CSE 573 Project 3: Face Detection in the Wild

Arvind Thirumurugan, 50289656
Department of Computer Science and Engineering
University at Buffalo
athirumu@buffalo.edu

1 Introduction

The Purpose of project 3 is to implement Viola-jones face detection algorithm to detect faces in the Wild. The algorithm detects faces in a given image using five different Haar features. The algorithm is trained using the FDDB dataset for positive images which contains more than 2800 images and associated bounding box annotation, for more than 5700 faces. And for the negative images the CBCL dataset provided by MIT is utilized

2 Implementation of the project

2.1 Processing the images and building the dataset

The first step in the implementation is to read the images and their corresponding bounding information from the files and extract faces as positive samples for the algorithm from FDDB. Then the negative images are obtained from the CBCL dataset.

2.2 Extracting features from dataset

From the dataset created we resize each image to 19*19 images and extract five different Haar features from the images at different positions in the image with different height and width.

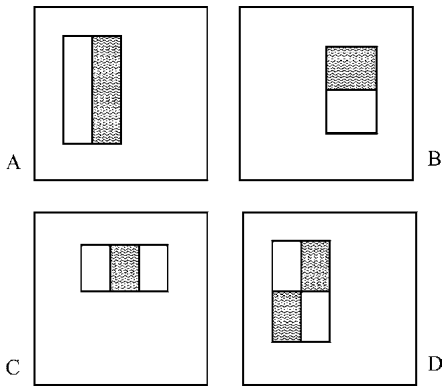

Fig 1: Haar features extracted from the image

In a 19*19 image there are more than 60,000 features per image and in order to extract these efficiently the Integral image values for each image is calculated where each value in the integral image denotes all the sum of all the pixel values to above and to the left of its position. The integral image can be calculated as follows,

$$s(x, y) = s(x, y-1) + i(x, y)$$
$$ii(x, y) = ii(x-1, y) + s(x, y)$$

Fig 2: Formula used for calculating Integral image

2.3 Training Strong Classifiers from weak classifiers

After the features are extracted from the images, we need to specify the number of weak classifiers required for each Strong classifier. Initially, we initialize the weights for each training example and normalize the weights for each iteration. Then for each feature we train a weak classifier using a single feature and find the error with respect to the weights.

Then choose the classifier with the lowest error, store the current weak classifier and update the weights used for the next weak classifier. After obtaining all the weak classifiers we form the Strong classifiers which are a linear combination of the weak classifiers obtained so far.

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively where m and l are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:
  1. Normalize the weights, $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$
  2. Select the best weak classifier with respect to the weighted error
     $$\epsilon_t = \min_{f,p,\theta} \sum_i w_i \left| h(x_i, f, p, \theta) - y_i \right|.$$
     See Section 3.1 for a discussion of an efficient implementation.
  3. Define $h_t(x) = h(x, f_t, p_t, \theta_t)$ where f_t, p_t, and θ_t are the minimizers of ϵ_t.
  4. Update the weights:
     $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$
     where $e_i = 0$ if example x_i is classified correctly, $e_i =$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- The final strong classifier is:

$$C(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

where $\alpha_t = \log \frac{1}{\beta_t}$

Fig 3: Steps involved in training the Strong Classifier

In this implementation two Strong classifiers will be utilized to detect faces on the test dataset, where the first Strong classifiers contains five weak classifiers and the second Strong classifier contains fifteen weak classifiers.

3 Results obtained by implementing the algorithm

Since the model predicts using only one strong classifier containing 15 weak classifiers the model doesn't predict all the faces and the false positive rate is also a bit high

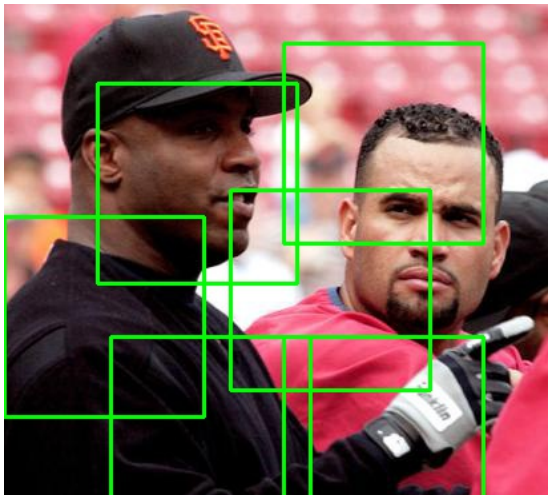

Fig 4: Regions Detected by the model with a one Strong Classifier

4 Possible Improvements

4.1 Attentional Cascade

In order to increase accuracy and get better results while detecting faces multiple cascades can be used to detect false positives where each Cascade contains a specific number of Strong Classifiers. The ensures that negative samples get rejected in the earlier cascades and only samples which might be faces go through all the cascades to be detected as a face. Cascading also makes sure that false positives are not classified as faces.

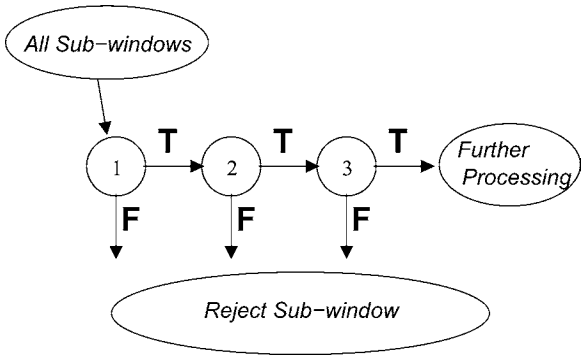

Fig 5: Schematic depiction of Detection Cascade

5 References:

[1] http://www.ai.mit.edu/courses/6.899/lectures/faces.tar.gz

[2] https://docs.opencv.org/3.4.1/d1/de0/tutorial_py_feature_homography.html

[3] https://docs.opencv.org/3.4/d9/dab/tutorial_homography.html

[4] https://www.pyimagesearch.com/2014/11/17/non-maximum-suppression-object-detection-python/