

# **COS30082**

## **Applied Machine Learning**

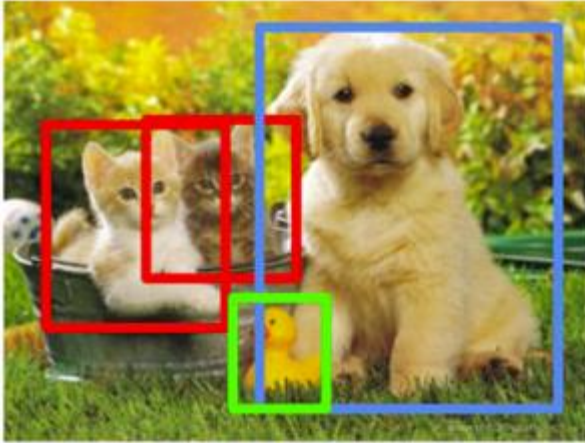


## Lecture 7

### Object Detection

# Object detection vs. Image classification

## Object detection



**cat, duck, dog**

Locate the presence of objects with a bounding box and types or classes of the located objects in an image.

## Image classification

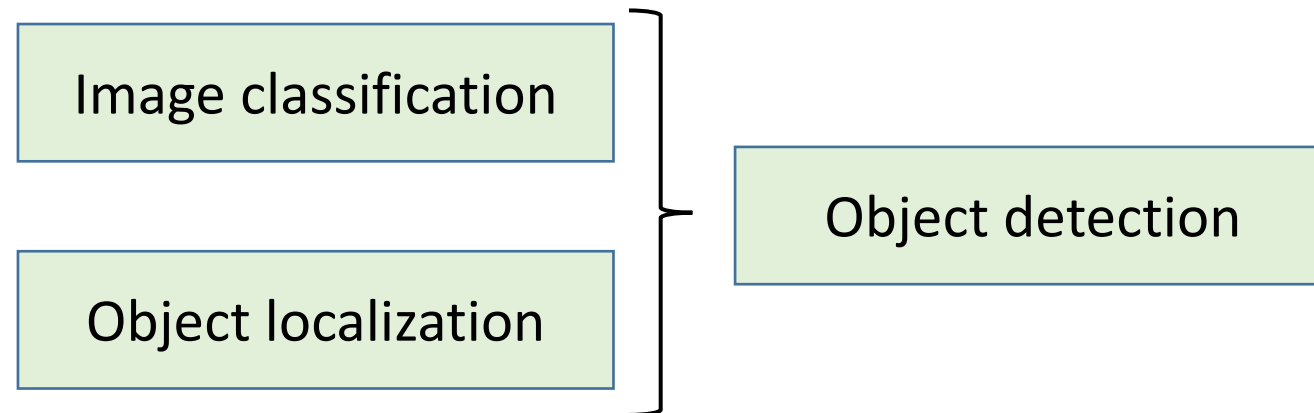


**cat**

Predict the type or class of an object in an image.

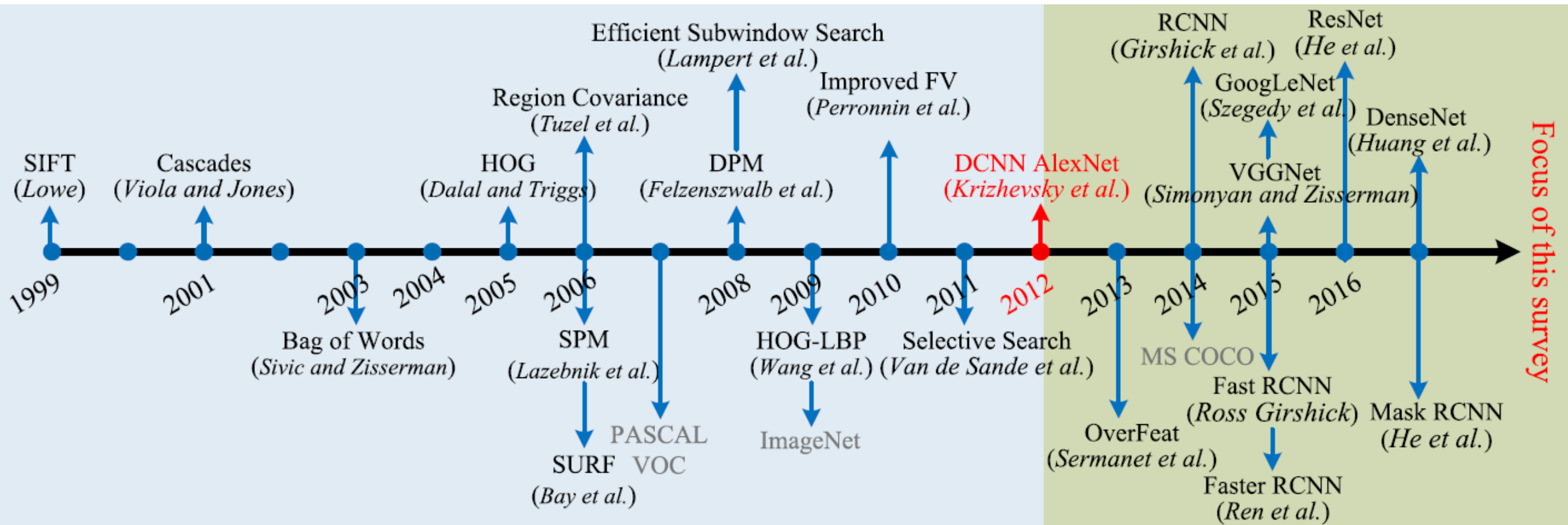
# Object detection

- **Image Classification:** The entire image is classified with a single label.
  - **Input:** An image
  - **Output:** A class label
- **Object Localization:** Locate the presence of objects in an image and indicate their location with a bounding box.
  - **Input:** An image with one or more objects
  - **Output:** One or more bounding boxes (e.g. defined by a point, width, and height).
- **Object Detection:** Locate the presence of objects with a bounding box and types or classes of the located objects in an image.
  - **Input:** An image with one or more objects.
  - **Output:** One or more bounding boxes (e.g. defined by a point, width, and height), and a class label for each bounding box.



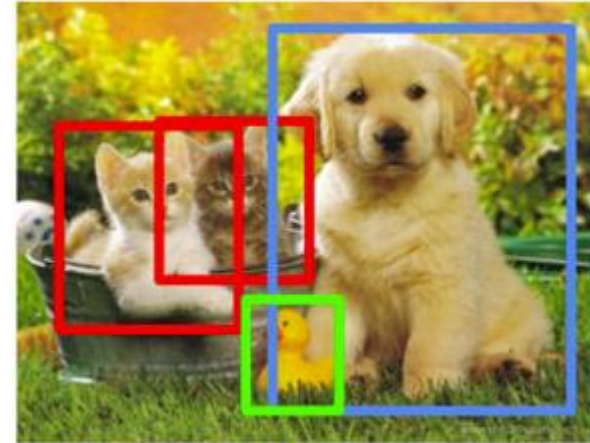
# Milestones of object detection and recognition

- The time period up to 2012 is dominated by handcrafted features, a transition took place in 2012 with the development of Deep CNNs for image classification by Krizhevsky et al. (2012a), with methods after 2012 dominated by related deep networks



# Why standard CNN framework fails?

- We cannot proceed with this problem by building a standard convolutional network followed by a fully connected layer is because the length of the output layer is variable — not constant
- Note that the number of occurrences of the objects of interest is not fixed



Cat (x, y, w, h)  
Duck (x, y, w, h)  
Dog (x, y, w, h)



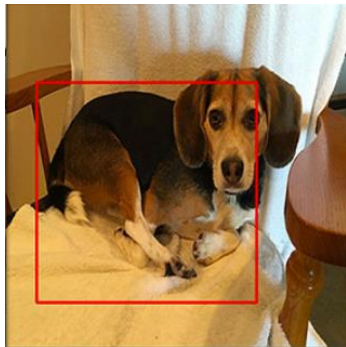
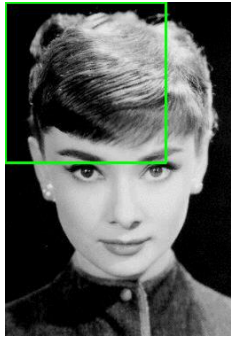
Cat (x, y, w, h)

# Object detection approach

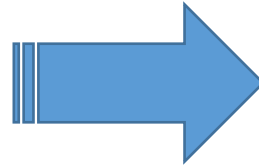
- Method 1: Traditional object detection approach
  - Involves standard, computer-vision based object detection methods (i.e., non-deep learning methods) such as **sliding windows** and **image pyramids**.
  - Typically used in **HOG + Linear SVM-based** object detectors.
  - Slow, tedious, and error-prone.
- Method 2: Pre-trained network as a base network
  - Use the **pre-trained network** as a **base network** in a deep learning **object detection framework** (i.e., Faster R-CNN, SSD, YOLO).
  - Requires intimate knowledge on how deep learning object detectors work.



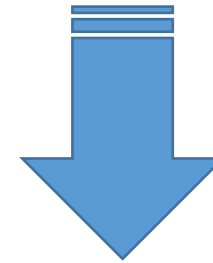
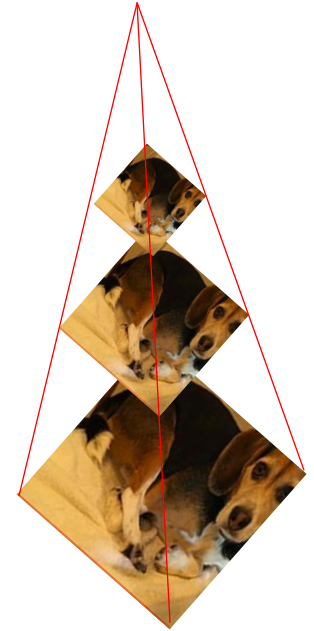
# Traditional object detection approach



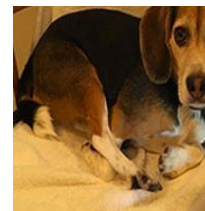
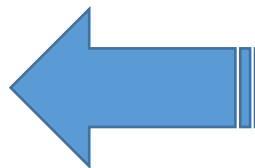
**Step 1:** Fixed size sliding windows, which slide from left-to-right and top-to-bottom to localize objects at different locations



**Step 2:** An image pyramid to detect objects at varying scales



**Step 4:** If the classification probability of label  $L$ :  $P(L) > T$ , mark the bounding box of the ROI as the label  $L$ .

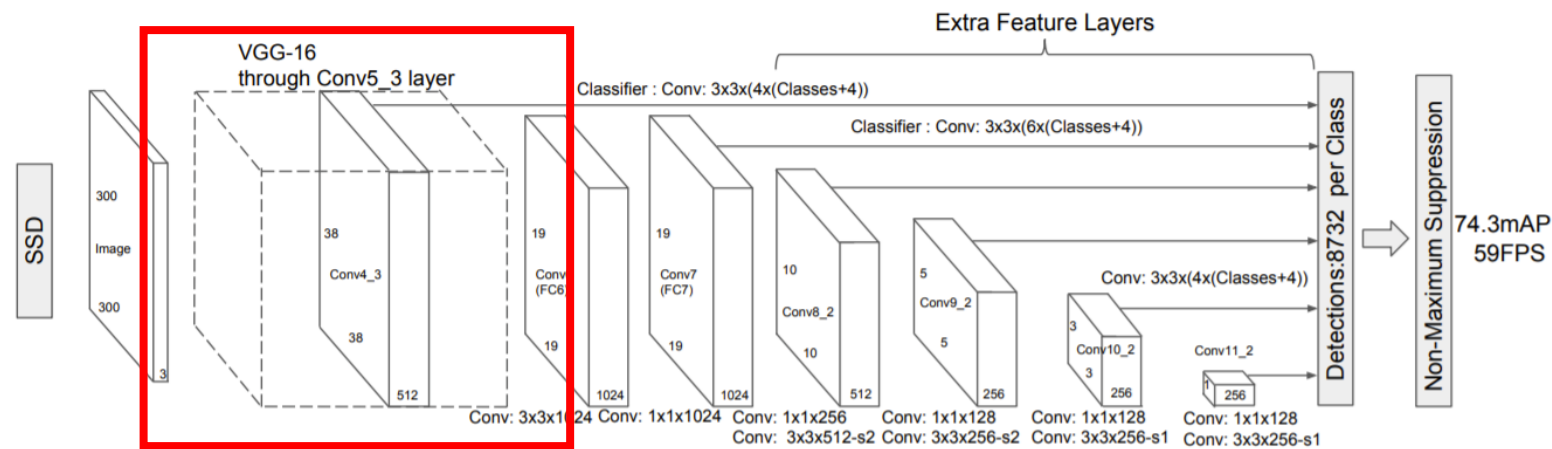


**Step 3:** Classification of each ROI (Region of Interest) via:

- A pre-trained CNN
- Traditional machine learning method (e.g., SVM)

# Pre-trained network as a base network

- Basically is to treat the pre-trained classification network as a **base network** in a deep learning **object detection framework** (such as Faster R-CNN, SSD, or YOLO).



The **VGG16 base network** is a component of the **SSD** deep learning object detection framework.



- The base networks are the common (classification) CNN architectures, including:
  - VGGNet
  - ResNet
  - MobileNet
  - DenseNet
- Typically these networks are pre-trained to perform classification on a large image dataset, such as ImageNet, to learn a rich set of discerning, discriminating filters.

- **Two-stage detector**

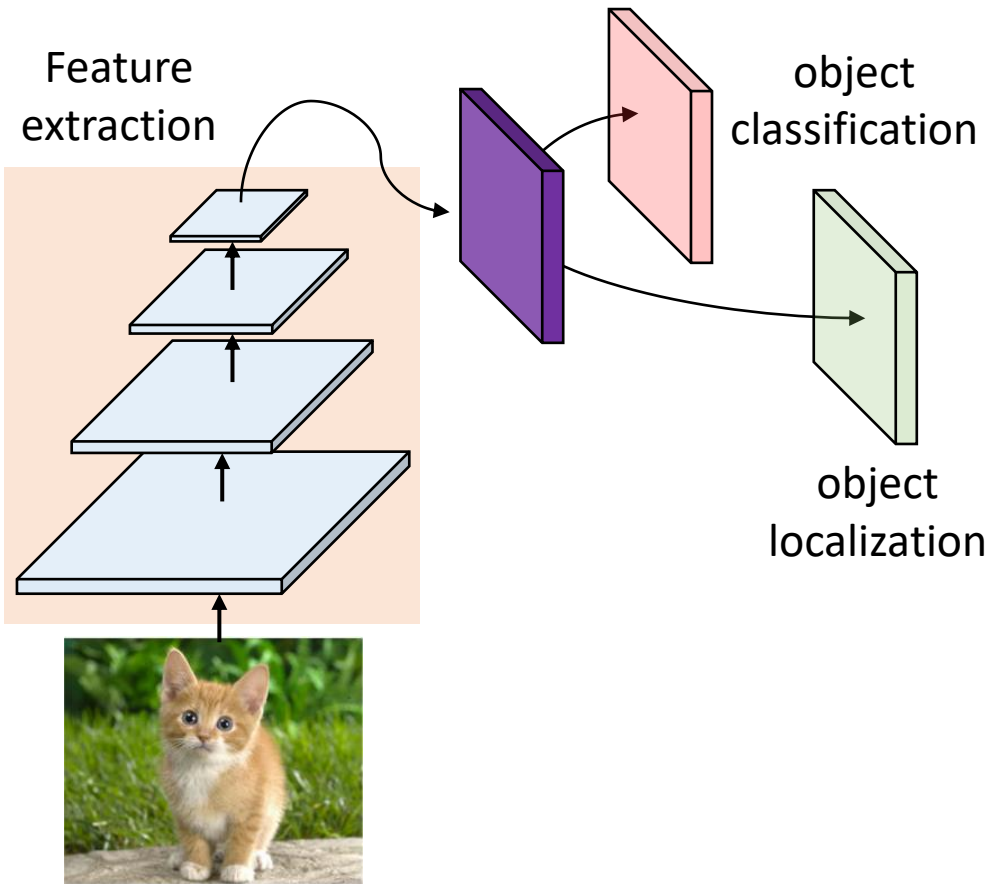
- First, the model proposes a set of regions of interests by **region proposal** methods (e.g. selective search, edge boxes) or network.
  - The proposed regions are sparse as the potential bounding box candidates can be infinite.
- Then a classifier only processes the region candidates.
- Models in the R-CNN family are all region-based: RCNN, Fast- RCNN, Faster-RCNN

