Object Detection

CSC2541, 2017 Winter
Bin Yang
13 Feb. 2017

“If I have seen further it is by standing on the shoulders of giants.” - Isaac Newton

slides adopted from Ross Girshick, Chris McIntosh, Sanja Fidler, Mubarak Shah and many others
Object detection

- Introduction
- Pre-CNN time
  - HOG detector
  - Deformable Part-based Model
- CNN time
  - Region-CNN
  - Fast versions of R-CNN
  - YOLO/SSD
- 3D object detection
- Devil’s in the details
Object detection

- Introduction
- Pre-CNN time
  - HOG detector
  - Deformable Part-based Model
- CNN time
  - Region-CNN
  - Fast versions of R-CNN
  - YOLO/SSD
- 3D object detection
- Devil’s in the details
Image understanding

Snack time in the lab

slide credit: Ross Girshick
What objects are where?

robot: “I see a table with twinkies, pretzels, fruit, and some mysterious chocolate things...”
Formalizing the object detection task

Many possible ways, this one is popular:

Input

Desired output

cat, dog, chair, cow, person, motorbike, car, ...

slide credit: Ross Girshick
Formalizing the object detection task

Many possible ways, this one is popular:

Input

Desired output

Performance summary:

Average Precision (AP)
0 is worst 1 is perfect

slide credit: Ross Girshick
Example 1: Find Waldo!

image /

slide credit: Chris McIntosh
1. Make the template as a filter

slide credit: Chris McIntosh
2. Result of normalized cross-correlation

slide credit: Chris McIntosh
3. Find the highest peak
4. Put a bounding box (the size of template) at the point
Example 2: Find all persons?

slide credit: Chris McIntosh
Example 2: Find all persons?

A template for all instances?

slide credit: Chris McIntosh
Example 2: Find all persons?

A template for all instances?

We need features!

slide credit: Chris McIntosh
Object detection

• Introduction
• Pre-CNN time
  • HOG detector
  • Deformable Part-based Model
• CNN time
  • Region CNN
  • Fast versions of RCNN
  • YOLO/SSD
• 3D object detection
• Devil’s in the details
The HOG Detector

N. Dalal and B. Triggs

*Histograms of oriented gradients for human detection*

CVPR, 2005


cited by 17,502
HOG detector: pipeline

1. Scan image(s) at all scales and locations
2. Extract features over windows
3. Run linear SVM classifier on all locations
4. Fuse multiple detections in 3-D position & scale space
5. Object detections with bounding boxes
I. Sliding window

- Scan image(s) at all scales and locations
- Extract features over windows
- Run linear SVM classifier on all locations
- Fuse multiple detections in 3-D position & scale space
- Object detections with bounding boxes

Scale-space pyramid
Detection window

locations
scales

Slide credit: Sanja Fidler
II. Histograms of Oriented Gradients

- Scan image(s) at all scales and locations
- Extract features over windows
- Run linear SVM classifier on all locations
- Fuse multiple detections in 3-D position & scale space
- Compute gradients
- Weighted vote into spatial & orientation cells
- Contrast normalize over overlapping spatial blocks

Slide credit: Sanja Fidler
II. Histograms of Oriented Gradients

- Scan image(s) at all scales and locations
- Extract features over windows
- Run linear SVM classifier on all locations
- Fuse multiple detections in 3-D position & scale space
- Object detections with bounding boxes

Slide credit: Sanja Fidler
II. Histograms of Oriented Gradients

Scan image(s) at all scales and locations

Extract features over windows

Run linear SVM classifier on all locations

Fuse multiple detections in 3-D position & scale space

Object detections with bounding boxes

Compute gradients

Weighted vote into spatial & orientation cells

Contrast normalize over overlapping spatial blocks

9-dim feature vector

Slide credit: Sanja Fidler
II. Histograms of Oriented Gradients

Scan image(s) at all scales and locations

Extract features over windows

Run linear SVM classifier on all locations

Fuse multiple detections in 3-D position & scale space

Object detections with bounding boxes

Compute gradients

Weighted vote into spatial & orientation cells

Contrast normalize over overlapping spatial blocks

Cell

Block

Overlap of Blocks

Feature vector $f = [\ldots, \ldots, \ldots]$ 

L2 normalization in each block:

$$f = \frac{f}{\sqrt{\|f\|_2^2 + \epsilon^2}}$$

Slide credit: Sanja Fidler
III. SVM classifier

Training:
- Train a classifier (e.g., person vs no person)

