note: black & white
Object Category Detection

- Focus on object search: “Where is it?”
- Build templates that quickly differentiate object patch from background patch
Object Detection

• Overview

• Dalal-Triggs (pedestrian detection)
  – Histogram of Oriented Gradients
  – Database learning with SVM
Histogram of Oriented Gradients

Orientation: 9 bins (for unsigned angles 0 - 180)

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

Rectangular HOG (R-HOG)

How to normalize?

- Concatenate all cell responses from block into vector.
- Normalize vector.
- Extract responses from cell of interest.
- Do this 4x for each overlapping block.

\[ f = \frac{v}{\sqrt{||v||^2_2 + e^2}} \]

*e* is a small constant (for empty bins)
Person detection with HoG’s & linear SVM’s

- Histograms of Oriented Gradients for Human Detection, Navneet Dalal, Bill Triggs, International Conference on Computer Vision & Pattern Recognition - June 2005
Dalal-Triggs uses a template with a rigid form – humans are boxed shaped.

But...is there a way to learn the spatial layout more fluidly?

  – Might help us capture more appearance variation...

What about faster, too?
FACE DETECTION
Consumer application: Apple iPhoto

Things iPhoto thinks are faces
"The Nikon S60 detects up to 12 faces."
“Flashed Face Distortion”
2nd Place in the 8th Annual
Best Illusion of the Year
Contest, VSS 2012
Keep your eyes on the cross.
Face detection and recognition

Detection

Recognition

“Sally”
Challenges of face detection

Sliding window = tens of thousands of location/scale evaluations

- One megapixel image has \(~10^6\) pixels, and a comparable number of candidate face locations

Faces are rare: 0–10 per image

- For computational efficiency, spend as little time as possible on the non-face windows.

- For 1 Mpix, to avoid having a false positive in every image, our false positive rate has to be less than \(10^{-6}\)
Sliding Window Face Detection with Viola-Jones


The Viola/Jones Face Detector

A seminal approach to real-time object detection. Training is slow, but detection is very fast.

Key ideas:
1. *Integral images* for fast feature evaluation
2. *Boosting* for feature selection
3. *Attentional cascade* for fast non-face window rejection
1. *Integral images* for fast feature evaluation

- The *integral image* computes a value at each pixel \((x,y)\) that is the sum of all pixel values above and to the left of \((x,y)\), inclusive.

- This can quickly be computed in one pass through the image.

- ‘Summed area table’

\[
I_\Sigma(x, y) = \sum_{x' \leq x} \sum_{y' \leq y} i(x', y')
\]
Computing the integral image

Region already computed

Current pixel
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: \( ii = 
cumsum(cumsum(double(i)), 2); \)
Computing sum within a rectangle

- Let A, B, C, D be the values of the integral image at the corners of a rectangle.

- 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!
Integral Images

- \( ii = \text{cumsum}(\text{cumsum}(\text{im}, 1), 2) \)

\( ii(x,y) = \text{Sum of the values in the grey region} \)

SUM within Rectangle D is
\( ii(4) - ii(2) - ii(3) + ii(1) \)
Features that are fast to compute

“Haar-like features”

- Differences of sums of intensity
- Computed at different positions and scales within sliding window

Two-rectangle features

Three-rectangle features

Etc.

CC BY-SA 3.0,
hits://commons.wikimedia.org/w/index.php?curid=801361
Image Features

“Rectangle filters”

\[
\text{Value} = \sum (\text{pixels in white area}) - \sum (\text{pixels in black area})
\]
Computing a rectangle feature
But these features are rubbish…!

Yes, individually they are ‘weak classifiers’

*Jargon:* ‘feature’ and ‘classifier’ are used interchangeably here. Also with ‘learner’.

But, what if we combine *thousands* of them…
How many features are there?

For a 24x24 detection region, the number of possible rectangle features is ~160,000!
How many features are there?

• 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 learn a ‘strong classifier’ using just a small subset of all possible features?
2. *Boosting* for feature selection

Initially, weight each training example equally.

Weight = size of point

Slide credit: Paul Viola
In each boosting round:

Find the weak classifier that achieves the lowest *weighted* training error.

Raise the weights of training examples misclassified by current weak classifier.
In each boosting round:

Find the weak classifier that achieves the lowest weighted training error.

Raise the weights of training examples misclassified by current weak classifier.
In each boosting round:

Find the weak classifier that achieves the lowest \textit{weighted} training error.

Raise the weights of training examples misclassified by current weak classifier.
In each boosting round:

Find the weak classifier that achieves the lowest *weighted* training error.

