Object detection
Outline

• Task definition and evaluation
• Two-stage detectors:
  • R-CNN
  • Fast R-CNN
  • Faster R-CNN
• Single-stage and multi-resolution detectors
• Other detectors: 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
  • But how?

Ground truth (GT)
Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?

Source: J. Johnson
Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU):

\[
\text{Area of Intersection} \quad \text{Area of Union}
\]

Source: J. Johnson
Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU):

\[
\frac{\text{Area of Intersection}}{\text{Area of Union}}
\]

Source: J. Johnson
Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU):

\[
\text{IoU} = \frac{\text{Area of Intersection}}{\text{Area of Union}}
\]

IoU > 0.5 is “decent”

Source: J. Johnson
Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU):

\[
\frac{\text{Area of Intersection}}{\text{Area of Union}}
\]

IoU > 0.5 is “decent”
IoU > 0.7 is “pretty good”

Source: [J. Johnson](https://example.com)
Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU):

\[
\text{IoU} = \frac{\text{Area of Intersection}}{\text{Area of Union}}
\]

- IoU $> 0.5$ is “decent”
- IoU $> 0.7$ is “pretty good”
- IoU $> 0.9$ is “almost perfect”

Source: J. Johnson
Non-maximum suppression

Problem: Detectors often output many overlapping detections

Solution: Post-process raw detections using Non-Max Suppression (NMS)

Source: J. Johnson
Non-maximum suppression

Problem: Detectors often output many overlapping detections

Solution: Post-process raw detections using Non-Max Suppression (NMS)

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1

Source: J. Johnson
Non-maximum suppression

Problem: Detectors often output many overlapping detections

Solution: Post-process raw detections using Non-Max Suppression (NMS)

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1

Source: J. Johnson
Non-maximum suppression

Problem: Detectors often output many overlapping detections

Solution: Post-process raw detections using Non-Max Suppression (NMS)

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1

Source: J. Johnson
Non-maximum suppression

Problem: Detectors often output many overlapping detections

Solution: Post-process raw detections using Non-Max Suppression (NMS)

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1

Source: J. Johnson
Non-maximum suppression

Problem: Detectors often output many overlapping detections

Solution: Post-process raw detections using Non-Max Suppression (NMS)

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1

Source: J. Johnson
Problem: Detectors often output many overlapping detections.

Solution: Post-process raw detections using Non-Max Suppression (NMS).

1. Select next highest-scoring box.
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7).
3. If any boxes remain, GOTO 1.

Source: J. Johnson
Non-maximum suppression

How would NMS do on an image like this?
- It will eliminate “good” boxes when objects are highly overlapping

Source: J. Johnson
Evaluating object detectors

1. Run object detector on all test images (with NMS)
2. For each category, compute **Average Precision (AP)** or area under **Precision vs. Recall Curve**

Source: J. Johnson
Evaluating object detectors

1. Run object detector on all test images (with NMS)
2. For each category, compute **Average Precision (AP)** or area under **Precision vs. Recall Curve**

1. For each detection (highest 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

\[
\text{Precision} = \frac{\text{true positive detections}}{\text{total detections so far}}
\]
\[
\text{Recall} = \frac{\text{true positive detections}}{\text{true positive test instances}}
\]

Source: [J. Johnson](https://www.johnson.com)
Evaluating object detectors

1. Run object detector on all test images (with NMS)
2. For each category, compute **Average Precision (AP)**
or area under **Precision vs. Recall Curve**
   
   1. For each detection (highest 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

   Precision = 1/1 = 1.0
   Recall = 1/3 = 0.33

Source: J. Johnson
Evaluating object detectors

1. Run object detector on all test images (with NMS)
2. For each category, compute **Average Precision (AP)** or area under **Precision vs. Recall Curve**

   1. For each detection (highest 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

All ground-truth boxes

Match: IoU > 0.5

Precision = 2/2 = 1.0
Recall = 2/3 = 0.67

Source: J. Johnson
Evaluating object detectors

1. Run object detector on all test images (with NMS)
2. For each category, compute **Average Precision (AP)** or area under **Precision vs. Recall Curve**