Detection:
- Use the trained classifier to predict presence/absence of object class in each window in the image

Slide credit: Sanja Fidler
III. SVM classifier - training

Learning phase

- Input: Annotations on training images
- Create fixed-resolution normalised training image data set
- Encode images into feature spaces
- Learn binary classifier

Train classifier

- Predict presence/absence of object class in each image window

Positive training examples

Negative training examples

- All image crops are scaled to the same size (for this example (15x8) x (7x8) pixels), where 8 is the width/height of each HOG cell in pixels
- Cool trick: take a bigger region than each annotated object to also capture context (works better!)

Pics: S. Lazebnik

Slide credit: Sanja Fidler
III. SVM classifier - training

Learning phase

Input: Annotations on training images
- Create fixed-resolution normalised training image data set
- Encode images into feature spaces
- Learn binary classifier

Train classifier
Predict presence/absence of object class in each image window

positive training examples

negative training examples

*** These are just feature visualizations. Each "picture" is really a 15x7x31 feature matrix.

Before training a classifier, we vectorize each of these examples: \( f = f(\cdot) \)

Slide credit: Sanja Fidler
III. SVM classifier - training

Learning phase

Input: Annotations on training images

Create fixed-resolution normalised training image data set

Encode images into feature spaces

Learn binary classifier

Train classifier. SVM (Support Vector Machines) is typically used.
III. SVM classifier - detection

- Computing the score $w^T \cdot x + b$ in every location is the same as performing cross-correlation with template $w$ (and add $b$ to result).

Detection Phase

Scan image(s) at all scales and locations

Extract features over windows

Run linear SVM classifier on all locations

Fuse multiple detections in 3-D position & scale space

Object detections with bounding boxes

Train classifier

Predict presence/absence of object class in each image window

$p$

$score(l, p) = w \cdot \phi(l, p)$

Image pyramid

HOG feature pyramid

Pic from: R. Girshik

Slide credit: Sanja Fidler
IV. Non-Maxima Suppression (NMS)

- Scan image(s) at all scales and locations
- Extract features over windows
- Run linear SVM classifier on all locations
- Fuse multiple detections in 3-D position & scale space
- Object detections with bounding boxes

Non-maxima suppression (NMS)

\[
\text{overlap} = \frac{\text{area}(\text{box}_1 \cup \text{box}_2)}{\text{area}(\text{box}_1 \cap \text{box}_2)} > 0.5
\]

- Remove all boxes that overlap more than XX (typically 50%) with the chosen box
Non-Maxima Suppression (NMS)

- Greedy algorithm.
- At each iteration pick the highest scoring box.
- Remove all boxes that overlap more than XX (typically 50%) with the chosen box.
HOG detector: summary

*Dalal & Triggs '05*
- Histogram of Oriented Gradients (HOG)
- SVM training
- Sliding window detection
Example 3: How can we deal with this guy?

Dalal & Triggs ’05
- Histrogram of Oriented Gradients (HOG)
- SVM training
- Sliding window detection

Slide credit: Sanja Fidler, Ross Girshick
HOG detector: limitations

Dalal & Triggs ’05
- Histogram of Oriented Gradients (HOG)
- SVM training
- Sliding window detection

Fischler & Elschlager ’73
Felzenszwalb & Huttenlocher ’00
- Pictorial structures
- Weak appearance models
- Non-discriminative training

