



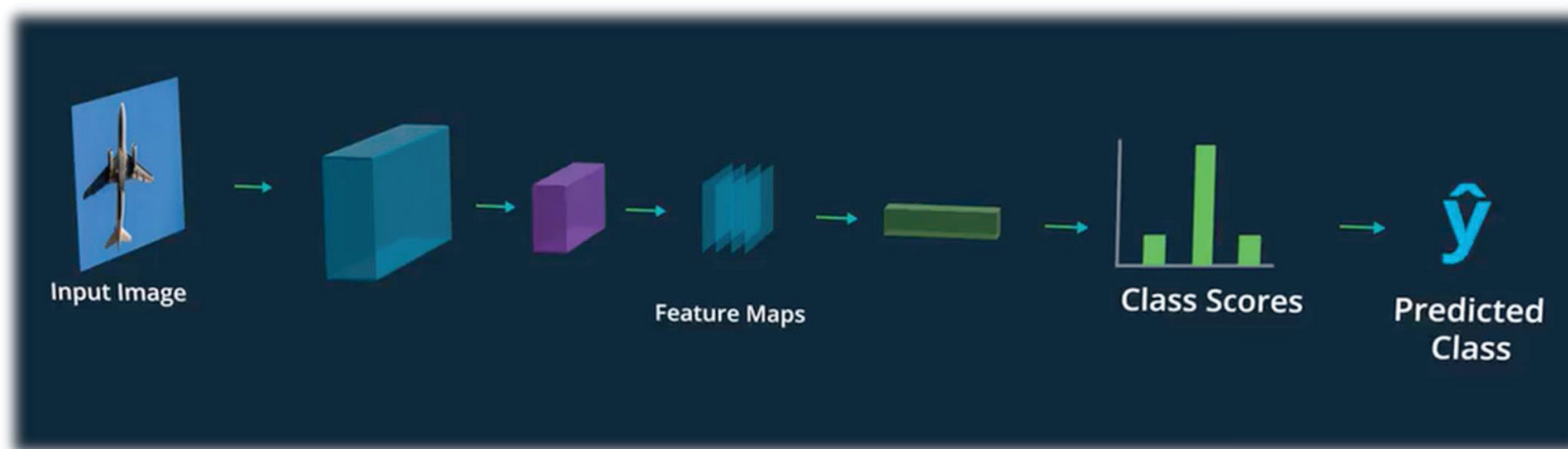
# Review

- Neural Network
  - Gradient Descent
  - Back propagation
- Convolutional Neural Network
  - Convolutional layers
  - Pooling
  - Transfer Learning



## Section 1 – Object Detection

# Review



The network we learned is to identify whether the image contains vehicle/building/pedestrian etc.



# Problems in Real World

- Only identifying an image is not quite useful for urban analytics
- Usually, we are more interested in the size, location, counts, types, and others.
- There are typically three types of CV problems that are useful in urban analytics
  - Object detection
  - Semantic segmentation
  - Instance segmentation

