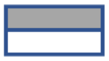
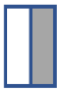
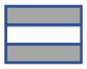
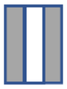
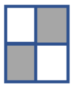
Project 3
Face Detection in the Wild

Submitted by –
Raj Kalpeshkumar Shah (50321013)
Sahil Kapahi (50317075)

An overview of the Viola-Jones algorithm-

Viola Jones algorithm is a machine learning algorithm which takes in a large number of linear classifiers to form a non-linear space for the purpose of face-detection. The algorithm as defined in the paper [1] is composed of three different training parts including – Haar feature calculation (based on integral image concept), Boosting and Cascading.

The entire pipeline can be explained as below:
1) Haar feature extraction – These are rectangular features which consider adjacent rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region and calculates the difference between these sums. Viola Jones algorithm includes 5 different categories of these simple features.

- Horizontal two-rectangle features 
- Vertical two-rectangle features 
- Horizontal three-rectangle features 
- Vertical three-rectangle features 
- Four-rectangle features 

These can be done very quickly using integral image which needs to be calculated only once and features can thus be extracted very efficiently over a dataset as large as 10,000 plus images of size 24x24. The extracted features have all the possibilities of window sizes over a 24x24 window which gives 162,336 possible features in total.

2) Boosting - Adaboost is used on extracted features to further filter out a set of relevant features which capture the features of human faces. These are simple rectangular features which can be used to differentiate between faces and non-faces based on thresholding to form a boundary of separation for each feature.

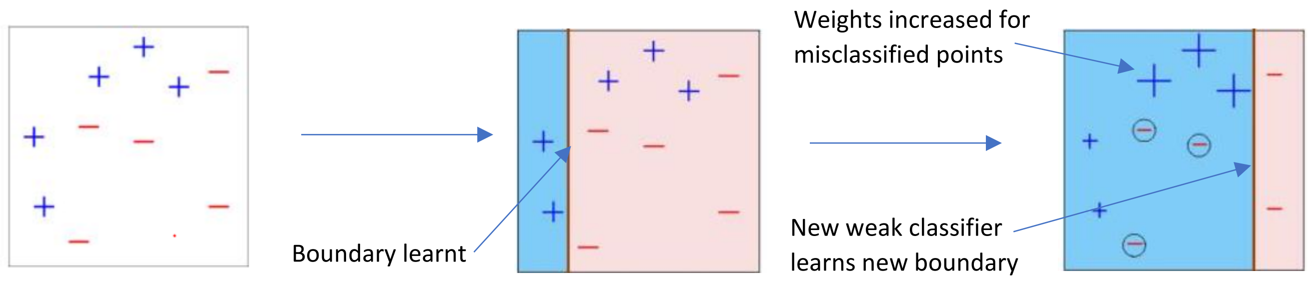

Boundary learnt
Weights increased for misclassified points
New weak classifier learns new boundary

Figure1: Adaboost algorithm explained diagrammatically (*Image Source: Leroy Samson*)

The thresholds are learnt on the basis that the sum of error on negative and positive misclassifications is minimum for each of the classifier and the classifier with the minimum error gets selected. Errors are calculated based on weights assigned to each image in the dataset such that the total sums up to one. Once a classifier gets selected based on minimum error, the weights for all the classifiers get updated based on the misclassifications of the best selected classifier to set a higher value for the misclassified images and a lower for the rightly classified. This is done to learn a new boundary (classifier in this case) which makes a minimum error on the updated weights. The process is repeated till a sufficient number of classifiers are extracted to rightly learn all the required features.

The final strong classifier obtained from the combination of weak classifiers is used for the final detection. Each weak classifier in the strong classifier is given weights based on its error. Lower the error, higher will be the weight of the weak classifier.

This can be calculated according to the formula $h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha h(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha \\ 0\ otherwise \end{cases}$, where α is the weight of the classifier t, and can be calculated as $\alpha = \log 1/\beta_t$.