- **One stage detector**

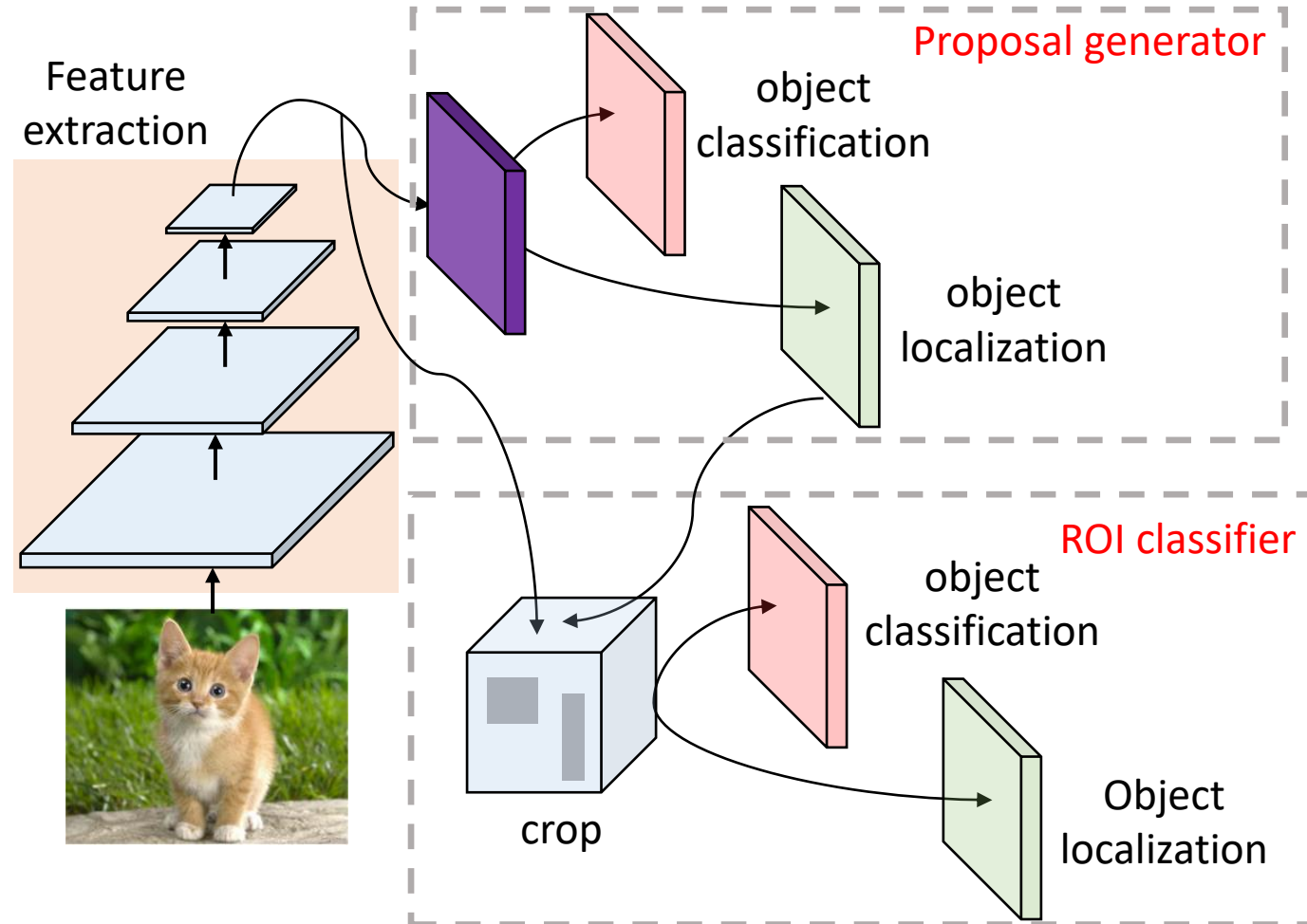
- Single convolutional network predicts the bounding boxes and the class probabilities for these boxes.
- YOLO, SSD

# One-stage vs two-stage detector

## One-stage detector



## Two-stage detectors



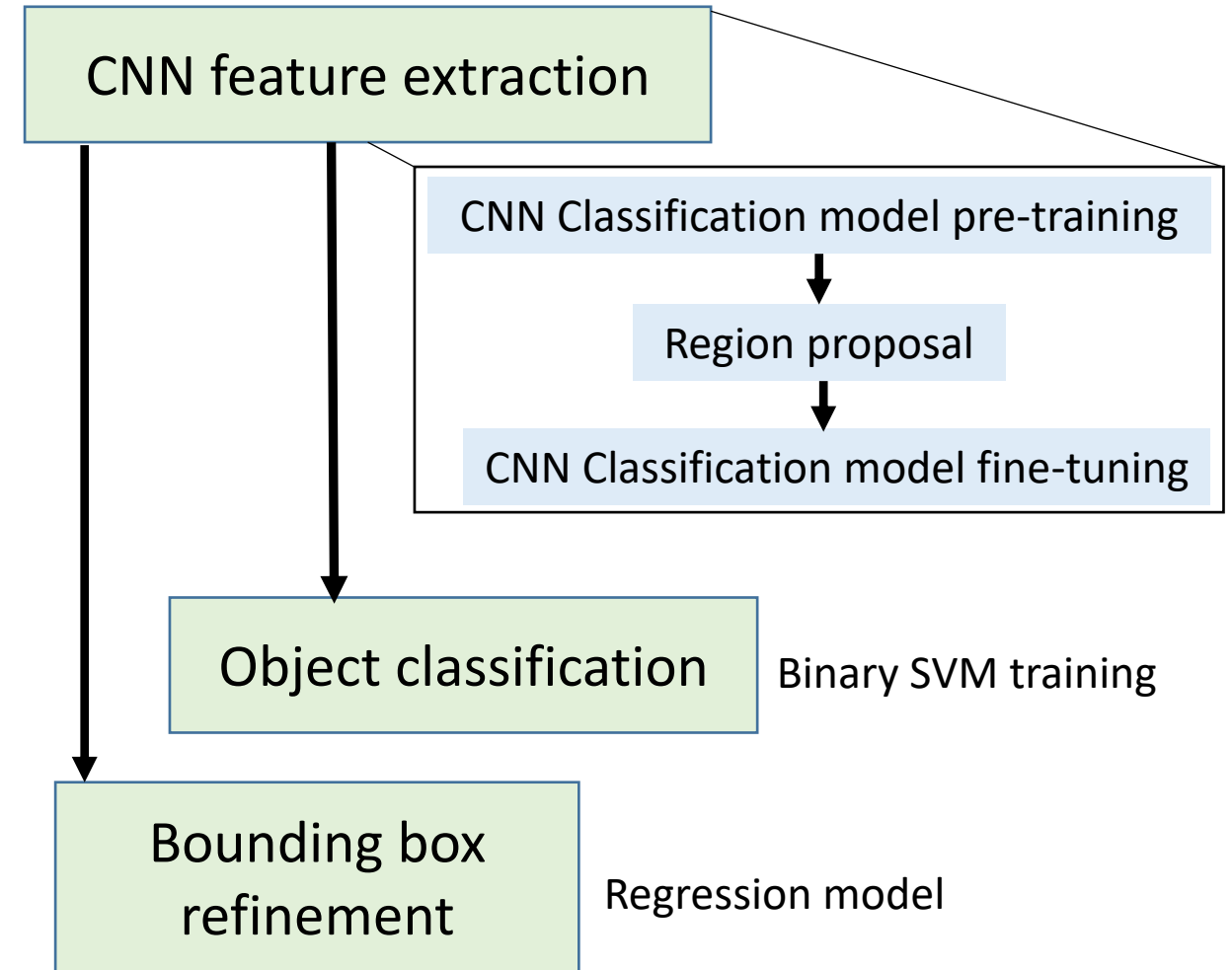
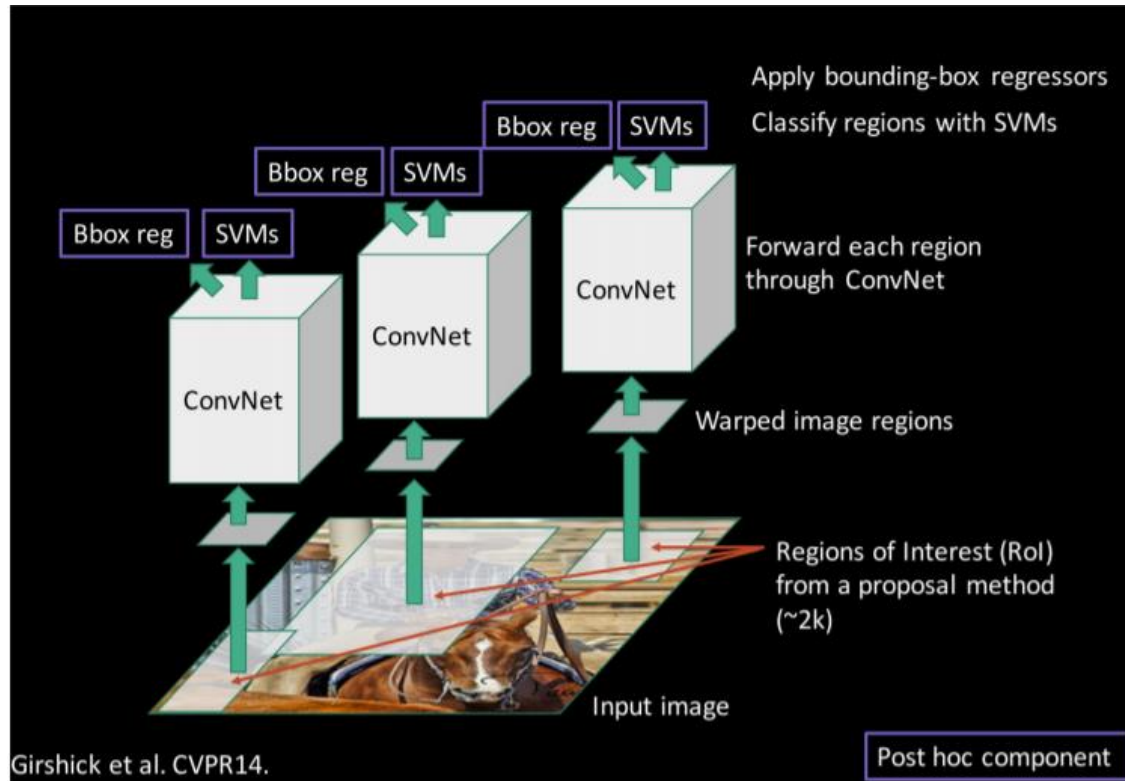
# One-stage vs two-stage detector

- **Two-stage detector:** This solution can be slow because we have to run predictions for every selected region.
- **One-stage detector:** They are commonly used for real-time object detection as, in general, they trade a bit of accuracy for large improvements in speed.

# Two-stage detector

- **RCNN (Region based CNN)**
- Fast-RCNN
- Faster-RCNN

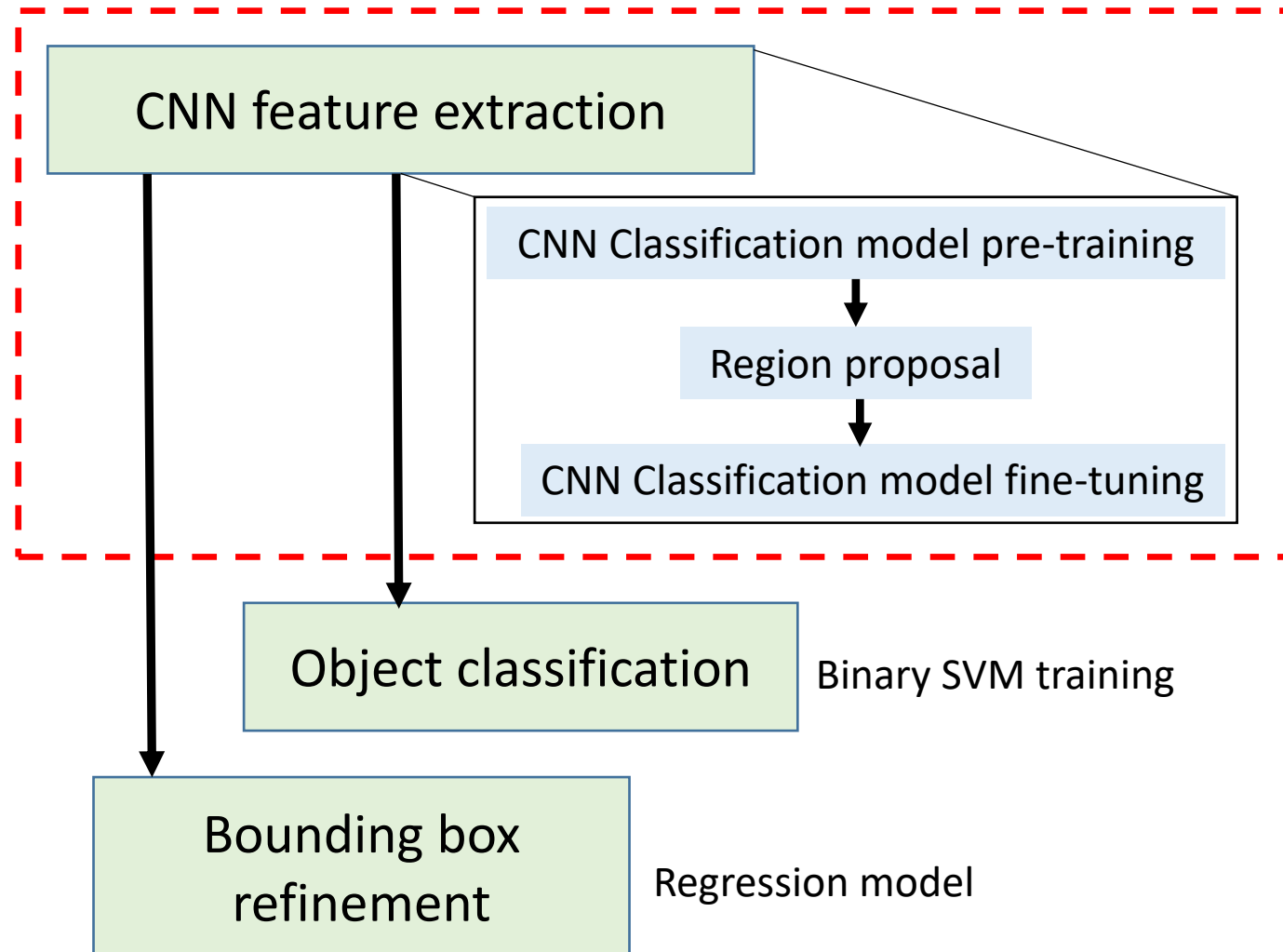
# RCNN



Girshick, R., Donahue, J., Darrell, T. and Malik, J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580-587).



