Object Recognition II

Linda Shapiro

ECE/CSE 576

with CNN slides from Ross Girshick
Outline

• Object detection
  • the task, evaluation, datasets

• Convolutional Neural Networks (CNNs)
  • overview and history

• Region-based Convolutional Networks (R-CNNs)

• You Only Look Once (YOLO)
Image classification

- $K$ classes
- Task: assign correct class label to the whole image

Digit classification (MNIST)  
Object recognition (Caltech-101)
Classification vs. Detection

✅ Dog

Dog

Dog
Problem formulation

{ airplane, bird, motorbike, person, sofa }
Evaluating a detector

Test image (previously unseen)
First detection ...

☐ ‘person’ detector predictions
Second detection ...

□ ‘person’ detector predictions
Third detection ...

□ ‘person’ detector predictions
Compare to ground truth

- 'person' detector predictions
- ground truth 'person' boxes
Sort by confidence

0.9 ✓
0.8 X
0.6 ✓
0.5 ✓
0.2 X
0.1 X

true positive
(high overlap)

false positive
(no overlap, low overlap, or duplicate)
Evaluation metric

\[
\text{precision}@t = \frac{\# \text{true positives}@t}{\# \text{true positives}@t + \# \text{false positives}@t}
\]

\[
\text{recall}@t = \frac{\# \text{true positives}@t}{\# \text{ground truth objects}}
\]
Evaluation metric

Average Precision (AP)
0% is worst
100% is best

mean AP over classes (mAP)
Pedestrians

AP ~77%
More sophisticated methods: AP ~90%

(a) average gradient image over training examples
(b) each “pixel” shows max positive SVM weight in the block centered on that pixel
(c) same as (b) for negative SVM weights
(d) test image
(e) its R-HOG descriptor
(f) R-HOG descriptor weighted by positive SVM weights
(g) R-HOG descriptor weighted by negative SVM weights
Overview of HOG Method

1. Compute gradients in the region to be described
2. Put them in bins according to orientation
3. Group the cells into large blocks
4. Normalize each block
5. Train classifiers to decide if these are parts of a human
Details

• **Gradients**
  
  [-1 0 1] and [-1 0 1]^T were good enough filters.

• **Cell Histograms**
  
  Each pixel within the cell casts a weighted vote for an orientation-based histogram channel based on the values found in the gradient computation. (9 channels worked)

• **Blocks**
  
  Group the cells together into larger blocks, either R-HOG blocks (rectangular) or C-HOG blocks (circular).
More Details

• **Block Normalization**

  They tried 4 different kinds of normalization.

  • L1-norm
  • sqrt of L1-norm
  • L2 norm
  • L2-norm followed by clipping

• If you think of the block as a vector $\mathbf{v}$, then the normalized block is $\mathbf{v}/\text{norm}(\mathbf{v})$
1. Extract fixed-sized (64x128 pixel) window at each position and scale
2. Compute HOG (histogram of gradient) features within each window
3. Score the window with a linear SVM classifier
4. Perform non-maxima suppression to remove overlapping detections with lower scores
Input image → Normalize gamma & colour → Compute gradients → Weighted vote into spatial & orientation cells → Contrast normalize over overlapping spatial blocks → Collect HOG’s over detection window → Linear SVM → Person / non-person classification

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05
<table>
<thead>
<tr>
<th>-1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>centered</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>uncentered</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>-8</td>
<td>0</td>
</tr>
<tr>
<td>cubic-corrected</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>diagonal</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>-1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Sobel</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
- Histogram of gradient orientations

Orientation: 9 bins (for unsigned angles)

Histograms in 8x8 pixel cells

- Votes weighted by magnitude
- Bilinear interpolation between cells
Normalize with respect to surrounding cells

$$L2 - \text{norm}: v \rightarrow \frac{v}{\sqrt{\|v\|_2^2 + \epsilon^2}}$$
Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05

# features = \(15 \times 7 \times 9 \times 4 = 3780\)

- # features
- # cells
- # normalizations by neighboring cells
- # orientations

\[ X = \]
Training set

+ 

-
$0.16 = w^T x - b$

$\text{sign}(0.16) = 1$

$\Rightarrow$ pedestrian
Detection examples
Deformable Parts Model

• Takes the idea a little further
• Instead of one rigid HOG model, we have multiple HOG models in a spatial arrangement
• One root part to find first and multiple other parts in a tree structure.
The Idea

Articulated parts model

• Object is configuration of parts
• Each part is detectable

Images from Felzenszwalb
Deformable objects

Images from Caltech-256
Deformable objects

Images from D. Ramanan’s dataset
How to model spatial relations?

