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
• Generic object detection before deep learning
• Zoo of deep detection approaches
  • R-CNN
  • Fast R-CNN
  • Faster R-CNN
  • Yolo
  • SSD
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 considered a true positive

Ground truth (GT)
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, plot Recall-Precision curve and compute Average Precision (area under the curve)
- Take mean of AP over classes to get mAP

**Precision:**
true positive detections / total detections

**Recall:**
true positive detections / total positive test instances
PASCAL VOC Challenge (2005-2012)

• 20 challenge classes:
  • Person
  • Animals: bird, cat, cow, dog, horse, sheep
  • Vehicles: aeroplane, 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

mean Average Precision (mAP)

year
Detection before deep learning
Conceptual approach: Sliding window detection

- 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
Histories of oriented gradients (HOG)

- Partition image into blocks and compute histogram of gradient orientations in each block

N. Dalal and B. Triggs, [Histograms of Oriented Gradients for Human Detection](http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.158.4005), CVPR 2005
Pedestrian detection with HOG

- Train a pedestrian template using a linear support vector machine

positive training examples

negative training examples

N. Dalal and B. Triggs, Histograms of Oriented Gradients for Human Detection, CVPR 2005
Pedestrian detection with HOG

- Train a pedestrian template using a linear support vector machine
- At test time, convolve feature map with template
- Find local maxima of response
- For multi-scale detection, repeat over multiple levels of a HOG pyramid

Example detections

[Dalal and Triggs, CVPR 2005]
Discriminative part-based models

- Single rigid template usually not enough to represent a category
  - Many objects (e.g. humans) are articulated, or have parts that can vary in configuration

- Many object categories look very different from different viewpoints, or from instance to instance
Discriminative part-based models

Discriminative part-based models

Multiple components

P. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan, Object Detection with Discriminatively Trained Part Based Models, PAMI 32(9), 2010
Discriminative part-based models

P. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan, Object Detection with Discriminatively Trained Part Based Models, PAMI 32(9), 2010
Progress on PASCAL detection

PASCAL VOC

Before CNNs

mean Average Precision (mAP)

year
Conceptual approach: Proposal-driven detection

- 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

J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders, **Selective Search for Object Recognition**, IJCV 2013
Another proposal method: EdgeBoxes

• Box score: number of edges in the box minus number of edges that overlap the box boundary
• Uses a trained edge detector
• Uses efficient data structures (incl. integral images) for fast evaluation
• Gets 75% recall with 800 boxes (vs. 1400 for Selective Search), is 40 times faster

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**
  • Accurate!
  • 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 is slow (84h), takes a lot of disk space
    • 2000 CNN passes per image
  • Inference (detection) is slow (47s / image with VGG16)
Fast R-CNN

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

Source: 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**

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.88</td>
<td>0.44</td>
<td>0.14</td>
<td>0.16</td>
<td>0.37</td>
<td>0.77</td>
<td>0.96</td>
<td>0.27</td>
<td></td>
</tr>
<tr>
<td>0.19</td>
<td>0.45</td>
<td>0.57</td>
<td>0.16</td>
<td>0.63</td>
<td>0.29</td>
<td>0.71</td>
<td>0.70</td>
<td></td>
</tr>
<tr>
<td>0.66</td>
<td>0.26</td>
<td>0.82</td>
<td>0.64</td>
<td>0.54</td>
<td>0.73</td>
<td>0.59</td>
<td>0.26</td>
<td></td>
</tr>
<tr>
<td>0.85</td>
<td>0.34</td>
<td>0.76</td>
<td>0.84</td>
<td>0.29</td>
<td>0.75</td>
<td>0.62</td>
<td>0.25</td>
<td></td>
</tr>
<tr>
<td>0.32</td>
<td>0.74</td>
<td>0.21</td>
<td>0.39</td>
<td>0.34</td>
<td>0.03</td>
<td>0.33</td>
<td>0.48</td>
<td></td>
</tr>
<tr>
<td>0.20</td>
<td>0.14</td>
<td>0.16</td>
<td>0.13</td>
<td>0.73</td>
<td>0.65</td>
<td>0.96</td>
<td>0.32</td>
<td></td>
</tr>
<tr>
<td>0.19</td>
<td>0.69</td>
<td>0.09</td>
<td>0.86</td>
<td>0.88</td>
<td>0.07</td>
<td>0.01</td>
<td>0.48</td>
<td></td>
</tr>
<tr>
<td>0.83</td>
<td>0.24</td>
<td>0.97</td>
<td>0.04</td>
<td>0.24</td>
<td>0.35</td>
<td>0.50</td>
<td>0.91</td>
<td></td>
</tr>
</tbody>
</table>
Prediction