We need flexible models!
The DPM Detector

P. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan

Object Detection with Discriminatively Trained Part Based Models

T-PAMI, 2010

Code: http://www.cs.berkeley.edu/~rbg/latent/

cited by 5,084
Deformable Part Model (DPM): key idea

Port the success of Dalal & Triggs into a part-based model

DPM

D&T

PS

<table>
<thead>
<tr>
<th>Year</th>
<th>AP</th>
<th>2005</th>
</tr>
</thead>
<tbody>
<tr>
<td>2005</td>
<td>12%</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Year</th>
<th>2008</th>
</tr>
</thead>
<tbody>
<tr>
<td>2008</td>
<td>27%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Year</th>
<th>2009</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009</td>
<td>36%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Year</th>
<th>2010</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010</td>
<td>45%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Year</th>
<th>2011</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011</td>
<td>49%</td>
</tr>
</tbody>
</table>

Slide credit: Ross Girshick
DPM: Model representation

- A model has a root filter $F_0$ and $n$ part models $(F_i, v_i, d_i)$
  - $F_i$: $i$-th part filter
  - $v_i$: anchor position of $i$-th part relative to the root
  - $d_i$: deformation parameters for $i$-th part
DPM: Object Hypothesis

- In HOG feature pyramid
  - root filter - coarser scale
  - part filters - finer scale

\[ z = (p_0, ..., p_n) \]

- \( p_0 \) : location of root
- \( p_1, ..., p_n \) : location of parts

Score is sum of filter scores minus deformation costs

Image pyramid

HOG feature pyramid

Slide credit: Mubarak Shah
DPM: Score of a Hypothesis

Score of a hypothesis $z$ is

$$\text{score}(z) = \beta \cdot \psi(H, z)$$

where

$$\beta = (F_0, ..., F_n, d_1, ..., d_n, b)$$

$$\psi(H, z) = (\phi(H, p_0), ..., \phi(H, p_n), -\phi_d(dx_1, dy_1), ..., -\phi_d(dx_n, dy_n), 1)$$

Filters
Feature of subwindow at location $p_i$
Spatial prior
Deformation parameters
Displacement of part $i$ relative to its anchor position
Bias

Slide credit: Mubarak Shah
DPM: Score of a Hypothesis

\[
\text{score}(p_0, ..., p_n) = \sum_{i=0}^{n} F_i \cdot \phi(H, p_i) - \sum_{i=1}^{n} d_i \cdot \phi_d(dx_i, dy_i) + b
\]

- Filters
- Feature of subwindow at location \( p_i \)
- Deformation parameters
- Displacement of part \( i \) relative to its anchor position

\[
\phi_d(dx, dy) = (dx, dy, dx^2, dy^2)
\]

Initial Value (to be learned)

Slide credit: Mubarak Shah
DPM: Detection

- The overall score of a root location is computed according to the best possible placement of the parts
  \[ \text{score}(p_0) = \max_{p_1, \ldots, p_n} \text{score}(p_0, \ldots, p_n) \]
- High-scoring root locations define detections
- Sliding-window approach
- Efficient computation (\(O(nk)\)): dynamic programming + generalized distance transforms

Slide credit: Mubarak Shah
DPM: Detection

• **Distance transform**
  - Response of the $i$-th part filter in the $l$-th level of the feature pyramid
    \[ R_{i,l}(x, y) = F_i \cdot \phi(H, (x, y, l)) \]
  - Transformed response, given root is at $(x, y)$
    \[ D_{i,l}(x, y) = \max_{dx, dy} (R_{i,l}(x + dx, y + dy) - d_i \cdot \phi_d(dx, dy)) \]

\[ \phi_d(dx, dy) = (dx, dy, dx^2, dy^2) \]

\[ d_i = (0, 0, 1, 1) \]
DPM: Detection

Slide credit: Mubarak Shah, Ross Girshick
DPM: Training

- Positive training examples are labeled with bounding boxes
- No part location is available during training (latent)
- Aim: learn model parameters $\beta = (F_0, \ldots, F_n, d_1, \ldots, d_n, b)$
DPM: Latent Variables

- The positions of the parts are not given in both the training and the testing images.
- The variables that exist but not known in training samples are called latent variables.
- The learning algorithm must be able to find/discover the optimal values for the latent variables, namely the position of the parts.

Slide credit: Mubarak Shah
DPM: Training

- The classifier scores an example $x$ by
  $f_\beta(x) = \max_{z \in Z(x)} \beta \cdot \Phi(x, z)$
- $\beta$: the model parameters
- $z$: latent values
- $Z(x)$: the possible latent values for example $x$
DPM: Training

- Minimize the objective function

\[ L_D(\beta) = \frac{1}{2} \| \beta \|^2 + C \sum_{i=1}^{n} \max(0, 1 - y_i f_\beta(x_i)) \]

- Labeled training examples \( D = \langle (x_1, y_1), \ldots, (x_n, y_n) \rangle \)
- \( y_i \in \{-1, 1\} \)
DPM: Latent SVM

- A latent SVM is semi-convex
  - $f_\beta(x) = \max_{z \in Z(x)} \beta \cdot \Phi(x, z)$ is convex in $\beta$
  - For negative examples ($y_i = -1$), the hinge loss is convex
    \[
    \max \left( 0, 1 - y_i f_\beta(x_i) \right) = \max(0, 1 + f_\beta(x_i))
    \]
    (the maximum of two convex function)
  - For positive examples ($y_i = 1$), the hinge loss is not convex
    \[
    \max \left( 0, 1 - y_i f_\beta(x_i) \right) = \max(0, 1 - f_\beta(x_i))
    \]
    (the maximum of a convex function and a concave function)
- If the latent value for positive examples are fixed, the hinge loss is convex

Slide credit: Mubarak Shah
DPM: Latent SVM

• Initialize $\beta$ using standard SVM by assuming the same parts locations for all the positive examples

• Iterative optimization:
  • Relabel positive examples: fix $\beta$, find the best $z$ for each positive example (exactly the same with detection!)
  • Optimize $\beta$: fix $z$, optimize $\beta$ by solving the convex problem

Slide credit: Mubarak Shah
DPM: Mixture model

- A mixture model consists of $m$ components
- Captures extreme intra-class variation
- Split the positive bounding boxes into $m$ groups by aspect ratio

Mixture Model Example - Person

Mixture Model Example - Bicycle

Slide credit: Mubarak Shah
DPM on PASCAL VOC

[Source: http://pascallin.ics.uci.edu/challenges/VOC/voc20(07,08,09,10,11,12)/results/index.html]

Slide credit: Ross Girshick

Ross Girshick

Lifetime Achievement Award by PASCAL VOC
Object detection

- Introduction
- Pre-CNN time
  - HOG detector
  - Deformable Part-based Model
- CNN time
  - Region CNN
  - Fast versions of RCNN
  - YOLO/SSD
- 3D object detection
- Devil’s in the details
Object detection renaissance (2013-present)
Deep object detection

Slide credit: https://handong1587.github.io/deep_learning/2015/10/09/object-detection.html
**R-CNN: Regions with CNN features**

1. Input image
2. Extract region proposals (~2k)
3. Compute CNN features
4. Classify regions

1. Pre-train CNN for **image classification**

large auxiliary dataset (ImageNet)
Training

1. Pre-train CNN for **image classification**
   - train CNN
   - large auxiliary dataset (ImageNet)

2. Fine-tune CNN on **target dataset** and **task**
   - fine-tune CNN
   - small target dataset (PASCAL VOC)

(optional)
Training

1. Pre-train CNN for **image classification**

   - Train CNN
   - Large auxiliary dataset (ImageNet)

2. Fine-tune CNN on **target dataset** and **task**
   - Fine-tune CNN
   - Small target dataset (PASCAL VOC)

   (optional)

3. Train linear predictor for **detection**

   - Region proposals
   - Small target dataset (PASCAL VOC)
   - ~2000 warped windows / image
   - CNN features
   - Per class SVM
   - Training labels
## R-CNN Results

