Arnav Pandey – 2020102016

Geet Dasani – 2020102001

Rajarshi Ray – 2020101127

SMAI Project:

Rapid Object Detection using a Boosted Cascade of Simple Features

Introduction

Goal:

In the project, we aim to build a model for visual object detection using Adaboost and Cascading with simple features which processes images extremely rapidly and achieves high detection rates.

There are three important contributions that make this project stand out.

- The first is the bringing of intermediate stage of a brand-new picture representation termed the "Integral Image," which enables the very speedy computation of the characteristics employed by our detector.

- The second is an AdaBoost-based learning algorithm that chooses a small number of crucial visual characteristics from a wider set and produces incredibly effective classifiers.
- The third contribution is a technique called "cascading" classifiers, which allows background portions of the picture to be swiftly eliminated while promising object-like regions receive longer processing time.

Dataset

Dataset is MIT-CBCL face recognition dataset.

We downloaded the dataset in pickel format, this was done for faster and easier computation, as pickle compresses the dataset(which consists of many images) into one file and data is stored in a file in such a way that it is faster for RAM to read and by pickel library in python we can unpack the file and extract data from it and further computations are done. Our database contains both face and non-face images for good training and testing.

Methodology

Features:

More technically called Haar features.

Our object detection process divides photos into categories depending on the importance of basic attributes. We employ features rather than pixels because they may be used to encapsulate ad-hoc domain knowledge that is challenging to acquire with a limited amount of training data. The fact that this system functions far quicker than a pixel-based system adds a second crucial justification for features.

We essentially use 3 kinds of features:

1) The value of a two-rectangle feature is the difference between the sum of the pixels within two rectangular regions.
2) A three-rectangle feature computes the sum within two outside rectangles subtracted from the sum in a center rectangle.
3) Finally, a four-rectangle feature computes the difference between diagonal pairs of rectangles.

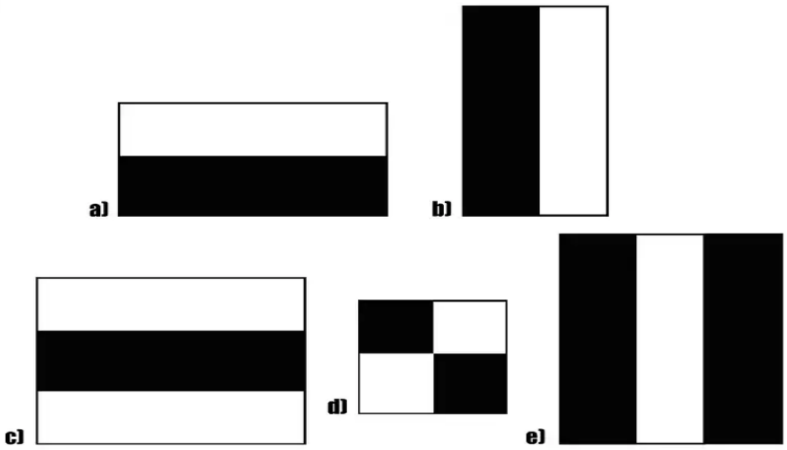
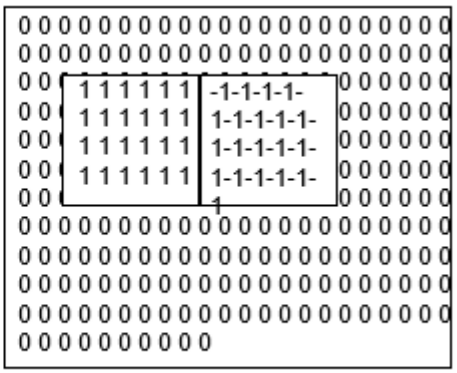


Fig. A sample of Haar features used in the Original Research Paper published by Viola and Jones.

The first pair of rectangular characteristics is in charge of determining whether the edges are horizontal or vertical. Finding out whether a lighter zone is flanked by darker parts on each side or the opposite is the responsibility of the second set of three rectangular characteristics. The variation in pixel intensities across diagonals is determined by the third group of four rectangle characteristics.



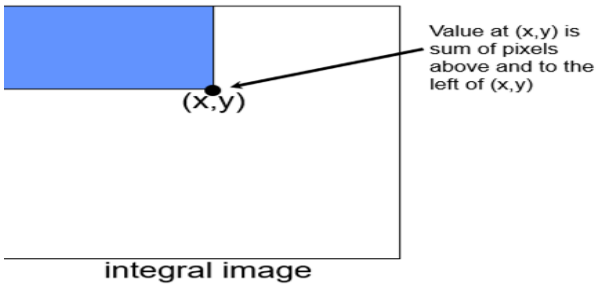
Rectangle features can be computed very rapidly using an intermediate representation of the image which we call the **integral image.**