Typical  
Image

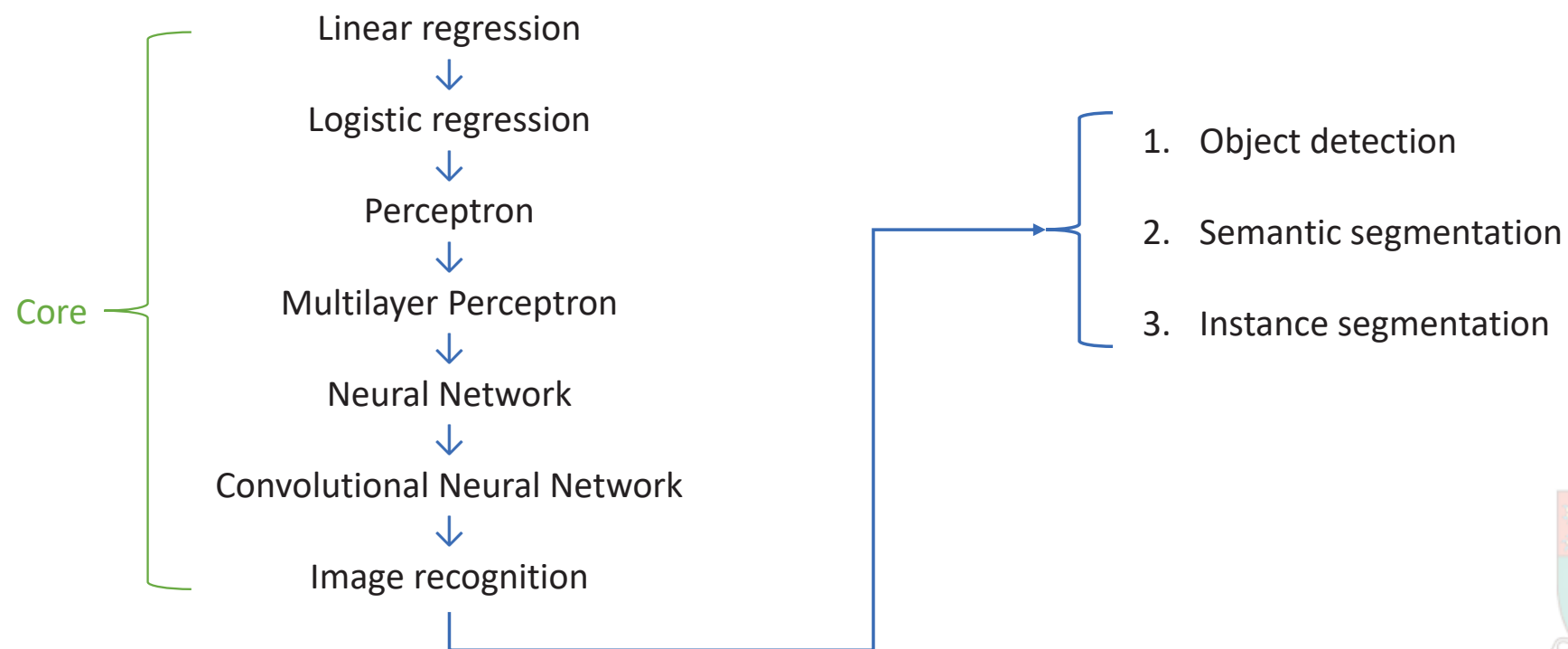


Real World  
Scene

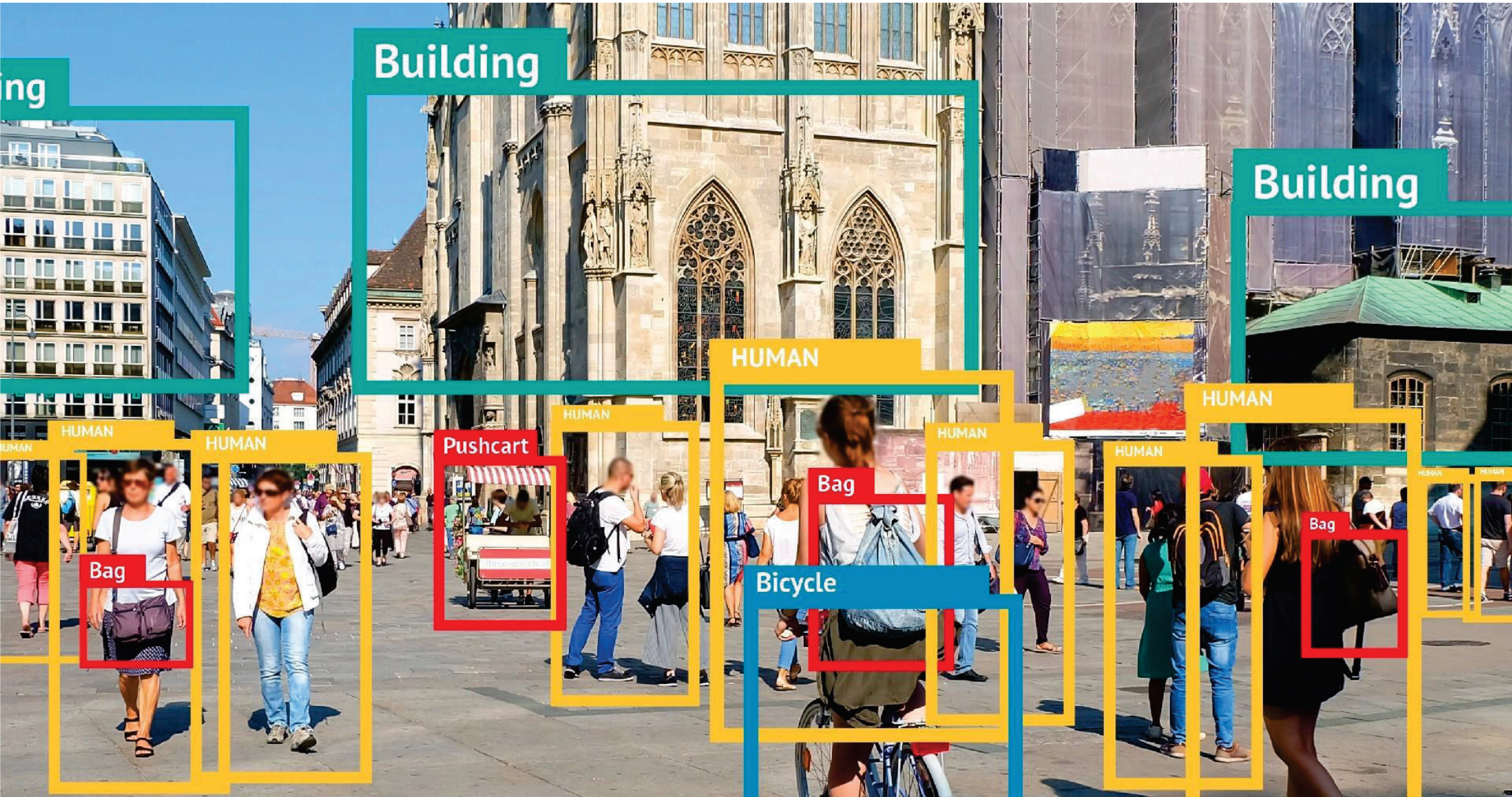




# The learning milestones







Object Detection





Semantic Segmentation





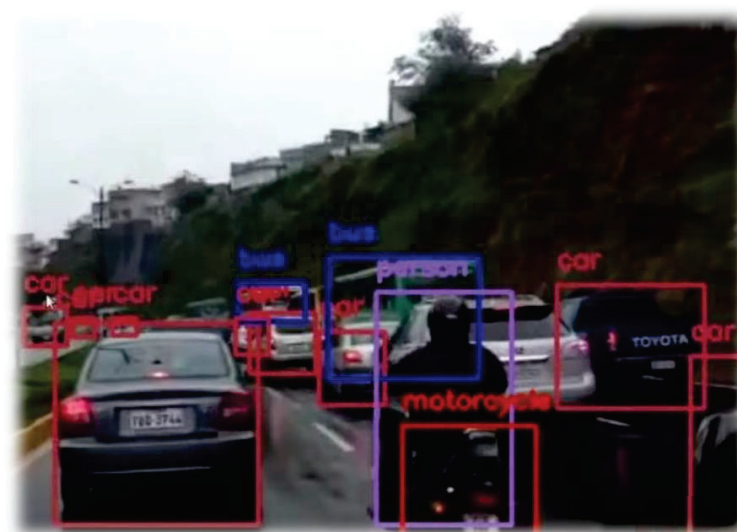
Instance Segmentation





# Object Detection

- Now let's start from the first and the most typical problem.
- Object detection is a computer technology related to computer vision and image processing that deals with **detecting instances of semantic objects of a certain class** (such as humans, buildings, or cars) in digital images and videos.
- Typical methods and networks
  - Faster R-CNN
  - YOLO





# Applications

- Parking cars
- Detecting vehicle speed
- Traffic violation
- Safe driving (eye bowl/face positions)
- Analyze pedestrian flow
- Analyze urban landscape/tree planning from satellite images

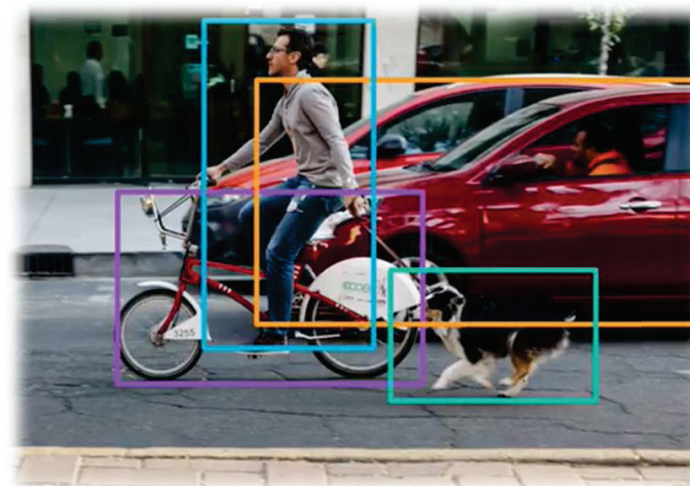
Key word: identify, count and location





# The Basic Method

- Faster RCNN (region based CNN)
  - Analyze different cropped areas of a single input image,
  - Decides which regions correspond to objects and then perform classification





# Localization of the objects in Image

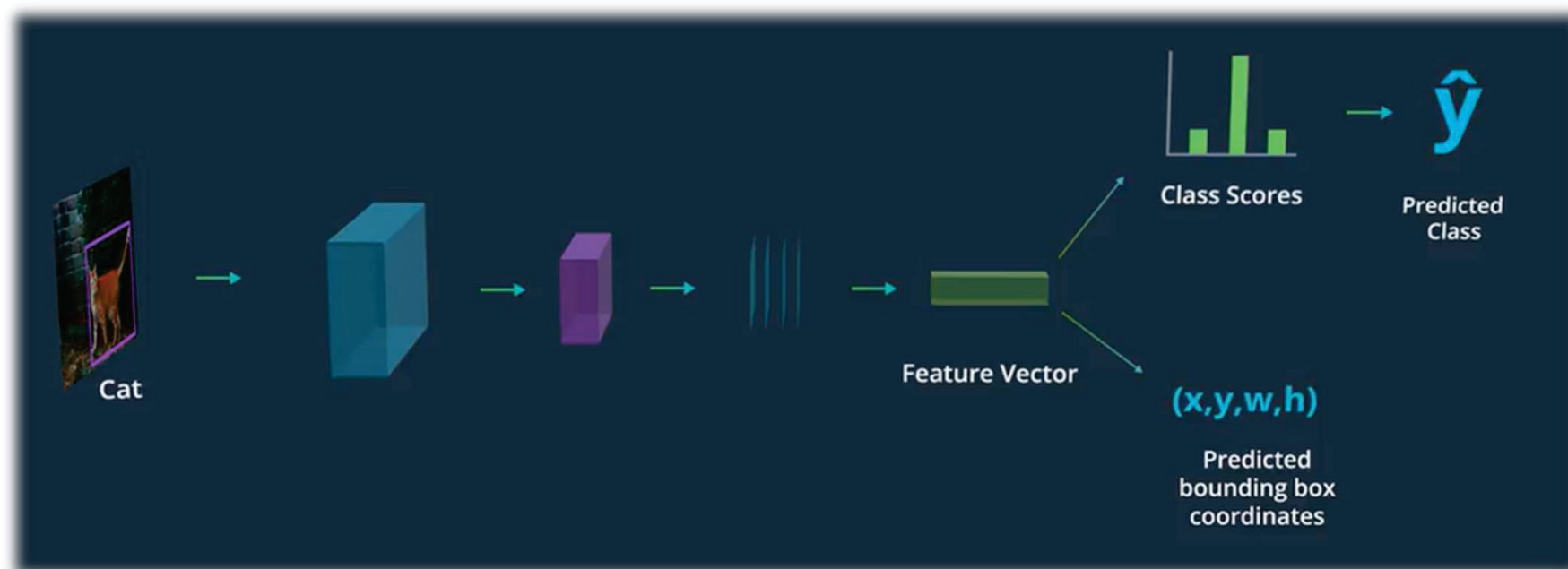
- Find the location of the cat in the image
- **Add a bounding box**
- Train the classifier to know the location of the object







# An easy-to-think strategy relies fully on CNN



Simply add more targets or dimensions in the outputs





# An easy-to-think strategy


- Prepare **the training datasets** first
  - If no existing datasets, you may have to do it manually
  - The “true bounding box”
- Train one model for the five targets
- Or train five model for the five targets
  - If cat
  - W
  - H
  - Location X
  - Location y
- Not a difficult task as we are predicting 10 targets in the fasion MNIST example







# An easy-to-think strategy

- Five targets
  - If cat
  - $W$
  - $H$
  - Location  $X$
  - Location  $y$
-  Another issue comes out, how to define the loss/error function.
- In typical CNN, we usually have one target/categorical targets. We can use accuracy/cross entropy/SSE, etc.
- Which should have higher weights?  $W$ ,  $H$ , location  $x$ , or location  $y$ ?





