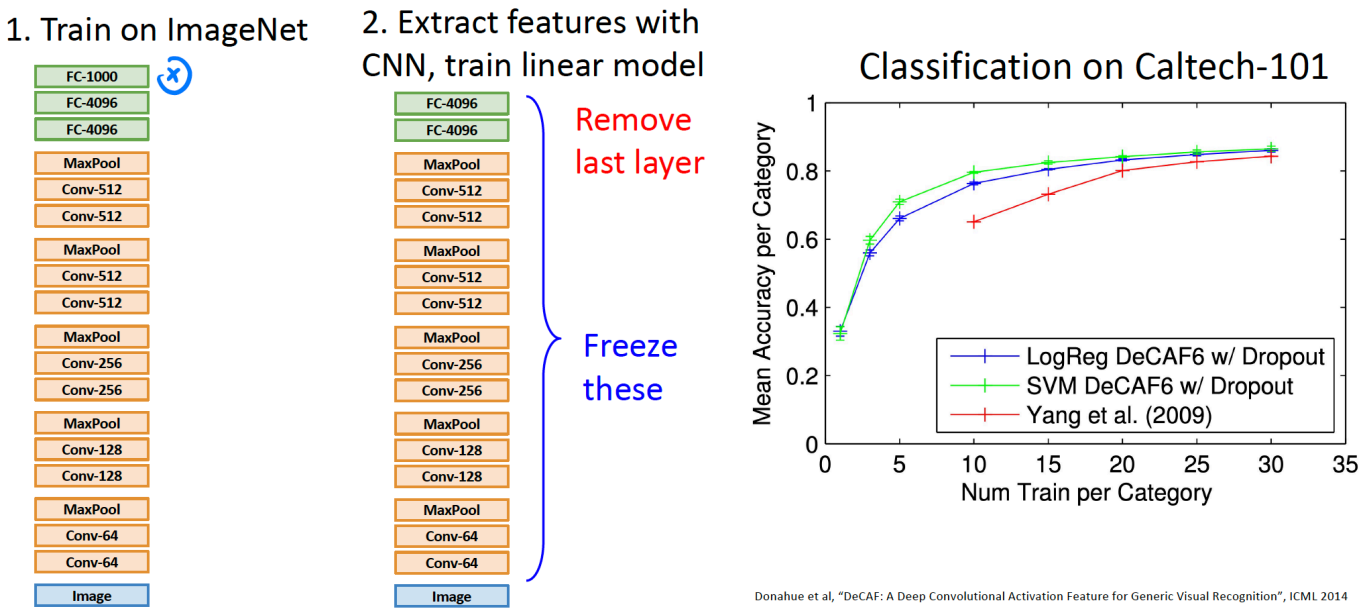
Lecture 13 - Object Detection

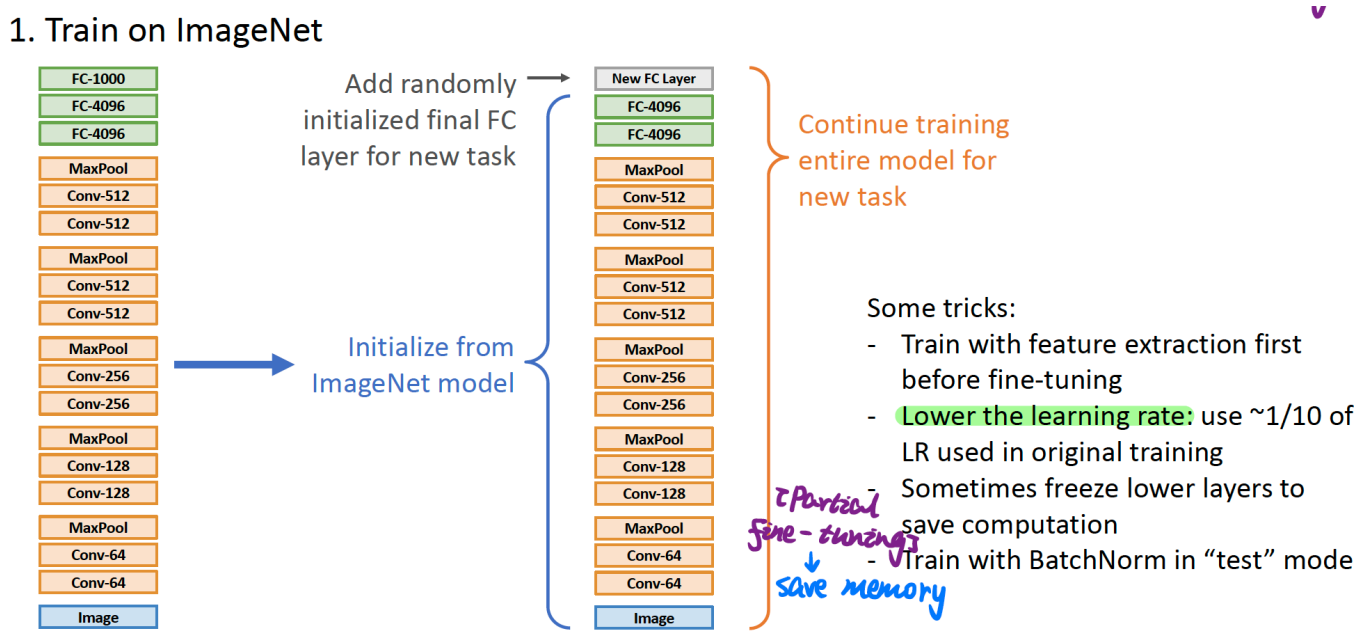
1. Transfer Learning: Generalizing to New Tasks