<table>
<thead>
<tr>
<th>Method</th>
<th>VOC2007</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPM v5 (Girshick et al. 2011)</td>
<td>33.7%</td>
</tr>
<tr>
<td>Regionlets (Wang et al. 2013)</td>
<td>41.7%</td>
</tr>
<tr>
<td>R-CNN (AlexNet)</td>
<td>54.2%</td>
</tr>
<tr>
<td>R-CNN (AlexNet) + BB</td>
<td>58.5%</td>
</tr>
<tr>
<td>R-CNN (VGGNet)</td>
<td>62.2%</td>
</tr>
<tr>
<td>R-CNN (VGGNet) + BB</td>
<td>66.0%</td>
</tr>
</tbody>
</table>
# R-CNN Results

<table>
<thead>
<tr>
<th>Method</th>
<th>VOC2007</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPM v5 (Girshick et al. 2011)</td>
<td>33.7%</td>
<td></td>
</tr>
<tr>
<td>Regionlets (Wang et al. 2013)</td>
<td>41.7%</td>
<td></td>
</tr>
<tr>
<td>R-CNN (AlexNet)</td>
<td>54.2%</td>
<td></td>
</tr>
<tr>
<td>R-CNN (AlexNet) + BB</td>
<td>58.5%</td>
<td></td>
</tr>
<tr>
<td>R-CNN (VGGNet)</td>
<td>62.2%</td>
<td></td>
</tr>
<tr>
<td>R-CNN (VGGNet) + BB</td>
<td>66.0%</td>
<td></td>
</tr>
<tr>
<td>R-CNN (VGGNet)</td>
<td></td>
<td>Test: 47 s/im</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Train: 84 hours</td>
</tr>
</tbody>
</table>
Slow R-CNN

Apply bounding-box regressors
Classify regions with SVMs
Forward each region through ConvNet
Warped image regions
Regions of Interest (RoI) from a proposal method (~2k)
Input image

Slide credit: Ross Girshick
Object Detection System

Getting Proposals  Feature Extraction  Classifier
Object Detection System

Getting Proposals → Feature Extraction → Classifier

- aeroplane? no.
- person? yes.
- tvmonitor? no.
Spatial Pyramid Pooling

Spatial Pyramid Pooling

SPP-net

Trainable (3 layers)

Bbox reg

SVMs

FCs

Frozen (13 layers)

ConvNet

Slide credit: Ross Girshick
SPP-net Results

<table>
<thead>
<tr>
<th></th>
<th>VOC2007</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-CNN (ZFNet)</td>
<td>59.2%</td>
<td>14.5 s/im</td>
</tr>
<tr>
<td>R-CNN (VGGNet)</td>
<td>66.0%</td>
<td>47.0 s/im</td>
</tr>
<tr>
<td>SPP (ZFNet)</td>
<td>59.2%</td>
<td>0.38 s/im</td>
</tr>
<tr>
<td>SPP (VGGNet)</td>
<td>63.1%</td>
<td>2.3 s/im</td>
</tr>
</tbody>
</table>
Object Detection System

Getting Proposals → Feature Extraction → Classifier

SPP
Object Detection System

Getting Proposals → Feature Extraction → Classifier
Fast R-CNN

Totally end-to-end!

Multi-task loss

Trainable

Log loss + smooth L1 loss

Linear + softmax

Linear

FCs

ConvNet

Slide credit: Ross Girshick
# Fast R-CNN Results

<table>
<thead>
<tr>
<th>Model</th>
<th>VOC2007</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPPNet BB</td>
<td>63.1%</td>
</tr>
<tr>
<td>R-CNN BB</td>
<td>66.0%</td>
</tr>
<tr>
<td>Fast RCNN</td>
<td>66.9%</td>
</tr>
<tr>
<td>Fast RCNN (07+12)</td>
<td>70.0%</td>
</tr>
</tbody>
</table>
Object Detection System

Getting Proposals -> Feature Extraction -> Classifier

- aeroplane? no.
- person? yes.
- tvmonitor? no.

Fast R-CNN
Object Detection System

Getting Proposals

Feature Extraction

Classifier

(e.g. selective search)
Faster R-CNN

Region Proposal Network

- Sliding window style
- Multi-scale predictions on fixed-sized window for efficiency (take advantage of the large receptive field of CNN features)
- Same loss as R-CNN (cls+bbox)
Figure 2: Recall vs. IoU overlap ratio on the PASCAL VOC 2007 test set.
Faster R-CNN Results

- Fewer and better proposals not only bring speed-up, but also detection performance boost.