# A new type of loss measurement should be added

- We need a new type of measurement to calculate the “closeness” to the truth.
- Examples:
  - L1 loss of the center positions
  - MSE of the area of the box
  - Class labels
  - Or weighted sum
    - $0.5 * \text{cross\_entropy\_loss} + 0.5 * \text{L1\_loss}$



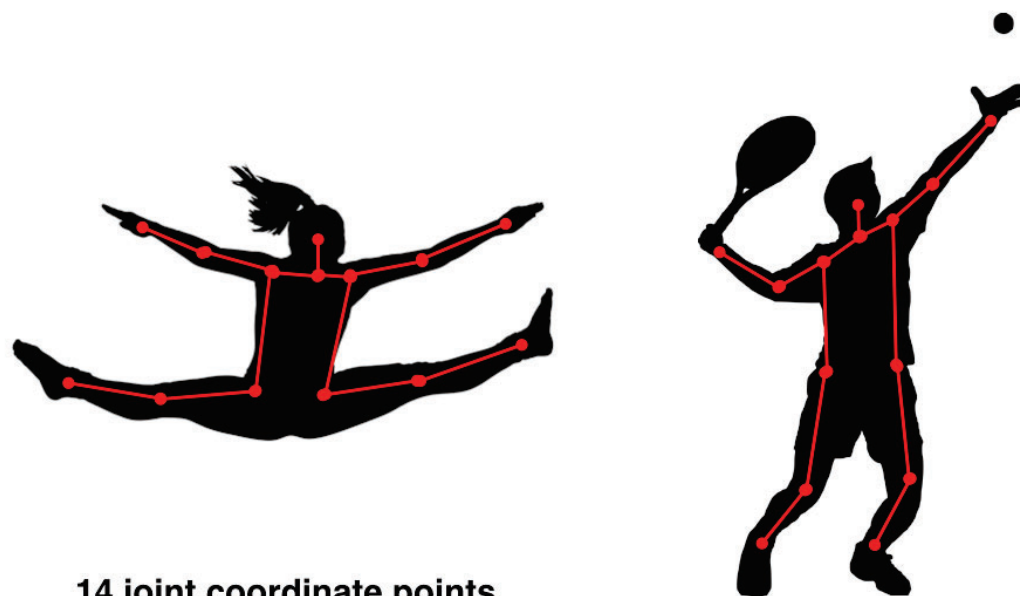
Green: truth  
Red: predicted







# A more complicated case but good for understanding (human pose estimation)

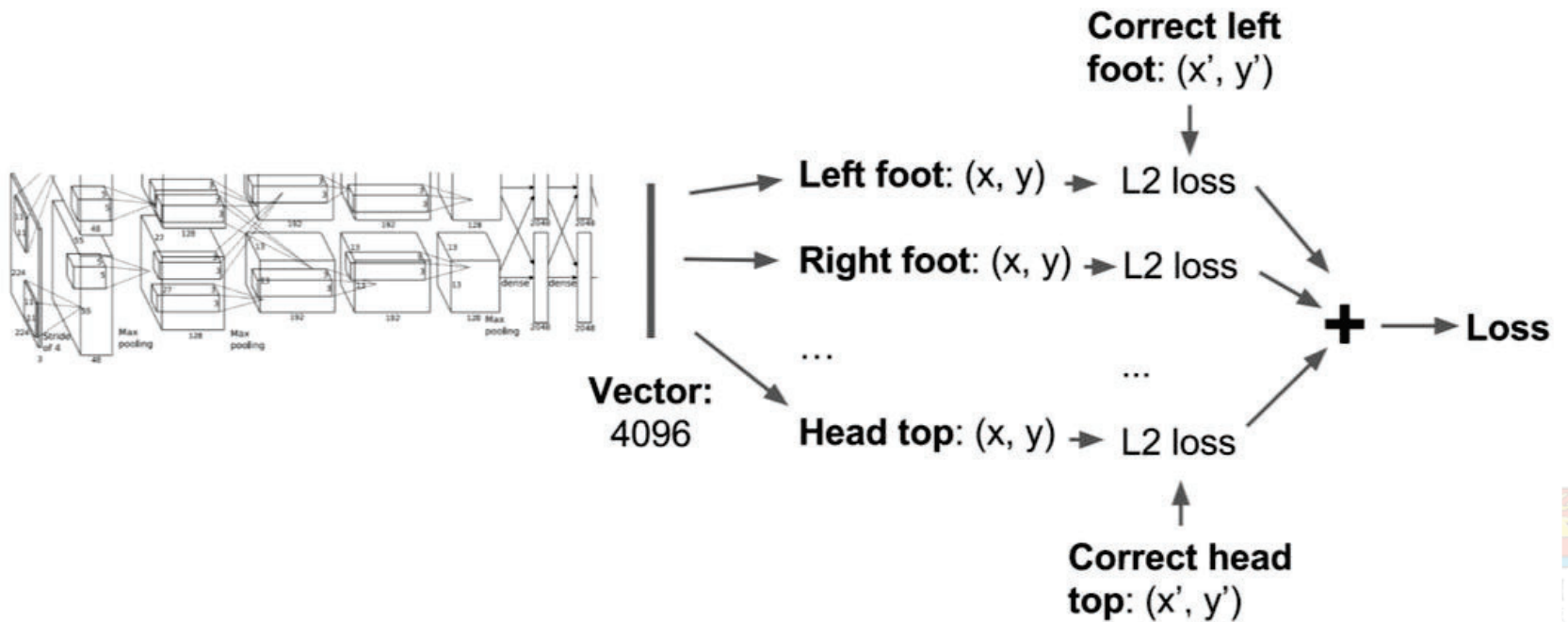


14 joint coordinate points

Sometime AI/deep learning is all about training data, loss function, and network structure.



# A more complicated case but good for understanding (human pose estimation)



Q



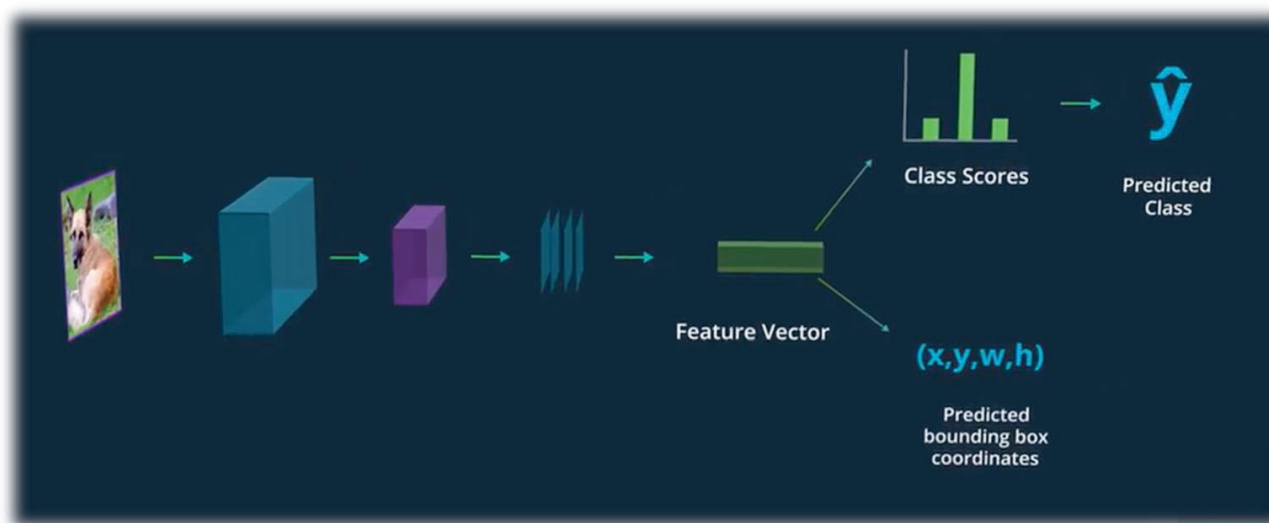


## Now let's think about two objects

- The previous “easy-to-think” strategy in fact works in practice. Probably just with lower accuracy.
- Then how about two?



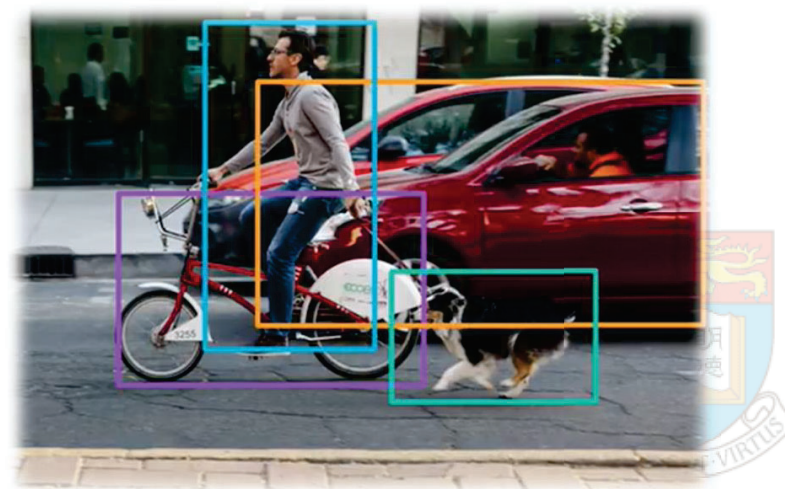
# Another “easy-to-think” strategy





# The real challenge is you don't know the number

- How about three or more?
- The real challenge is **you don't know the exact number of objects in the image.**
- Also, CNN and most neural networks have predefined input/output dimensions. E.g.,  $16*16*3$ ,  $28*28*3$ , etc.
- How to get adaptive to the variations?