Raise the weights of training examples misclassified by current weak classifier.
In each boosting round:

Find the weak classifier that achieves the lowest weighted training error.

Raise the weights of training examples misclassified by current weak classifier.
Compute final classifier as linear combination of all weak classifier.

Weight of each classifier 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 for face detection

- First two features selected by boosting:

  This feature combination can yield 100% recall and 50% false positive rate
Feature selection with boosting

- Create a large pool of features (180K)
- Select discriminative features that work well together

\[ h(x) = \text{sign} \left( \sum_{j=1}^{M} \alpha_j h_j(x) \right) \]

- “Weak learner” = feature + threshold + ‘polarity’

\[ h_j(x) = \begin{cases} -s_j & \text{if } f_j < \theta_j \\ s_j & \text{otherwise} \end{cases} \]

‘polarity’ = black or white region flip

- Choose weak learner that minimizes error on the weighted training set, then reweight
1. Input the positive and negative training examples along with their labels \( \{(x_i, y_i)\} \), where \( y_i = 1 \) for positive (face) examples and \( y_i = -1 \) for negative examples.

2. Initialize all the weights to \( w_{i,1} \leftarrow \frac{1}{N} \), where \( N \) is the number of training examples. (Viola and Jones (2004) use a separate \( N_1 \) and \( N_2 \) for positive and negative examples.)

3. For each training stage \( j = 1 \ldots M \):
   
   (a) Renormalize the weights so that they sum up to 1 (divide them by their sum).

   (b) Select the best classifier \( h_j(x; f_j, \theta_j, s_j) \) by finding the one that minimizes the weighted classification error

   \[
   e_j = \sum_{i=0}^{N-1} w_{i,j} e_{i,j},
   \]

   \[
   e_{i,j} = 1 - \delta(y_i, h_j(x_i; f_j, \theta_j, s_j)).
   \]

   For any given \( f_j \) function, the optimal values of \( (\theta_j, s_j) \) can be found in linear time using a variant of weighted median computation (Exercise 14.2).

   (c) Compute the modified error rate \( \beta_j \) and classifier weight \( \alpha_j \),

   \[
   \beta_j = \frac{e_j}{1 - e_j} \quad \text{and} \quad \alpha_j = -\log \beta_j.
   \]

   (d) Update the weights according to the classification errors \( e_{i,j} \)

   \[
   w_{i,j+1} \leftarrow w_{i,j} \beta_j^{1 - e_{i,j}},
   \]

   i.e., downweight the training samples that were correctly classified in proportion to the overall classification error.

4. Set the final classifier to

\[
h(x) = \text{sign} \left[ \sum_{j=0}^{m-1} \alpha_j h_j(x) \right].
\]
Boosting vs. SVM

**Advantages of boosting**
- Integrates classifier training 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

**Disadvantages**
- Needs many training examples
- Training is slow
- Often doesn’t work as well as SVM (especially for many-class problems)
Fast classifiers early in cascade which reject many negative examples but detect almost all positive examples.

Slow classifiers later, but most examples don’t get there.
Attentional cascade

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

![Diagram of attentional cascade]

Receiver operating characteristic

\[
\begin{array}{c}
\text{% False Pos} \\
\hline
0 & 0 \quad 100 \quad 50 \\
0 & \text{% True positive}
\end{array}
\]
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 boosting 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
Viola-Jones details

- 38 stages with 1, 10, 25, 50 ... features
  - 6061 total used out of 180K candidates
  - 10 features evaluated on average

- Training Examples
  - 4916 positive examples
  - 10000 negative examples collected after each stage

- Scanning
  - Scale detector rather than image
  - Scale steps = 1.25 (factor between two consecutive scales)
  - Translation 1*scale (# pixels between two consecutive windows)

- Non-max suppression: average coordinates of overlapping boxes

- Train 3 classifiers and take vote
System performance

• Training time: “weeks” on 466 MHz Sun workstation
• 38 cascade layers, total of 6061 features
• Average of 10 features evaluated per window on test set

“The 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)
Viola Jones Results
Speed = 15 FPS (in 2001)

MIT + CMU face dataset
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
Output of Face Detector on Test Images
Other detection tasks

Facial Feature Localization

Profile Detection

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

- Rectangle features
- Integral images for fast computation
- Boosting for feature selection
- Attentional cascade for fast rejection of negative windows
Things to remember

• Sliding window for search

• Features based on differences of intensity (gradient, wavelet, etc.)
  – Excellent results = careful feature design

• Boosting for feature selection

• Integral images, cascade for speed

• Bootstrapping to deal with many, many negative examples