Object detection
(part 1 recap)
Types of object detectors

• One-stage detectors

Image → Feature extraction → Classification → Localization

Class score (cat, dog, person)
Bounding box (x, y, w, h)

• Two-stage detectors

Image → Feature extraction → Extraction of object proposals → Classification → Localization

Class score (cat, dog, person)
Refine bounding box (Δx, Δy, Δw, Δh)
Types of object detectors

- One-stage detectors
  - YOLO, SSD, RetinaNet
  - CenterNet, CornerNet, ExtremeNet

- Two-stage detectors
  - R-CNN, Fast R-CNN, Faster R-CNN
  - SPP-Net, R-FCN, FPN
Two-stage detectors
R-CNN

Regression head to refine the bounding box location

Classification head

Extract features

Warping to a fix size 227 x 227

R-CNN

• Training scheme:
  – 1. Pre-train the CNN on ImageNet
  – 2. Finetune the CNN on the number of classes the detector is aiming to classify (softmax loss)
  – 3. Train a linear Support Vector Machine classifier to classify image regions. One SVM per class! (hinge loss)
  – 4. Train the bounding box regressor (L2 loss)
Fast R-CNN

- Softmax classifier
- Linear + softmax
- Linear
- Bounding-box regressors
- FCs
- Fully-connected layers
- “RoI Pooling” (single-level SPP) layer
- “conv5” feature map of image
- Forward whole image through ConvNet
- Regions of Interest (RoIs) from a proposal method

Shared computation at test time (like SPP)
Fast R-CNN

- Softmax classifier
- Linear + softmax
- Linear
- Bounding-box regressors
- Fully-connected layers
- “Roi Pooling” (single-level SPP) layer
- “conv5” feature map of image
- Forward whole image through ConvNet

Region of Interest Pooling

- Regions of Interest (RoIs) from a proposal method
- ConvNet
- Input image
Fast R-CNN Results

• Problem to be solved:
  – Proposal generation is still computed in a separate step, which makes it time consuming
Faster R-CNN:

- Solution: Have the proposal generation integrated with the rest of the pipeline.

- Region Proposal Network (RPN) trained to produce region proposals directly.

- After RPN, everything is like Fast R-CNN.

Object detection
(part 2)
Region proposal network

• How to extract proposals

- How many proposals?
  ✓ We need to decide a fixed number

- Where are they placed?
  ✓ Densely

Image
(N x M x 3)

(H x W x 4096)

Extract proposals

Zoom in
Region proposal network

- We fix the number of proposals by using a set of $n=9$ anchors per location.
- 9 anchors = 3 scales and 3 aspect ratios

Ren et al., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015
Region proposal network

- We fix the number of proposals by using a set of n=9 anchors per location.
- 9 anchors = 3 scales and 3 aspect ratios
- We extract a descriptor per location

Ren et al., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015
Region proposal network

- How to extract proposals

Image
\((N \times M \times 3)\)

\((H \times W \times 4096)\) → 3x3 conv → \((H \times W \times 256)\)

#anchors per image? \((H \times W \times n)\)
Region proposal network

• How to extract proposals

Image
(N x M x 3)

(H x W x 4096)

3x3 conv

(H x W x 256)

1x1 conv

(H x W x (2n+4n))

Anchor regression to proposal box

1 classification score per proposal (object/non-object)

#anchors per image?

(H x W x n)
Region proposal network

- How to extract proposals

Image
\((N \times M \times 3)\)

\((H \times W \times 4096)\)

3x3 conv

\((H \times W \times 256)\)

1x1 conv

\((H \times W \times (2n+4n))\)

RPN

Anchor regression to proposal box

1 classification score per proposal (object/non-object)

Per feature map location, I get a set of anchor correction and classification into object/non-object
RPN: training and losses

• Classification ground truth: We compute $p^*$ which indicates how much an anchor overlaps with the ground truth bounding boxes

$$p^* = 1 \quad if \quad \text{IoU} > 0.7$$

$$p^* = 0 \quad if \quad \text{IoU} < 0.3$$

• 1 indicates the anchor represent an object (foreground) and 0 indicates background object. The rest do not contribute to the training.
RPN: training and losses

• For an image, we randomly sample 256 anchors to form a mini-batch (balanced objects vs. non-objects).
• We calculate the classification loss (binary cross-entropy).
• Those anchors that do contain an object are used to compute the regression loss.
RPN: training and losses

- Each anchor is described by the center position, width and height \( x_a, y_a, w_a, h_a \)
RPN: training and losses

- Each anchor is described by the center position, width and height \( x_a, y_a, w_a, h_a \)
- What the network actually predicts are \( t_x, t_y, t_w, t_h \)

Normalized x \( t_x = (x - x_a)/w_a \), \( t_y = (y - y_a)/h_a \),

Normalized y

Normalized width \( t_w = \log(w/w_a) \), \( t_h = \log(h/h_a) \),

Normalized height

- Smooth L1 loss on regression targets
Faster R-CNN: Training

- First implementation, training of RPN separate from the rest.
- Now we can train jointly!

