From image classification to object detection

**Image classification**

**Object detection**

*Image source*
Outline

• Task definition and evaluation
• Detection strategies: sliding windows vs. region proposals
• Two-stage detectors:
  • R-CNN
  • Fast R-CNN
  • Faster R-CNN
• Single-stage and multi-resolution detectors
• Recent trends
What are some challenges of object detection?

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

**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: 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

mean Average Precision (mAP)

Before CNNs

After CNNs

year

More challenging 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

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Precision (AP)</td>
<td>% AP at IoU=.50:.05:.95 (primary challenge metric)</td>
</tr>
<tr>
<td>( \text{AP}_{\text{IoU=.50}} )</td>
<td>% AP at IoU=.50 (PASCAL VOC metric)</td>
</tr>
<tr>
<td>( \text{AP}_{\text{IoU=.75}} )</td>
<td>% AP at IoU=.75 (strict metric)</td>
</tr>
<tr>
<td>AP Across Scales:</td>
<td></td>
</tr>
<tr>
<td>( \text{AP}^{\text{small}} )</td>
<td>% AP for small objects: area &lt; 32(^2)</td>
</tr>
<tr>
<td>( \text{AP}^{\text{medium}} )</td>
<td>% AP for medium objects: 32(^2) &lt; area &lt; 96(^2)</td>
</tr>
<tr>
<td>( \text{AP}^{\text{large}} )</td>
<td>% AP for large objects: area &gt; 96(^2)</td>
</tr>
<tr>
<td>Average Recall (AR)</td>
<td>% AR for 1 detection per image</td>
</tr>
<tr>
<td>( \text{AR}^{\text{max}=1} )</td>
<td>% AR for 10 detections per image</td>
</tr>
<tr>
<td>( \text{AR}^{\text{max}=10} )</td>
<td>% AR for 100 detections per image</td>
</tr>
<tr>
<td>AR Across Scales:</td>
<td></td>
</tr>
<tr>
<td>( \text{AR}^{\text{small}} )</td>
<td>% AR for small objects: area &lt; 32(^2)</td>
</tr>
<tr>
<td>( \text{AR}^{\text{medium}} )</td>
<td>% AR for medium objects: 32(^2) &lt; area &lt; 96(^2)</td>
</tr>
<tr>
<td>( \text{AR}^{\text{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
Outline

• Task definition and evaluation
• Detection strategies: sliding windows vs. region proposals
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

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
Detection strategies: Summary

• Before ~2010, dominated by sliding windows
• 2010-2013: proposal-driven
• Deep learning approaches started as proposal-driven (two-stage), but have evolved back toward sliding windows (single-stage)
• Most recent methods are converging with dense prediction
Outline

• Task definition and evaluation
• Detection strategies: sliding windows vs. region proposals
• Two-stage detectors:
  • R-CNN
  • Fast R-CNN
  • Faster R-CNN
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

ConvNet

Region proposals

Softmax classifier

Linear + softmax

Bounding-box regressors

Fully-connected layers

RoI Pooling layer

Conv5 feature map of image

Forward whole image through ConvNet

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
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

[Diagram showing ground truth box, region proposal (a.k.a default box, prior, reference, anchor), predicted box, target offset to predict, predicted offset, and loss. The text notes that the target offset is usually predicted in transformed, normalized coordinates.]
Fast R-CNN training

Log loss + smooth L1 loss

Multi-task loss

Linear + softmax

Linear

FCs

Trainable

ConvNet

Source: R. Girshick

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

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

(vs. 53.7% for AlexNet)

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

Region proposal network (RPN)

- Idea: put an “anchor box” of fixed size over each position in the feature map and try to predict whether this box is likely to contain an object

Figure source: J. Johnson
Region proposal network (RPN)

• Idea: put an “anchor box” of fixed size over each position in the feature map and try to predict whether this box is likely to contain an object

Figure source: J. Johnson
Region proposal network (RPN)

- Idea: put an “anchor box” of fixed size over each position in the feature map and try to predict whether this box is likely to contain an object

Figure source: J. Johnson
Region proposal network (RPN)