- **Feature Extraction**



Donahue et al, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014

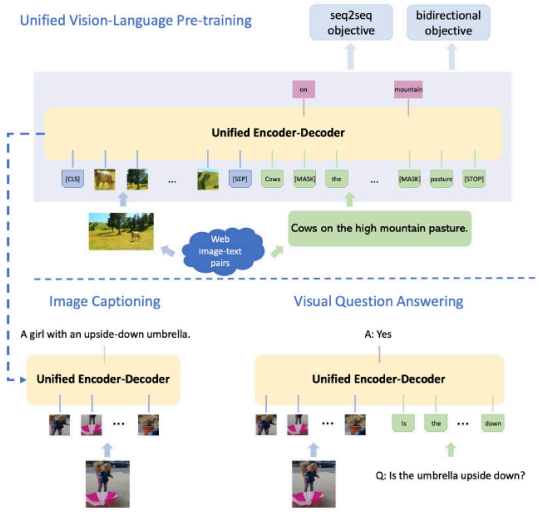
- **Fine-Tuning**



  - Compared with feature-extraction, fine-tuning:
    - Require more data
    - More computationally expensive
    - Can give higher accuracies

- **Properties**

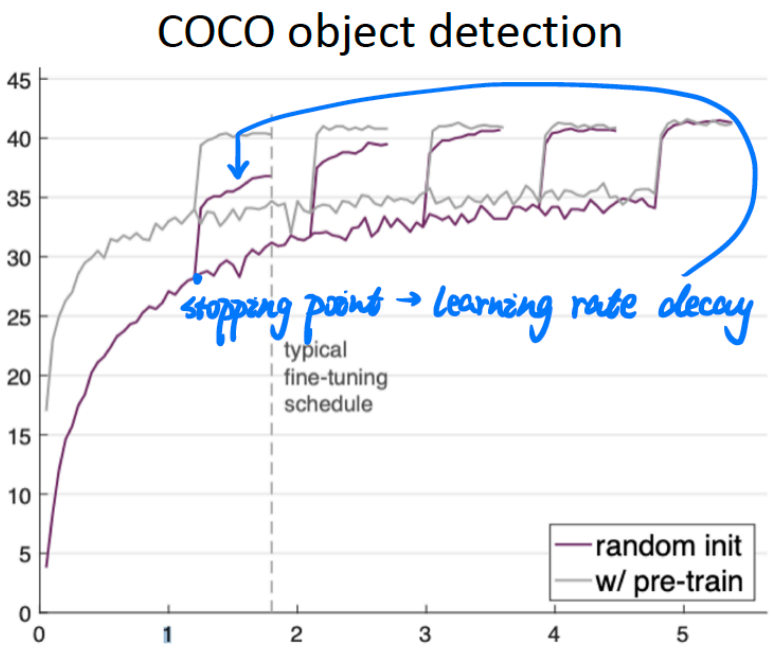
- Architecture matters! --> improvement in CNN architectures lead to improvements in many downstrean tasks (current task) thanks to transfer learning (same algorithms with different underlying CNN structure)
- Very pervasive/popular --> become the norm not the exception



1. Train CNN on ImageNet
2. Fine-Tune (1) for object detection on Visual Genome
3. Train BERT language model on lots of text
4. Combine (2) and (3), train for joint image / language modeling
5. Fine-tune (5) for image captioning, visual question answering, etc.

