From image classification to object detection

**Image classification**

**Object detection**

**Image source**
What are the challenges of object detection?

- Images may contain more than one class, multiple instances from the same class
- Bounding box localization
- Evaluation
Outline

• Task definition and evaluation
• Detection strategies
• Zoo of deep detection approaches
  • R-CNN
  • Fast R-CNN
  • Faster R-CNN
  • YOLO
  • SSD
  • Some recent entries: RetinaNet, CornerNet, DETR
Object detection evaluation

- At test time, predict bounding boxes, class labels, and confidence scores
- For each detection, determine whether it is a true or false positive
  - PASCAL criterion: $\frac{\text{Area}(\text{GT} \cap \text{Det})}{\text{Area}(\text{GT} \cup \text{Det})} > 0.5$
  - For multiple detections of the same ground truth box, only one is considered a true positive
Object detection evaluation

• At test time, predict bounding boxes, class labels, and confidence scores
• For each detection, determine whether it is a true or false positive
• For each class, sort detections from highest to lowest confidence, plot Recall-Precision curve and compute Average Precision (area under the curve)
• Take mean of AP over classes to get mAP

\[ \text{Precision:} \quad \frac{\text{true positive detections}}{\text{total detections}} \]
\[ \text{Recall:} \quad \frac{\text{true positive detections}}{\text{total positive test instances}} \]
PASCAL VOC Challenge (2005-2012)

• 20 challenge classes:
  • *Person*
  • *Animals*: bird, cat, cow, dog, horse, sheep
  • *Vehicles*: airplane, bicycle, boat, bus, car, motorbike, train
  • *Indoor*: bottle, chair, dining table, potted plant, sofa, tv/monitor

• Dataset size (by 2012): 11.5K training/validation images, 27K bounding boxes, 7K segmentations

http://host.robots.ox.ac.uk/pascal/VOC/
Progress on PASCAL detection

PASCAL VOC

Before CNNs

After CNNs

mean Average Precision (mAP)

year

Current benchmark: COCO

What is COCO?

COCO is a large-scale object detection, segmentation, and captioning dataset. COCO has several features:

- Object segmentation
- Recognition in context
- Superpixel stuff segmentation
- 330K images (>200K labeled)
- 1.5 million object instances
- 80 object categories
- 91 stuff categories
- 5 captions per image
- 250,000 people with keypoints

http://cocodataset.org/#home
COCO dataset: Tasks

- image classification
- object detection
- semantic segmentation
- instance segmentation

- Also: keypoint prediction, captioning, question answering…
COCO detection metrics

### Average Precision (AP):

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP</td>
<td>% AP at IoU=.50:.05:.95 (primary challenge metric)</td>
</tr>
<tr>
<td>$AP_{IoU=.50}$</td>
<td>% AP at IoU=.50 (PASCAL VOC metric)</td>
</tr>
<tr>
<td>$AP_{IoU=.75}$</td>
<td>% AP at IoU=.75 (strict metric)</td>
</tr>
</tbody>
</table>

### AP Across Scales:

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AP_{small}$</td>
<td>% AP for small objects: area &lt; $32^2$</td>
</tr>
<tr>
<td>$AP_{medium}$</td>
<td>% AP for medium objects: $32^2 &lt; \text{area} &lt; 96^2$</td>
</tr>
<tr>
<td>$AP_{large}$</td>
<td>% AP for large objects: area &gt; $96^2$</td>
</tr>
</tbody>
</table>

### Average Recall (AR):

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AR_{max=1}$</td>
<td>% AR given 1 detection per image</td>
</tr>
<tr>
<td>$AR_{max=10}$</td>
<td>% AR given 10 detections per image</td>
</tr>
<tr>
<td>$AR_{max=100}$</td>
<td>% AR given 100 detections per image</td>
</tr>
</tbody>
</table>

### AR Across Scales:

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AR_{small}$</td>
<td>% AR for small objects: area &lt; $32^2$</td>
</tr>
<tr>
<td>$AR_{medium}$</td>
<td>% AR for medium objects: $32^2 &lt; \text{area} &lt; 96^2$</td>
</tr>
<tr>
<td>$AR_{large}$</td>
<td>% AR for large objects: area &gt; $96^2$</td>
</tr>
</tbody>
</table>

- **Leaderboard:** [http://cocodataset.org/#detection-leaderboard](http://cocodataset.org/#detection-leaderboard)
- **Official COCO challenges no longer include detection**
  - Emphasis has shifted to instance segmentation and dense semantic segmentation
Approaches to detection: Sliding windows