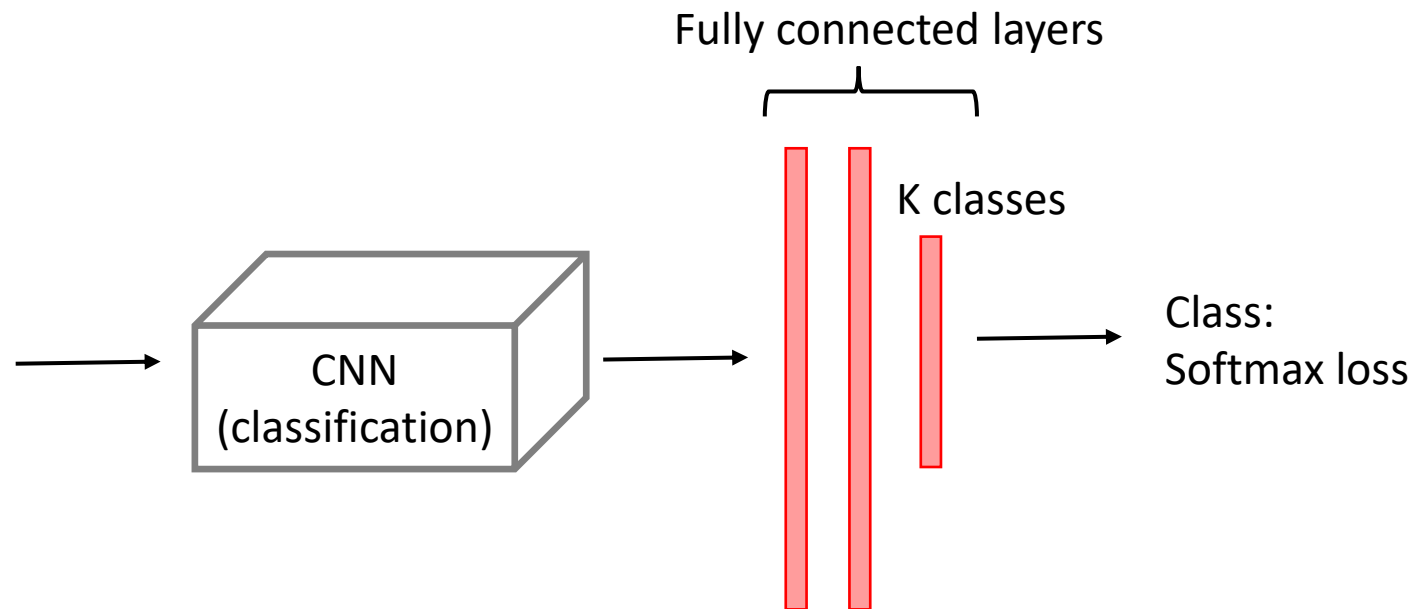
# CNN feature extraction



# CNN Classification model pre-training

- **Pre-train** a CNN network on image classification tasks, for example, VGG or ResNet trained on ImageNet dataset

Training images



# Region proposal

Target images




Edge  
boxes

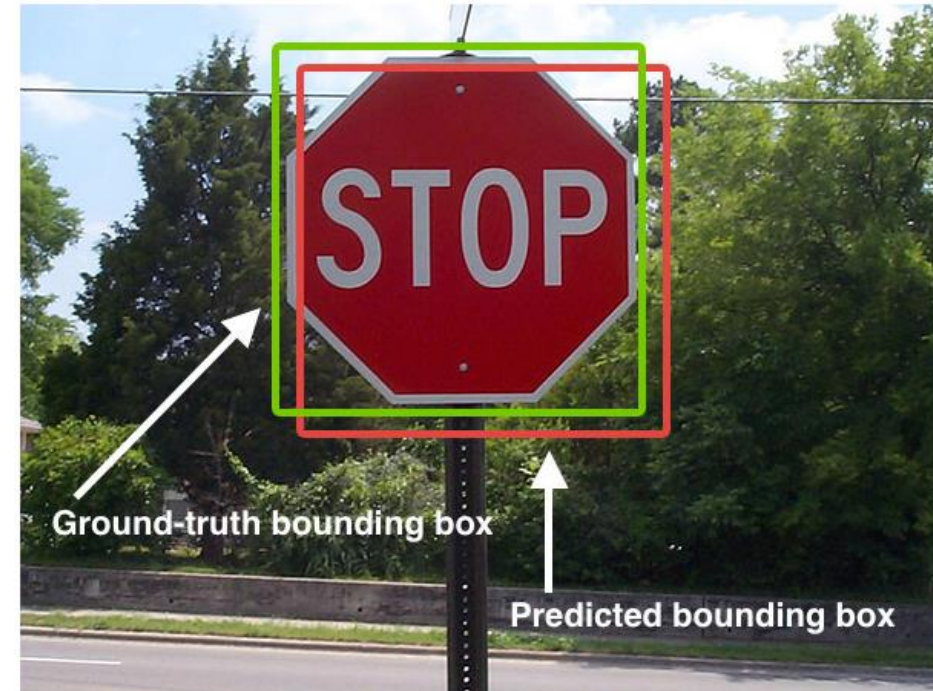
Region proposal



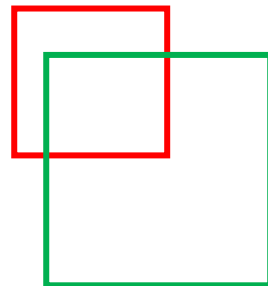
- We first generate region proposals using an algorithm such as Edge Boxes. (~2k candidate proposals per image)
- Those regions may contain target objects and they are of different sizes.
- Region proposals with  $\geq 0.5$  IoU (Intersection of Union) overlap with a ground-truth box are treated as positives for that box's class and the rest as negatives.

# Intersection of Union (IoU)

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


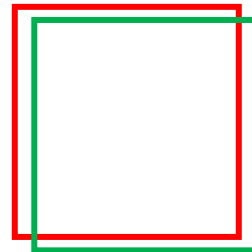


Poor



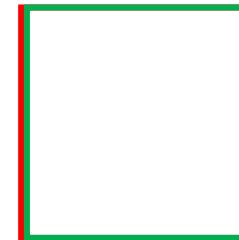
IoU = 0.421

Good



IoU = 0.748

Excellent



IoU = 0.952

# CNN Classification model fine-tuning

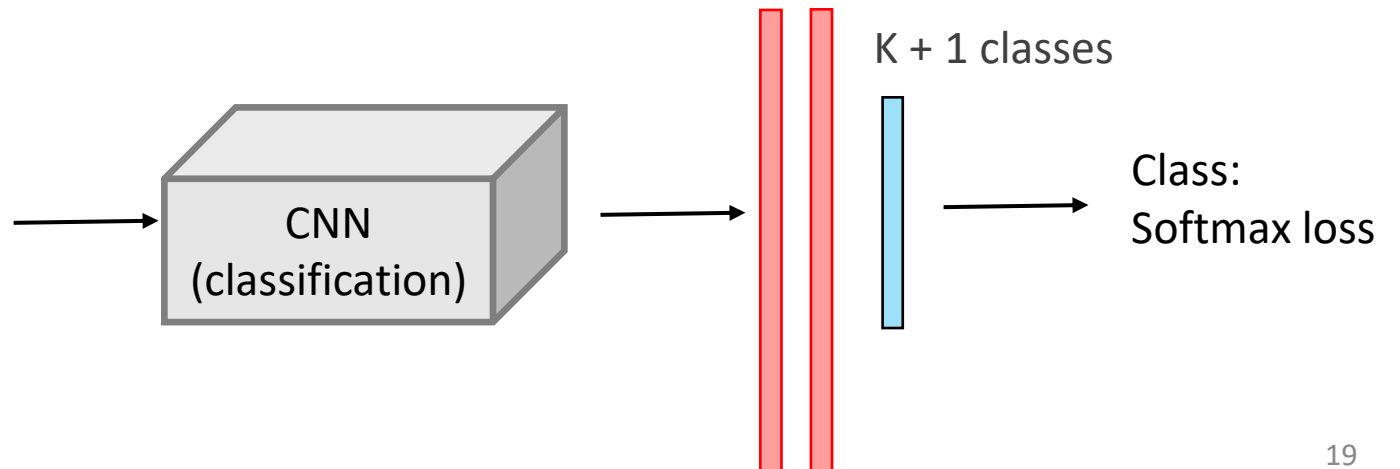
## Region proposal



Resize and  
warped

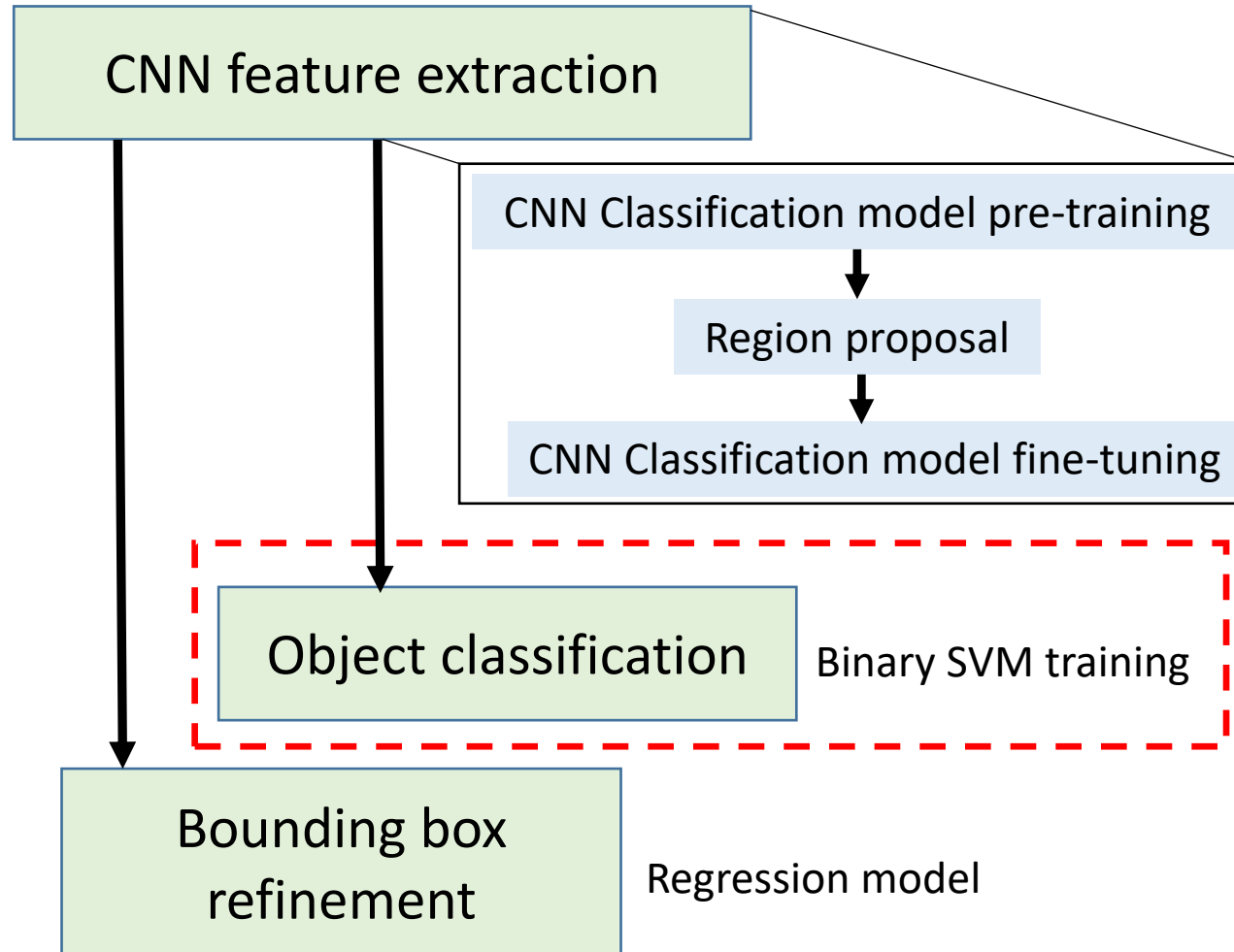


- The proposal regions are cropped out of the image, resize and warped to have a fixed size as required by CNN.
- Fine-tuning the CNN on the wrapped proposal regions for  $K + 1$  classes.
  - The additional one class refers to the background
- In the fine-tuning stage, we use a much smaller learning rate and the mini-batch oversamples the positive cases because most proposed regions are just background.





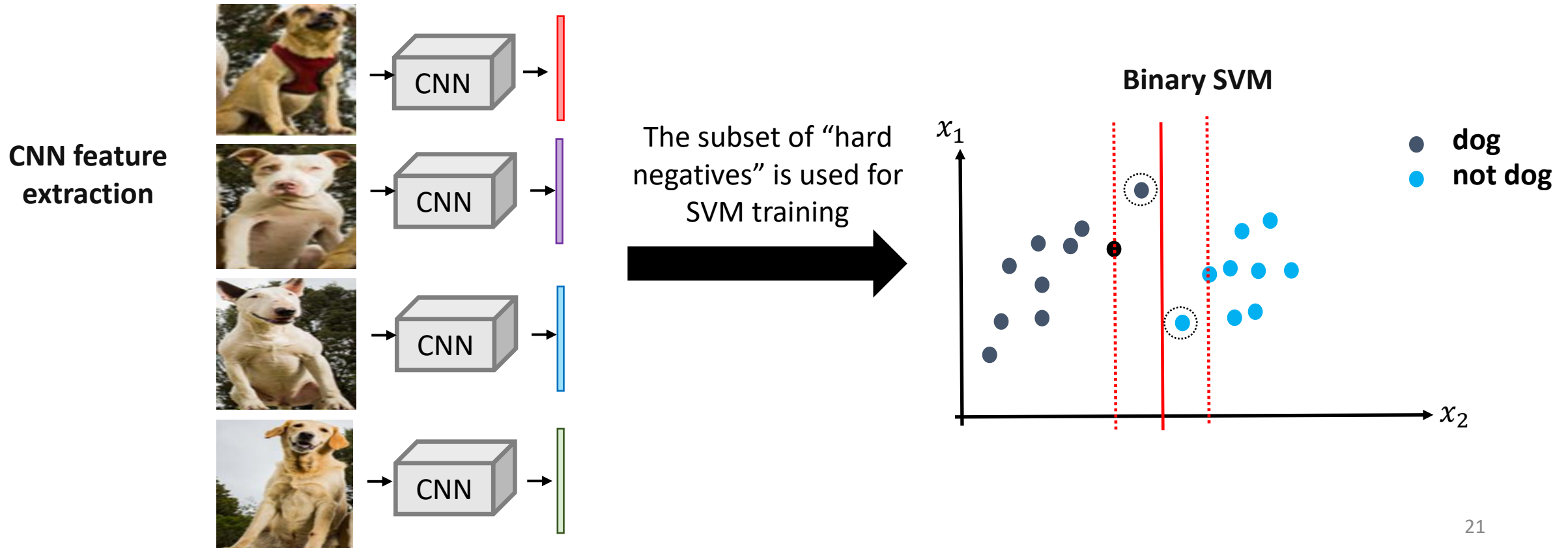
# Object classification



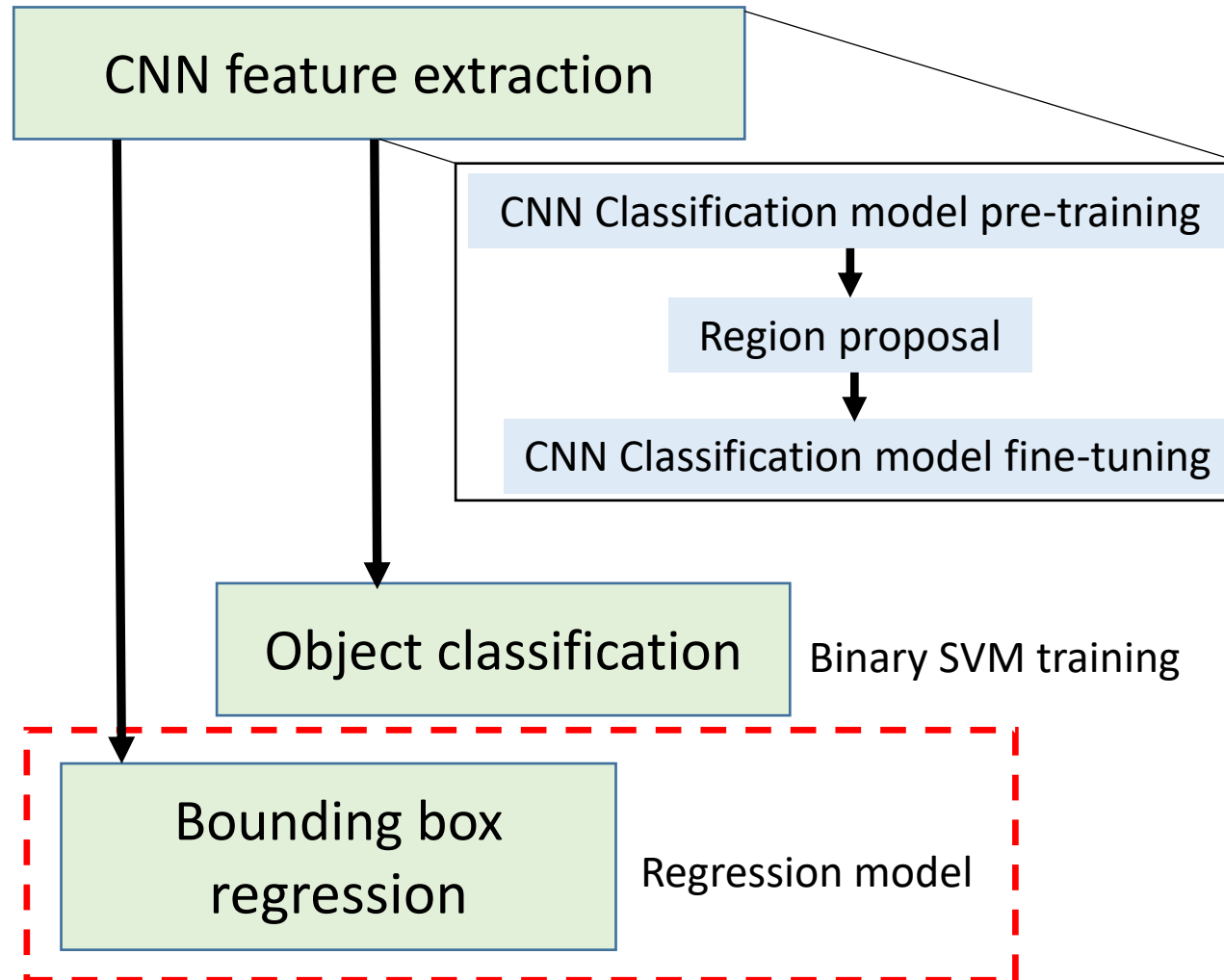


# Object classification

- Given every image region, one forward propagation through the CNN generates a feature vector.
- This feature vector is then consumed by a **binary SVM** trained for **each class** independently.
- The positive samples are proposed regions with IoU (intersection over union) overlap threshold  $\geq 0.3$ , and negative samples are irrelevant others.