<table>
<thead>
<tr>
<th>method</th>
<th># proposals</th>
<th>data</th>
<th>mAP (%)</th>
<th>time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS</td>
<td>2k</td>
<td>07</td>
<td>66.9</td>
<td>1830</td>
</tr>
<tr>
<td>SS</td>
<td>2k</td>
<td>07+12</td>
<td>70.0</td>
<td>1830</td>
</tr>
<tr>
<td>RPN+VGG, unshared</td>
<td>300</td>
<td>07</td>
<td>68.5</td>
<td>342</td>
</tr>
<tr>
<td>RPN+VGG, shared</td>
<td>300</td>
<td>07</td>
<td>69.9</td>
<td>196</td>
</tr>
<tr>
<td>RPN+VGG, shared</td>
<td>300</td>
<td>07+12</td>
<td>73.2</td>
<td>196</td>
</tr>
</tbody>
</table>
Object Detection System

Getting Proposals

Feature Extraction

Classifier

Faster R-CNN
Efficient Object Detection System

Getting Proposals: Faster R-CNN
Feature Extraction: SPP
Classifier: Fast R-CNN

66.0% → 73.2%
47 s/im → 0.2 s/im
Example 4: Driving car

<table>
<thead>
<tr>
<th></th>
<th>Pascal 2007 mAP</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPM v5</td>
<td>33.7</td>
<td>0.07 FPS</td>
</tr>
<tr>
<td>R-CNN</td>
<td>66.0</td>
<td>0.05 FPS</td>
</tr>
</tbody>
</table>

\( \frac{1}{3} \) Mile, 1760 feet

Slide credit: Joseph Chet Redmon
Example 4: Driving car

<table>
<thead>
<tr>
<th></th>
<th>Pascal 2007 mAP</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPM v5</td>
<td>33.7</td>
<td>.07 FPS</td>
</tr>
<tr>
<td>R-CNN</td>
<td>66.0</td>
<td>.05 FPS</td>
</tr>
<tr>
<td>Fast R-CNN</td>
<td>70.0</td>
<td>.5 FPS</td>
</tr>
</tbody>
</table>

176 feet

Slide credit: Joseph Chet Redmon
#### Example 4: Driving car

<table>
<thead>
<tr>
<th></th>
<th>Pascal 2007 mAP</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPM v5</td>
<td>33.7</td>
<td>.07 FPS</td>
</tr>
<tr>
<td>R-CNN</td>
<td>66.0</td>
<td>.05 FPS</td>
</tr>
<tr>
<td>Fast R-CNN</td>
<td>70.0</td>
<td>.5 FPS</td>
</tr>
<tr>
<td>Faster R-CNN</td>
<td>73.2</td>
<td>7 FPS</td>
</tr>
</tbody>
</table>

Slide credit: Joseph Chet Redmon
Real-time object detectors?

<table>
<thead>
<tr>
<th>Method</th>
<th>mAP (%)</th>
<th>Speed (fps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-CNN, Girshick 2014</td>
<td>66%</td>
<td>0.02</td>
</tr>
<tr>
<td>Faster R-CNN, Ren 2015</td>
<td>73%</td>
<td>7</td>
</tr>
<tr>
<td>Fast R-CNN, Girshick 2015</td>
<td>70%</td>
<td>0.4</td>
</tr>
<tr>
<td>YOLO, Redmon 2016</td>
<td>66%</td>
<td>21</td>
</tr>
<tr>
<td>SSD300</td>
<td>77%</td>
<td>46</td>
</tr>
<tr>
<td>SSD512</td>
<td>80%</td>
<td>19</td>
</tr>
</tbody>
</table>

**Two-Stage**

- box proposal + postclassify

**Single Shot**

Slide credit: Wei Liu
YOLO: You Only Look Once

locations

class prob.