• Slide a window across the image and evaluate a detection model at each location
  • Thousands of windows to evaluate: efficiency and low false positive rates are essential
  • Difficult to extend to a large range of scales, aspect ratios
Approaches to detection: Object proposals

- Generate and evaluate a few hundred region proposals
  - Proposal mechanism can take advantage of low-level perceptual organization cues
  - Proposal mechanism can be category-specific or category-independent, hand-crafted or trained
  - Classifier can be slower but more powerful
Selective search for detection

• Use hierarchical segmentation: start with small superpixels and merge based on diverse cues

Selective search for detection

Evaluation of region proposals

J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders, Selective Search for Object Recognition, IJCV 2013
Selective search for detection

- Feature extraction: color SIFT, codebook of size 4K, spatial pyramid with four levels = 360K dimensions

Approaches to detection

• Before ~2010, dominated by sliding windows
• 2010-2013: proposal-driven
• Deep learning approaches started as proposal-driven, but have evolved back toward sliding windows
• Most recently, “global” methods are becoming more common
R-CNN: Region proposals + CNN features

Source: R. Girshick

R-CNN details

- **Regions**: ~2000 Selective Search proposals
- **Network**: AlexNet *pre-trained* on ImageNet (1000 classes), *fine-tuned* on PASCAL (21 classes)
- **Final detector**: warp proposal regions, extract fc7 network activations (4096 dimensions), classify with linear SVM
- **Bounding box regression** to refine box locations
- **Performance**: mAP of 53.7% on PASCAL 2010 (vs. 35.1% for Selective Search and 33.4% for Deformable Part Models)
R-CNN pros and cons

- **Pros**
  - Much more accurate than previous approaches!
  - Any deep architecture can immediately be “plugged in”

- **Cons**
  - Not a single end-to-end system
    - Fine-tune network with softmax classifier (log loss)
    - Train post-hoc linear SVMs (hinge loss)
    - Train post-hoc bounding-box regressions (least squares)
  - Training was slow (84h), took up a lot of storage
    - 2000 CNN passes per image
  - Inference (detection) was slow (47s / image with VGG16)
Fast R-CNN

- Forward whole image through ConvNet
- Conv5 feature map of image
- RoI Pooling layer
- Fully-connected layers
- Linear
- Linear + softmax
- Softmax classifier
- Region proposals
- Bounding-box regressors

Source: R. Girshick

R. Girshick, Fast R-CNN, ICCV 2015
RoI pooling

• “Crop and resample” a fixed-size feature representing a region of interest out of the outputs of the last conv layer
  • Use nearest-neighbor interpolation of coordinates, max pooling

Source: R. Girshick, K. He
RoI pooling illustration

![RoI pooling illustration](Image source)
Prediction

- For each RoI, network predicts probabilities for $C + 1$ classes (class 0 is background) and four bounding box offsets for $C$ classes

R. Girshick, Fast R-CNN, ICCV 2015
Fast R-CNN training

Log loss + smooth L1 loss

Multi-task loss

Linear + softmax

Linear

FCs

ConvNet

Trainable

Source: R. Girshick

R. Girshick, Fast R-CNN, ICCV 2015
Multi-task loss

- Loss for ground truth class $y$, predicted class probabilities $P(y)$, ground truth box $b$, and predicted box $\hat{b}$:

$$L(y, P, b, \hat{b}) = -\log P(y) + \lambda \mathbb{I}[y \geq 1]L_{\text{reg}}(b, \hat{b})$$

  - Softmax loss
  - Regression loss

- Regression loss: smooth $L_1$ loss on top of log space offsets relative to proposal

$$L_{\text{reg}}(b, \hat{b}) = \sum_{i=\{x,y,w,h\}} \text{smooth}_{L_1}(b_i - \hat{b}_i)$$

$$\text{smooth}_{L_1}(x) = \begin{cases} 
0.5x^2 & \text{if } |x| < 1 \\
|x| - 0.5 & \text{otherwise}
\end{cases}$$
Bounding box regression

Ground truth box

Target offset to predict*

Predicted offset

Loss

Region proposal (a.k.a default box, prior, reference, anchor)

Predicted box

*Typically in transformed, normalized coordinates
ROI pooling: Backpropagation

- Similar to max pooling, but has to take into account overlap of pooling regions

Source: Ross Girshick
ROI pooling: Backpropagation

• Similar to max pooling, but has to take into account overlap of pooling regions

\[
\frac{\partial e}{\partial x_i} = \sum_r \sum_j \frac{\partial e}{\partial z_{rj}} \frac{\partial z_{rj}}{\partial x_i} = \sum_r \sum_j \mathbb{I}[i = i^*(r,j)] \frac{\partial e}{\partial z_{rj}}
\]