- Four losses:
  1. RPN classification (object/non-object)
  2. RPN regression (anchor -> proposal)
  3. Fast R-CNN classification (type of object)
  4. Fast R-CNN regression (proposal -> box)
Faster R-CNN

• 10x faster at test time wrt Fast R-CNN
• Trained end-to-end including feature extraction, region proposals, classifier and regressor
• More accurate, since proposals are learned. RPN is fully convolutional
Faster R-CNN: Results

<table>
<thead>
<tr>
<th>Test time per image (with proposals)</th>
<th>R-CNN</th>
<th>Fast R-CNN</th>
<th>Faster R-CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>50 seconds</td>
<td>2 seconds</td>
<td>0.2 seconds</td>
</tr>
<tr>
<td>(Speedup)</td>
<td>1x</td>
<td>25x</td>
<td>250x</td>
</tr>
<tr>
<td>mAP (VOC 2007)</td>
<td>66.0</td>
<td>66.9</td>
<td>66.9</td>
</tr>
</tbody>
</table>
Related works

One-stage detectors
Types of object detectors

- One-stage detectors
  - Feature extraction
  - Classification
  - Localization

- Two-stage detectors
  - Feature extraction
  - Extraction of object proposals
  - Classification
  - Localization
  - Refine bounding box: $(\Delta x, \Delta y, \Delta w, \Delta h)$
  - Class score (cat, dog, person)
YOLO: You Only Look Once

- Recall sliding window object detection
- To make it efficient, we will “slide our window” only on certain locations of the image
- We divide our image in a grid.

YOLO: You Only Look Once

- We will place a box at the center of each cell in the grid, and this will be our initial box guess for that object.
YOLO: You Only Look Once

- Direct regression from image to box coordinates

YOLO: You Only Look Once

• In YOLOv2 we use anchor boxes

For each grid location we predict $n$ boxes

Green = anchors
Blue = predictions

YOLO: You Only Look Once

• In YOLOv2 we use anchor boxes

For each grid location we predict $n$ boxes

Similar to Faster R-CNN region proposal network

YOLO: You Only Look Once

• In YOLOv2 we use anchor boxes

(7 x 7 x C) → 3x3 conv → (7 x 7 x (25 x n))

• 1 class prediction (20)
• Anchor relative box regression (4)
• Object/non-object (1)

**SSD: Single Shot multibox Detector**

- SSD predicts at different scales

YOLO and SSD

• PROS:
  – Very fast
  – End-to-end trainable and fully convolutional
  – SSD detects more objects than YOLO

• CONS:
  – Performance is not as good as two-stage detectors
  – Difficulty with small objects
Problem with one-stage detectors?

• Two-stage detectors:
  – Classification only work on “interesting” foreground regions (proposals, ~1-2k). Most background examples are already filtered out.
  – Class balance between foreground and background objects is manageable.
  – Classifier can concentrate on analyzing proposals with rich information content
Problem with one-stage detectors?

- One-stage detectors:
  - Many locations need to be analyzed (100k) densely covering the image → foreground-background imbalance
  - Hard negative mining is useful, but not sufficient

Many negative examples, no useful signal
Few positive examples, rich information for learning
RetinaNet

- Solution: change the loss function!
- Recall cross-entropy loss

TY Lin et al. “Focal Loss for Dense Object Detection”. ICCV 2017

Very hard example, we get loss 2.5

Well classified (easy) examples, we get loss 0.1
RetinaNet

- Solution: change the loss function!
- Recall cross-entropy loss

100 hard examples * 2.5 = 250
100000 easy examples * 0.1 = 10000

TY Lin et al. “Focal Loss for Dense Object Detection”. ICCV 2017
RetinaNet

- Solution: change the loss function!
- Recall cross-entropy loss

100 hard examples * 2.5 = 250

100000 easy examples * 0.1 = 10000

TY Lin et al. “Focal Loss for Dense Object Detection”. ICCV 2017
RetinaNet

- Proposed: Focal loss

\[
\text{CE}(p_t) = -\log(p_t)
\]

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

- When \( \gamma = 0 \) it is equivalent to the cross-entropy loss

- As \( \gamma \) goes towards 1, the easy examples are down-weighted.