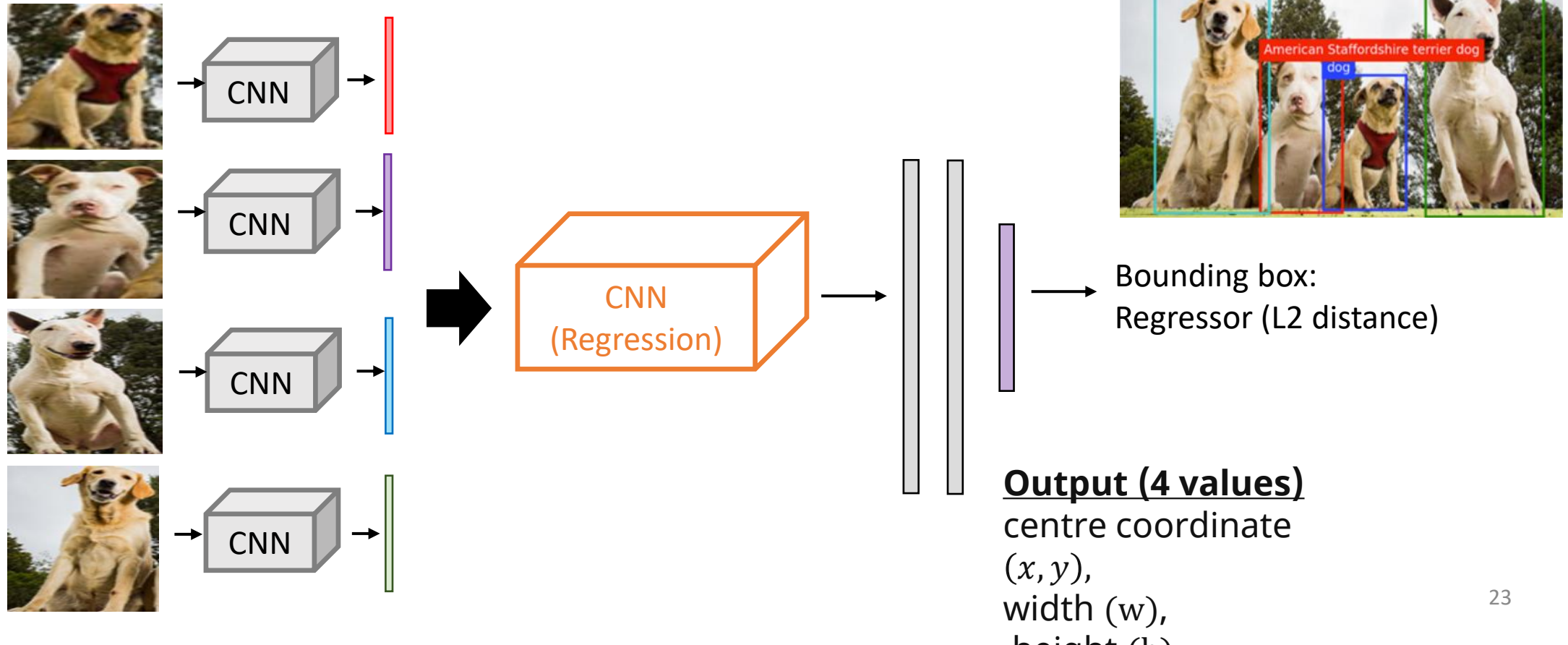
# Bounding box regression



# Bounding box regression

- Bounding box regression stage is to improve the localization performance. It predicts a new bounding box for the detection for each object class using the features computed by CNN.
- Only the predicted box with  $\geq 0.6$  is used to train the class-specific bounding box regressor.

## CNN feature extraction



# Bounding box regression

The bounding box regression is to adjust an initial, rough guess of an object's location (proposal box) to more accurately match its actual position and size (groundtruth box).

# Problems with RCNN

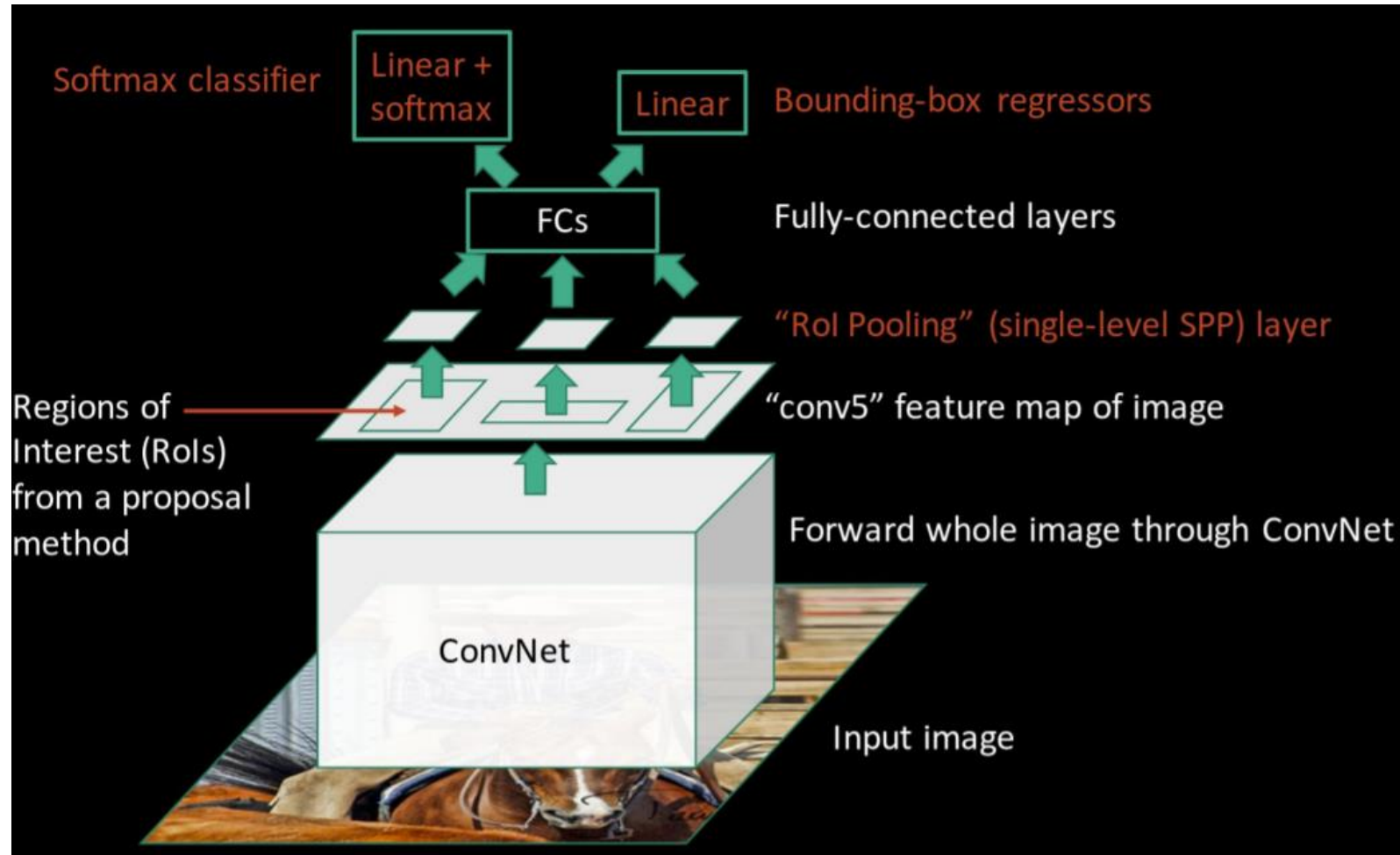
- **Slow at test-time**
  - Due to *independent forward passes* of the CNN.
  - Need to run full forward pass of CNN for *each region proposal*.
- **Post-hoc training of SVMs and regressors**
  - CNN features *not updated* in response to SVMs and regressors.
  - This could lead to the generation of bad candidate region proposals.
- **Complex multistage training pipeline**
  - Training is *expensive* and *slow*.
  - The whole process involves *three* models separately without much shared computation:
    - 1) CNN for image classification and feature extraction
    - 2) SVM classifier for object classification
    - 3) Regression model for bounding boxes correction

# Two-stage detector

- RCNN
- **Fast-RCNN**
- Faster-RCNN



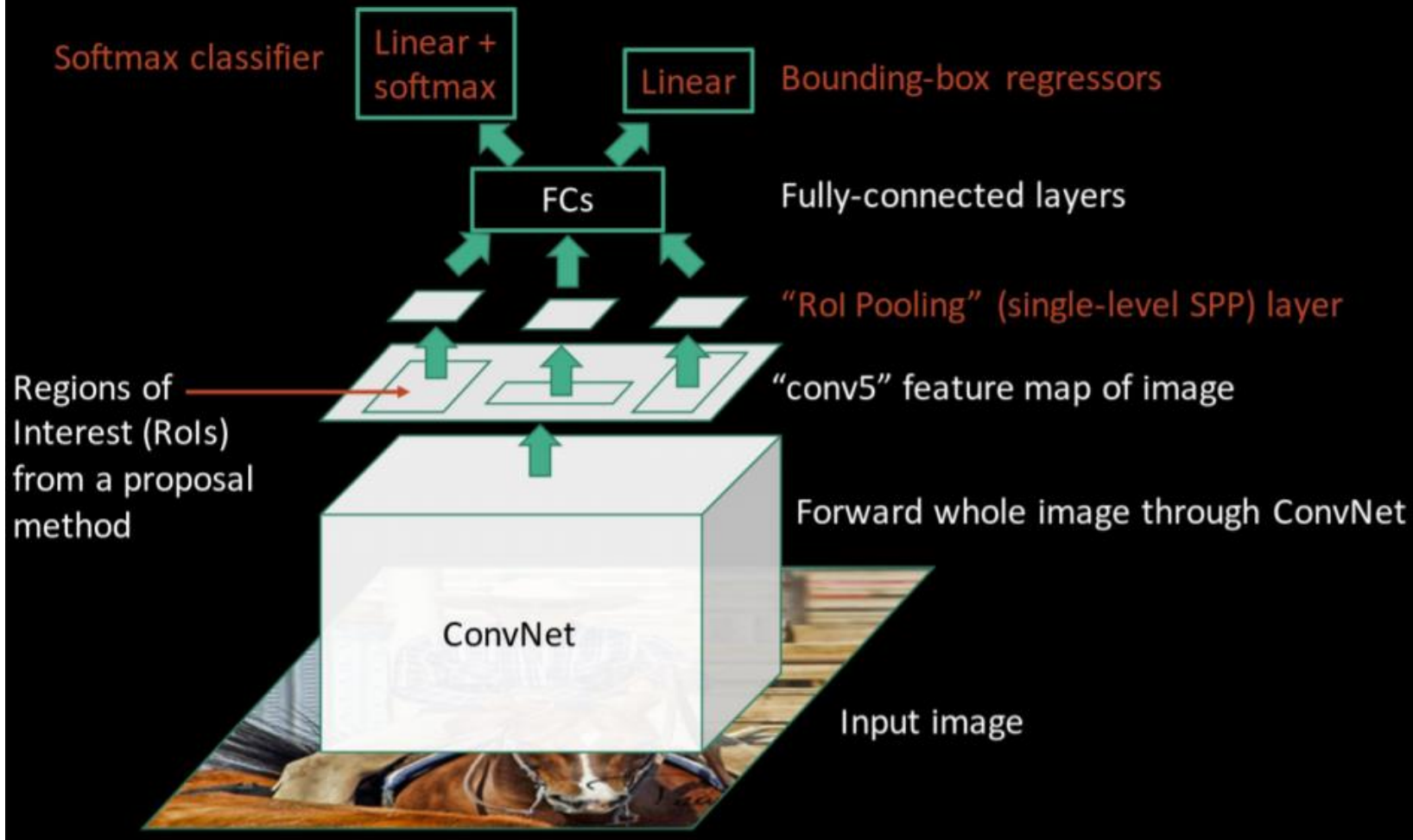
# Fast-RCNN



- Fast-RCNN makes R-CNN **faster** by improving the training procedure by **unifying three independent models** into one jointly trained framework and increasing shared computation results.

