Two-stage detectors: short recap
Types of object detectors

- One-stage detectors
  - Feature extraction
  - Classification
  - Localization
  - Class score (cat, dog, person)
  - Bounding box (x, y, w, h)

- Two-stage detectors
  - Feature extraction
  - Extraction of object proposals
  - Classification
  - Localization
  - Refine bounding box (Δx, Δy, Δw, Δh)
R-CNN

Regression head to refine the bounding box location

Classification head

Extract features

Warping to a fix size 227 x 227

Fast R-CNN

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

Region of Interest Pooling

ConvNet

Input image

"conv5" feature map of image

"RoI Pooling" (single-level SPP) layer

Fully-connected layers

Bounding-box regressors

Linear + softmax

Softmax classifier

FCs

Regions of Interest (RoIs) from a proposal method


Slide credit: Ross Girschick
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


Slide credit: Ross Girshick
Region proposal network

• How to extract proposals

Image
(N x M x 3)

(H x W x 4096)

Extract proposals

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

• Where are they placed?
✓ Densely

Zoom in
Region proposal network

• How to extract proposals

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

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

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

Anchor regression to proposal box

1x1 conv
\((H \times W \times 256)\)

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

RPN
\((H \times W \times (2n+4n))\)
One-stage detectors
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)
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

Convolutional neural network ➔ Detections

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

- 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

Very hard example, we get loss 2.5

Well classified (easy) examples, we get loss 0.1

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

Very hard example, we get loss 2.5

Well classified (easy) examples, we get loss 0.1

TY Lin et al. “Focal Loss for Dense Object Detection”. ICCV 2017
• Solution: change the loss function!
• Recall cross-entropy loss

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

Very hard example, we get loss 2.5
Well classified (easy) examples, we get loss 0.1

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

![Graph showing performance comparison between RetinaNet and YOLOv3 across different methods and inference times. The graph illustrates mAP and time versus inference time (ms).]
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’. ICCV 2019
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
ExtremeNet

• 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

Henderson and Ferrari, "End-to-End Training of Object Class Detectors for Mean Average Precision". ACCV 2016
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>0.17</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

\[
\begin{align*}
p & = \frac{t}{k} = \frac{1}{1} = 1.00 \\
n & = 6 \\
r & = \frac{t}{n} = \frac{1}{6} = 0.17
\end{align*}
\]
Average Precision

- Plot Precision vs Recall

\[ k = 2 \quad p = \frac{t}{k} = \frac{2}{2} = 1.00 \]
\[ n = 6 \quad r = \frac{t}{n} = \frac{2}{6} = 0.33 \]
Average Precision

- Plot Precision vs Recall

\[ p = \frac{t}{k} = \frac{2}{3} = 0.67 \]
\[ r = \frac{t}{n} = \frac{2}{6} = 0.33 \]
Average Precision

- Plot Precision vs Recall

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

\[
k = 4 \quad t = 3
\]

\[
p = \frac{t}{k} = \frac{3}{4} = 0.75
\]

\[
n = 6 \quad r = \frac{t}{n} = \frac{3}{6} = 0.50
\]
Average Precision

- Plot Precision vs Recall

\[
\begin{array}{cccccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\
T & T & F & T & F & F & T & F & T & T & F & F
\end{array}
\]

\[
k = 12 \quad p = \frac{t}{k} = \frac{6}{12} = 0.50 \\
n = 6 \quad r = \frac{t}{n} = \frac{6}{6} = 1.00
\]
Average Precision

• Average precision = area under curve
Average Precision

- Average precision = area under curve (filled)
Recall non-maximum suppression
NMS

• Many bounding boxes are predicted by the CNN

Images from Y. Avrithis, Object detection lecture. INRIA
NMS

• Most confident region remains
NMS

• Second most confident region remains
NMS

• Third most confident region remains
NMS

• Region 4 overlap with 1 more than threshold → remove
NMS

• The same will happen to all the regions in black

Region 8 does not overlap with green boxes (confirmed detections), hence we keep it.
NMS

• In the end, we have 4 confirmed predicted boxes.
NMS

• Problem with highly overlapping objects

With an NMS threshold of 0.4, we remove both boxes, hence we have a False Negative.
NMS

- Problem with highly overlapping objects

With an NMS threshold of 0.6, we keep both boxes, hence we have a False Positive.
Detection beyond 2D boxes
Prediction “as points”