- Example: \( \gamma = 2 \), if \( p_t = 0.9 \), FL is 100 lower than CE.

TY Lin et al. “Focal Loss for Dense Object Detection”. ICCV 2017
RetinaNet

- Proposed: Focal loss
- Powerful feature extraction: ResNet
- Multi-scale prediction
- 9 anchors per level, each one with a classification and regression target

TY Lin et al. “Focal Loss for Dense Object Detection”. ICCV 2017
RetinaNet

TY Lin et al. “Focal Loss for Dense Object Detection”. ICCV 2017
## RetinaNet

The diagram illustrates the performance comparison between RetinaNet, YOLOv3, and other related models. The x-axis represents inference time (ms), while the y-axis shows COCO AP. The table below lists the methods, mAP, and time in milliseconds for different models:

<table>
<thead>
<tr>
<th>Method</th>
<th>mAP</th>
<th>time (ms)</th>
</tr>
</thead>
<tbody>
<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-50-500</td>
<td>32.5</td>
<td>73</td>
</tr>
<tr>
<td>RetinaNet-101-500</td>
<td>34.4</td>
<td>90</td>
</tr>
<tr>
<td>RetinaNet-101-800</td>
<td>37.8</td>
<td>198</td>
</tr>
<tr>
<td>YOLOv3-320</td>
<td>28.2</td>
<td>22</td>
</tr>
<tr>
<td>YOLOv3-416</td>
<td>31.0</td>
<td>29</td>
</tr>
<tr>
<td>YOLOv3-608</td>
<td>33.0</td>
<td>51</td>
</tr>
</tbody>
</table>
One-stage (point-based) detectors
Getting rid of anchors?

- CornerNet: express bounding boxes with 2 points, the top-left and bottom-right corners.

H. Law and J. Deng. “CornerNet: Detecting Objects as Paired Keypoints”. ECCV 2018
CornerNet

1. Probability map for each corner type

2. Box identification with an embedding (needed to match top and bottom keypoints)

3. Class

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

• Hourglass network, originally published in:

We predict corners at a lower resolution and then regress an offset (bounding box correction as we have seen for all methods)

H. Law and J. Deng. „CornerNet: Detecting Objects as Paired Keypoints“. ECCV 2018
CornerNet

• Corner pooling

H. Law and J. Deng. ‘CornerNet: Detecting Objects as Paired Keypoints’. ECCV 2018
CornetNet

• What is the problem with CornetNet?
• Many incorrect bounding boxes (especially small) → too many False Positives
• Hypothesis: It is hard to infer the class of the box if the network is focused on the boundaries
CenterNet

- Idea: focus on the center of the object to infer its class
- Use the corners as proposals, and the center to verify the class of the object and filter out outliers

K. Duan et al. „CenterNet: Keypoint Triplets for Object Detection“. arXiv 2018
ExtremeNet

- Bounding box corner representation is not ideal, why?

X. Zhou et al. „Bottom-up object detection by grouping extreme and center points“. CVPR 2019
• Bounding box corner representation is not ideal, why?

- Corner lies on the object
- Corner does not lie on the object
- Hard for a CNN to predict as a corner

X. Zhou et al. „Bottom-up object detection by grouping extreme and center points“. CVPR 2019
ExtremeNet

- Represent objects by their extreme points

X. Zhou et al. „Bottom-up object detection by grouping extreme and center points“. CVPR 2019
ExtremeNet

- No need to predict embeddings for the box computation

X. Zhou et al. „Bottom-up object detection by grouping extreme and center points“. CVPR 2019
Extreme points

- Extreme points are used commonly for annotation

  - D. P. Papadopoulos et al. „Extreme clicking for efficient object annotation“. ICCV 2017

  - K. Maninis et al. „Deep extreme cut: From extreme points to object segmentation“. CVPR 2018
Detection evaluation
Detection evaluation

TP = True positives.  FP = False positives.  FN = False negatives
Detection evaluation

- Precision: how accurate your predictions are.

\[ \text{Precision} = \frac{TP}{TP + FP} \]

- Recall: how good you are at finding all positives

\[ \text{Recall} = \frac{TP}{TP + FN} \]

TP = True positives. FP = False positives. FN = False negatives.
Detection evaluation

- What is a positive?
- We use the Intersection over Union (IoU) or Jaccard Index
- If IoU > 0.5 → positive match
- Depending on the dataset it is 0.5 or 0.7

TP = True positives. FP = False positives. FN = False negatives
Detection evaluation: AP

- Mean Average Precision (mAP)
  - For each image and class independently, rank the predicted boxes by confidence score. Assign the boxes to the corresponding ground truth if IoU > 0.5.
  - Each ground truth box can only be matched to one predicted box.
  - For each class independently, compute Average Precision
  - Mean over all classes to obtain mAP
### Average Precision