Zhou et al, "Unified Vision-Language Pre-Training for Image Captioning and VQA", AAAI 2020

- Help to converge faster!



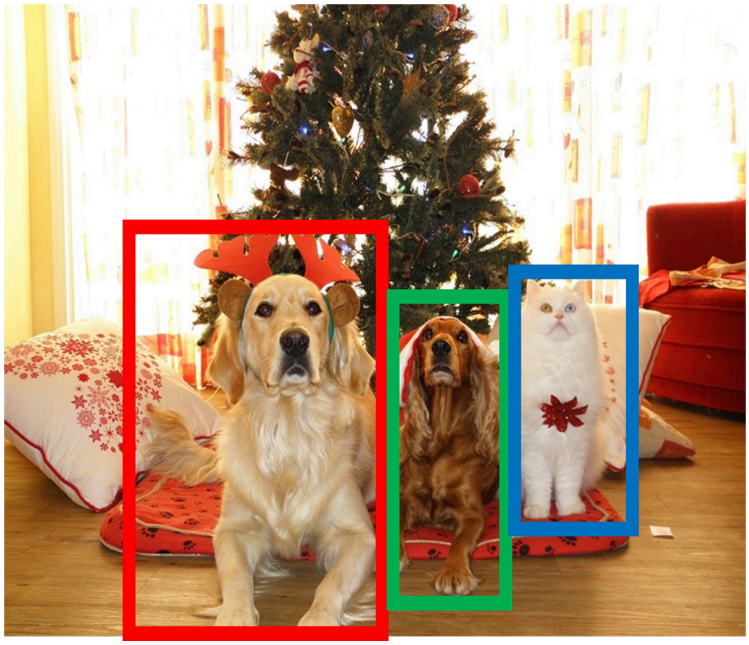
(With enough data and train for much longer, random initialization can sometimes do well as transfer learning)

2. Object Detection

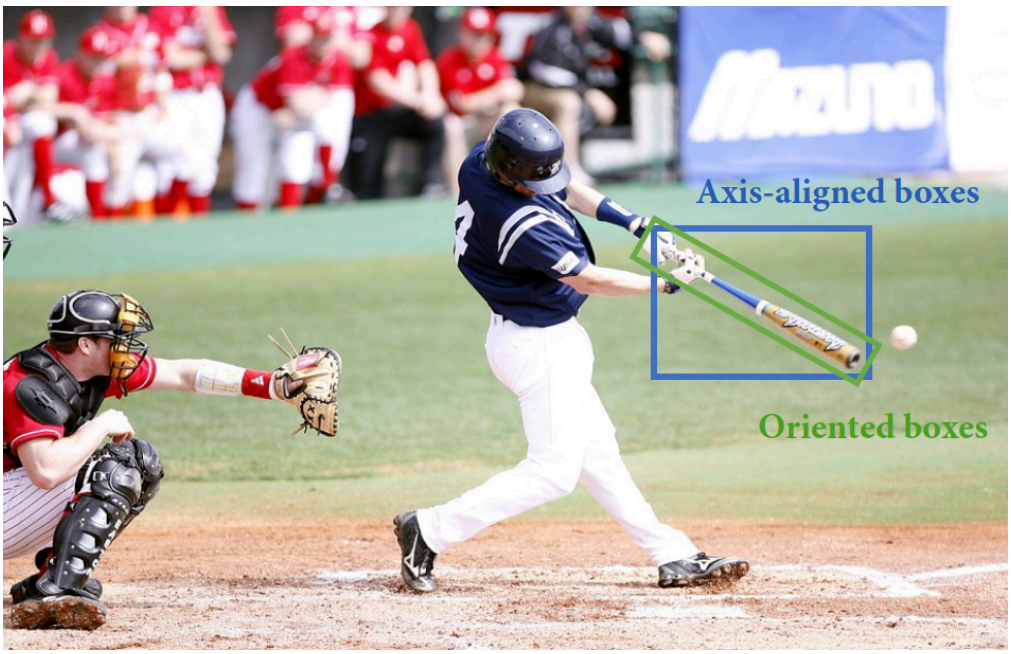
--> localizing objects with bounding boxes

- **Task Definition**
  - Input: single RGB image

- Output: a set of detected objects --> for each object predict:
    1. Category label (from fixed, known set of categories)
    2. Bounding box (four numbers: x, y, width, height)