• 3D bounding boxes: prediction of the center + orientation + width + height + depth
• Facial landmarks

• Human pose estimation: body joints
Human pose estimation

- Estimate a 2D pose (x,y) coordinates for each joint from a RGB image.
- Parameterization:
  - 17 joints*
  - Left/right ankle
  - Left/right knee
  - etc

* As defined in http://cocodataset.org/#keypoints-2019

Human pose estimation

- It might seem an easy problem, but it is challenging: occlusions, clothing, extreme poses, viewpoint changes

Human pose estimation

• It might seem an easy problem, but it is challenging: occlusions, clothing, extreme poses, viewpoint changes

Toshev and Szegedy, "DeepPose: Human Pose Estimation via Deep Neural Networks", CVPR 2014
Human pose estimation

• Approach 1: direct regression

Image (221 x 221 x 3) → Convolutional Neural Network → Feature map (13 x 13 x 192) → Now the output are body joints (17 x 2)

Human pose estimation

• Approach 2: heatmap prediction
  – Instead of prediction by regression, for each joint one predicts a full image with a heatmap of the joint location

Human pose estimation

- Approach 2: heatmap prediction
  - Instead of prediction by regression, for each joint one predicts a full image with a heatmap of the joint location.
  - Powerful representation, easier to predict a confidence per location, rather than regress a value for the position.

Human pose estimation

- Approach 2: heatmap prediction
  - Ground truth (GT) heatmap is constructed by placing a 2D Gaussian around the joint position (e.g. variance 1.5 pixels)
  - Loss: MSE between predicted and GT heatmap

Human pose estimation

- Bringing the structure of the problem
  - Body parts are linked to each other
  - Body symmetries
  - Joint limits, e.g., elbow cannot bend backwards
  - Physical connectivity: elbow connected to wrist

Human pose estimation

- Using graphical models also allows us to find the pose of several targets


All joint detections as provided directly by the CNN
Human pose estimation

- Using graphical models also allows us to find the pose of several targets

Cluster the joints based on joint type and target ID

Human pose estimation

- Using graphical models also allows us to find the pose of several targets

Human pose estimation

• Using graphical models also allows us to find the pose of several targets

• Alternatively, 1. Object detection + 2. Pose estimation

Human pose estimation

• Improving the architecture: Stacked hourglass
  – Repeated bottom-up (high-to-low-res) and top-down processing (low-to-high-res)

Hourglass – Unet – Autoencoder

• Architecture that first scales down the resolution and then upsamples it to get to the initial resolution

Hourglass – Unet - Autoencoder

- **Encoder**: normal convolutional filters + pooling
- **Decoder**: Upsampling + convolutional filters

Hourglass – Unet - Autoencoder

• **Encoder:** normal convolutional filters + pooling

• **Decoder:** Upsampling + convolutional filters

Hourglass – Unet - Autoencoder

- **Encoder**: normal convolutional filters + pooling

- **Decoder**: Upsampling + convolutional filters

- The convolutional filters in the decoder are learned using backprop and their goal is to refine the upsampling

Upsampling in hourglass

• Interpolation
Upsampling in hourglass

- Interpolation

Original image  x 10

Nearest neighbor interpolation  Bilinear interpolation  Bicubic interpolation

Image: Michael Guerzhoy
Residual connections

- Improving the architecture: Stacked hourglass

Preserve spatial knowledge at each resolution

Human pose estimation

• Improving the architecture: Stacked hourglass
  – Repeated bottom-up (high-to-low-res) and top-down processing (low-to-high-res)

Human pose estimation

Make a prediction

Refine your prediction

Final prediction

Human pose estimation

• Intermediate supervision

Human pose estimation

- Intermediate supervision

The prediction is reintegrated into the network with 1x1 convs

Human pose estimation

• Intermediate supervision

Human pose estimation

A. Newell et al. „Stacked Hourglass Networks for Human Pose Estimation” ECCV, 2016.
Human pose estimation

Human pose estimation

- Open pose – the code to use


https://github.com/CMU-Perceptual-Computing-Lab/openpose
Next lectures

Lectures 2-3
Object Detection

Lectures 4-5
Object Tracking

Lectures 7-8
Object Segmentation

Lecture 9
Video Object Segmentation
One-stage detectors
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