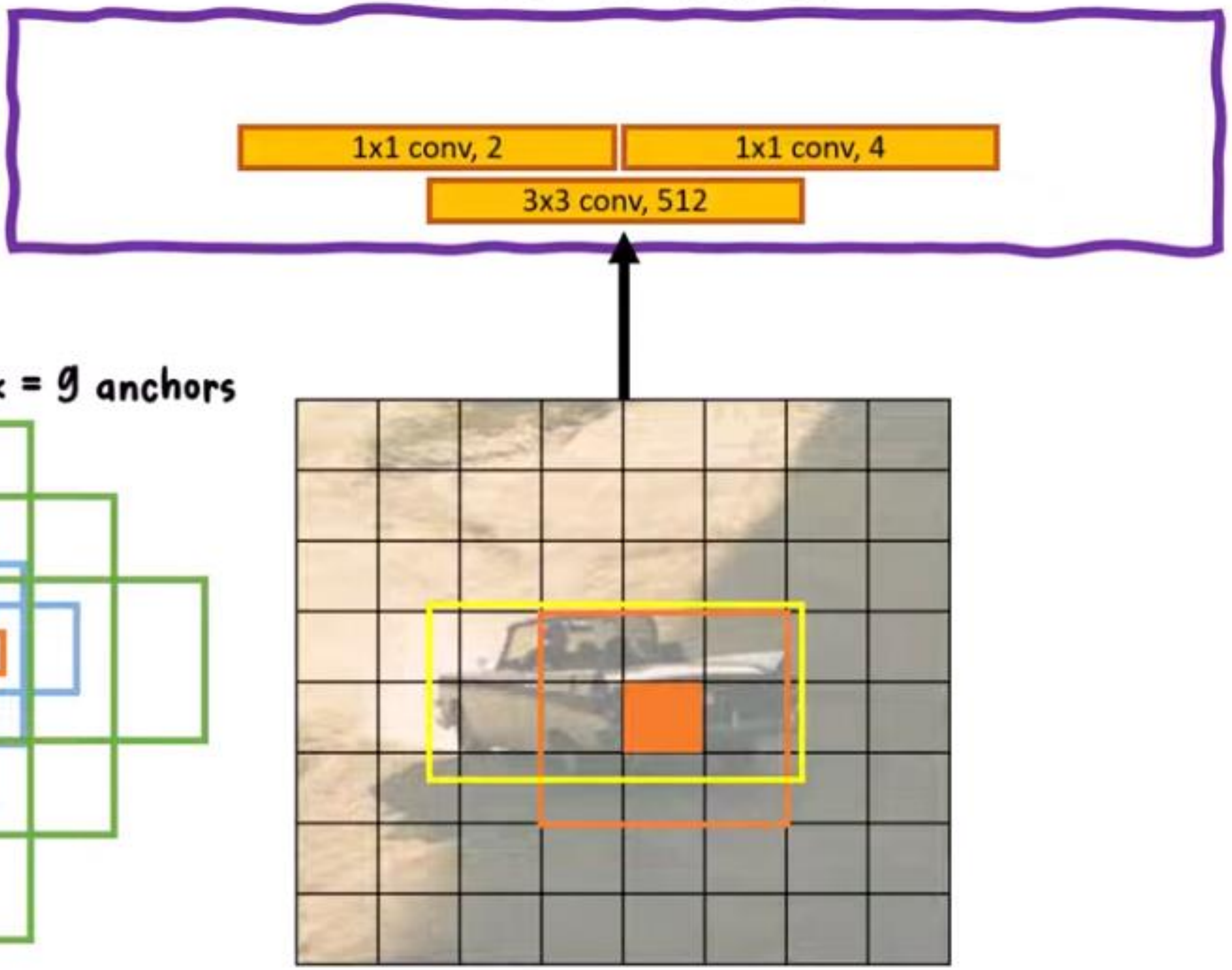
Inside RPN

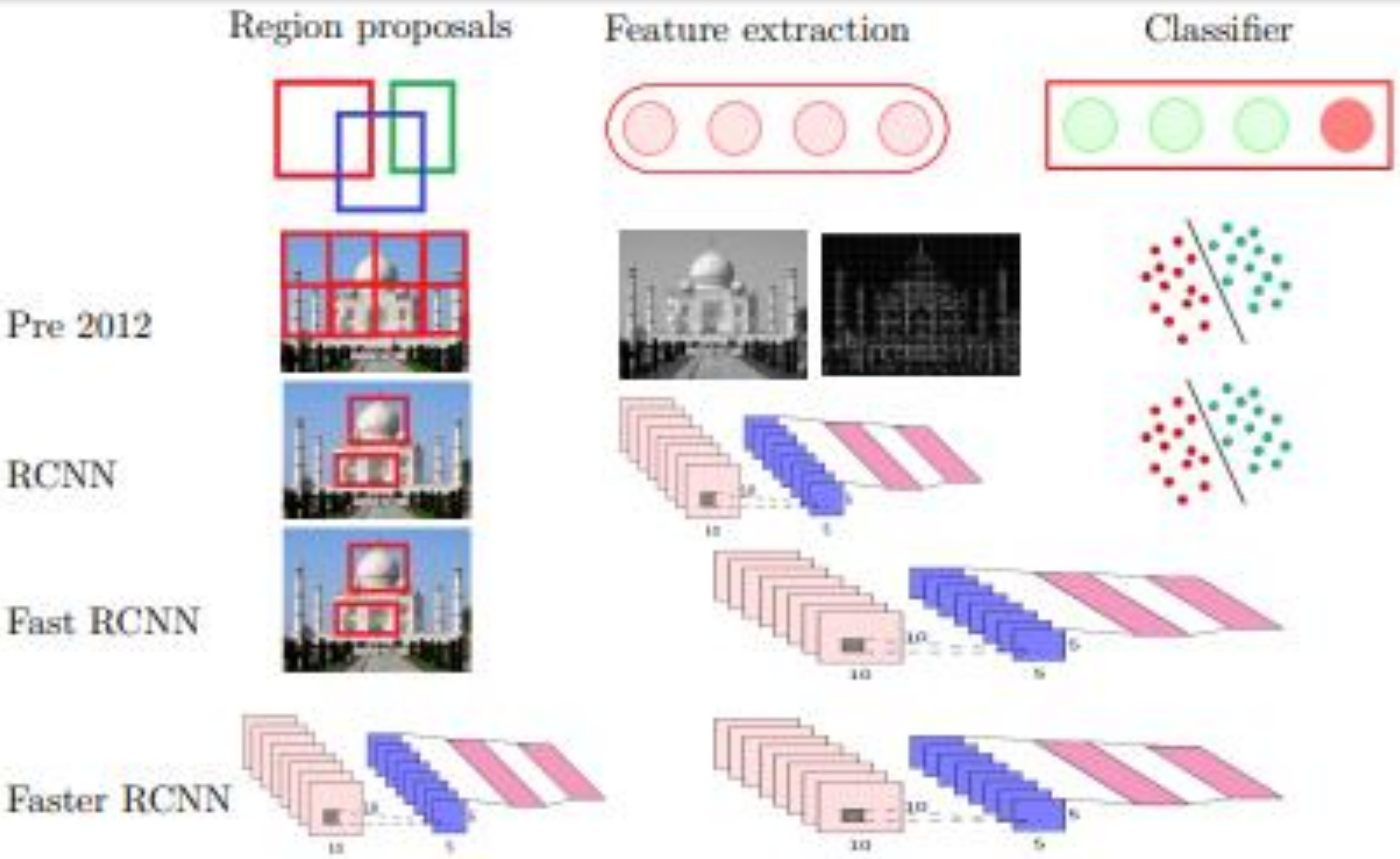

600
600
Car



Inside RPN

In practice we use k = 9 anchors



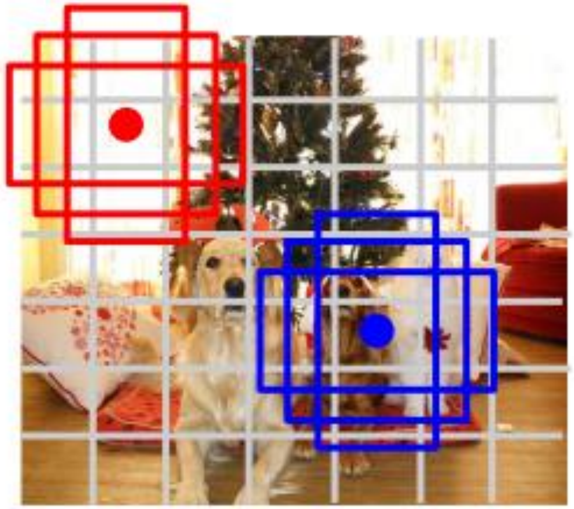


Region proposals
Feature extraction
Classifier
Pre 2012
RCNN
Fast RCNN
Faster RCNN

(You Only Look Once

Single-Stage Object Detectors: YOLO / SSD / RetinaNet

Input image
3 x H x W



Divide image into grid
7 x 7

Image a set of **base boxes** centered at each grid cell
Here B = 3

Within each grid cell:

- Regress from each of the B base boxes to a final box with 5 numbers: (dx, dy, dh, dw, confidence)
- Predict scores for each of C classes (including background as a class)
- Looks a lot like RPN, but category-specific!

Output:
$7 \times 7 \times (5 * B + C)$

Redmon et al, "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016
Liu et al, "SSD: Single-Shot MultiBox Detector", ECCV 2016
Lin et al, "Focal Loss for Dense Object Detection", ICCV 2017

divides the image into grid- For each grid cell, YOLO predicts multiple bounding boxes-

Object Detection: Lots of variables ...

Backbone Network
VGG16
ResNet-101
Inception V2
Inception V3
Inception ResNet
MobileNet

“Meta-Architecture”
Two-stage: Faster R-CNN
Single-stage: YOLO / SSD
Hybrid: R-FCN

Image Size
 # Region Proposals
...