   1. For each detection (highest 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

Source: J. Johnson
Evaluating object detectors

1. Run object detector on all test images (with NMS)
2. For each category, compute **Average Precision (AP)** or area under **Precision vs. Recall Curve**

   1. For each detection (highest 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

   

Source: J. Johnson
Evaluating object detectors

1. Run object detector on all test images (with NMS)
2. For each category, compute **Average Precision (AP)** or area under **Precision vs. Recall Curve**

   1. For each detection (highest 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

   All ground-truth boxes

   - Precision = 3/5 = 0.6
   - Recall = 3/3 = 1.0

Source: J. Johnson
Object detection: Outline

- Task definition and evaluation
- Two-stage detectors:
  - R-CNN
  - Fast R-CNN
  - Faster R-CNN
- Single-stage and multi-resolution detectors
- Other detectors: CornerNet, DETR
Evaluating object detectors

1. Run object detector on all test images (with NMS)
2. For each category, compute **Average Precision (AP)** or area under **Precision vs. Recall Curve**
   
   1. For each detection (highest 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

Source: J. Johnson
1. Run object detector on all test images (with NMS)
2. For each category, compute **Average Precision (AP)** or area under **Precision vs. Recall 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”

Source: J. Johnson
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)

0% 10% 20% 30% 40% 50% 60% 70% 80%

year

Before CNNs

After CNNs

More recent 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>Average Precision (AP):</th>
<th>% AP at IoU=.50:.05:.95 (primary challenge metric)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP</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:      | % AP for small objects: area < 32²                    |
| AP_small               | % AP for medium objects: 32² < area < 96²            |
| AP_medium              | % AP for large objects: area > 96²                   |

| Average Recall (AR):   | % AR given 1 detection per image                      |
| AR_max=1               | % AR given 10 detections per image                    |
| AR_max=10              | % AR given 100 detections per image                   |

| AR Across Scales:      | % AR for small objects: area < 32²                    |
| AR_small               | % AR for medium objects: 32² < area < 96²            |
| AR_large               | % AR for large objects: area > 96²                   |

- Leaderboard: [http://cocodataset.org/#detection-leaderboard](http://cocodataset.org/#detection-leaderboard)
- Not updated since 2020
Object detection: Outline

- Task definition and evaluation
- Two-stage detectors
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

Softmax classifier

Linear + softmax

Linear

Bounding-box regressors

Fully-connected layers

RoI Pooling layer

Conv5 feature map of image

Forward whole image through ConvNet

Region proposals

Source: R. Girshick

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

Input Image
(e.g., 3 x 640 x 480)

Source: J. Johnson
RoI pooling

Input Image (e.g., 3 x 640 x 480)

Feature map (e.g., 512 x 20 x 15)

Source: J. Johnson
RoI pooling

Input Image
(e.g., 3 x 640 x 480)

Feature map
(e.g., 512 x 20 x 15)

Project proposal onto features

Source: J. Johnson
RoI pooling

Project proposal onto features

“Snap” proposal to feature grid

Want fixed-size RoI representation regardless of its size

Input Image (e.g., 3 x 640 x 480)

Feature map (e.g., 512 x 20 x 15)

Source: J. Johnson
RoI pooling

Project proposal onto features

Divide into (roughly) equal sub-regions

Input Image (e.g., 3 x 640 x 480)

Feature map (e.g., 512 x 20 x 15)

Source: J. Johnson
RoI pooling

Project proposal onto features

Divide into (roughly) equal sub-regions

Max-pool within each sub-region

Region features (here 512 x 2 x 2; In practice 512 x 7 x 7)

Input Image (e.g., 3 x 640 x 480)

Feature map (e.g., 512 x 20 x 15)

Source: J. Johnson
RoI pooling illustration

```
<table>
<thead>
<tr>
<th></th>
<th>0.88</th>
<th>0.44</th>
<th>0.14</th>
<th>0.16</th>
<th>0.37</th>
<th>0.77</th>
<th>0.96</th>
<th>0.27</th>
</tr>
</thead>
<tbody>
<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>
```

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

Trainable

ConvNet

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

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

- Ground truth box
- Region proposal (a.k.a default box, prior, reference, anchor)
- Predicted box
- Target offset to predict
- Predicted offset
- Loss

*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

Backward Pass:
Have \( \frac{\partial e}{\partial z} \), want \( \frac{\partial e}{\partial x} \)

Source: Ross Girshick
## 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
Object detection: Outline

- Task definition and evaluation
- Two-stage detectors:
  - R-CNN
  - Fast R-CNN
  - Faster R-CNN
- Single-stage and multi-resolution detectors
- Other detectors: CornerNet, DETR
Faster R-CNN

Region proposal network (RPN)

- Idea: tile the image with “anchor boxes” of a set size and try to predict how likely each anchor is to contain an object.