Over regions $r$, RoI indices $j$

1 if $r, j$ "pooled" input $i$; 0 o/w

Source: Ross Girshick
Mini-batch sampling

- Sample a few images (e.g., 2)
- Sample many regions from each image (64)

Source: R. Girshick, K. He
Fast R-CNN results

<table>
<thead>
<tr>
<th></th>
<th>Fast R-CNN</th>
<th>R-CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train time (h)</td>
<td>9.5</td>
<td>84</td>
</tr>
<tr>
<td>- Speedup</td>
<td>8.8x</td>
<td></td>
</tr>
<tr>
<td>Test time / image</td>
<td>0.32s</td>
<td>47.0s</td>
</tr>
<tr>
<td>- Test speedup</td>
<td>146x</td>
<td></td>
</tr>
<tr>
<td>mAP</td>
<td>66.9%</td>
<td>66.0%</td>
</tr>
</tbody>
</table>

(vs. 53.7% for AlexNet)

Timings exclude object proposal time, which is equal for all methods.
All methods use VGG16.

Source: R. Girshick
Faster R-CNN

Region proposal network (RPN)

- Slide a small window (3x3) over the conv5 layer
  - Predict object/no object
  - Regress bounding box coordinates with reference to anchors (3 scales x 3 aspect ratios)
One network, four losses

Source: R. Girshick, K. He
Faster R-CNN results

<table>
<thead>
<tr>
<th>system</th>
<th>time</th>
<th>07 data</th>
<th>07+12 data</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-CNN</td>
<td>~50s</td>
<td>66.0</td>
<td>-</td>
</tr>
<tr>
<td>Fast R-CNN</td>
<td>~2s</td>
<td>66.9</td>
<td>70.0</td>
</tr>
<tr>
<td>Faster R-CNN</td>
<td>198ms</td>
<td>69.9</td>
<td>73.2</td>
</tr>
</tbody>
</table>

detection mAP on PASCAL VOC 2007, with VGG-16 pre-trained on ImageNet
Object detection progress

![Graph showing object detection progress over years, with annotations for Before CNNs, R-CNNv1, After CNNs, Faster R-CNN, and Fast R-CNN. The x-axis represents years from 2006 to 2016, and the y-axis represents mean Average Precision (mAP) in percentage.](image-url)
Review: R-CNN

Review: Fast R-CNN

- Forward whole image through ConvNet
- "conv5" feature map of image
- "RoI Pooling" layer
- Fully-connected layers
- Bounding-box regressors
- Linear
- Linear + softmax
- Softmax classifier
- Region proposals

R. Girshick, Fast R-CNN, ICCV 2015
Review: Faster R-CNN

Streamlined detection architectures

- The Faster R-CNN pipeline separates proposal generation and region classification:

- Is it possible do detection in one shot?
YOLO

• Divide the image into a coarse grid and directly predict class label and a few candidate boxes for each grid cell

1. Take conv feature maps at 7x7 resolution
2. Add two FC layers to predict, at each location, a score for each class and 2 bboxes w/ confidences
   • For PASCAL, output is 7x7x30 (30 = 20 + 2*(4+1))

YOLO

- Objective function:

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

- **Objective function:**

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

- Cell i contains object, predictor j is responsible for it
- Small deviations matter less for larger boxes than for smaller boxes
- Confidence for object
- Confidence for no object
- Down-weight loss from boxes that don’t contain objects ($\lambda_{\text{noobj}} = 0.5$)
- Class probability
YOLO: Results

- Each grid cell predicts only two boxes and can only have one class – this limits the number of nearby objects that can be predicted.
- Localization accuracy suffers compared to Fast(er) R-CNN due to coarser features, errors on small boxes.
- 7x speedup over Faster R-CNN (45-155 FPS vs. 7-18 FPS).