Takeaways
Faster R-CNN is slower but more accurate

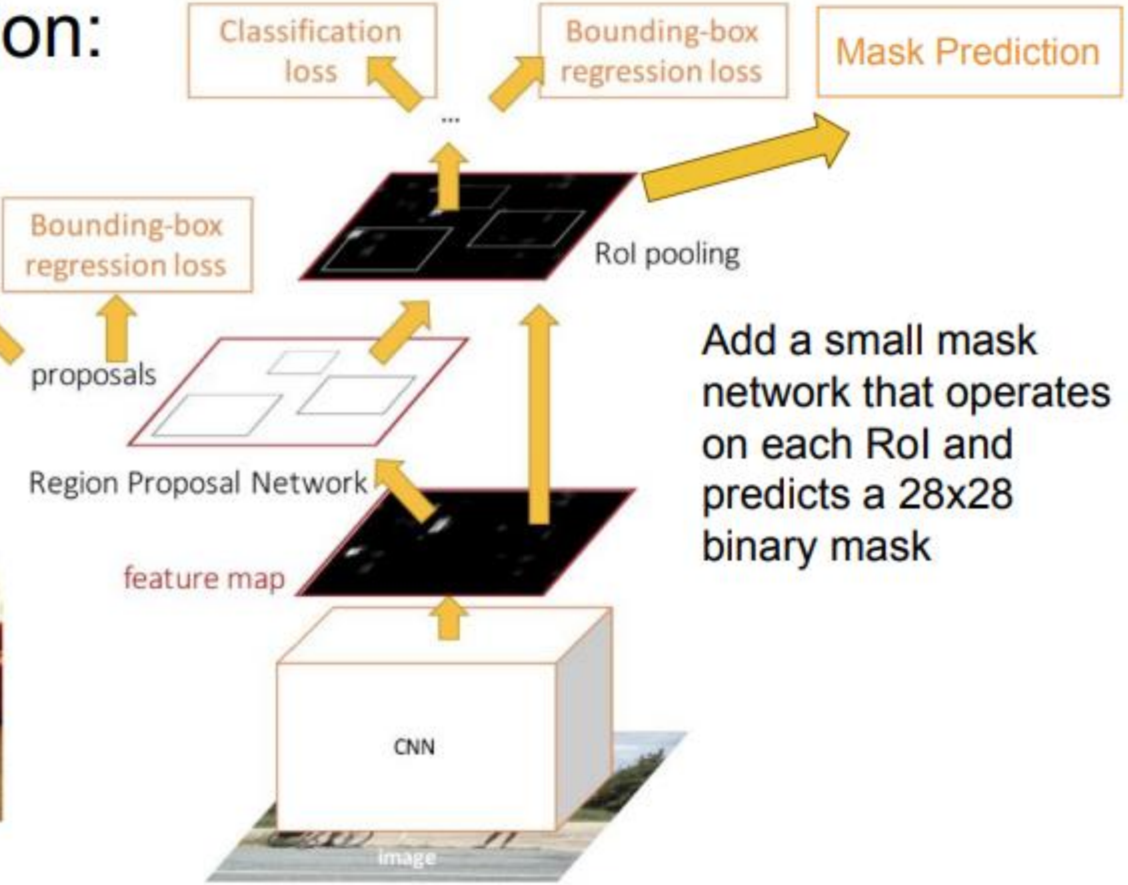
SSD is much faster but not as accurate

Bigger / Deeper backbones work better

Huang et al, “Speed/accuracy trade-offs for modern convolutional object detectors”, CVPR 2017
Zou et al, “Object Detection in 20 Years: A Survey”, arXiv 2019

R-FCN: Dai et al, "R-FCN: Object Detection via Region-based Fully Convolutional Networks", NIPS 2016
Inception-V2: Ioffe and Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", ICML 2015
Inception V3: Szegedy et al, "Rethinking the Inception Architecture for Computer Vision", arXiv 2016
Inception ResNet: Szegedy et al, "Inception-V4, Inception-ResNet and the Impact of Residual Connections on Learning", arXiv 2016
MobileNet: Howard et al, "Efficient Convolutional Neural Networks for Mobile Vision Applications", arXiv 2017

Instance Segmentation: Mask R-CNN



Add a small mask network that operates on each RoI and predicts a 28x28 binary mask