# Advantage of Fast-RCNN

## Fast R-CNN (test time)

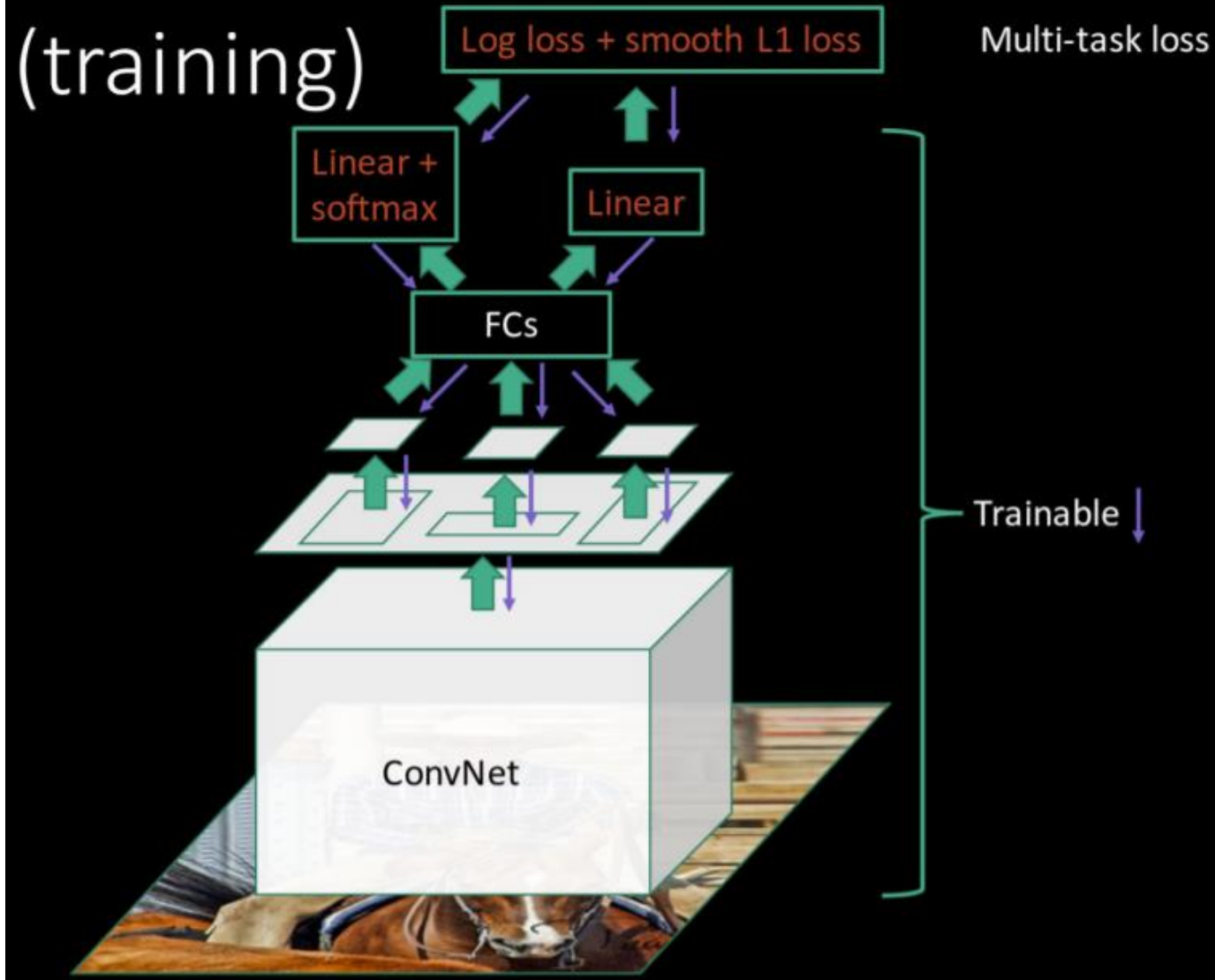


**R-CNN Problem #1:**  
Slow at test-time due to independent forward passes of the CNN

**Solution:**  
Share computation of convolutional layers between proposals for an image

# Advantage of Fast-RCNN

## Fast R-CNN (training)



### R-CNN Problem #2:

Post-hoc training: CNN not updated in response to final classifiers and regressors

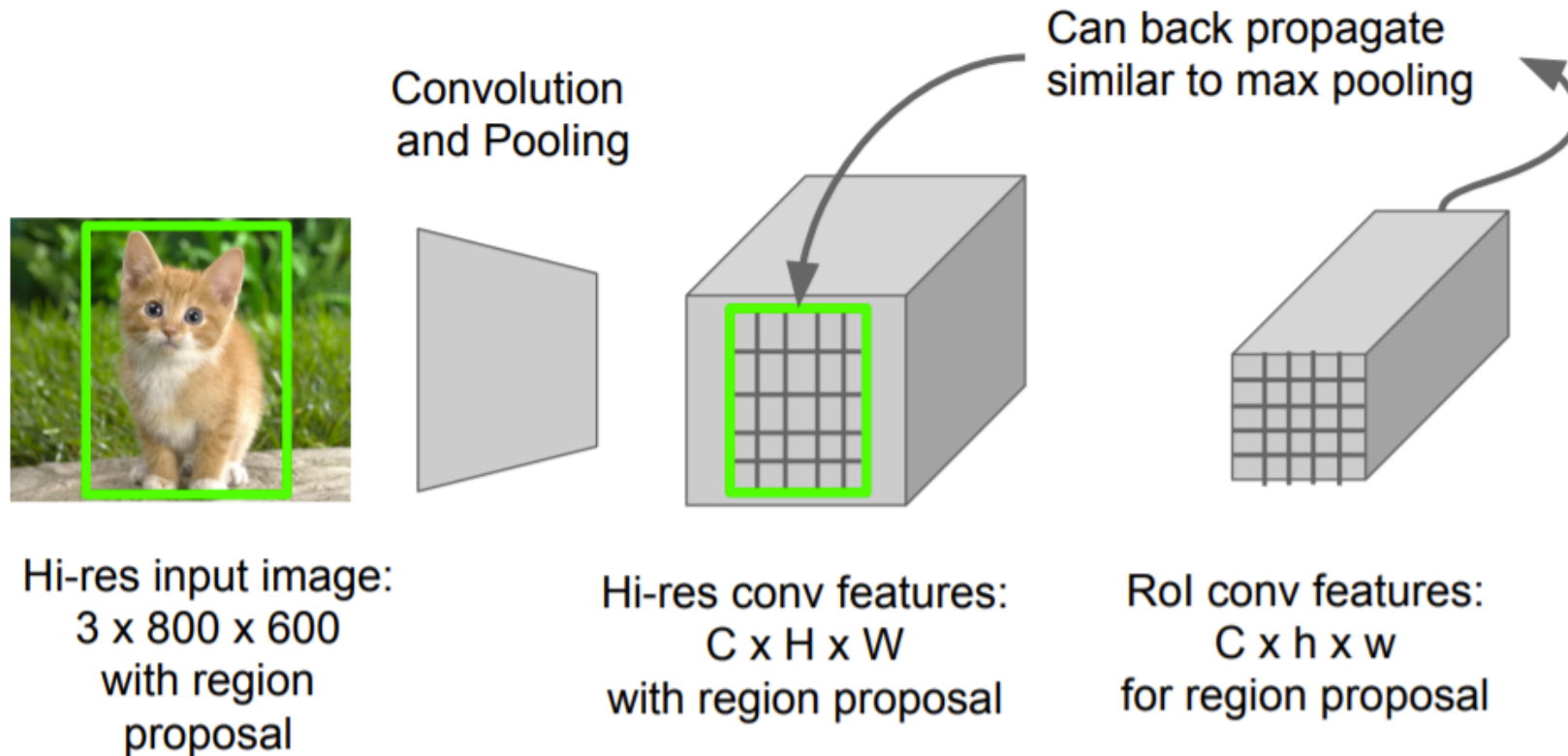
### R-CNN Problem #3:

Complex training pipeline

### Solution:

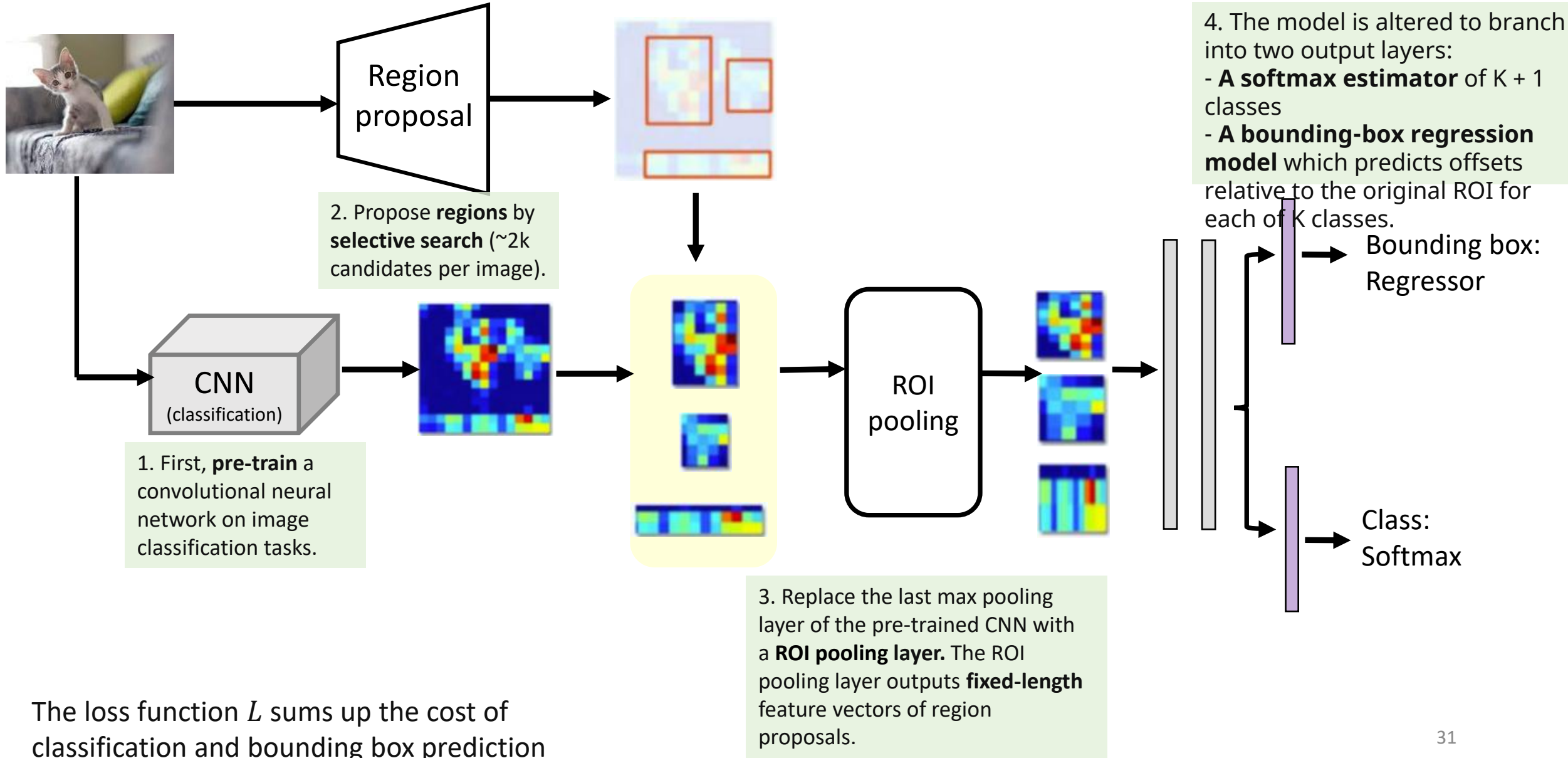
Just train the whole system end-to-end all at once!

# ROI pooling



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

# Model workflow



# Speed Bottleneck

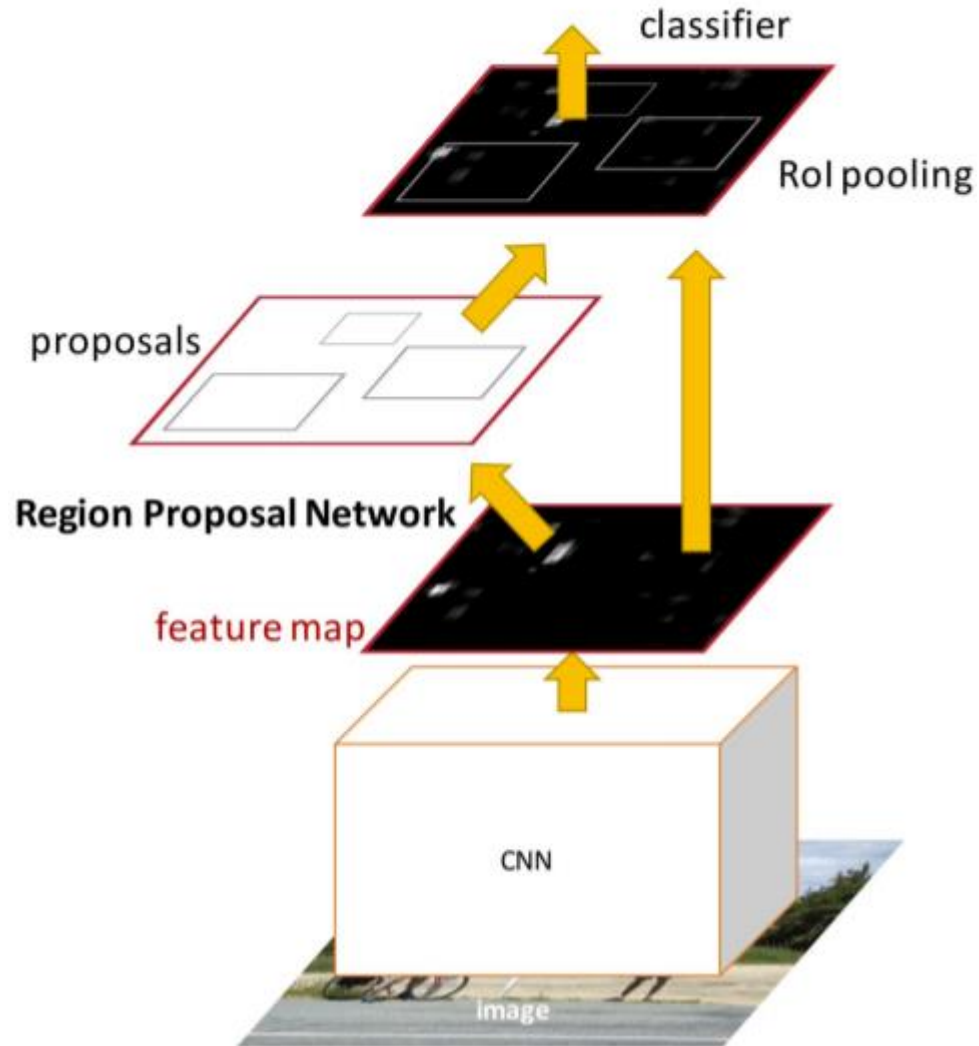
- Fast R-CNN is much faster in both training and testing time as compared to R-CNN.
- However, the improvement is not dramatic because the region proposals are generated separately by another model and that is very expensive.