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

- Idea: tile the image with “anchor boxes” of a set size and try to predict how likely each anchor is to contain an object.

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

- Idea: tile the image with “anchor boxes” of a set size and try to predict how likely each anchor is to contain an object

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

• Idea: tile the image with “anchor boxes” of a set size and try to predict how likely each anchor is to contain an object.

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

- Idea: tile the image with "anchor boxes" of a set size and try to predict how likely each anchor is 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
Region proposal network (RPN)

- Implementation: put conv layers over low-resolution feature grid, for each grid location predict “object/no object” scores and bounding box regression coordinates
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

- Mean Average Precision (mAP)
- Year: 2006 to 2016
- Before CNNs
- After CNNs
- Faster R-CNN
- Fast R-CNN
- R-CNNv1
Outline

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

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

![Diagram showing the Faster R-CNN pipeline]

- Is it possible to 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

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:

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

Regression
Object/no object confidence
Class prediction
YOLO

- Objective function:

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

- 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
- Each level of resolution has its own predictor

Multi-resolution prediction: SSD

- Predict boxes of different size from different conv maps
- Each level of resolution has its own predictor

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

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

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

Fully convolutional one-stage detector (FCOS)

- “Anchor-free” approach

Run backbone CNN to get features aligned to input image

Figure source: J. Johnson

Tian et al., FCOS: Fully Convolutional One-Stage Object Detection, ICCV 2019
Fully convolutional one-stage detector (FCOS)

- “Anchor-free” approach

For each class, predict whether location falls inside a GT bounding box

Figure source: J. Johnson

Tian et al., FCOS: Fully Convolutional One-Stage Object Detection, ICCV 2019
Fully convolutional one-stage detector (FCOS)

• “Anchor-free” approach

For positive points, also regress distance to left, right, top, and bottom of GT box (with L2 loss)

Figure source: J. Johnson

Tian et al., FCOS: Fully Convolutional One-Stage Object Detection, ICCV 2019
Fully convolutional one-stage detector (FCOS)

- “Anchor-free” approach

For positive points, also regress distance to left, right, top, and bottom of GT box (with L2 loss)

Weight detections by “centerness” and confidence, perform NMS

Tian et al., FCOS: Fully Convolutional One-Stage Object Detection, ICCV 2019

Figure source: J. Johnson
Fully convolutional one-stage detector (FCOS)

Tian et al., FCOS: Fully Convolutional One-Stage Object Detection, ICCV 2019
Outline

- Task definition and evaluation
- Two-stage detectors
  - R-CNN
  - Fast R-CNN
  - Faster R-CNN
- Single-stage and multi-resolution detectors
- Other detectors: CornerNet, DETR
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
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, ECCV 2018
CenterNet

- Use an additional center point to verify predictions:

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
## CenterNet

<table>
<thead>
<tr>
<th>Method</th>
<th>FD</th>
<th>FD$_5$</th>
<th>FD$_{25}$</th>
<th>FD$_{50}$</th>
<th>FD$_S$</th>
<th>FD$_M$</th>
<th>FD$_L$</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><strong>35.1</strong></td>
<td><strong>30.7</strong></td>
<td><strong>34.2</strong></td>
<td><strong>40.8</strong></td>
<td><strong>53.0</strong></td>
<td><strong>31.3</strong></td>
<td><strong>24.4</strong></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><strong>32.4</strong></td>
<td><strong>28.2</strong></td>
<td><strong>31.6</strong></td>
<td><strong>37.5</strong></td>
<td><strong>50.7</strong></td>
<td><strong>27.1</strong></td>
<td><strong>23.0</strong></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
Detection Transformer (DETR)

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