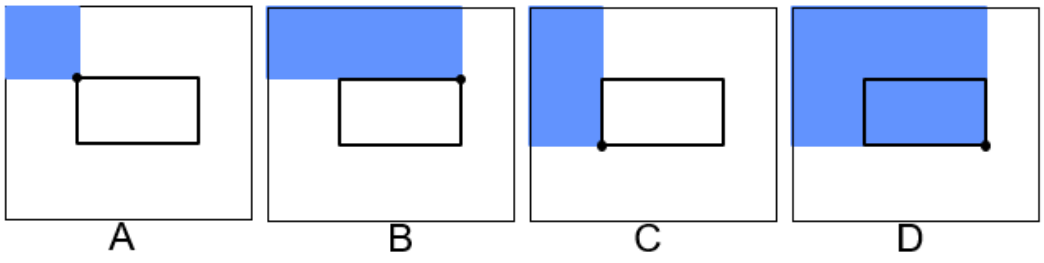
Integral Image:

In order to compute these features very rapidly at many scales we introduce the integral image representation for images. Once computed, any one of these Haar-like features can be computed at any scale or location in constant time.

The integral image at location (x,y) contains the sum of the pixels above and to the left of (x,y) inclusive.

$$ii(x,y) = \sum_{x' \le x, y' \le y} i(x',y')$$





Any rectangular sum may be calculated in four array references using the integral image. The two-rectangle characteristics listed above can be calculated in six array

references, eight for three-rectangle features, and nine for four-rectangle features since they include neighbouring rectangular sums.

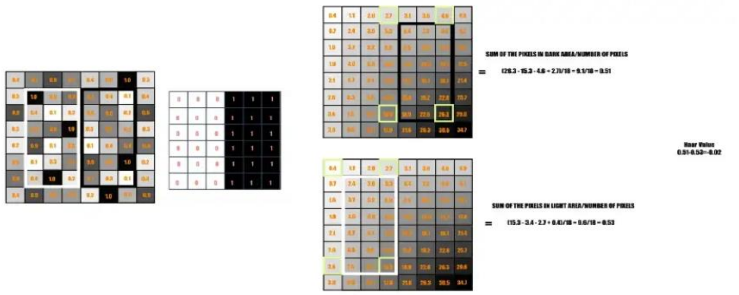


Fig. Integral Image is used here to calculate the haar value.

In this case, the goal is to calculate the total of all the picture pixels located in the haar feature's darker and lighter areas, respectively. then ascertain what makes them different. The haar value will be closer to 1 if there is an edge in the image dividing bright pixels on the left from dark pixels on the right. In other words, if the haar value is closer to 1, we declare that an edge has been found. Since the haar number is distant from 1.

Learning Classification Purposes:

Any variety of machine-learning techniques might be used to develop a classification function if there were a feature set and a training set of positive and negative pictures.

The system will employ a modified version of AdaBoost to choose a constrained set of features and train the classifier.

The weak learning algorithm we plan on designing is to be made to select the single rectangle feature which best separates the positive and negative examples.

For each feature, the weak learner will determine the optimal threshold classification function, such that the minimum number of examples is misclassified.

A weak classifier $h_j(x)$ thus consists of a feature f_j, a threshold, θ_j and a parity p_j indicating the direction of the inequality sign.

$$h_j(x) = \{1 \ , if\ p_j f_j(x) < p_j \theta_j$$

$$0 \ , otherwise \}$$

AdaBoost Algorithm:

Steps(Brief):

I. Start with a uniform weights on training examples.

II. For T rounds:

Calculate Weighted error for each feature and pick best.