• Tree-shaped model
Model Overview

Model has a root filter plus deformable parts
Hybrid template/parts model

Detections

Template Visualization

root filters
coarse resolution

part filters
finer resolution

deformation models

Felzenszwalb et al. 2008
Pictorial Structures Model

\[ P(L|I, \theta) \propto \left( \prod_{i=1}^{n} p(I|l_i, u_i) \right) \prod_{(v_i, v_j) \in E} p(l_i, l_j|c_{ij}) \]

Appearance likelihood

Geometry likelihood
Results for person matching
Results for person matching
2012 State-of-the-art Detector: Deformable Parts Model (DPM)

1. Strong low-level features based on HOG
2. Efficient matching algorithms for deformable part-based models (pictorial structures)
3. Discriminative learning with latent variables (latent SVM)

Why did gradient-based models work?
Generic categories

Can we detect people, chairs, horses, cars, dogs, buses, bottles, sheep ...?
PASCAL Visual Object Categories (VOC) dataset
Generic categories

Why doesn’t this work (as well)?

Can we detect people, chairs, horses, cars, dogs, buses, bottles, sheep ...?
PASCAL Visual Object Categories (VOC) dataset
Quiz time
(Back to Girshick)
Warm up

This is an average image of which object class?
Warm up

pedestrian
A little harder
A little harder

Hint: airplane, bicycle, bus, car, cat, chair, cow, dog, dining table
A little harder

bicycle (PASCAL)
A little harder, yet
A little harder, yet

?  

Hint: white blob on a green background
A little harder, yet

sheep (PASCAL)
Impossible?
Impossible?

dog (PASCAL)
Impossible?

Why does the mean look like this?
There’s no alignment between the examples!
How do we combat this?

dog (PASCAL)
PASCAL VOC detection history

year

mean Average Precision (mAP)


DPM, MKL, Selective Search, DPM++, MKL

41% 41%

DPM++, MKL

37%

DPM++

28%

DPM, HOG, BOW

23%

DPM

17%
Part-based models & multiple features (MKL)

<table>
<thead>
<tr>
<th>Year</th>
<th>DPM</th>
<th>DPM, HOG+BOW</th>
<th>DPM, MKL</th>
<th>DPM++, MKL</th>
<th>Selective Search</th>
<th>DPM++, MKL</th>
</tr>
</thead>
<tbody>
<tr>
<td>2006</td>
<td>17%</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2007</td>
<td>23%</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2008</td>
<td>28%</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2009</td>
<td>37%</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2010</td>
<td>41%</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2011</td>
<td>41%</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2012</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2013</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2014</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2015</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Kitchen-sink approaches

mean Average Precision (mAP) vs year

increasing complexity & plateau

- DPM
- HOG + BOW
- DPM, MKL
- DPM++, Selective Search
- DPM++, MKL

17%, 23%, 28%, 37%, 41%, 41%
Region-based Convolutional Networks (R-CNNs)

mean Average Precision (mAP)

year


DPM, HOG + BOW

DPM, MKL

DPM++, MKL, Selective Search

DPM++, MKL

[DPM, Girshick et al. CVPR 2014]
Region-based Convolutional Networks (R-CNNs)

mean Average Precision (mAP)

year

~1 year

~5 years

[R-CNN. Girshick et al. CVPR 2014]
Convolutional Neural Networks

• Overview
Standard Neural Networks

\[ x = (x_1, \ldots, x_{784})^T \]

\[ z_j = g(w_j^T x) \]

\[ g(t) = \frac{1}{1 + e^{-t}} \]
From NNs to Convolutional NNs

• Local connectivity
• Shared ("tied") weights
• Multiple feature maps
• Pooling
Convolutional NNs

- Local connectivity

- Each green unit is only connected to (3) neighboring blue units
Convolutional NNs

• Shared ("tied") weights

• All green units share the same parameters $w$

• Each green unit computes the same function, but with a different input window
Convolutional NNs

• Convolution with 1-D filter: \([w_3, w_2, w_1]\]

• All green units share the same parameters \(w\)

• Each green unit computes the same function, but with a different input window
Convolutional NNs

• Convolution with 1-D filter: \([w_3, w_2, w_1]\)

• All green units share the same parameters \(w\)

• Each green unit computes the same function, but with a different input window
Convolutional NNs

- Convolution with 1-D filter: \([w_3, w_2, w_1]\]

- All green units share the same parameters \(w\)

- Each green unit computes the same function, but with a different input window
Convolutional NNs

• Convolution with 1-D filter: \([w_3, w_2, w_1]\)

- All green units share the same parameters \(w\)
- Each green unit computes the same function, but with a different input window
Convolutional NNs

• Convolution with 1-D filter: \([w_3, w_2, w_1]\)

• All green units **share** the same parameters \(w\)

• Each green unit computes the **same function**, but with a **different input window**
Convolutional NNs

- Multiple feature maps

- All orange units compute the *same function* but with a *different input windows*