- Rank predictions by confidence

<table>
<thead>
<tr>
<th>Rank</th>
<th>True/false</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>F</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

My method predicts 12 boxes, 6 of them are true boxes present in the image.
## Average Precision

- Rank predictions by confidence

<table>
<thead>
<tr>
<th>Rank</th>
<th>True/false</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>T</td>
<td>1.0</td>
<td>0.17</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>F</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Average Precision**

- Rank predictions by confidence

<table>
<thead>
<tr>
<th>Rank</th>
<th>True/false</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>T</td>
<td>1.0</td>
<td>0.17</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>1.0</td>
<td>0.33</td>
</tr>
<tr>
<td>3</td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>F</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Average Precision

- Rank predictions by confidence

<table>
<thead>
<tr>
<th>Rank</th>
<th>True/false</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>T</td>
<td>1.0</td>
<td>0.17</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>1.0</td>
<td>0.33</td>
</tr>
<tr>
<td>3</td>
<td>F</td>
<td>0.67</td>
<td>0.33</td>
</tr>
<tr>
<td>4</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>F</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Average Precision

- Rank predictions by confidence

<table>
<thead>
<tr>
<th>Rank</th>
<th>True/false</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>T</td>
<td>1.0</td>
<td>0.17</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>1.0</td>
<td>0.33</td>
</tr>
<tr>
<td>3</td>
<td>F</td>
<td>0.67</td>
<td>0.33</td>
</tr>
<tr>
<td>4</td>
<td>T</td>
<td>0.75</td>
<td>0.5</td>
</tr>
<tr>
<td>5</td>
<td>F</td>
<td>0.6</td>
<td>0.5</td>
</tr>
<tr>
<td>6</td>
<td>F</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>7</td>
<td>T</td>
<td>0.57</td>
<td>0.67</td>
</tr>
<tr>
<td>8</td>
<td>F</td>
<td>0.5</td>
<td>0.67</td>
</tr>
<tr>
<td>9</td>
<td>T</td>
<td>0.56</td>
<td>0.83</td>
</tr>
<tr>
<td>10</td>
<td>T</td>
<td>0.6</td>
<td>1.0</td>
</tr>
<tr>
<td>11</td>
<td>F</td>
<td>0.55</td>
<td>1.0</td>
</tr>
<tr>
<td>12</td>
<td>F</td>
<td>0.5</td>
<td>1.0</td>
</tr>
</tbody>
</table>
Average Precision

• Plot Precision vs Recall

![Graph showing precision vs recall](image-url)
Average Precision

- Average precision = area under curve
Average Precision

- Average precision = area under curve (filled)
Interesting read

• Overview paper that analyzes backbone architecture selection, number of proposals used, etc. to see which has an effect on object detection accuracy.

• Huang et al. “Speed/accuracy trade-offs for modern convolutional object detectors”. CVPR 2017
CV3DST competition

• Dates:
  – 26.11.19: Competition starts! Starting code and training set are released.
  – Before 25.12.19: Test set is released.
  – 02.02.20 (midnight): Competition closes
  – 03.02.20 (midnight): Abstract and code submission deadline
  – 04.02.20: Presenters are announced
  – 07.02.20: Presentation of selected methods
CV3DST competition

• Rules:
  – No off-the-shelf detectors or trackers can be used.
  – You can use off-the-shelf methods that compute additional features that you might use, e.g., optical flow. Nonetheless, this cannot be the only contribution of your method.
  – This is an individual effort, but sharing ideas between participants in the class is allowed, as long as you come up with different methods and implementations to solve the MOT problem.
  – We will not provide help to solve coding issues nor to decide what methods to use to improve your tracker. This is a real competition, so you are on your own!
CV3DST competition

• I get the bonus if…
  – You reach a minimum MOTA value (will be announced together with the test set).
  – You respect the rules of the challenge (this will be judged at our discretion. ➔ make your own code, try to come up with ideas, show initiative)

• Presentations on February 7th:
  – Presence mandatory
  – GOAL: find out what has worked best/worst to solve the MOT problem
  – It is going to be an active discussion!
Next lectures

- 29.11: Detection 2
- 06.12: Single/Multiple object tracking
- 13:12: Multiple object tracking
- 20.12: Trajectory prediction
- Christmas Break
- 10.01: Video object segmentation
- 17.01: Semantic/Instance Segmentation
- 24.01: Semantic/Instance Segmentation
- 31.01: Going towards 3D tracking and segmentation
- 07.02: Project Presentations