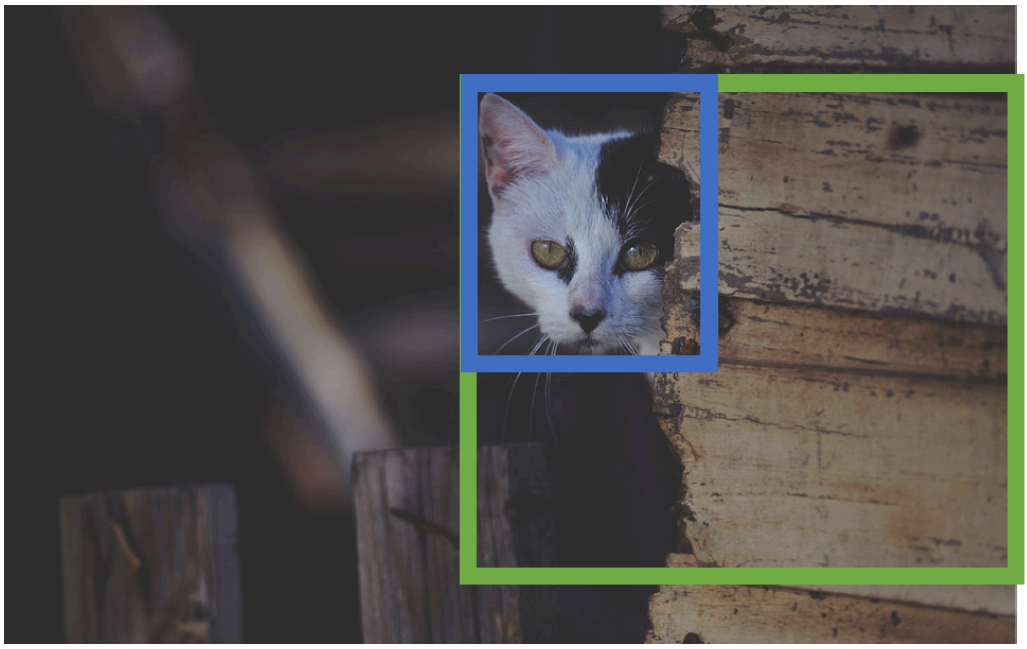


- **Challenges**
    - Multiple outputs: Need to output variable numbers of objects per image
    - Multiple types of output: Need to predict **"what" (category label)** as well as **"where" (bounding box)**
    - Large images: Classification works at 224x224; need **higher resolution for detection**, often ~800x600
- **Bounding Boxes**
    - Typically axis-aligned; Oriented boxes are much less common;



    - **Modal Detection:** Usually cover only the visible portion of the object

Amodal Detection: box covers the entire extent of the object, even occluded parts（遮挡的部分）



- Comparing Boxes: **Intersection over Union (IoU)**

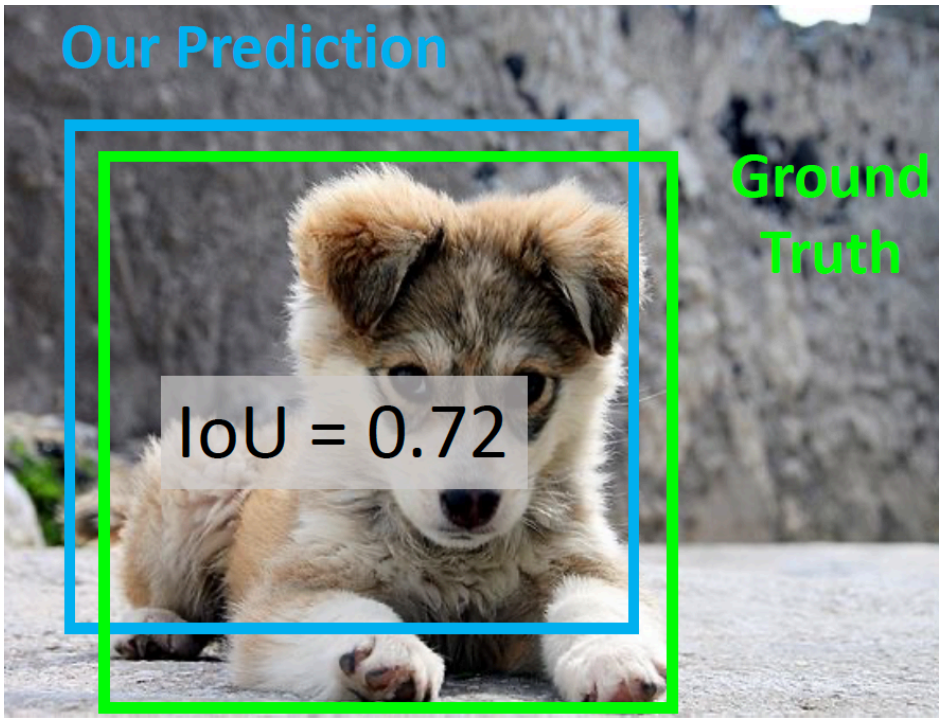
  --> compare our prediction to the ground-truth box