- Orange and green units *compute* *different functions*
Convolutional NNs

• Pooling (max, average)

- Pooling area (how much to pool): 2 units
- Pooling stride (how far to move): 2 units
- **Subsamples** feature maps
2D input

Pooling

Convolution

Image
Historical perspective – 1980

Hubel and Wiesel

1962

Included basic ingredients of ConvNets, but no supervised learning algorithm
Supervised learning – 1986

Gradient descent training with error backpropagation

Learning Internal Representations by Error Propagation

D. E. RUMELHART, G. E. HINTON, and R. J. WILLIAMS

Early demonstration that error backpropagation can be used for supervised training of neural nets (including ConvNets)
Backpropagation applied to handwritten zip code recognition, Lecun et al., 1989
Practical ConvNets

Gradient-Based Learning Applied to Document Recognition, Lecun et al., 1998
Core idea of “deep learning”

• Input: the “raw” signal (image, waveform, ...)

• Features: hierarchy of features is *learned* from the raw input
What’s new since the 1980s?

• More layers
  • LeNet-3 and LeNet-5 had 3 and 5 learnable layers
  • Current models have 8 – 20+

• “ReLU” non-linearities (Rectified Linear Unit)
  • \( g(x) = \max(0, x) \)
  • Gradient doesn’t vanish

• “Dropout” regularization (randomly selects neurons to remove during training epochs, reduces overfitting)

• Fast GPU implementations

• More data
Demo

• [http://cs.stanford.edu/people/karpathy/convnetjs/demo/mnist.html](http://cs.stanford.edu/people/karpathy/convnetjs/demo/mnist.html)

• ConvNetJS by Andrej Karpathy (Ph.D. student at Stanford)

Software libraries

• Caffe (C++, python, matlab)

• Torch7 (C++, lua)

• Theano (python)
What else? **Object Proposals**

- Sliding window based object detection

- Object proposals
  - Fast execution
  - High recall with low # of candidate boxes

Iterate over window size, aspect ratio, and location
The number of contours wholly enclosed by a bounding box is indicative of the likelihood of the box containing an object.
Ross’s Own System: Region CNNs

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

**R-CNN: Regions with CNN features**

warped region
Competitive Results

<table>
<thead>
<tr>
<th>VOC 2010 test</th>
<th>aero</th>
<th>bike</th>
<th>bird</th>
<th>boat</th>
<th>bottle</th>
<th>bus</th>
<th>car</th>
<th>cat</th>
<th>chair</th>
<th>cow</th>
<th>table</th>
<th>dog</th>
<th>horse</th>
<th>mbike</th>
<th>person</th>
<th>plant</th>
<th>sheep</th>
<th>sofa</th>
<th>train</th>
<th>tv</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPM v5 [20]*</td>
<td>49.2</td>
<td>53.8</td>
<td>13.1</td>
<td>15.3</td>
<td>35.5</td>
<td>53.4</td>
<td>49.7</td>
<td>27.0</td>
<td>17.2</td>
<td>28.8</td>
<td>14.7</td>
<td>17.8</td>
<td>46.4</td>
<td>51.2</td>
<td>47.7</td>
<td>10.8</td>
<td>34.2</td>
<td>20.7</td>
<td>43.8</td>
<td>38.3</td>
<td>33.4</td>
</tr>
<tr>
<td>UVA [39]</td>
<td>56.2</td>
<td>42.4</td>
<td>15.3</td>
<td>12.6</td>
<td>21.8</td>
<td>49.3</td>
<td>36.8</td>
<td>46.1</td>
<td>12.9</td>
<td>32.1</td>
<td>30.0</td>
<td>36.5</td>
<td>43.5</td>
<td>52.9</td>
<td>32.9</td>
<td>15.3</td>
<td>41.1</td>
<td>31.8</td>
<td>47.0</td>
<td>44.8</td>
<td>35.1</td>
</tr>
<tr>
<td>Regionlets [41]</td>
<td>65.0</td>
<td>49.8</td>
<td>25.9</td>
<td>24.6</td>
<td>24.5</td>
<td>56.1</td>
<td>54.5</td>
<td>51.2</td>
<td>17.0</td>
<td>28.9</td>
<td>30.2</td>
<td>35.8</td>
<td>40.2</td>
<td>55.7</td>
<td>43.5</td>
<td>14.3</td>
<td>43.9</td>
<td>32.6</td>
<td>54.0</td>
<td>45.9</td>
<td>39.7</td>
</tr>
<tr>
<td>SegDPM [18]*</td>
<td>61.4</td>
<td>53.4</td>
<td>25.6</td>
<td>25.2</td>
<td>35.5</td>
<td>51.7</td>
<td>50.6</td>
<td>50.8</td>
<td>19.3</td>
<td>33.8</td>
<td>26.8</td>
<td>40.4</td>
<td>48.3</td>
<td>54.4</td>
<td>47.1</td>
<td>14.8</td>
<td>38.7</td>
<td>35.0</td>
<td>52.8</td>
<td>43.1</td>
<td>40.4</td>
</tr>
<tr>
<td>R-CNN</td>
<td>67.1</td>
<td>64.1</td>
<td>46.7</td>
<td>32.0</td>
<td>30.5</td>
<td>56.4</td>
<td>57.2</td>
<td>65.9</td>
<td>27.0</td>
<td>47.3</td>
<td>40.9</td>
<td>66.6</td>
<td>57.8</td>
<td>63.9</td>
<td>53.6</td>
<td>26.7</td>
<td>56.5</td>
<td>38.1</td>
<td>52.8</td>
<td>50.2</td>
<td>50.2</td>
</tr>
<tr>
<td>R-CNN BB</td>
<td>71.8</td>
<td>65.8</td>
<td>53.0</td>
<td>36.8</td>
<td>35.9</td>
<td>59.7</td>
<td>60.0</td>
<td>69.9</td>
<td>27.9</td>
<td>50.6</td>
<td>41.4</td>
<td>70.0</td>
<td>62.0</td>
<td>69.0</td>
<td>58.1</td>
<td>29.5</td>
<td>59.4</td>
<td>39.3</td>
<td>61.2</td>
<td>52.4</td>
<td>53.7</td>
</tr>
</tbody>
</table>