# A two-step strategy

- To solve the problem, scholars proposed the two-step strategy. The idea is also driven from the “easy-to-think” strategy:
  1. Break down the images into different regions
  2. Detect the objects.
- It is easy to understand step2. Then how to achieve Step 1?

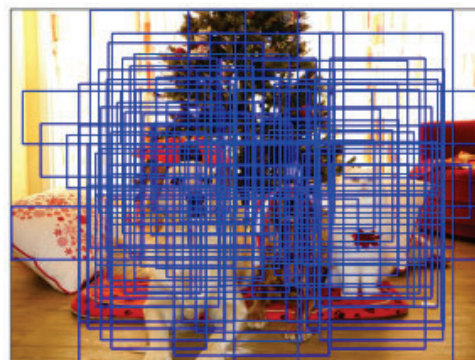


# Slide Window Concept

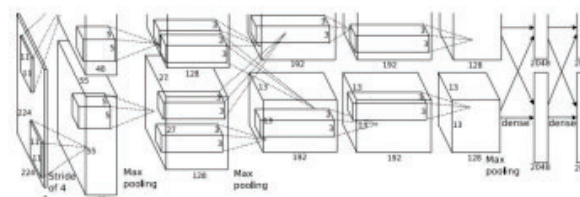
- One possible way is to set up a feasible window and stride, and slide through the original image. (or try different values)



# Problem



Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO  
Cat? YES  
Background? NO

**Problem: Need to apply CNN to huge number of locations, scales, and aspect ratios, very computationally expensive!**

- we have to try (1) different window sizes. (2) many windows do not contain objects.
- Both issues cause large amounts of calculations (not efficient).

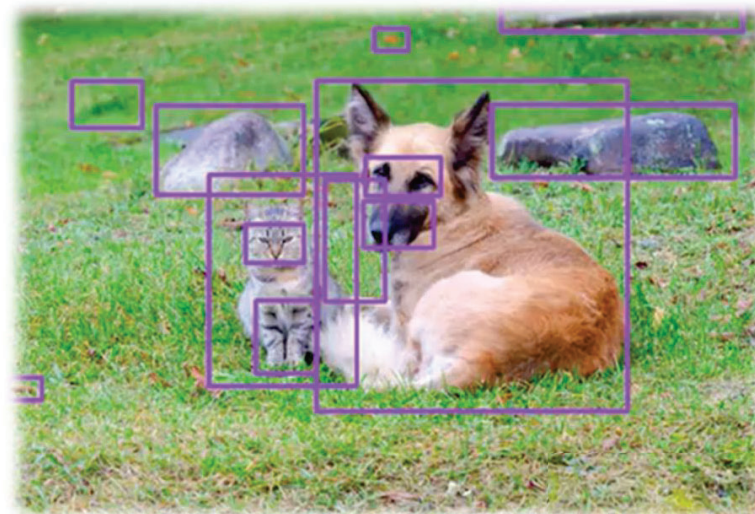






# Region Proposal

- To simplify the number of tested regions,
- We can use traditional CV techniques such as edge detection/hough transfer/color adjustment/clustering/crop, etc. (called selective search)
- The related concept is called Region Proposal. The purpose is to generate region of interest (ROI).





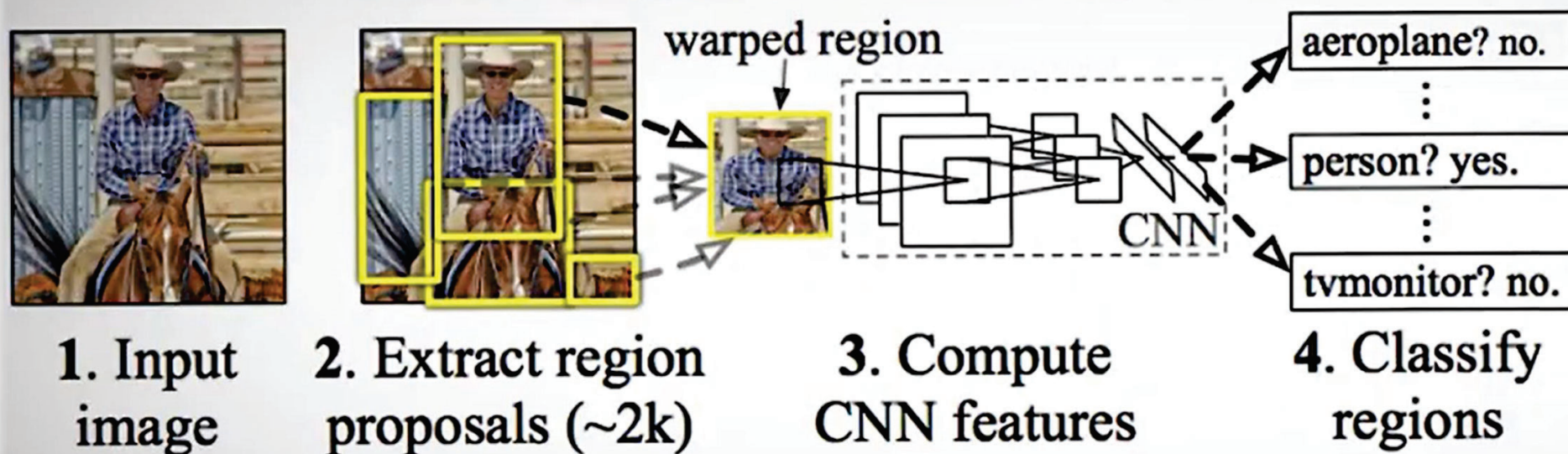
# R-CNN

- The two-step strategy **based on selective search** is called R-CNN (**Region-CNN**).
- The R-CNN is the least sophisticated region-based architecture,
- but it is the **basis** for understanding how multiple object recognition algorithms work!
- It outputs a class score and **bounding box coordinates** for every input ROI.
- An R-CNN feeds an image into a CNN with regions of interest (RoI's) already identified. **Since these RoI's are of varying sizes, they often need to be warped to be a standard size**
- **In image recognition tasks, you run CNN on one image once. In R-CNN, you will probably run ~2000 times of CNN on one image.**