  $IoU = \frac{Area\ of\ Interest}{Area\ of\ Union}$

  - IoU > 0.5 is “decent”,

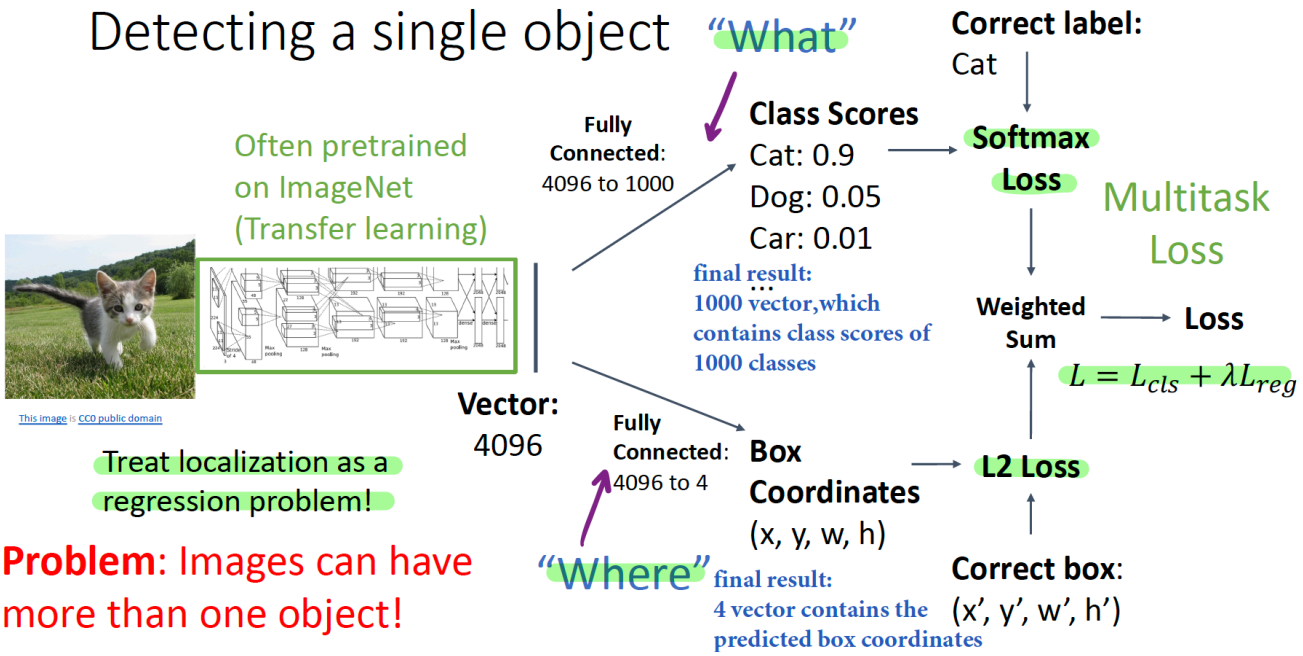
    IoU > 0.7 is “pretty good”,

    IoU > 0.9 is “almost perfect”



- Detect a Single Object

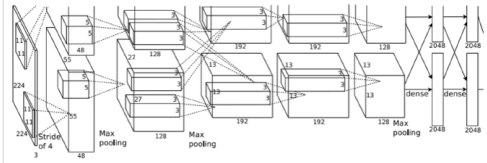
conduct both classification task for "what" and regression task for "where"



- Detect Multiple Objects

  --> using **Sliding Window** technique to classify each sub-region

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



Dog? YES
Cat? NO
Background? NO

  - How many possible boxes are there in an image of size H x W?

Consider a box of size h x w:
Possible x positions: W – w + 1
Possible y positions: H – h + 1
Possible positions:
(W – w + 1) * (H – h + 1)

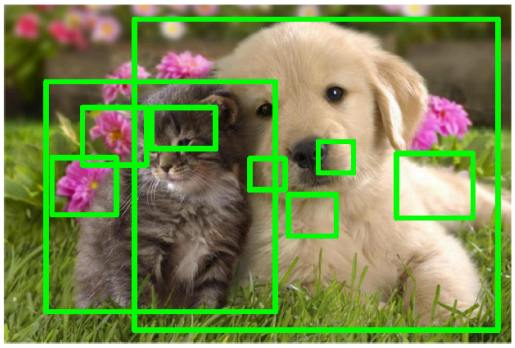
Total possible boxes:

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

$$=\frac{H(H+1)}{2}\frac{W(W+1)}{2}$$

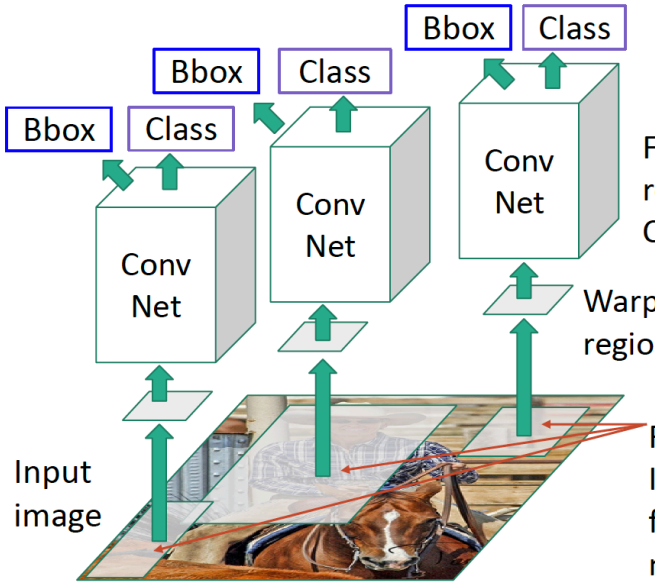
- Region Proposals

  - Find a small set of boxes that are likely to **cover all objects**
  - Often based on heuristics/intuition: e.g. look for "blob-like" image regions（斑点状区域）
  - Relatively **fast to run**; e.g. Selective Search gives 2000 region proposals in a few seconds on CPU



- **R-CNN: Region-Based CNN** --> processes region proposals with a CNN

R-CNN: Region-Based CNN



Classify each region

Bounding box regression: Predict "transform" to correct the RoI: 4 numbers (t_x, t_y, t_h, t_w)

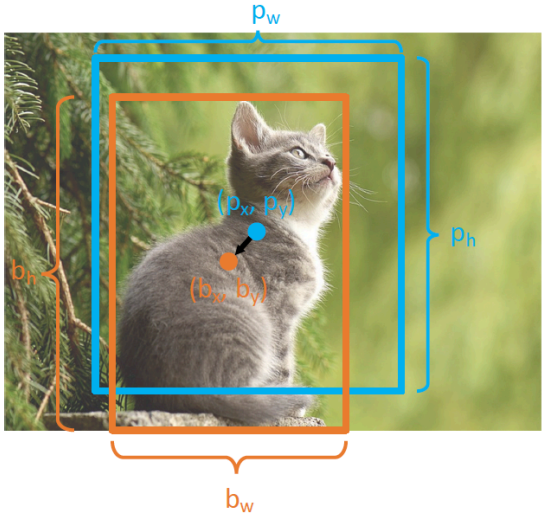
Forward each region through ConvNet

Warped image regions (224x224)

扭曲的图像区域 --> reshape the original RoI to 224*224 to feed into ConvNet

Regions of Interest (RoI) from a proposal method (~2k)

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.



Consider a region proposal with center (p_x, p_y), width p_w, height p_h

Model predicts a transform (t_x, t_y, t_w, t_h) to correct the region proposal

The output box is defined by:

$$b_x = p_x + p_w t_x$$
$$b_y = p_y + p_h t_y$$
$$b_w = p_w \exp(t_w)$$
$$b_h = p_h \exp(t_h)$$

Shift center by amount relative to proposal size

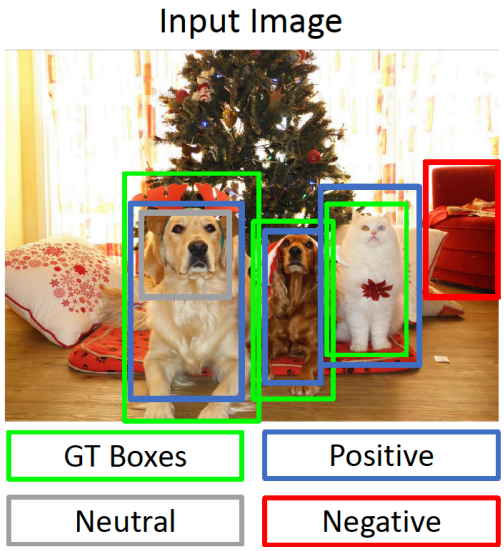
Scale proposal; exp ensures that scaling factor is > 0

Given proposal and target output, we can solve for the transform the network should output:

$$t_x = (b_x - p_x)/p_w$$
$$t_y = (b_y - p_y)/p_h$$
$$t_w = \log(b_w/p_w)$$
$$t_h = \log(b_h/p_h)$$

- When transform is 0 ($t_x = t_y = t_w = t_h$), output = proposal
- L2-regularization encourages leaving proposal unchanged
- **Scale / Translation invariance:** Transform **encodes relative difference between proposal and output**; important since CNN doesn't see absolute size or position after cropping
- R-CNN Training:

1. Run proposal method to find region proposals and categorize each region proposal, ignore neural proposal

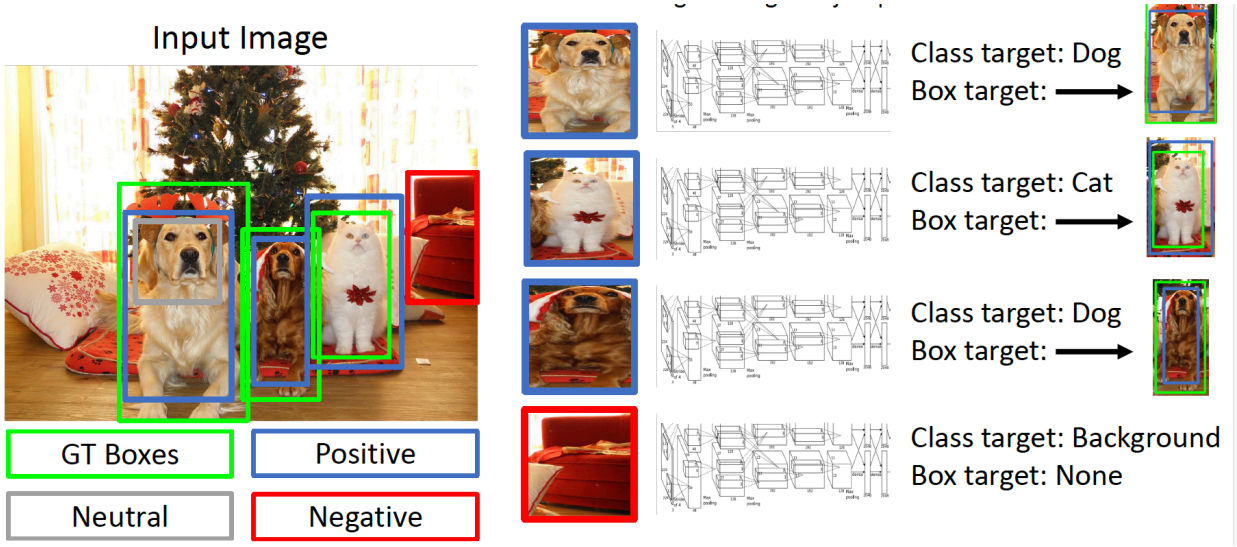


Categorize each region proposal as positive, negative, or neutral based on overlap with ground-truth boxes:

Positive: > 0.5 IoU with a GT box → high overlap
Negative: < 0.3 IoU with all GT boxes
Neutral: between 0.3 and 0.5 IoU with GT boxes
△ ignore neutral proposal

2. Crop pixels from each positive and negative proposal and resize to 224×224
3. Run each region through CNN to get class scores and transforms
   - Positive regions: predict class and transform bounding boxes
   - Negative regions: just predict class
4. **Threshold** class scores to get a set of detections

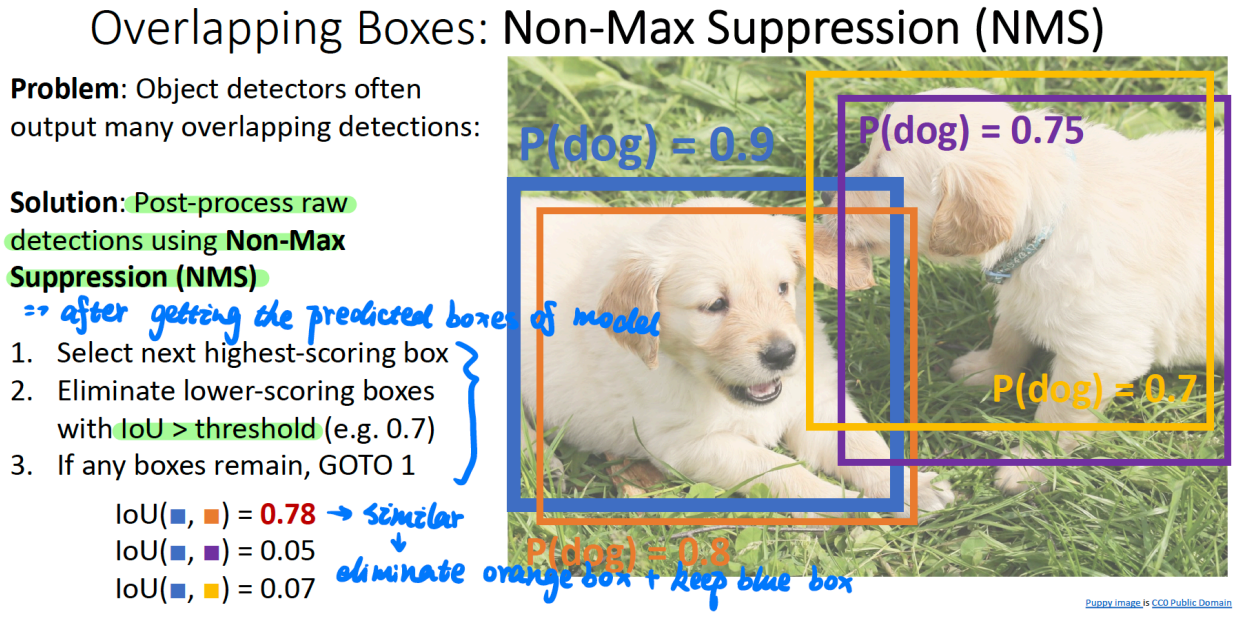
Negative regions: just predict class



- **Problems**

1. Overlapping Boxes output by CNN --> **Non-Max Suppression (NMS)**

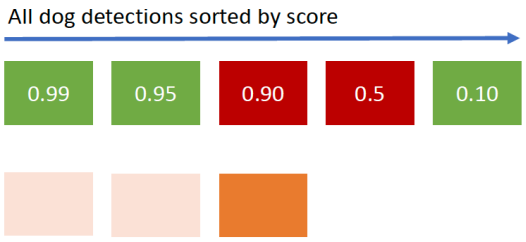
   [Eliminate overlapping detections using NMS]



   Current Problem: NMS may eliminate "good" boxes when objects are highly overlapping... no good solution :(

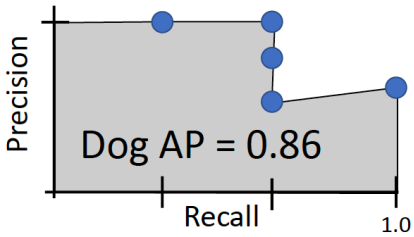
2. Set Thresholds --> --> **Evaluate Object Detectors: Mean Average Precision (mAP)**

Evaluating Object Detectors: Mean Average Precision (mAP)



1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve
    2. Average Precision (AP) = area under PR curve

How to get AP = 1.0: Hit all GT boxes with IoU > 0.5, and have no "false positive" detections ranked above any "true positives"



3. Mean Average Precision (mAP) = average of AP for each category
4. For "COCO mAP": Compute mAP@thresh for each IoU threshold (0.5, 0.55, 0.6, …, 0.95) and take average

$mAP@+IoU$ means the mean average precision at each IoU threshold

Car AP = 0.65

Cat AP = 0.80

Dog AP = 0.86

mAP@0.5 = 0.77

mAP@0.5 = 0.77

mAP@0.55 = 0.71

mAP@0.60 = 0.65

…

mAP@0.95 = 0.2

COCO mAP = 0.4