Performance on PASCAL 2007
SSD

- Similarly to YOLO, predict bounding boxes directly from conv maps
- Unlike YOLO, do not use FC layers and predict different size boxes from conv maps at different resolutions
- Similarly to RPN, use anchors

SSD: Results (PASCAL 2007)

- More accurate and faster than YOLO and Faster R-CNN

<table>
<thead>
<tr>
<th>Method</th>
<th>mAP</th>
<th>FPS</th>
<th>batch size</th>
<th># Boxes</th>
<th>Input resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faster R-CNN (VGG16)</td>
<td>73.2</td>
<td>7</td>
<td>1</td>
<td>~ 6000</td>
<td>~ 1000 × 600</td>
</tr>
<tr>
<td>Fast YOLO</td>
<td>52.7</td>
<td>155</td>
<td>1</td>
<td>98</td>
<td>448 × 448</td>
</tr>
<tr>
<td>YOLO (VGG16)</td>
<td>66.4</td>
<td>21</td>
<td>1</td>
<td>98</td>
<td>448 × 448</td>
</tr>
<tr>
<td>SSD300</td>
<td>74.3</td>
<td>46</td>
<td>1</td>
<td>8732</td>
<td>300 × 300</td>
</tr>
<tr>
<td>SSD512</td>
<td>76.8</td>
<td>19</td>
<td>1</td>
<td>24564</td>
<td>512 × 512</td>
</tr>
<tr>
<td>SSD300</td>
<td>74.3</td>
<td>59</td>
<td>8</td>
<td>8732</td>
<td>300 × 300</td>
</tr>
<tr>
<td>SSD512</td>
<td>76.8</td>
<td>22</td>
<td>8</td>
<td>24564</td>
<td>512 × 512</td>
</tr>
</tbody>
</table>
YOLO v2

- Remove FC layer, do convolutional prediction with anchor boxes instead
- Increase resolution of input images and conv feature maps
- Improve accuracy using batch normalization and other tricks

VOC 2007 results

YouTube demo

J. Redmon and A. Farhadi, YOLO9000: Better, Faster, Stronger, CVPR 2017
Multi-resolution prediction

- SSD predicts boxes of different size from different conv maps, but each level of resolution has its own predictors and higher-level context does not get propagated back to lower-level feature maps.
- Can we have a more elegant multi-resolution prediction architecture?
Feature pyramid networks

- Improve predictive power of lower-level feature maps by adding contextual information from higher-level feature maps
- Predict different sizes of bounding boxes from different levels of the pyramid (but share parameters of predictors)

RetinaNet

- Combine feature pyramid network with focal loss to reduce the standard cross-entropy loss for well-classified examples

T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollar, Focal loss for dense object detection, ICCV 2017
RetinaNet

- Combine feature pyramid network with focal loss to reduce the standard cross-entropy loss for well-classified examples

\[ CE(p_t) = - \log(p_t) \]
\[ FL(p_t) = -(1 - p_t)^{\gamma} \log(p_t) \]

RetinaNet: Results

T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollar, Focal loss for dense object detection, ICCV 2017
YOLO v3

YOLOv3: An Incremental Improvement

Joseph Redmon, Ali Farhadi
University of Washington

Abstract

We present some updates to YOLO! We made a bunch of little design changes to make it better. We also trained this new network that's pretty swell. It's a little bigger than last time but more accurate. It's still fast though, don't worry. At 320 × 320 YOLOv3 runs in 22 ms at 28.2 mAP, as accurate as SSD but three times faster. When we look at the old .5 IOU mAP detection metric YOLOv3 is quite good. It achieves 57.9 AP₅₀ in 51 ms on a Titan X, compared to 57.5 AP₅₀ in 198 ms by RetinaNet, similar performance but 3.8× faster. As always, all the code is online at https://pjreddie.com/yolo/.

1. Introduction

Summary so far

• R-CNN: region proposals + CNN on cropped, resampled regions
• Fast R-CNN: region proposals + RoI pooling on top of a conv feature map
• Faster R-CNN: RPN + RoI pooling
• Next generation of detectors: YOLO, SSD, RetinaNet
  • Direct prediction of BB offsets, class scores on top of conv feature maps
  • Get better context by combining feature maps at multiple resolutions
• Most recent developments: architectures borrowed from dense prediction, transformers
**CornerNet**

**Fig. 1.** We detect an object as a pair of bounding box corners grouped together. A convolutional network outputs a heatmap for all top-left corners, a heatmap for all bottom-right corners, and an embedding vector for each detected corner. The network is trained to predict similar embeddings for corners that belong to the same object.

H. Law and J. Deng, *CornerNet: Detecting Objects as Paired Keypoints*, ECCV 2018
**CornerNet**

**Fig. 2.** Often there is no local evidence to determine the location of a bounding box corner. We address this issue by proposing a new type of pooling layer.

H. Law and J. Deng, [CornerNet: Detecting Objects as Paired Keypoints](https://arxiv.org/abs/1808.01244), ECCV 2018
Detection Transformer (DETR)

N. Carion et al. End-to-end object detection with transformers, arXiv 2020