YOLO: output parameterization

Each cell predicts:
- For each bounding box:
  - 4 coordinates \((x, y, w, h)\)
  - 1 confidence value
- Some number of class probabilities

For Pascal VOC:
- 7x7 grid
- 2 bounding boxes / cell
- 20 classes

\[
7 \times 7 \times (2 \times 5 + 20) = 7 \times 7 \times 30 \text{ tensor} = 1470 \text{ outputs}
\]
YOLO: limitations

- Small objects
- Objects with different shapes/sizes
- Occluded objects
SSD: Single Shot MultiBox Detector

SSD: YOLO + default box shape + multi-scale

(a) Image with GT boxes  (b) 8 × 8 feature map  (c) 4 × 4 feature map

loc: \(\Delta(cx, cy, w, h)\)
conf: \((c_1, c_2, \cdots, c_p)\)

SSD: YOLO + default box shape + multi-scale

Object detection

• Introduction
• Pre-CNN time
  • HOG detector
  • Deformable Part-based Model
• CNN time
  • Region CNN
  • Fast versions of RCNN
  • YOLO/SSD
• 3D object detection
• Devil’s in the details
3D object detection: camera model

\[
\begin{bmatrix}
  u \\
  v \\
  1
\end{bmatrix}
= \begin{bmatrix}
  f_x & 0 & c_x \\
  0 & f_y & c_y \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  r_{11} & r_{12} & r_{13} & t_1 \\
  r_{21} & r_{22} & r_{23} & t_2 \\
  r_{31} & r_{32} & r_{33} & t_3 \\
  0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  X \\
  Y \\
  Z \\
  1
\end{bmatrix}
\]

\[
w \begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
= \begin{bmatrix}
  X \\
  Y \\
  Z \\
  1
\end{bmatrix} P
\]

\[
P = \begin{bmatrix}
  R \\
  t
\end{bmatrix} K
\]

Scale factor, Image points, World points

Camera matrix, Extrinsic matrix, Intrinsic matrix

Rotation and translation
3D object detection: pipeline

3D object detection: demo

Object detection

- Introduction
- Pre-CNN time
  - HOG detector
  - Deformable Part-based Model
- CNN time
  - Region CNN
  - Fast versions of RCNN
  - YOLO/SSD
- 3D object detection
- Devil’s in the details
Trick: Pre-trained model

<table>
<thead>
<tr>
<th>Faster R-CNN baseline</th>
<th>mAP@.5</th>
<th>mAP@.5:.95</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG-16</td>
<td>41.5</td>
<td>21.5</td>
</tr>
<tr>
<td>ResNet-101</td>
<td>48.4</td>
<td>27.2</td>
</tr>
</tbody>
</table>

COCO detection results (ResNet has 28% relative gain)

Gentlemen, our learner overgeneralizes because the VC-Dimension of our Kernel is too high. Get some experts and minimize the structural risk in a new one. Rework our loss function, make the next kernel stable, unbiased and consider using a soft margin.

Trick: Sampling

1. Use ‘ignore’ labels:

   Difficulties are defined as follows:
   - Easy: Min. bounding box height: 40 Px, Max. occlusion level: Fully visible, Max. truncation: 15 %
   - Moderate: Min. bounding box height: 25 Px, Max. occlusion level: Partly occluded, Max. truncation: 30 %
   - Hard: Min. bounding box height: 25 Px, Max. occlusion level: Difficult to see, Max. truncation: 50 %

2. Use hard-example mining:
   - Heuristics
   - Offline
   - Online[1]

Trick: Multi-region ensemble

Trick: Multi-scale feature fusion
Trick: Iterative localization

- Iterative bounding box regression
- Voting NMS

Object detection

- Introduction
- Pre-CNN time
  - HOG detector
  - Deformable Part-based Model
- CNN time
  - Region-CNN
  - Fast versions of R-CNN
  - YOLO/SSD
- 3D object detection
- Devil’s in the details

we need features!
we need flexible models!
we need better features!
we want to be fast!
we want to be real-time!
we like 3D!
we hack!
“The only stupid question is the one you never asked.” - Rich Sutton