# R-CNN

## R-CNN: *Regions with CNN features*

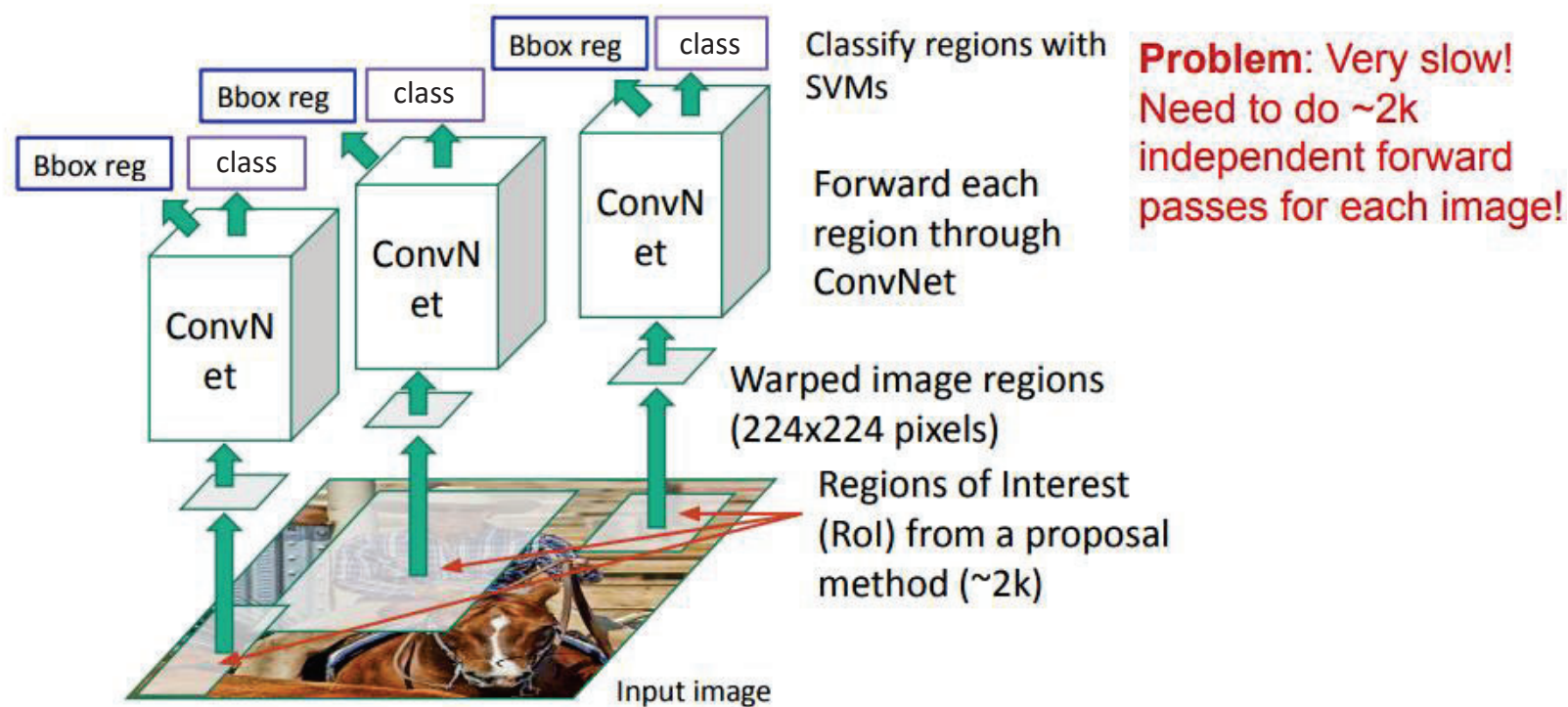


Note: CNN means CNN types of networks, such as typical CNN, AlexNet, ResNet, VGG, etc.



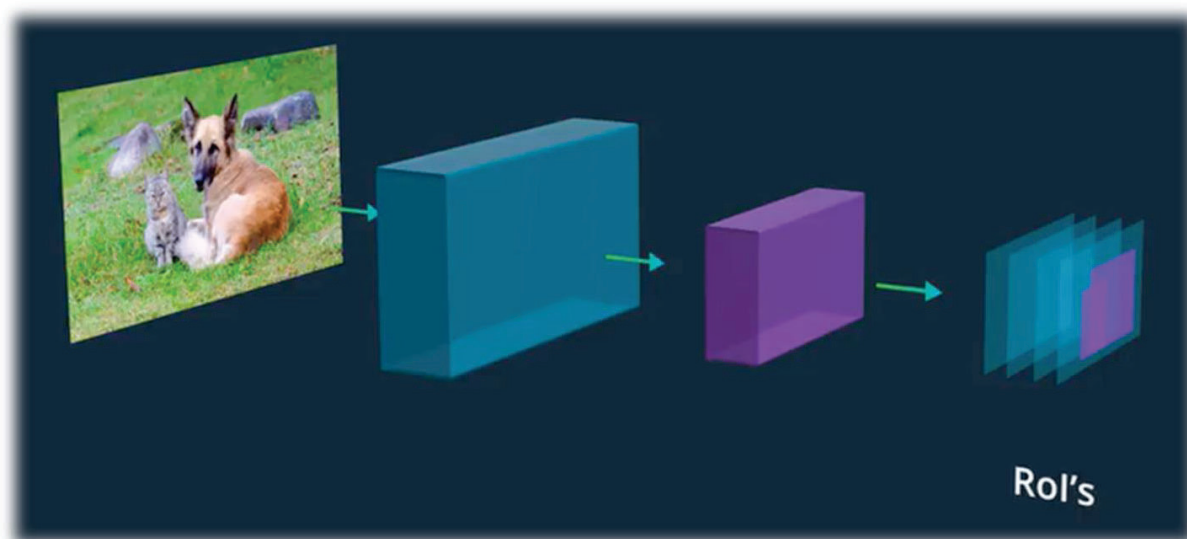


# “Slow” R-CNN

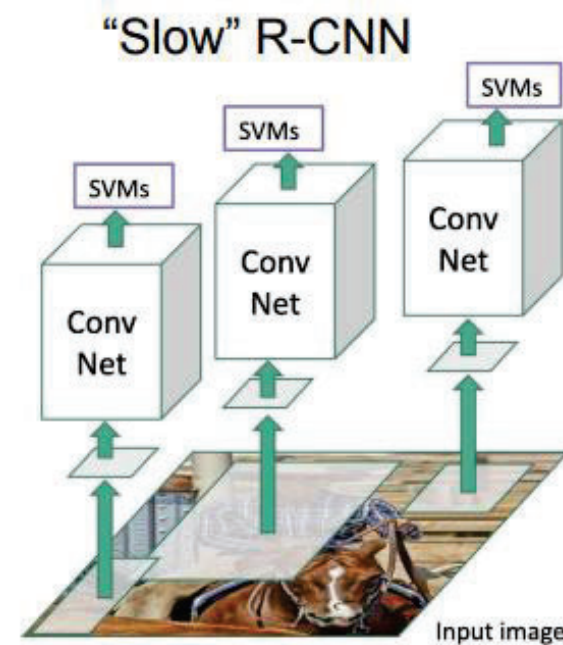
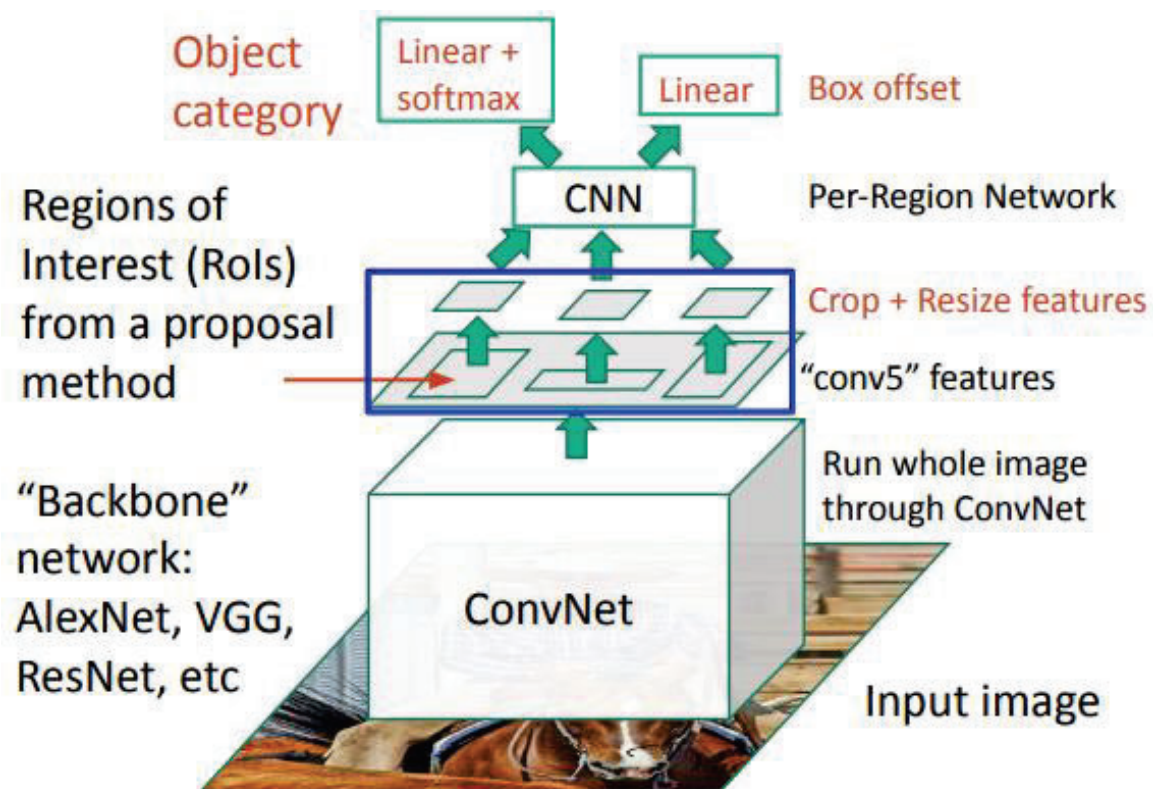


# Fast R-CNN

- Stead of sliding/searching through the original image (with different size/stride)
- Fast R-CNN slide/searching through feature maps.
- ROI pooling layer to adjust difference sized inputs from Rol layer.