III. Re-write the examples as:

Incorrectly Classified-More Weight

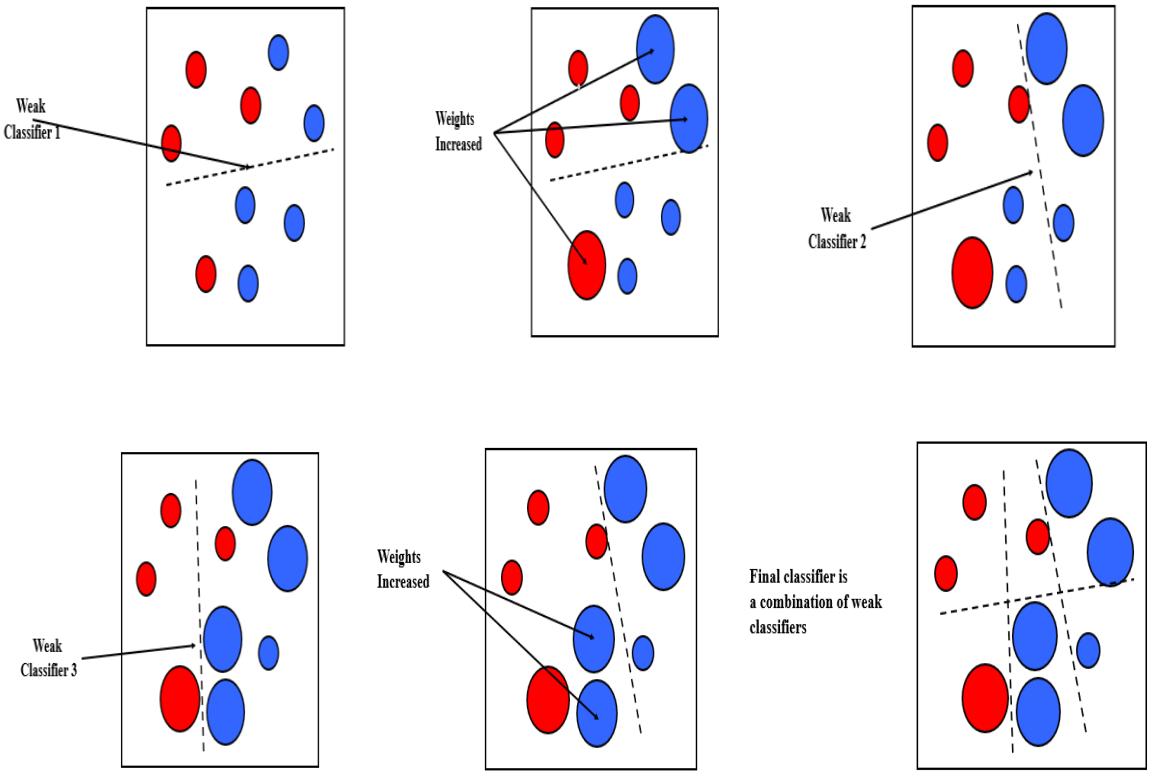
Correctly Classified- Less Weight

IV. Final Classifier is the combination of the weak ones, weighted according to the error they had.

Full Procedure:

- 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,
  $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
  so that w_t is a probability distribution.
  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$.
  3. Choose the classifier, h_t, with the lowest error 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, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- The final strong classifier is:
$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$
where $\alpha_t = \log \frac{1}{\beta_t}$

Classifiers through each updation:



Cascading Classifiers for Detection:

I. Form a cascade with low false negative rates early on.

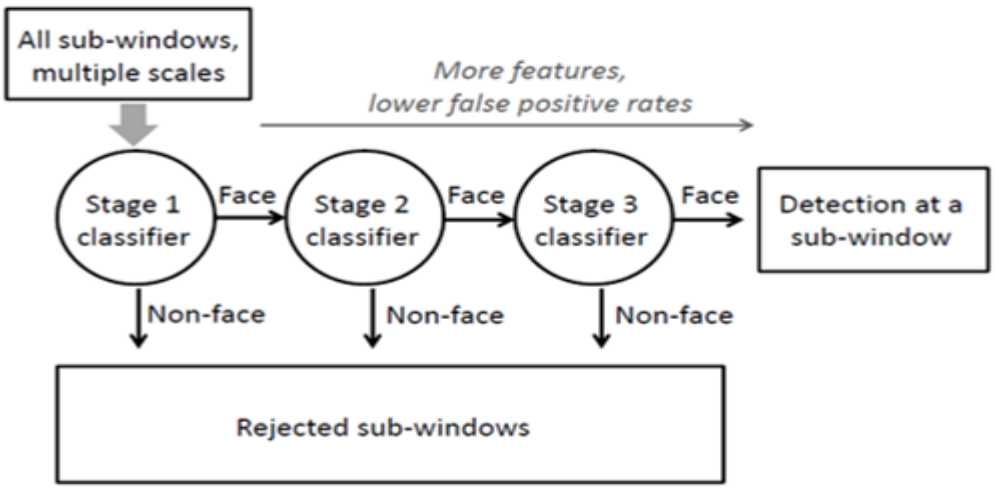
II. Apply less accurate but faster classifiers first to immediately discard windows that clearly appear to be negative.

III. Train with 5K positives, 350M negatives.

IV. Real-time detector using 38 layer cascade.

V. 6061 features in all layers.

Block Diagram of Cascading Procedure:



Thus, a random object can be classified as a face or a non-face using these cascading classifiers.

Results

False Positive Rate: 1045/4548 **(0.229771)**

False Negative Rate: 2324/2429 **(0.956772)**

Accuracy found was 3608/6977 (0.517128) = **51.7128%**

Average Classification Time: **0.000819**

A 38 layer cascaded classifier that can recognise frontal upright faces was taught to do so. A collection of face and nonface training photos were utilised to train the detector. 4916 manually labelled faces were scaled and aligned to a base resolution of 24 by 24 pixels for the face training set. The faces were taken out of pictures that were retrieved through a random crawl of

the worldwide web. Figure 5 displays a few instances of normal faces. The 9544 photos that were personally examined and determined to be faceless are the source of the non-face subwindows that were utilised to train the detector. These non-face pictures include roughly 350 million subwindows each.

The choice of delta affects both the speed of the detector as well as accuracy. The results we present are for delta = 1.0.

We can achieve a significant speedup by setting delta = 1.5 with only a slight decrease in accuracy.

Conclusion

We have provided an object detection method that reduces computing time while obtaining high detection accuracy. The method was utilised to create a face detection system that is around 15 times quicker than any previous approach available during its time.

References

https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola