CSCE 479/879 Lecture 10: Object Detection

Stephen Scott

sscott@cse.unl.edu

Introduction

- We know that CNNs are useful in image classification
- Now consider **object detection**
  - Given an input image, identify what objects (plural) are in it and where they are
  - Output **bounding box** of each object

Outline

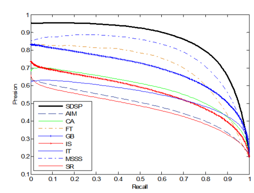
- Performance measures
- RCNN
- SPP-net
- Fast RCNN
- YOLO

Performance Measures

Mean Average Precision

Mean average precision (mAP) to measure how well objects are identified

- Recall from Lecture 3
  - **Precision** is fraction of those labeled positive that are positive
  - **Recall** is fraction of the true positives that are labeled positive
  - **Precision-recall curve** plots precision vs recall



Performance Measures

Mean Average Precision (2)

- Given a ranking (by confidence values) of n items, **average precision at** n (AP@n) is average of precision values at each position in the ranking:

$$AP = \sum_{k=1}^{n} P(k)\Delta r(k) \ ,$$

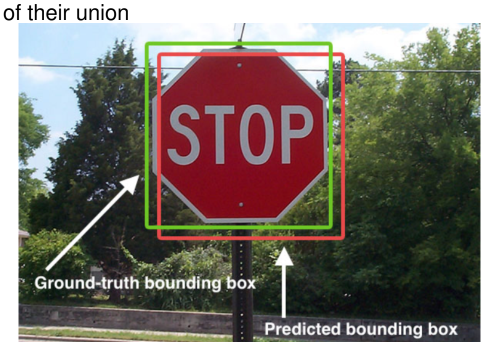
where $P(k)$ is precision at position k and $\Delta r(k)$ is change in recall: $r(k) - r(k-1)$ ($= 0$ if instance k is negative, $= 1/N_p$ if k is one of N_p positives)
  - E.g., if ranking $= \langle +, +, -, +, - \rangle$, AP@5 $= (1)(1/3) + (1)(1/3) + (2/3)(0) + (3/4)(1/3) + (3/5)(0)$
  - Larger as more positives ranked above negatives
- mAP is mean of average precision across all classes

Performance Measures

Intersection Over Union

Intersection over union (IoU) to measure quality of bounding boxes

- Divide the size of the two boxes' intersection by the size of their union



Basic Idea of Object Detection

Split input image into **regions** and classify each region with a CNN and other machinery

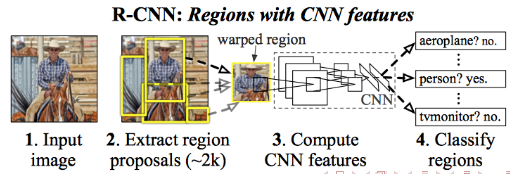
- Region boundary is bounding box
- Object detected in region is object in BB

Issues:

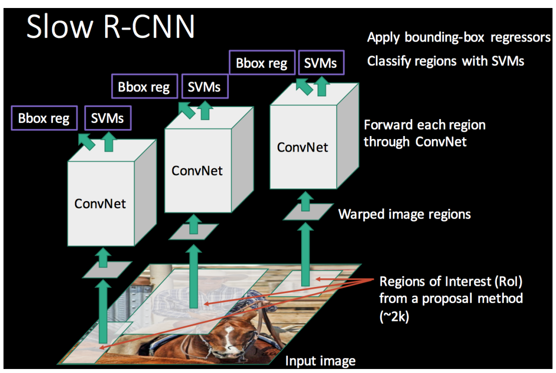
- Limited to bounding boxes of fixed sizes and locations
- An object could span regions

Region CNN (Girshick et al. 2014)

- R-CNN **proposes** collection of 2000 regions in image
- Warps each region to match input dimensions $(227 \times 227 \times 3)$ of CNN to get 4096-dimensional embedded representation
- Classifies each embedded vector with class-specific binary SVMs
- Apply class-specific regressors to fine-tune bounding boxes



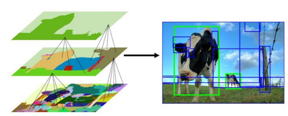
Region CNN (Girshick et al. 2014)
Example from Girshick (2015)



Region CNN (Girshick et al. 2014)
Selective Search

Popular method to propose RoIs: **selective search**

1. **Segment** the image
2. Compute bounding boxes of segments
3. Iteratively merge adjacent segments based on similarity
   - Linear combination of similarities of: color, texture, size, shape
4. Goto 2

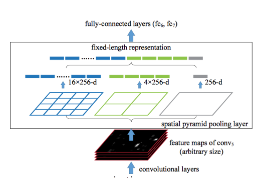


Region CNN (Girshick et al. 2014)
Issues

- Training and detection are slow
- Detection: 13s/image on GPU, 53s/image on CPU
- Due to large number of regions proposed, each run through CNN and classifier

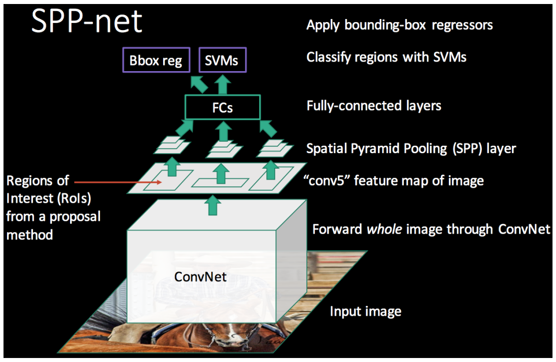
Spatial Pyramid Pooling (He et al. 2015)

- Part of R-CNN's slowdown at test time is running each RoI through ConvNet separately
- To speed up test time, instead put entire image through **single ConvNet**



- Choose RoIs from ConvNet output and run through **spatial pyramid pooling** (SPP) layer
  - Max/avg pooling with **fixed** number of bins
  - Produces fixed-length vector regardless of input size
- Fixed-length vectors feed to fully connected layers, then SVMs

Spatial Pyramid Pooling (He et al. 2015)
Example from Girshick (2015)



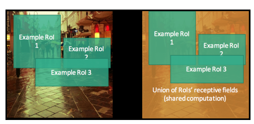
Spatial Pyramid Pooling (He et al. 2015)
Drawbacks

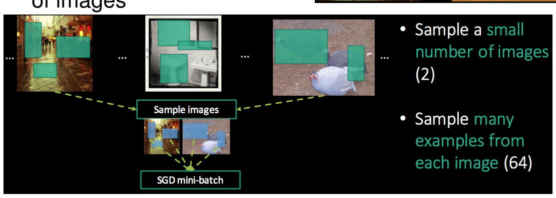
- While training is faster then R-CNN, is still slow and disk-intensive
- Cannot efficiently update ConvNet parameters, so kept frozen
  - Each RoI's receptive field covers most of entire image, so forward pass expensive across all images of mini-batch

Fast R-CNN (Girshick 2015)
Hierarchical Sampling

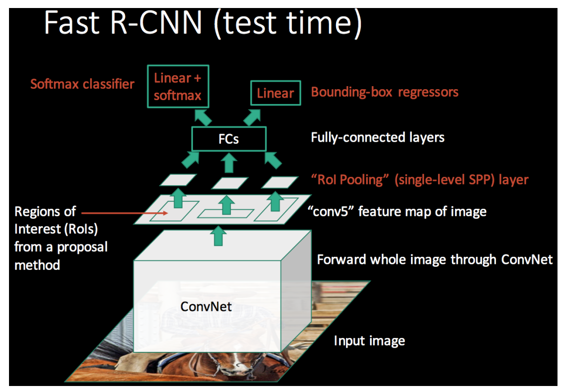
Similar architecture to SPP-net

- Mini-batches constructed via **hierarchical sampling**: Sample a similar number of RoIs over a **smaller** number of images

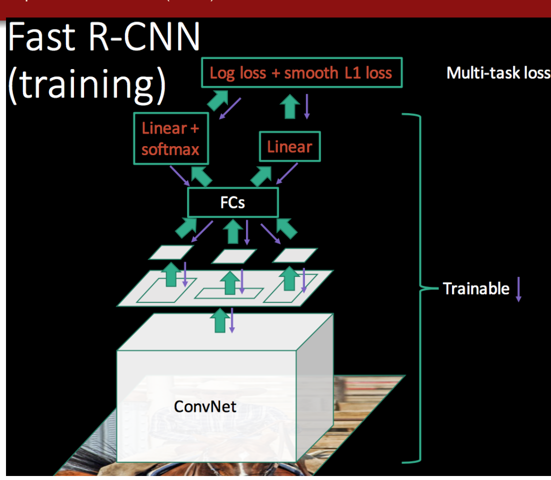




Fast R-CNN (Girshick 2015)
Example from Girshick (2015)



Fast R-CNN (Girshick 2015)
Example from Girshick (2015)

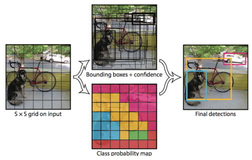


You Only Look Once (Redmon et al. 2016)

- A single, unified network
- Can process 45 frames per second on a GPU (155 fps for Fast YOLO)
- Lower mAP than some R-CNN variants, but much faster
- Highest mAP of real-time detectors (≥ 30 fps)

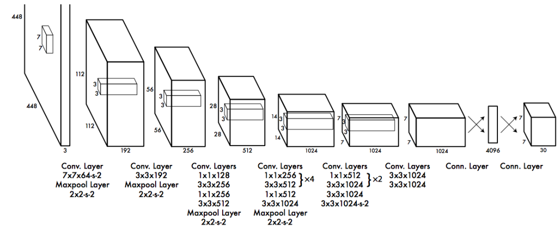
You Only Look Once (Redmon et al. 2016)
Idea

- Divides image into $S \times S$ grid
- Each grid cell predicts B bounding boxes, each as (x, y, w, h) (coordinates, width, height), and a confidence (five total predictions)



- $x, y, w, h \in [0, 1]$ (relative to image dimensions and grid cell location)
- Each cell also predicts C class probabilities
- Output is $S \times S \times (5B + C)$ tensor

You Only Look Once (Redmon et al. 2016)
Architecture



Leaky ReLU for all layers except output, which is linear

You Only Look Once (Redmon et al. 2016)
Training

- Pretrained 20 convolutional layers on ImageNet 1000
- Added 4 convolutional layers and 2 connected layers
- Trained to optimize weighted square loss function

$\lambda_{coord} = 5$ times more weight on (x, y, w, h) predictions

$$
\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}
$$