# Two-stage detector

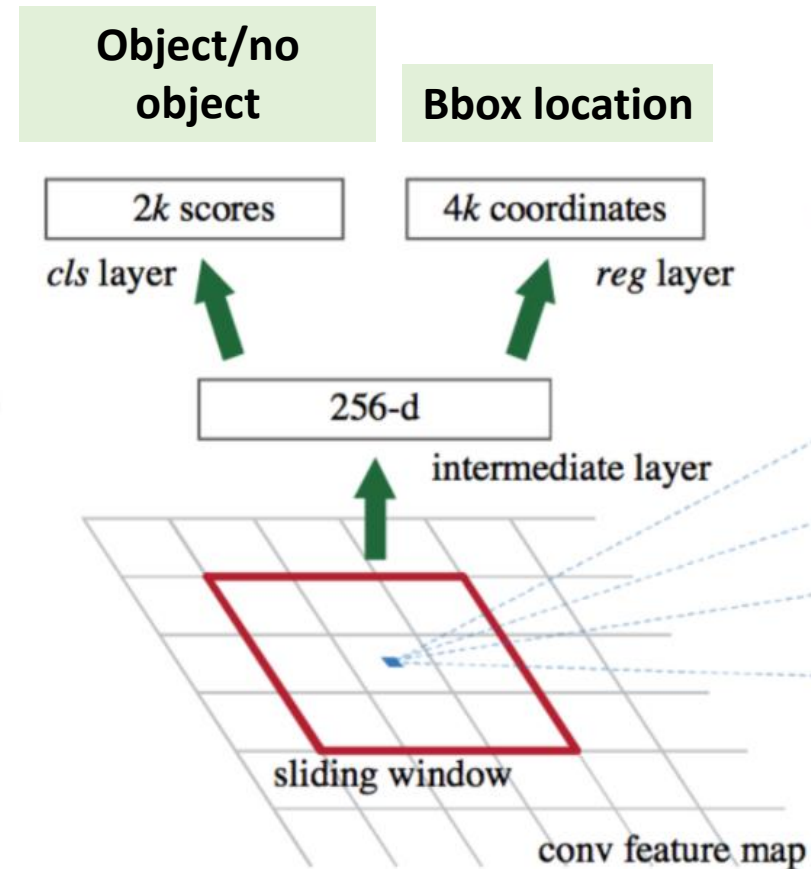
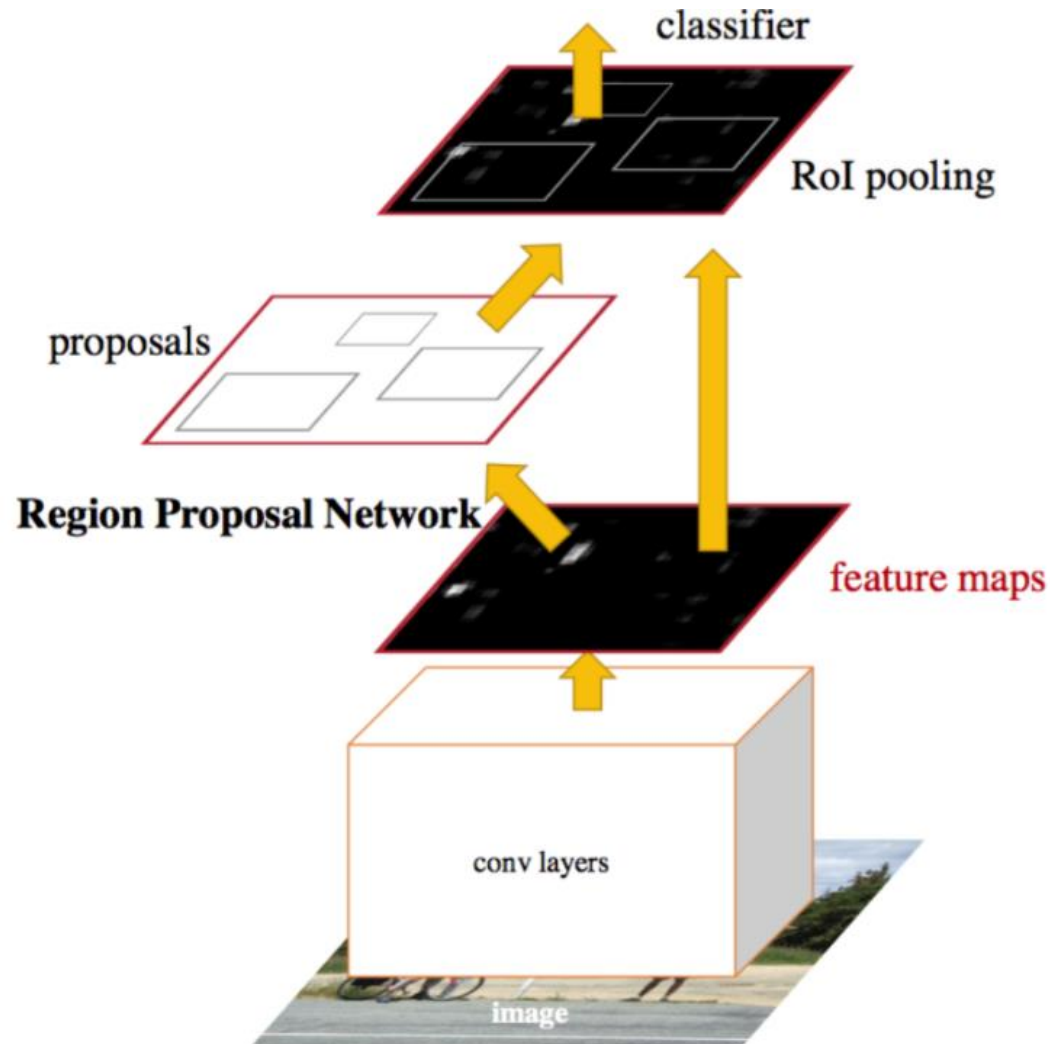
- RCNN
- Fast-RCNN
- **Faster-RCNN**

# Faster R-CNN

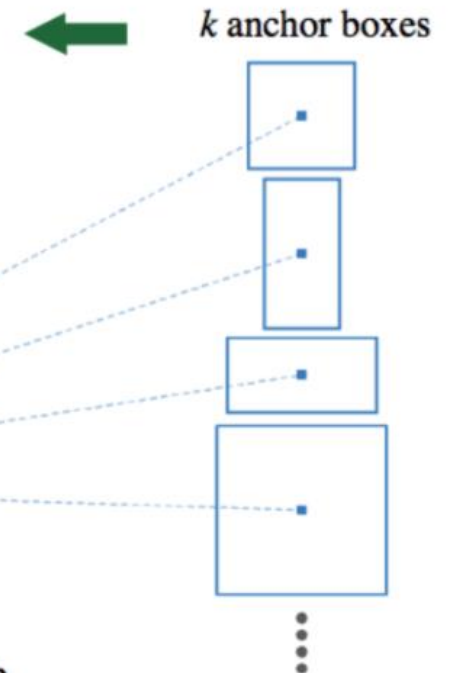


- Faster R-CNN inserts a **Region Proposal Network (RPN)** after the last convolutional layer.
- The RPN is trained to produce region proposals directly (no need for external region proposals)
- After RPN, it uses ROI Pooling and an upstream classifier and bounding box regressor just like Fast R-CNN.

# Region Proposal Network (RPN)

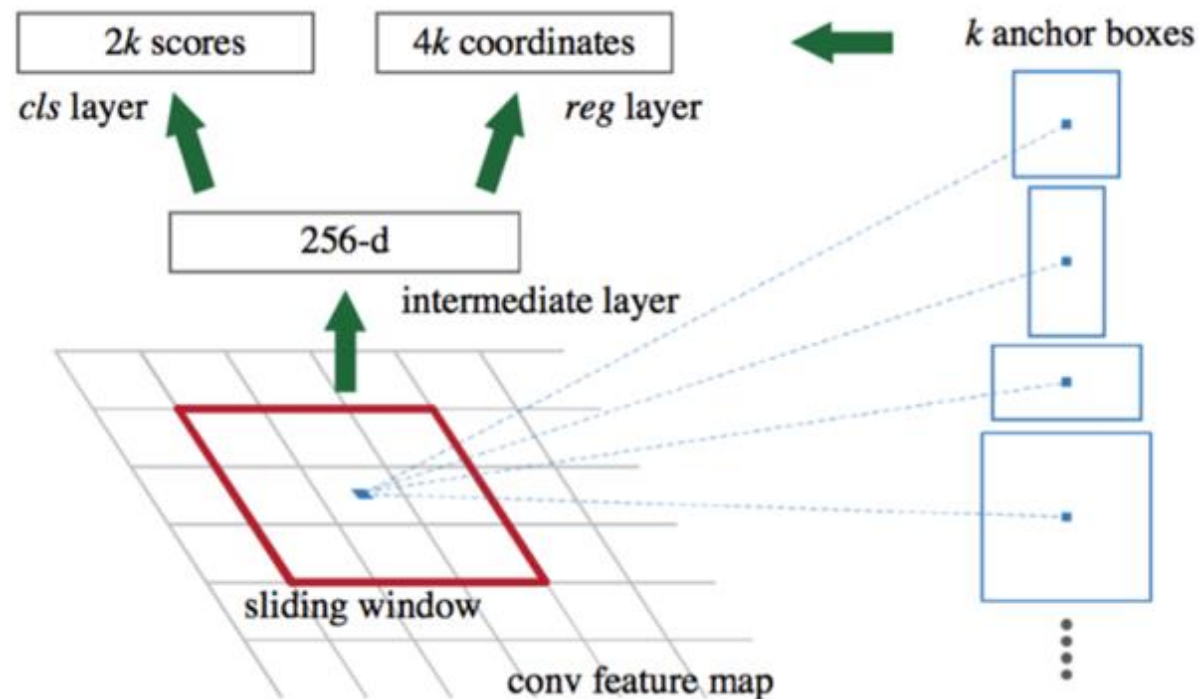


**Anchors** are translation invariant: use the same ones at every location



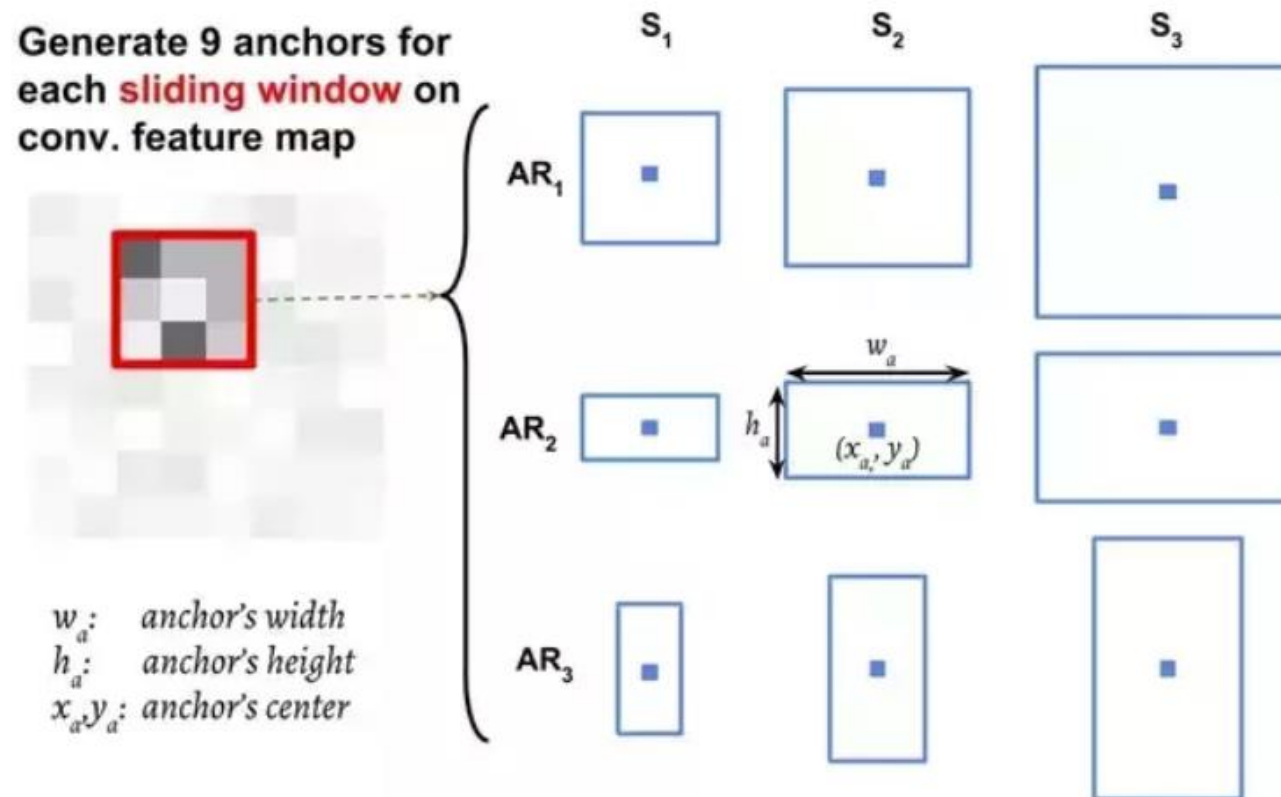
# Region Proposal Network (RPN)

- The RPN takes all the anchor boxes as input and then generates the object score for each box and performs regression to find a more accurate boundary box.
- The **Anchor Boxes** are predefined and fixed-size boxes, which are spread over the entire image. These boxes come in different sizes and different aspect ratios. Different sizes and ratios help to capture different types of objects.



# Region Proposal Network (RPN)

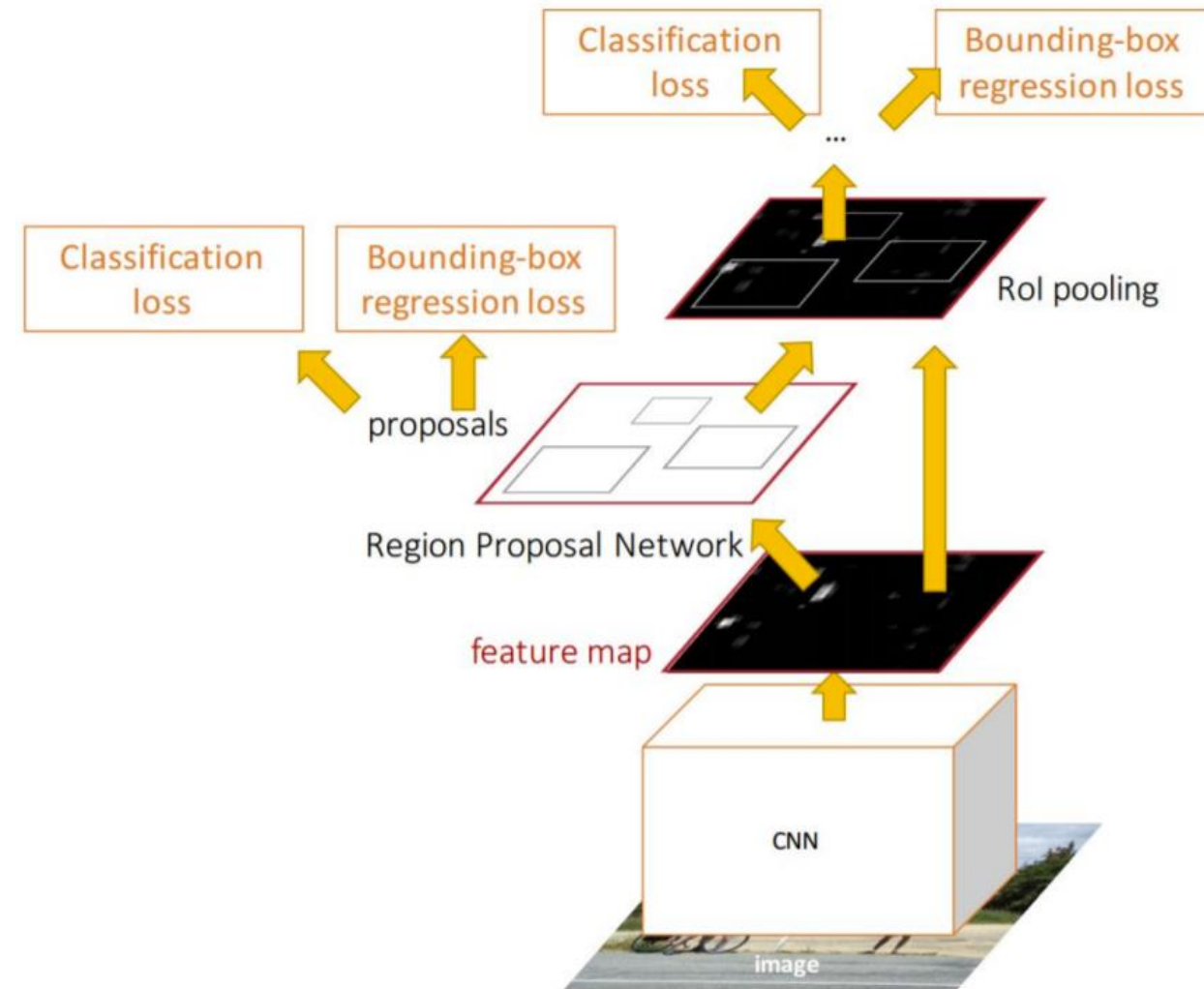
- RPN works on the feature map (output of CNN), and each feature of this map is called **Anchor Point**. For each anchor point, we place 9 anchor boxes (the combinations of different sizes and ratios) over the image. These anchor boxes are centered at the point in the image which is corresponding to the anchor point of the feature map.



# Training of RPN

- So for training, we assign a label to each anchor box, based on its Intersection over Union (IoU) with given ground truth. We basically assign either of the three (1, -1, 0) labels to each anchor box.
- After assigning the labels, it creates the mini-batch of 256 randomly picked anchor boxes, all of these anchor boxes are picked from the same image.
- The ratio of the number of positive and negative anchor boxes should be 1:1 in the mini-batch, but if there are less than 128 positive anchor boxes then we pad the mini-batch with negative anchor boxes.
- Now the RPN can be trained end-to-end by backpropagation and stochastic gradient descent (SGD).

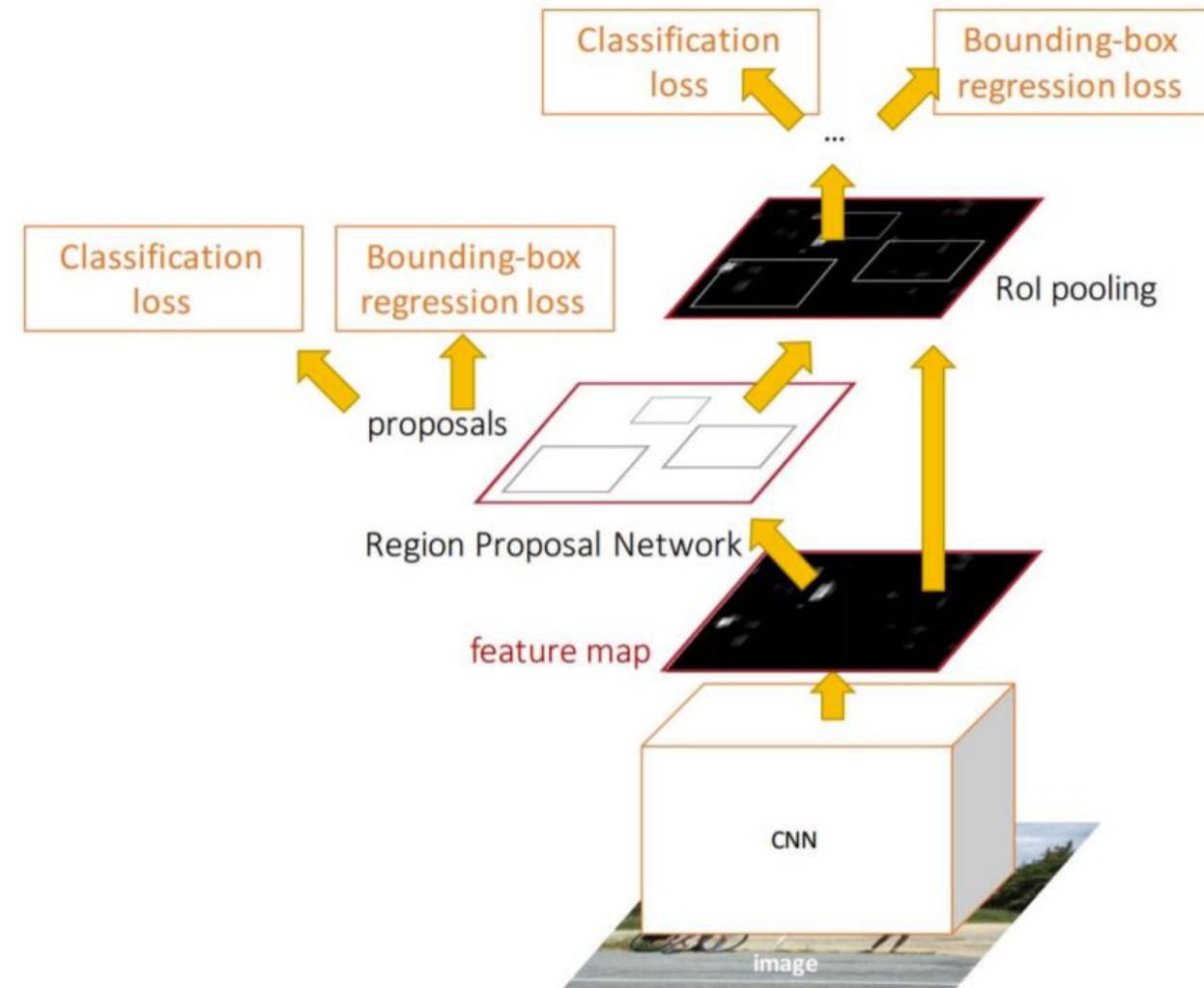
# Training of Faster-RCNN



1. First, we **train** the end to end **RPN**, where the CNN is been initialized by a pre-trained model, which has been trained on Image Net dataset. This RPN training fine-tune the weights end to end for the region proposal task.
2. In the second step, we **train** a separate **Fast R-CNN**, here also it initialized the CNN by pre-trained model (trained on Image Net). Till now there is no sharing of convolution layers between RPN and Fast R-CNN.
3. Now we **initialized** the RPN from the weights of Fast R-CNN, after initialization we freeze (exclude from training) the shared convolution layers and **fine-tune** the layers which are specific to the **RPN**.
4. In the final step we keep the shared convolution layers frozen and **fine-tune** the layers specific to **Fast R-CNN**.



# Training of Faster-RCNN



- In the paper: Ugly pipeline
  - Use **alternating optimization** to train RPN, then Fast R-CNN with RPN proposals, etc. - More complex than it has to be.
- Since publication: Joint training!
  - One network, four losses
    - RPN classification (anchor good / bad)
    - RPN regression (anchor -> proposal)
    - Fast R-CNN classification (over classes)
    - Fast R-CNN regression (proposal -> box)

# One-stage detector

- **YOLO**
- **SSD**

# YOLO: You Only Look Once

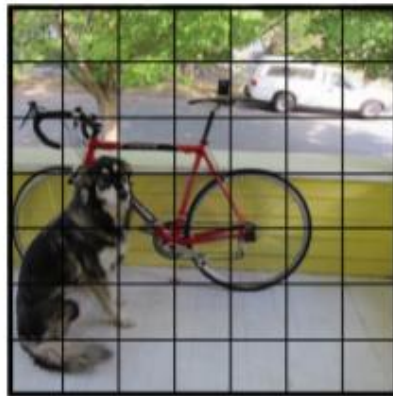
- The **YOLO** model is the very **first attempt** in building a **fast real-time** object detector.
- YOLO super fast, but it is not good at recognizing irregularly shaped objects or a group of small objects due to a limited number of bounding box candidates (or we can call them anchor boxes).

Redmon, J., Divvala, S., Girshick, R. and Farhadi, A., 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).

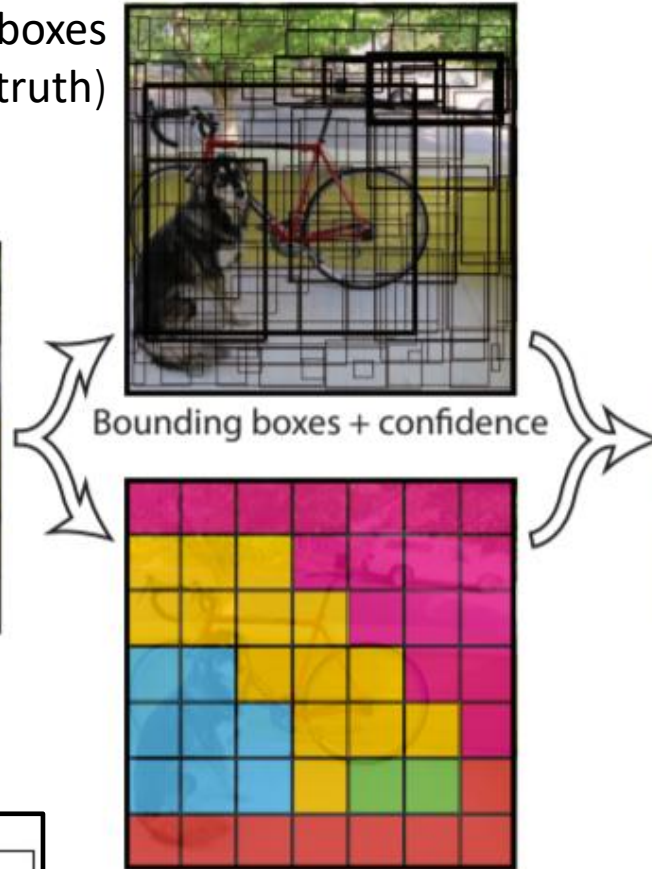
# YOLO

$S \times S \times B$  bounding boxes  
 $Confidence = P(\text{object}) \times IOU(\text{pred}, \text{truth})$

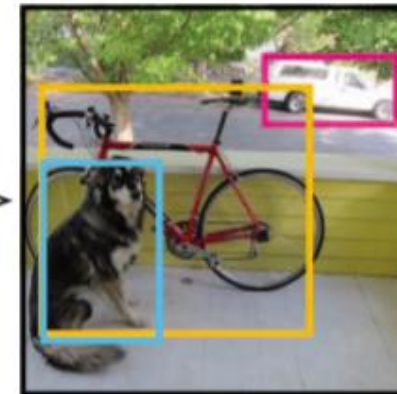
- Divides the image into an  $S \times S$  grid



$S \times S$  grid on input

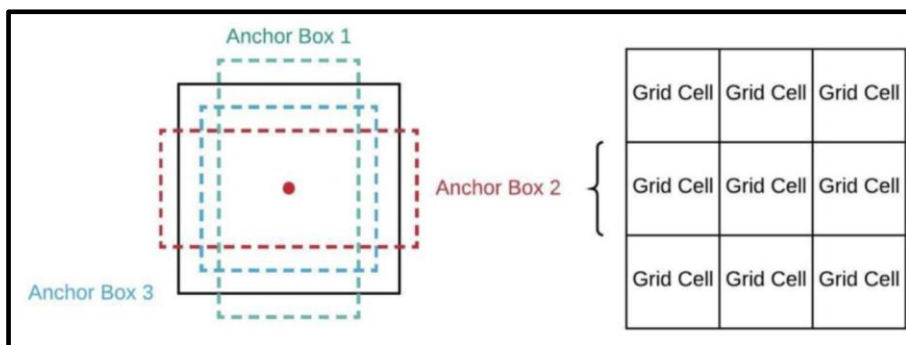


Class probability map



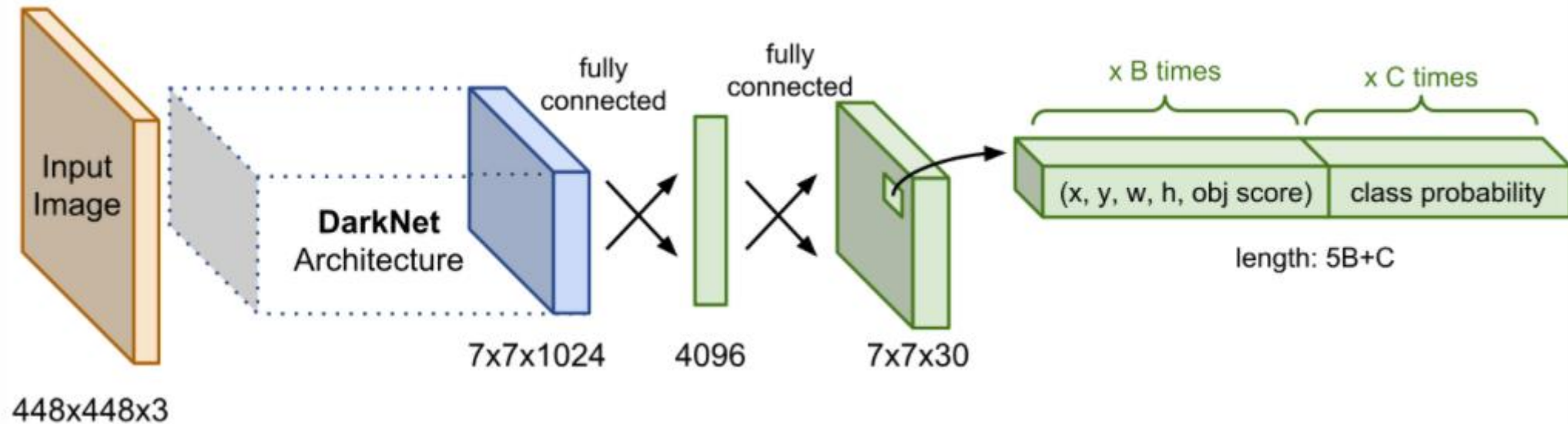
Final detections

- The total prediction values for one image is  $S \times S \times (B * 5 + C)$ , which is the tensor shape of the final conv layer of the model.
- Each bounding box consists of 5 predictions:  $x, y, w, h$ , and confidence



- For each grid cell predicts  $B$  bounding boxes (anchor boxes), confidence for those boxes, and  $C$  class probabilities

# Network architecture



- The base model is similar to GoogLeNet with inception module replaced by  $1 \times 1$  and  $3 \times 3$  conv layers. The final prediction of shape  $S \times S \times (B * 5 + C)$  is produced by two fully connected layers over the whole conv feature map.
- For example:  $S = 7$ ,  $B = 2$  with  $C = 20$  for 20 labelled classes. The final prediction is a  $7 \times 7 \times 30$  tensor.

# YOLO vs Faster-RCNN

- YOLO offers a significant increase in speed over Faster-RCNN which is 45 fps versus 7 fps.
- However, YOLO faces a decrease in detection accuracy compared to Faster-RCNN which is 63.4% versus 73.2%.

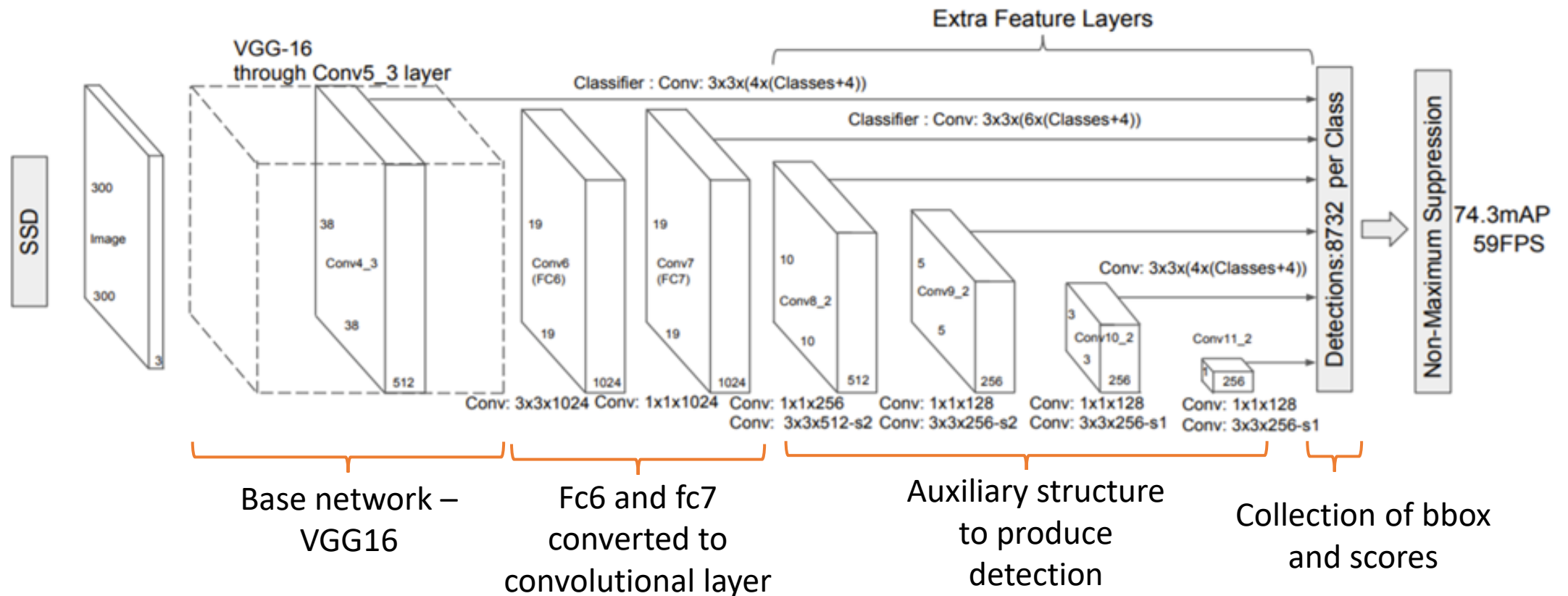
# One-stage detector

- YOLO
- **SSD**



# Single Shot Detector (SSD)

- The Single Shot Detector (SSD) is one of the first attempts at using convolutional neural network's *pyramidal feature* hierarchy for efficient detection of objects of various sizes.
- The SSD network ran both faster and had superior performance to YOLO

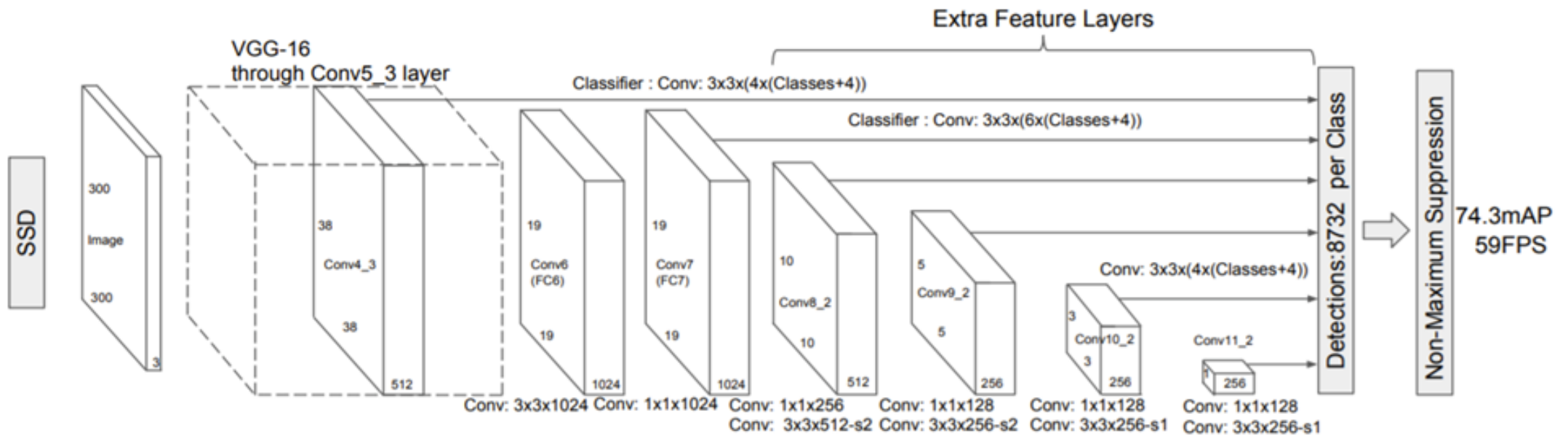


# Multi-scale feature maps for detection

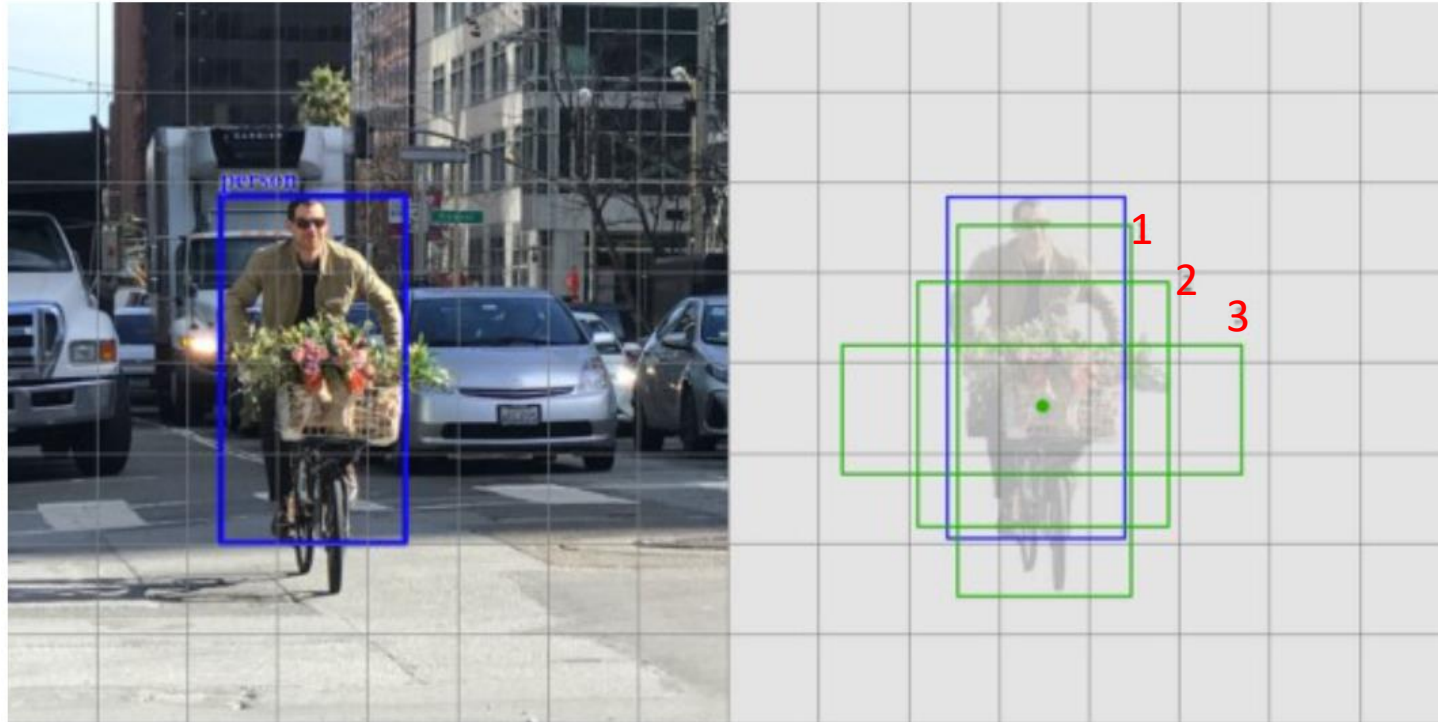
- SSD uses the VGG16 model pre-trained on ImageNet as its base model for extracting useful image features.
- On top of VGG16, SSD adds several conv feature layers of decreasing sizes in which allow predictions of detecting at multiple scales.
  - YOLO use only single scale feature map
- Intuitively large fine-grained feature maps at earlier levels are good at capturing small objects and small coarse-grained feature maps can detect large objects well.

# Convolutional predictors for detection

- For a feature map of size  $m \times n$  with  $p$  channels the basic elements for predicting parameters of a potential detection is a  $3 \times 3 \times p$  small kernel that produces either a score for a category, or a shape offset relative to the default box coordinates
  - YOLO has a fully connected layer for a score and a bbox coordinates.

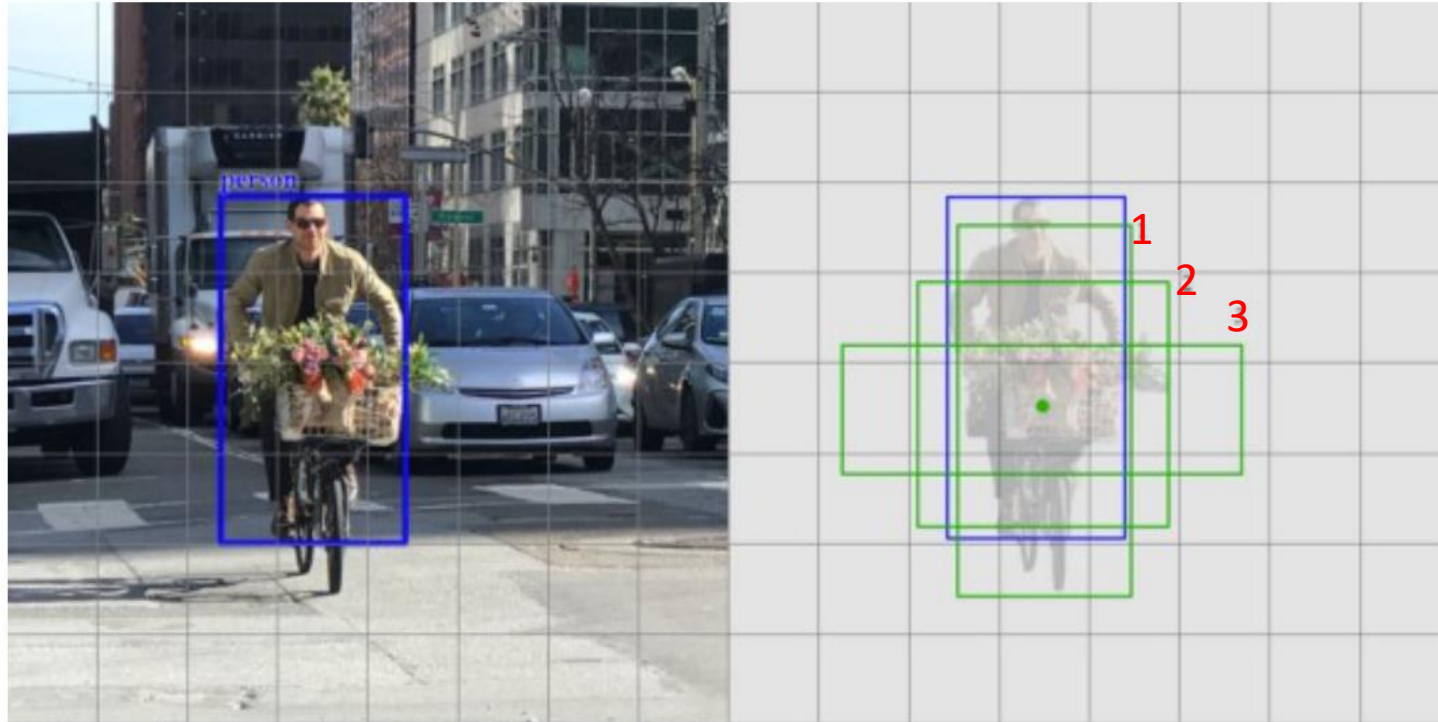


# Matching strategy



- SSD predictions are classified as positive matches or negative matches.
- SSD only uses positive matches in calculating the localization cost  $L_{loc}$
- If the corresponding **default boundary box** (not the predicted boundary box) has an  $IoU$  greater than 0.5 with the ground truth, the match is positive. Otherwise, it is negative.

# Matching strategy



- In the above example, only box 1 and 2 are positive matches.
- Once we identify the positive matches, we use the corresponding predicted boundary boxes to calculate the cost.
- This matching strategy nicely partitions what shape of the ground truth that a prediction is responsible for.

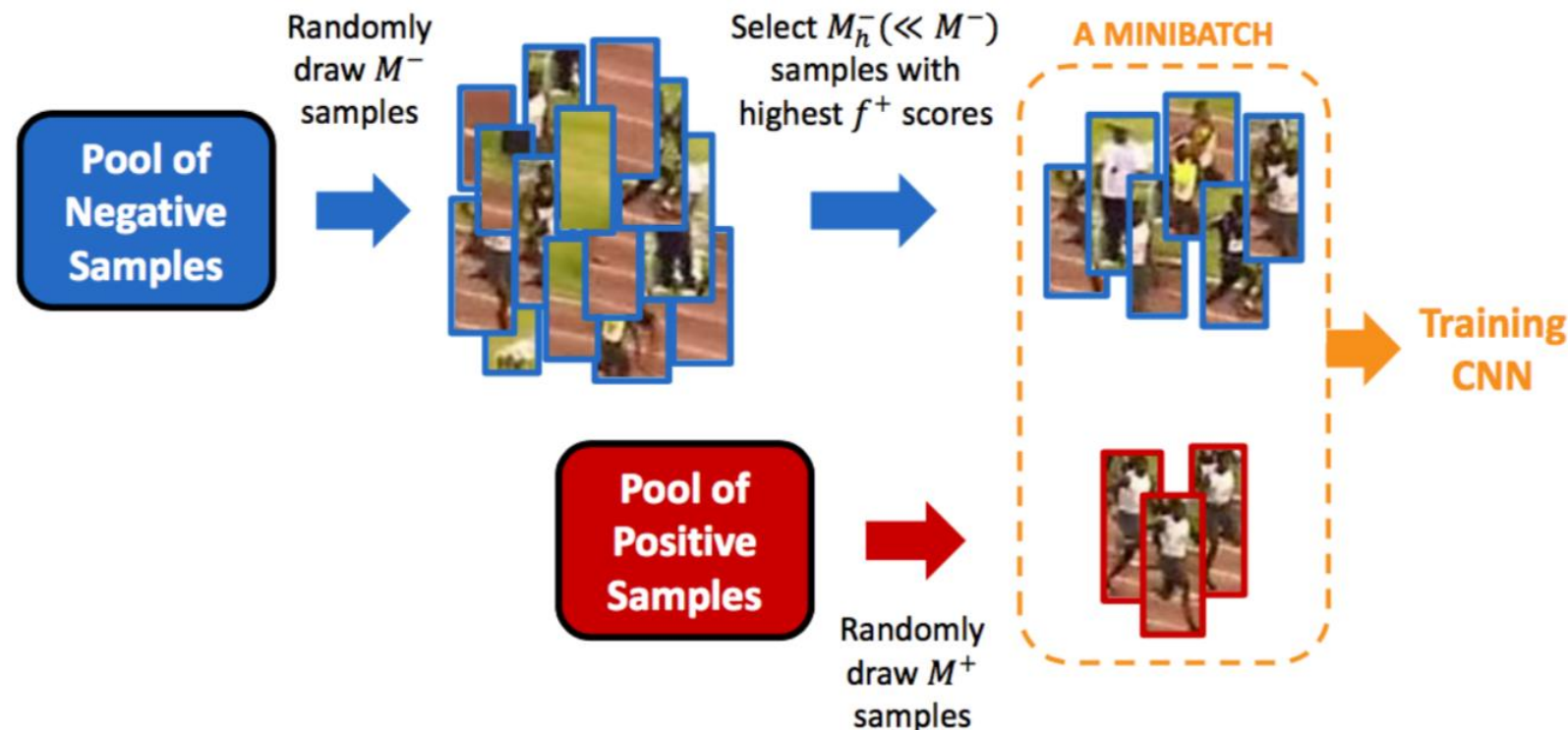
# Object detection common tricks

- Several tricks are commonly used in RCNN and other detection models.
  - Hard negative mining
  - Non-Maximum suppression



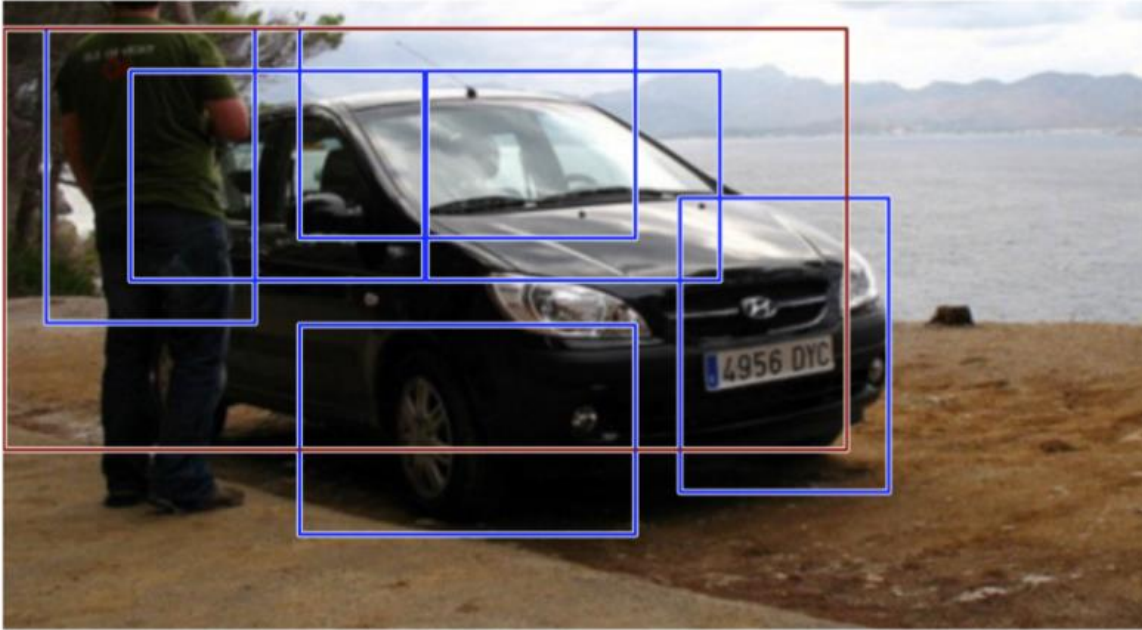
# Hard negative mining

- Not all the negative examples are equally hard to be identified. For example, if it holds pure **empty background**, it is likely an “**easy negative**”
- If the box contains weird noisy texture or **partial object**, it could be hard to be recognized and these are “**hard negative**”.
- We can explicitly find those false positive samples during the training loops and include them in the training data so as to improve the classifier.





# Non-max suppression (NMS)



Before non-max suppression



After non-max suppression

- The purpose of non-max suppression is to select the best bounding box for an object and reject or “suppress” all other bounding boxes.
- The NMS takes two things into account. The confidence score given by the model and the IOU of the bounding boxes.

# Next lecture

## ❖ Face recognition with Deep Learning