Table 1: Detection average precision (%) on VOC 2010 test. R-CNN is most directly comparable to UVA and Regionlets since all methods use selective search region proposals. Bounding-box regression (BB) is described in Section C. At publication time, SegDPM was the top-performer on the PASCAL VOC leaderboard. *DPM and SegDPM use context rescoring not used by the other methods.

Figure 3: (Left) Mean average precision on the ILSVRC2013 detection test set. Methods preceded by * use outside training data (images and labels from the ILSVRC classification dataset in all cases). (Right) Box plots for the 200 average precision values per method. A box plot for the post-competition OverFeat result is not shown because per-class APs are not yet available (per-class APs for R-CNN are in Table 8 and also included in the tech report source uploaded to arXiv.org; see R-CNN-ILSVRC2013-APs.txt). The red line marks the median AP, the box bottom and top are the 25th and 75th percentiles. The whiskers extend to the min and max AP of each method. Each AP is plotted as a green dot over the whiskers (best viewed digitally with zoom).
Figure 4: Top regions for six pool$_2$ units. Receptive fields and activation values are drawn in white. Some units are aligned to concepts, such as people (row 1) or text (4). Other units capture texture and material properties, such as dot arrays (2) and specular reflections (6).
What came Next?

- Faster R-CNN (Girshick, 2016)
- YOLO (Redmon, Girshick, Divvala, Farhadi, 2016) *You Only Look Once* (People’s Choice Award)
- YOLO9000: Better, Faster, Stronger (Redmon, Farhadi, CVPR 2017) Runner up for Best Paper
- YOLOv3: more improvements
Accurate object detection is slow!

<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>
<tr>
<td>YOLO</td>
<td>63.4 69.0</td>
<td>45 FPS</td>
</tr>
</tbody>
</table>
With YOLO, you only look once at an image to perform detection

YOLO: You Only Look Once

1. Resize image.
2. Run convolutional network.
3. Threshold detections.
Each cell predicts boxes and confidences: $P(\text{Object})$
Each cell predicts boxes and confidences: $P(\text{Object})$
Each cell also predicts a class probability.
Then we combine the box and class predictions.
Finally we do NMS and threshold detections
This parameterization fixes the output size

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

\[7 \times 7 \times (2 \times 5 + 20) = 7 \times 7 \times 30 \text{ tensor} = 1470 \text{ outputs}\]
Thus we can train one neural network to be a whole detection pipeline
More YOLO (second paper)

• At 67 FPS, YOLOv2 gets 76.8 mAP on VOC 2007.
• At 40 FPS, YOLOv2 gets 78.6 mAP, outperforming state-of-the-art methods like Faster R-CNN with ResNet and SSD while still running significantly faster
• A new methodology that jointly trains for detection and classification produced YOLO9000.
• YOLO9000 can detect more than 9000 object categories in real time.
• And YOLOv3 has come out.
Finale

• Object recognition has moved rapidly in the last 12 years to becoming very appearance based.
• The HOG descriptor lead to fast recognition of specific views of generic objects, starting with pedestrians and using SVMs.
• Deformable parts models extended that to allow more objects with articulated limbs, but still specific views.
• CNNs have become the method of choice; they learn from huge amounts of data and can learn multiple views of each object class.
• YOLO is the current winner (I think). CVPR 19 is coming up in June.