Object Detection

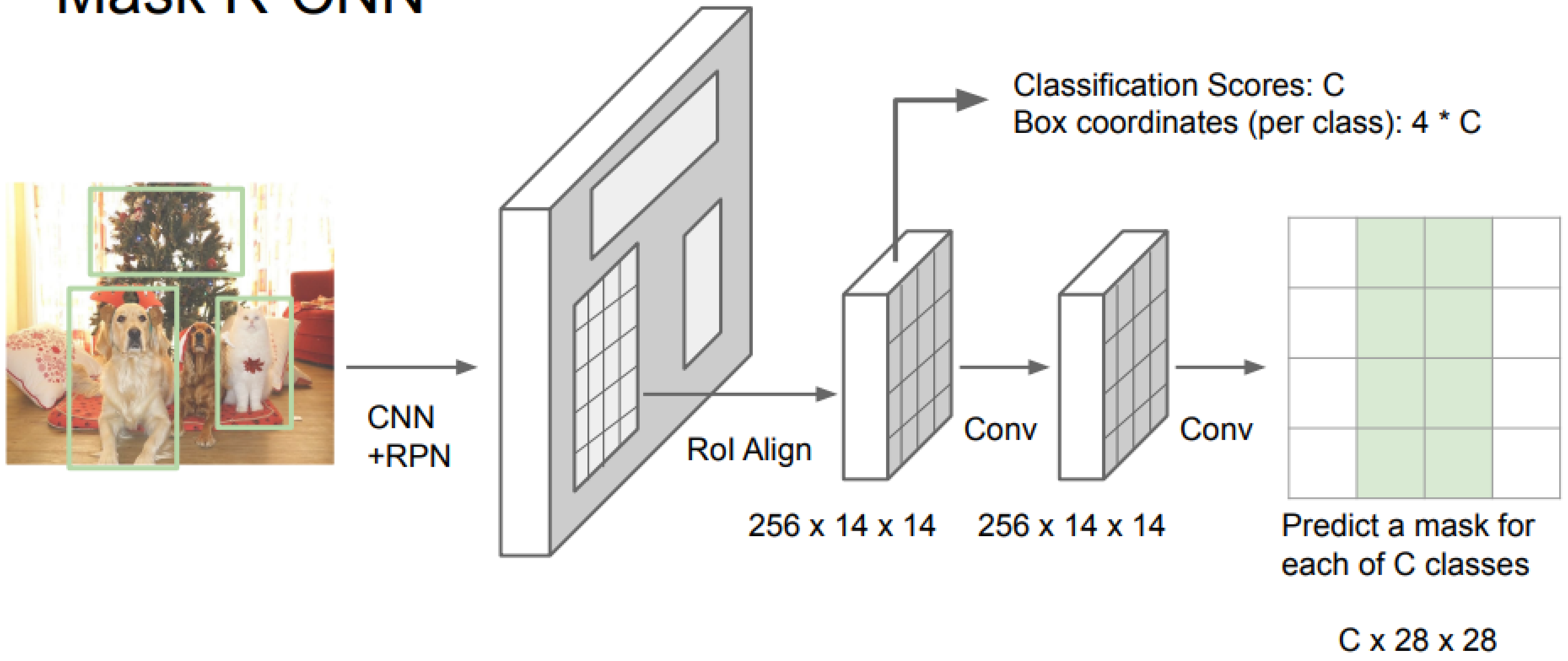
DOG, DOG, CAT

Instance Segmentation

DOG, DOG, CAT

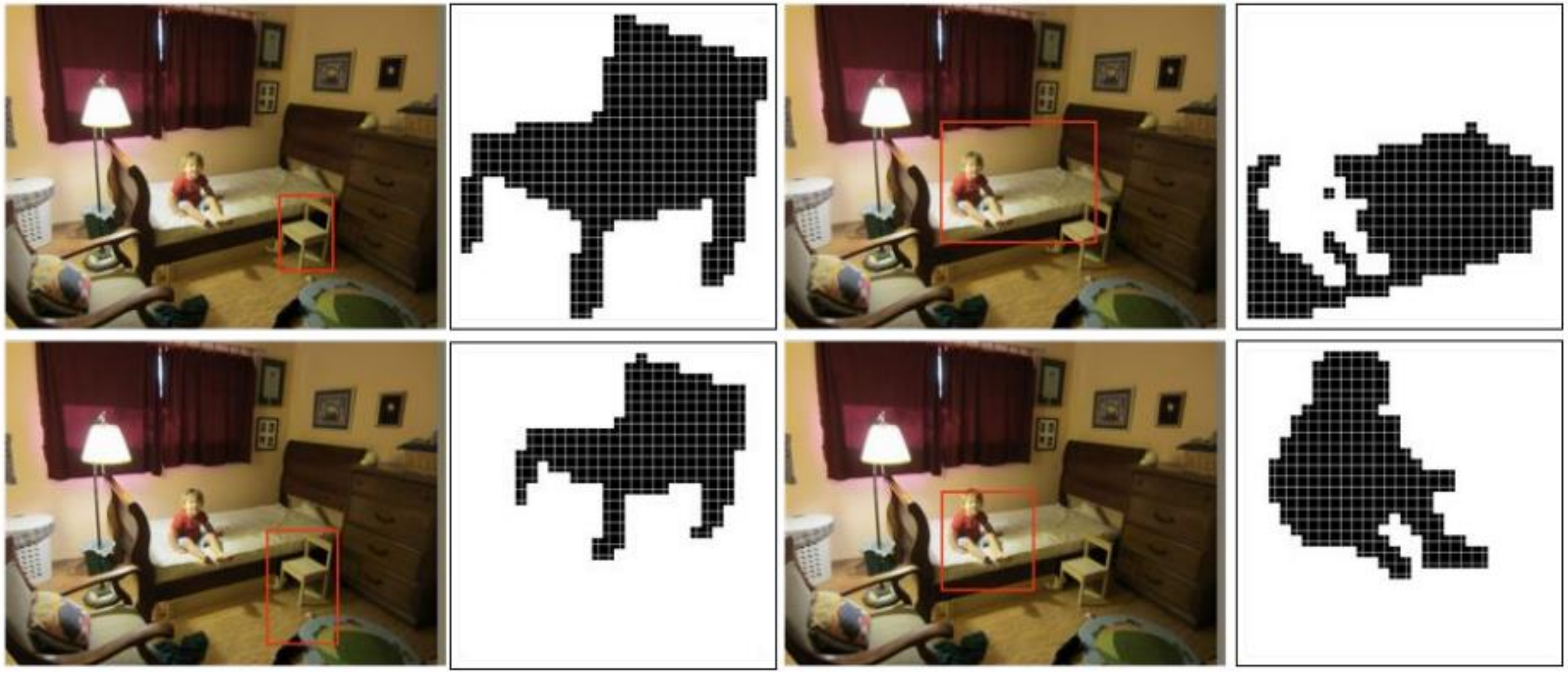
He et al, "Mask R-CNN", ICCV 2017

Mask R-CNN

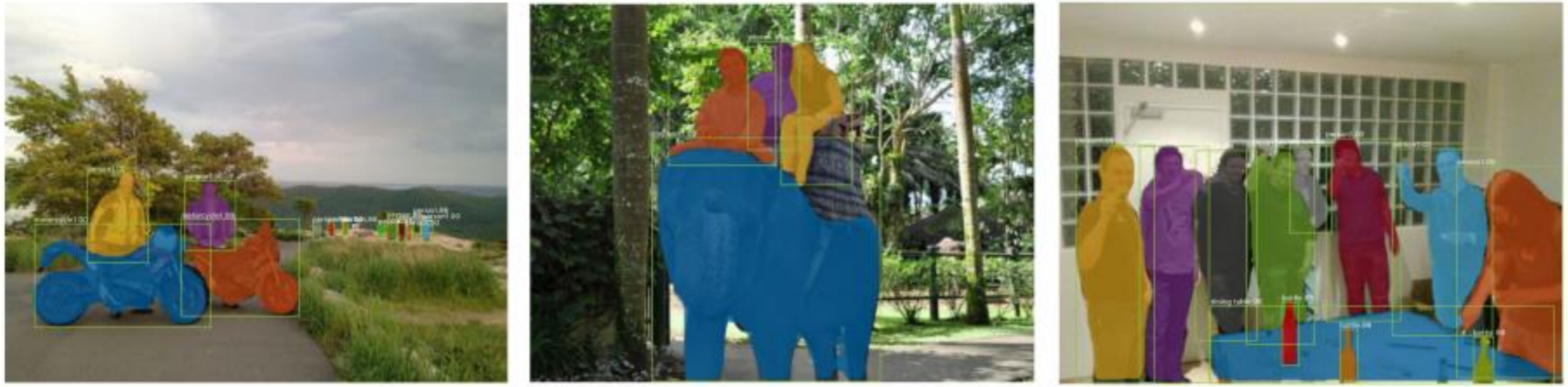


He et al, "Mask R-CNN", arXiv 2017

Mask R-CNN: Example Mask Training Targets



Mask R-CNN: Very Good Results!



He et al, "Mask R-CNN", ICCV 2017

References

- https://arxiv.org/pdf/1311.2524
- https://arxiv.org/pdf/1504.08083
- https://www.youtube.com/watch?v=5gAq6BZ87aA
- https://cs231n.stanford.edu/slides/2021/lecture_15.pdf