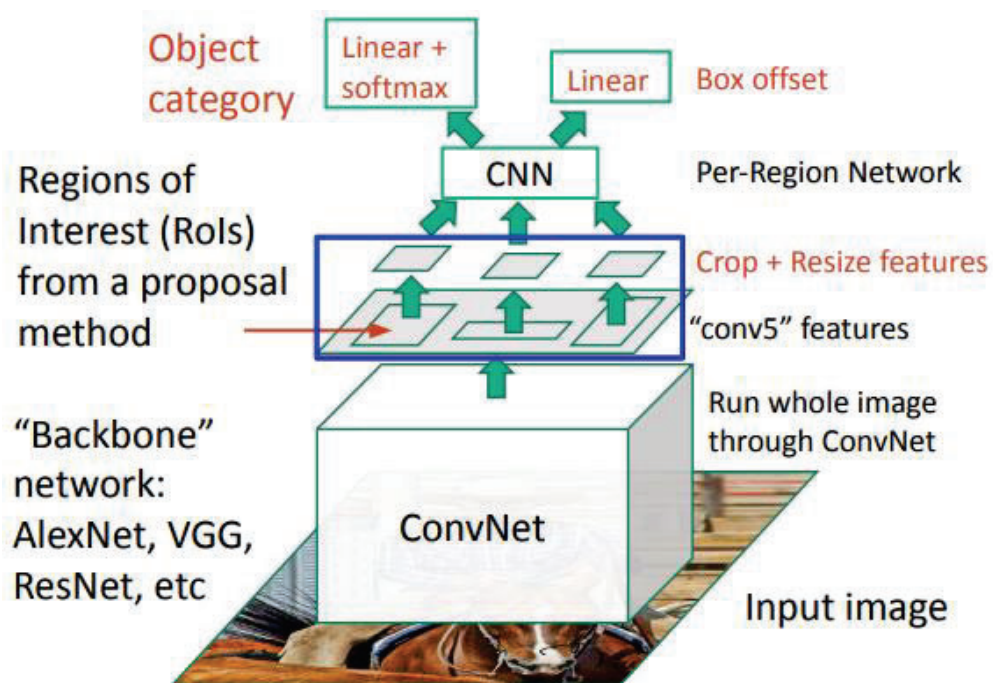
# Fast R-CNN





# Rol Pooling and Rol Align

- Two special techniques/terminology
- Rol Polling: To warp regions of interest (Rol) into a consistent size for further analysis
- Rol Align: map bounding box from feature map to the original image



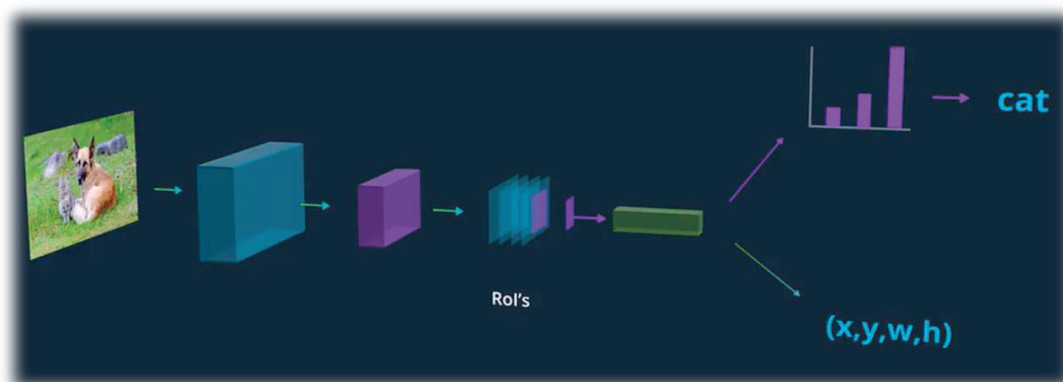
Note: Not necessary to know the detail of Rol Pooling and Align at this stage.



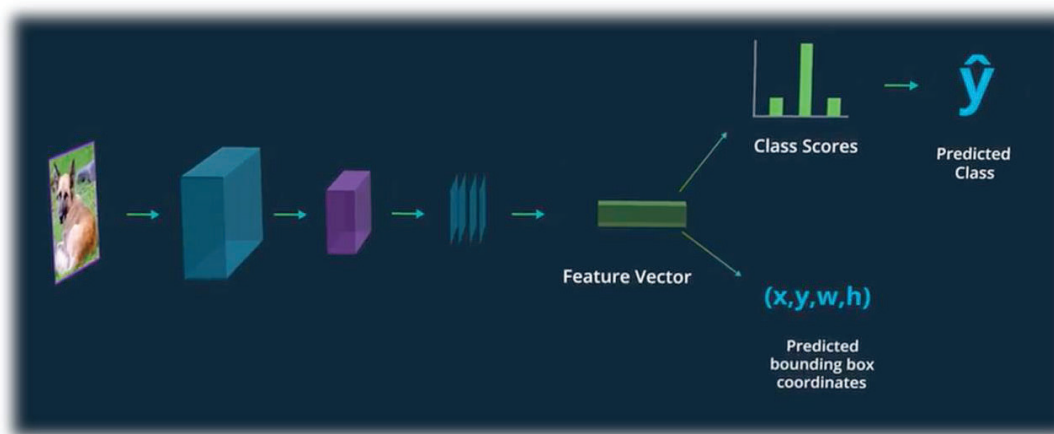
# Another View on Fast R-CNN

Difference:

1. Slide through image/feature map
2. The feature map layer to flattened layer are different (one is fully connected, the other is region proposal layer)



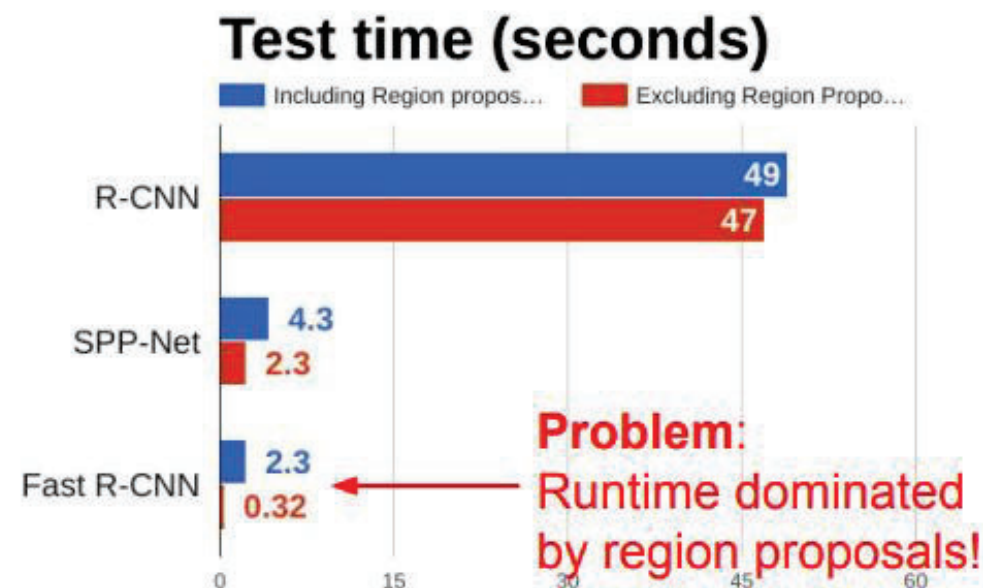
Fast R-CNN  
(separate CNN into two parts)



R-CNN  
(window slide on  
images + CNN)



# R-CNN vs Fast R-CNN



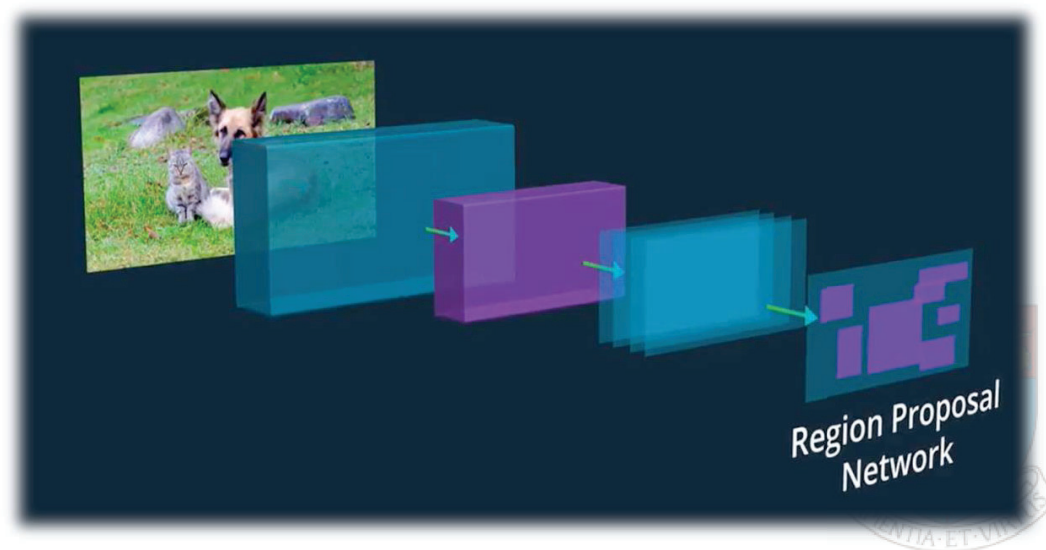
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.  
He et al, "Spatial pyramid pooling in deep convolutional networks for visual recognition", ECCV 2014  
Girshick, "Fast R-CNN", ICCV 2015





# Faster R-CNN

- Now that region proposal consumes the largest amount of time. Can we replace it using faster methods?
- Faster R-CNN using additional neural network layers to “recognize” region proposals,
- It is called region proposal network

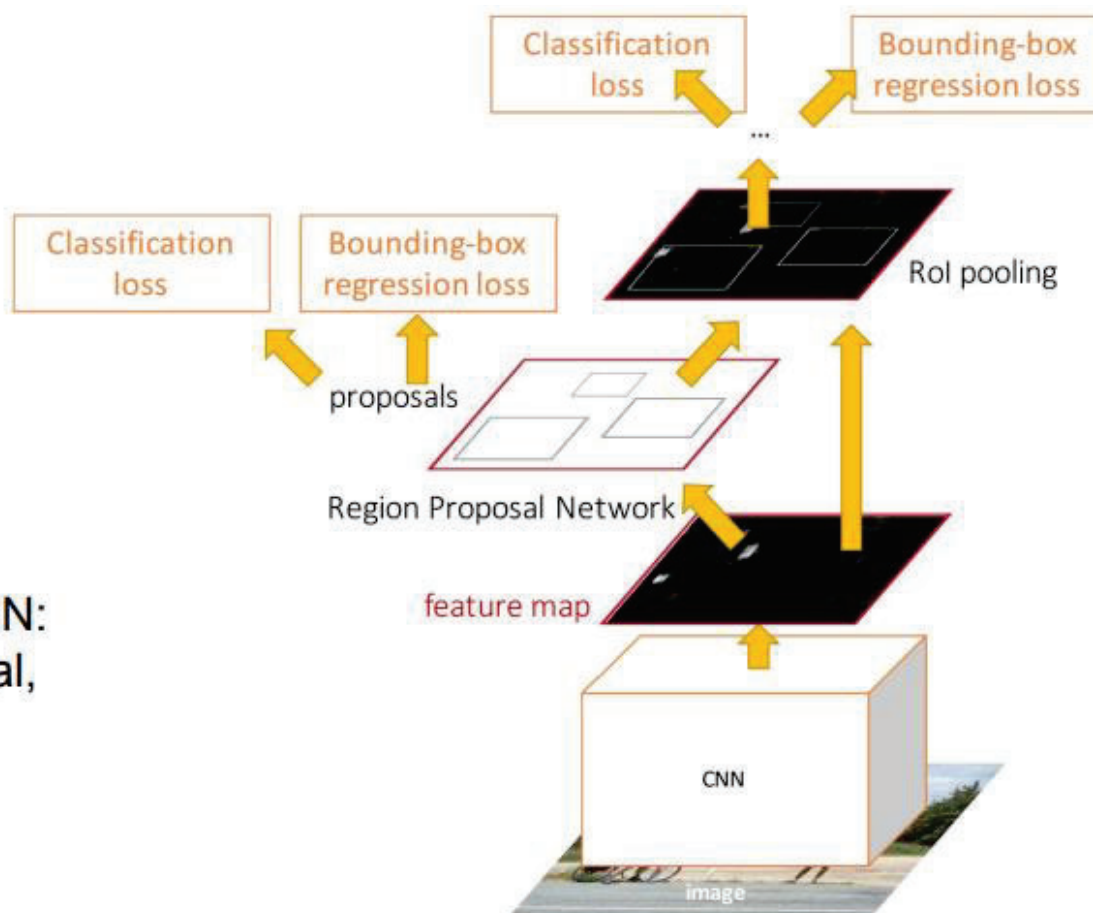


# Faster R-CNN

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





# Faster R-CNN

Make CNN do proposals!

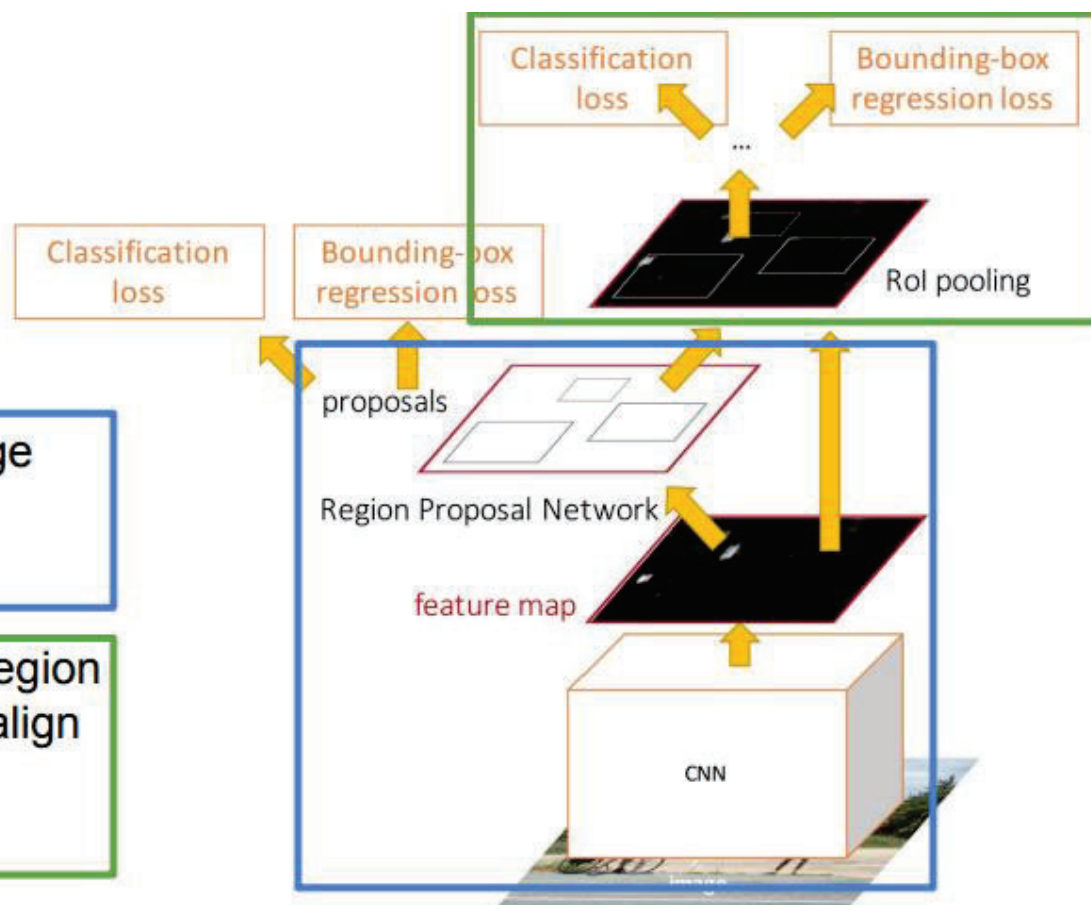
Faster R-CNN is a  
**Two-stage object detector**

First stage: Run once per image

- Backbone network
- Region proposal network

Second stage: Run once per region

- Crop features: RoI pool / align
- Predict object class
- Prediction bbox offset

