Viola and Jones Face Detection

Slides adapted from Dr. S. Lazebnik and Dr. P. Viola
Face detection
Basic idea: slide a window across image and evaluate a face model at every location.
Challenges of face detection

- Sliding window detector must evaluate tens of thousands of location/scale combinations
- Faces are rare: 0–10 per image
  - For computational efficiency, we should try to spend as little time as possible on the non-face windows
  - A megapixel image has ~$10^6$ pixels and a comparable number of candidate face locations
  - To avoid having a false positive in every image, our false positive rate has to be less than $10^{-6}$
Viola/Jones Face Detector

- A seminal approach to real-time object detection
- Training is slow, but detection is very fast
- Key ideas
  - Integral images for fast feature evaluation
  - Boosting for feature selection
  - Attentional cascade for fast rejection of non-face windows

Image Features

“Rectangle filters”

Value = 

\[ \sum \text{(pixels in white area)} - \sum \text{(pixels in black area)} \]
Fast computation with integral images

- The *integral image* computes a value at each pixel \((x,y)\) that is the sum of the pixel values above and to the left of \((x,y)\), inclusive.
- This can quickly be computed in one pass through the image.
Computing the integral image
Computing the integral image

- Cumulative row sum: \( s(x, y) = s(x-1, y) + i(x, y) \)
- Integral image: \( ii(x, y) = ii(x, y-1) + s(x, y) \)

MATLAB: \( ii = 	ext{cumsum}(	ext{cumsum}(	ext{double}(i)), 2); \)
Computing sum within a rectangle

- Let A, B, C, D be the values of the integral image at the corners of a rectangle.
- Then the sum of original image values within the rectangle can be computed as:
  \[ \text{sum} = A - B - C + D \]
- Only 3 additions are required for any size of rectangle!
Example

Integral Image
Feature selection

- For a 24x24 detection region, the number of possible rectangle features is ~160,000!
Feature selection

- For a 24x24 detection region, the number of possible rectangle features is \(~160,000\)!
- At test time, it is impractical to evaluate the entire feature set.
- Can we create a good classifier using just a small subset of all possible features?
- How to select such a subset?
Boosting

- Boosting is a classification scheme that works by combining weak learners into a more accurate ensemble classifier
  - A weak learner need only do better than chance
- Training consists of multiple boosting rounds
  - During each boosting round, we select a weak learner that does well on examples that were hard for the previous weak learners
  - “Hardness” is captured by weights attached to training examples

Training procedure

- Initially, weight each training example equally
- In each boosting round:
  - Find the weak learner that achieves the lowest weighted training error
  - Raise the weights of training examples misclassified by current weak learner
- Compute final classifier as linear combination of all weak learners (weight of each learner is directly proportional to its accuracy)
- Exact formulas for re-weighting and combining weak learners depend on the particular boosting scheme (e.g., AdaBoost)

Boosting illustration

Weak Classifier 1
Boosting illustration

Weights Increased
Boosting illustration
Boosting illustration

Weights Increased
Boosting illustration

Weak Classifier 3
Final classifier is a combination of weak classifiers
Boosting vs. SVM

- Advantages of boosting
  - Integrates classification with feature selection
  - Complexity of training is linear instead of quadratic in the number of training examples
  - Flexibility in the choice of weak learners, boosting scheme
  - Testing is fast
  - Easy to implement

- Disadvantages
  - Needs many training examples
  - Often doesn’t work as well as SVM (especially for many-class problems)
Boosting for face detection

- Define weak learners based on rectangle features

\[
h_t(x) = \begin{cases} 
1 & \text{if } p_t f_t(x) > p_t \theta_t \\
0 & \text{otherwise}
\end{cases}
\]
Boosting for face detection

- Define weak learners based on rectangle features
- For each round of boosting:
  - Evaluate each rectangle filter on each example
  - Select best threshold for each filter
  - Select best filter/threshold combination
  - Reweight examples
- Computational complexity of learning: $O(MNK)$
  - $M$ rounds, $N$ examples, $K$ features
Boosting for face detection

- First two features selected by boosting:

This feature combination can yield 100% detection rate and 50% false positive rate
Boosting for face detection

- A 200-feature classifier can yield 95% detection rate and a false positive rate of 1 in 14084

Not good enough!

Receiver operating characteristic (ROC) curve
Attentional cascade

- We start with simple classifiers which reject many of the negative sub-windows while detecting almost all positive sub-windows.
- Positive response from the first classifier triggers the evaluation of a second (more complex) classifier, and so on.
- A negative outcome at any point leads to the immediate rejection of the sub-window.

![Diagram showing the flow of information through three classifiers: Classifier 1, Classifier 2, and Classifier 3, leading to the classification of a sub-window as a face or non-face.](image-url)
Attentional cascade

Chain classifiers that are progressively more complex and have lower false positive rates:

![Diagram showing the cascade of classifiers with a Receiver operating characteristic graph]
The detection rate and the false positive rate of the cascade are found by multiplying the respective rates of the individual stages.

A detection rate of 0.9 and a false positive rate on the order of $10^{-6}$ can be achieved by a 10-stage cascade if each stage has a detection rate of 0.99 ($0.99^{10} \approx 0.9$) and a false positive rate of about 0.30 ($0.3^{10} \approx 6 \times 10^{-6}$).
Training the cascade

- Set target detection and false positive rates for each stage
- Keep adding features to the current stage until its target rates have been met
  - Need to lower AdaBoost threshold to maximize detection (as opposed to minimizing total classification error)
  - Test on a validation set
- If the overall false positive rate is not low enough, then add another stage
- Use false positives from current stage as the negative training examples for the next stage
The implemented system

- **Training Data**
  - 5000 faces
    - All frontal, rescaled to 24x24 pixels
  - 300 million non-faces
    - 9500 non-face images
  - Faces are normalized
    - Scale, translation

- **Many variations**
  - Across individuals
  - Illumination
  - Pose
System performance

- Training time: “weeks” on 466 MHz Sun workstation
- 38 layers, total of 6061 features
- Average of 10 features evaluated per window on test set
- “On a 700 Mhz Pentium III processor, the face detector can process a 384 by 288 pixel image in about .067 seconds”
  - 15 Hz
  - 15 times faster than previous detector of comparable accuracy (Rowley et al., 1998)
Face Detector Results
Other detection tasks

Facial Feature Localization

Profile Detection

Male vs. female
Profile Detection
Profile Features
Summary: Viola/Jones detector

- Rectangle features
- Integral images for fast computation
- Boosting for feature selection
- Attentional cascade for fast rejection of negative windows
Finding tiny faces*

* Demo code: https://www.cs.cmu.edu/~peiyunh/tiny/
Overview of the detection pipeline

- Large context and scale-variant representations were used.
- Massively-large receptive fields captures both coarse context and high-resolution image features, which are necessary for detecting small objects.

Receptive fields:
- Res-block 1: 27x27
- Res-block 2: 99x99
- Res-block 3: 291x291
- Res-block 4: 483x483
Results (WiderFace dataset)

<table>
<thead>
<tr>
<th>Face size set</th>
<th>Size range in height</th>
<th>AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Large</td>
<td>Size &gt; 140 p</td>
<td>0.92</td>
</tr>
<tr>
<td>Medium</td>
<td>40 p &gt; Size &gt; 140 p</td>
<td>0.91</td>
</tr>
<tr>
<td>Small</td>
<td>Size &lt; 40 p</td>
<td>0.83</td>
</tr>
</tbody>
</table>