- Idea: put an “anchor box” of fixed size over each position in the feature map and try to predict whether this box is likely to contain an object
- Introduce anchor boxes at multiple scales and aspect ratios to handle a wider range of object sizes and shapes

Figure source: J. Johnson
Faster R-CNN RPN design

- 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 notable improvements after the introduction of CNNs. The x-axis represents the years from 2006 to 2016, while the y-axis shows the mean average precision (mAP) percentage. Key points include:

- Before CNNs:
  - 2006: Around 10%
  - 2009: Around 20%

- After CNNs:
  - 2013: Faster R-CNN: Approximately 60%
  - 2014: Fast R-CNN: Approximately 65%
  - 2015: R-CNNv1: Approximately 70%
  - 2016: After CNNs: Improvements continue to rise.]

- Faster R-CNN
- Fast R-CNN
- R-CNNv1
- After CNNs
Outline

• Task definition and evaluation
• Detection strategies
• Two-stage detectors
  • R-CNN
  • Fast R-CNN
  • Faster R-CNN
• Single-stage and multi-resolution detectors
Single-stage detection architectures

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

  RPN → Region Proposals → RoI pooling → Classification + Regression → Detections

  Conv feature map of the entire image → Region Proposals

• 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 $7 \times 7 \times 30$ ($30 = 20 + 2 \times (4 + 1)$)

YOLO

- **Objective function:**

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

- **Regression**
- **Object/no object confidence**
- **Class prediction**
**YOLO**

- **Objective function:**

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

- 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
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: predict 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

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

\[
\begin{align*}
CE(p_t) &= -\log(p_t) \\
FL(p_t) &= -(1 - p_t)^\gamma \log(p_t)
\end{align*}
\]

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 $320 \times 320$ 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 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
• More recent trends: architectures borrowed from dense prediction, transformers
Outline

• Task definition and evaluation
• Detection strategies
• Two-stage detectors
  • R-CNN
  • Fast R-CNN
  • Faster R-CNN
• Single-stage and multi-resolution detectors
• Recent trends
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/1802.05774), ECCV 2018
CenterNet

- Use an additional center point to verify predictions:

![Diagram showing use of center point for verification]

<table>
<thead>
<tr>
<th>Method</th>
<th>FD</th>
<th>FD_5</th>
<th>FD_25</th>
<th>FD_50</th>
<th>FD_8</th>
<th>FD_m</th>
<th>FD_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>CornerNet511-52</td>
<td>40.4</td>
<td>35.2</td>
<td>39.4</td>
<td>46.7</td>
<td>62.5</td>
<td>36.9</td>
<td>28.0</td>
</tr>
<tr>
<td>CenterNet511-52</td>
<td>35.1</td>
<td>30.7</td>
<td>34.2</td>
<td>40.8</td>
<td>53.0</td>
<td>31.3</td>
<td>24.4</td>
</tr>
<tr>
<td>CornerNet511-104</td>
<td>37.8</td>
<td>32.7</td>
<td>36.8</td>
<td>43.8</td>
<td>60.3</td>
<td>33.2</td>
<td>25.1</td>
</tr>
<tr>
<td>CenterNet511-104</td>
<td>32.4</td>
<td>28.2</td>
<td>31.6</td>
<td>37.5</td>
<td>50.7</td>
<td>27.1</td>
<td>23.0</td>
</tr>
</tbody>
</table>

Table 3: Comparison of the false discovery rates (%) of CornerNet and CenterNet on the MS-COCO validation dataset. The results suggest that CenterNet avoids a large number of incorrect bounding boxes, especially for small incorrect bounding boxes.

K. Duan et al. [CenterNet: Keypoint Triplets for Object Detection], ICCV 2019
Figure 2: Architecture of CenterNet. A convolutional backbone network applies cascade corner pooling and center pooling to output two corner heatmaps and a center keypoint heatmap, respectively. Similar to CornerNet, a pair of detected corners and the similar embeddings are used to detect a potential bounding box. Then the detected center keypoints are used to determine the final bounding boxes.

K. Duan et al. CenterNet: Keypoint Triplets for Object Detection, ICCV 2019
Detection Transformer (DETR)

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