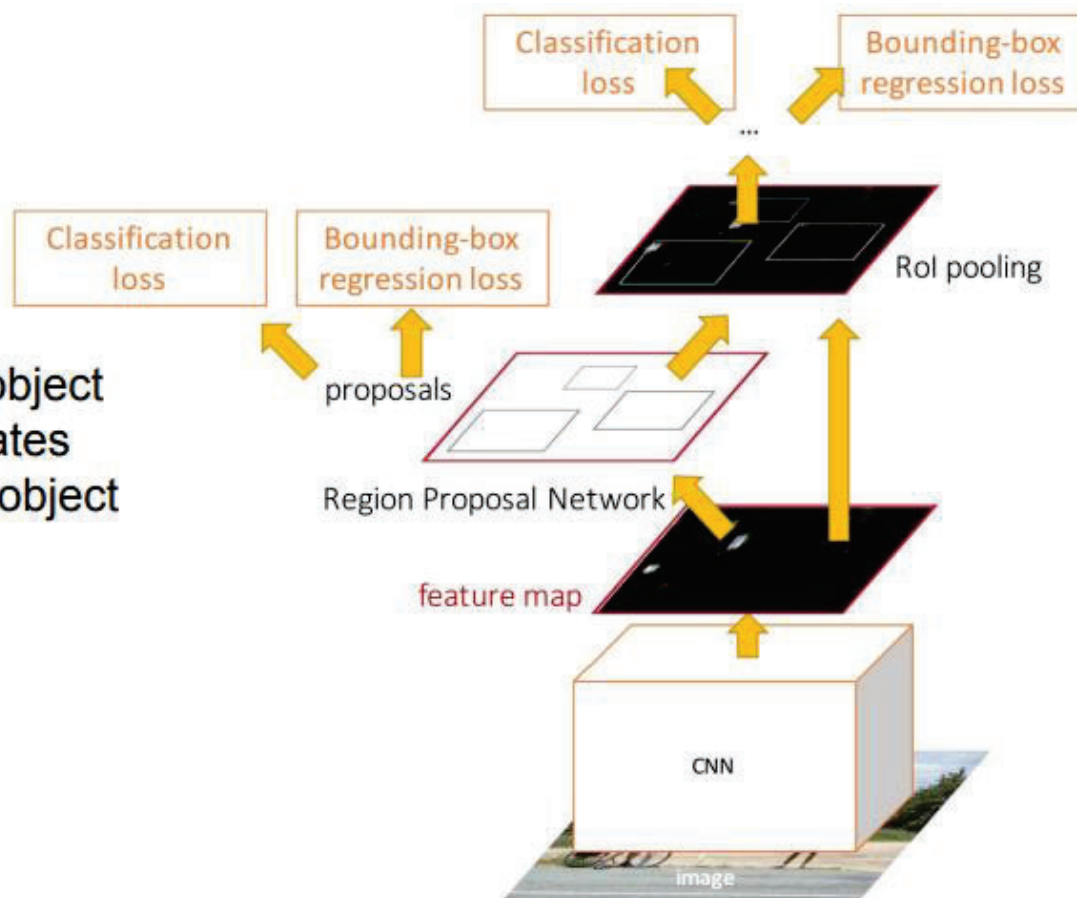


# Faster R-CNN

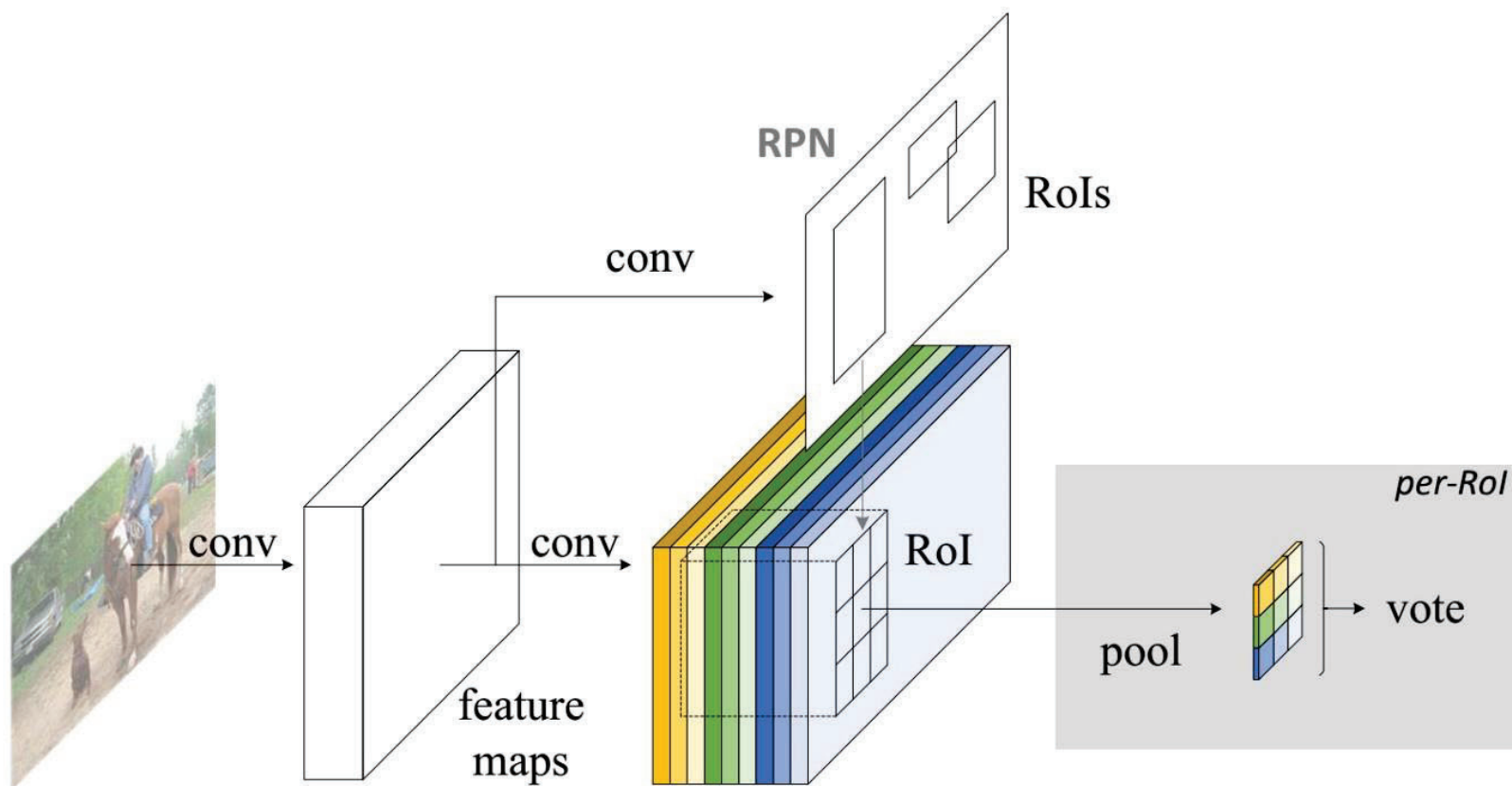
Make CNN do proposals!

Jointly train with 4 losses:

1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classification score (object classes)
4. Final box coordinates

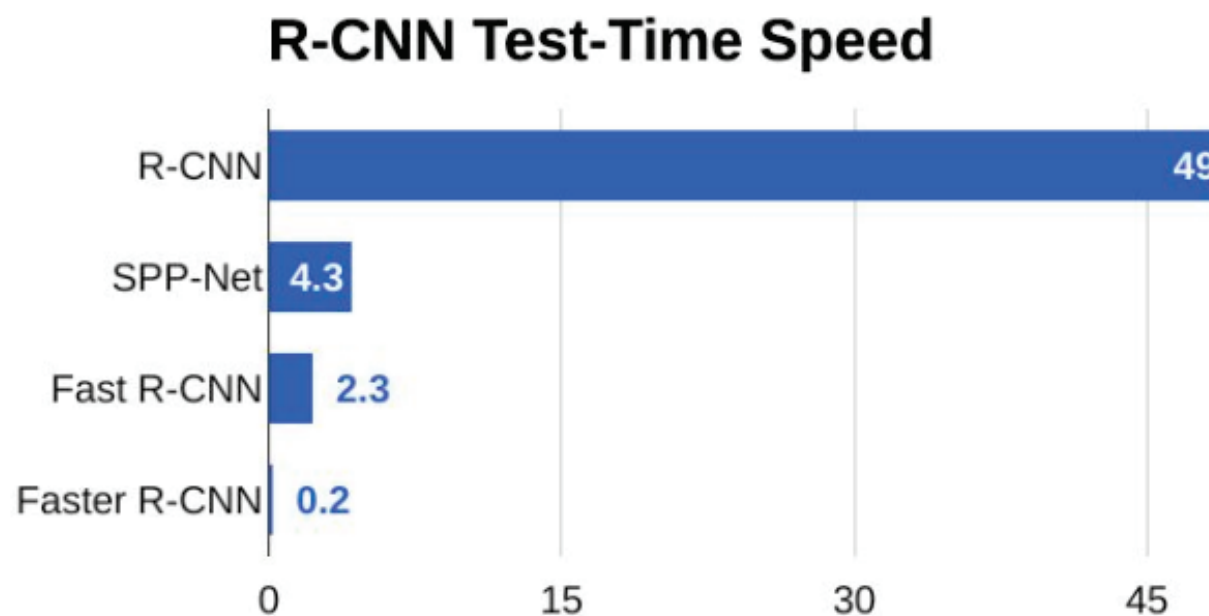


# Another View on the Network Structure





# Speed Comparison





Q

**Thanks for listening.**