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

Fast R-CNN training

Log loss + smooth L1 loss

Linear + softmax

Linear

FCs

ConvNet

Multi-task loss

Trainable

Source: R. Girshick

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 L1 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
## 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>1x</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>1x</td>
</tr>
<tr>
<td>mAP</td>
<td>66.9%</td>
<td>66.0%</td>
</tr>
</tbody>
</table>

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

(vs. 53.7% for AlexNet)

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

mean Average Precision (mAP)

year


Before CNNs

R-CNNv1

Fast R-CNN

Faster R-CNN

After CNNs
Streamlined detection architectures

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

  - Proposal generation:
    - RPN
    - Region Proposals
  - Region classification:
    - RoI pooling
    - Region of Interest (RoI) features
    - Classification + Regression
    - Detections

- Is it possible to do detection in one shot?

  - Conv feature map of the entire image
  - Classification + Regression
  - Detections
YOLO

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

YOLO

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:

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

- **Objective function:**

\[
\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{i,j} \left[ (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] 
\]

\[
+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{i,j} \left[ (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right] 
\]

\[
+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{i,j} \left( C_i - \hat{C}_i \right)^2 
\]

\[
+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{i,j} \left( C_i - \hat{C}_i \right)^2 
\]

\[
+ \sum_{i=0}^{S^2} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 
\]

- 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

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

YouTube demo

J. Redmon and A. Farhadi, YOLO9000: Better, Faster, Stronger, CVPR 2017
Newer 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
Newer benchmark: COCO

J. Huang et al., Speed/accuracy trade-offs for modern convolutional object detectors, CVPR 2017
COCO detection metrics

- Leaderboard: [http://cocodataset.org/#detection-leaderboard](http://cocodataset.org/#detection-leaderboard)
- Current best mAP: ~52%
- Official COCO challenges no longer include detection
- More emphasis on instance segmentation and dense segmentation
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

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

![Graph showing RetinaNet-50 and RetinaNet-101 performance](image)

<table>
<thead>
<tr>
<th>Method</th>
<th>AP</th>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[A] YOLOv2†</td>
<td>21.6</td>
<td>25</td>
</tr>
<tr>
<td>[B] SSD321</td>
<td>28.0</td>
<td>61</td>
</tr>
<tr>
<td>[C] DSSD321</td>
<td>28.0</td>
<td>85</td>
</tr>
<tr>
<td>[D] R-FCN‡</td>
<td>29.9</td>
<td>85</td>
</tr>
<tr>
<td>[E] SSD513</td>
<td>31.2</td>
<td>125</td>
</tr>
<tr>
<td>[F] DSSD513</td>
<td>33.2</td>
<td>156</td>
</tr>
<tr>
<td>[G] FPN FRCN</td>
<td>36.2</td>
<td>172</td>
</tr>
<tr>
<td>RetinaNet-50</td>
<td>32.5</td>
<td>73</td>
</tr>
<tr>
<td>RetinaNet-101</td>
<td>34.4</td>
<td>90</td>
</tr>
<tr>
<td>RetinaNet-101-800</td>
<td>37.8</td>
<td>198</td>
</tr>
</tbody>
</table>

†Not plotted ‡Extrapolated time

Deconvolutional SSD

- Improve performance of SSD by increasing resolution through learned “deconvolutional” layers

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 AP_{50} in 51 ms on a Titan X, compared to 57.5 AP_{50} 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: Object detection with CNNs

- 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
  - Direct prediction of BB offsets, class scores on top of conv feature maps
  - Get better context by combining feature maps at multiple resolutions
Beyond detection

- Image classification
- Object detection
- Semantic segmentation
- Instance segmentation

- Also: keypoint prediction, captioning, question answering...
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 + softmax
- Linear
- Softmax classifier
- Region proposals

R. Girshick, Fast R-CNN, ICCV 2015