3) Cascading – Finally, when the total number of features are reduced from 162,336 to around 6000, it still takes time to calculate all the 6000 features on all the possible sub-windows in an image. For this purpose, cascading is done to divide the entire detection process into stages. The cascading allows the classifiers to work in a certain order so as to minimize the detection time. This helps in very rapid rejection of the irrelevant background information to allow more computation time on the relevant face windows. The entire set of classifiers can be divided into stages such that each stage has a set true positive and false positive rate. Starting with the first stage, a false positive rate is decided for each stage and the classifiers are added to that stage till it meets its false positive target while also maintaining the true positive target. A new stage is started once the previous stage satisfies all the desired metrics expected from it. The stages are added till the desired false positive rate target for the entire detector is met. Thus, breaking the entire process in stages helps in the fast detection of the face data in any given image.

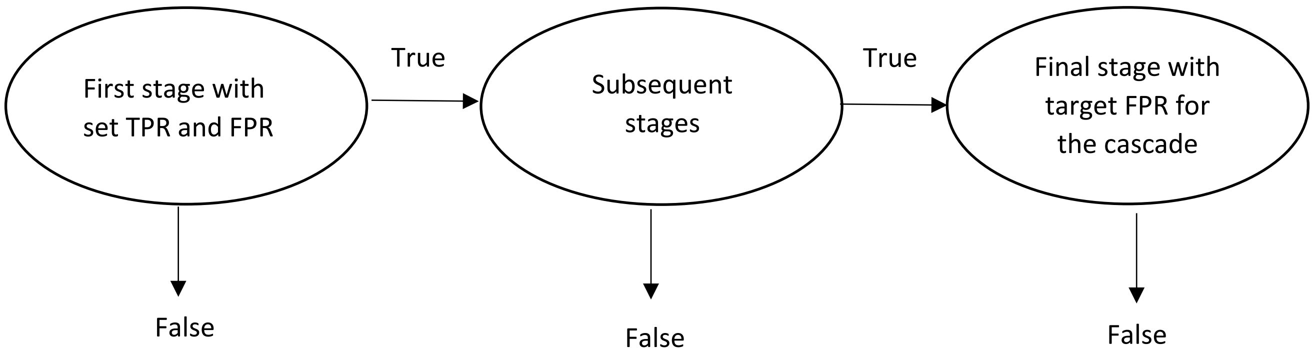

First stage with set TPR and FPR
True
Subsequent stages
True
Final stage with target FPR for the cascade
False
False
False

Figure2: Cascade stages

Description of our Implementation-

Our implementation is similar to that described in the paper.

Dataset –

From a total of 5000+ face data downloaded, **3800 faces were used along with their vertical flips giving a total of 7600 images**. The non-face data was extracted from the complete set of images by cropping the non-face patches from the complete image. This was done very carefully to avoid any faces. For this purpose, we completely neglected the face bounding boxes in the image dataset as we knew the coordinates and window sizes of the faces from the available text files. **A total of 7966 negative images were used.**

Feature Extraction –

The extracted image data was variance normalized in order to avoid any illumination issues. Integral images were then computed to help compute the haar features efficiently. The computed integral images were then used to extract features based on simple addition and subtraction operations on the different pixel locations defining the dimensions of the features. This gave us a total of 2x(43200) two-rectangle features, 2x(27600) three-rectangle features and 20736 four-rectangle features. A total of 162,336 features were thus extracted over a 24x24 window of face and non-face data.

Adaboost –

Once we got the features calculated and stored in the form of a matrix, we went on with implementing adaboost to **extract the relevant 1100 features**. We kept the weak classifier count to 1100 as it was computationally very expensive to handle a large dataset and we had to shift our computations to cloud computing station in order to complete it successfully.

To speed up the optimal threshold calculation in the first adaboost iteration we computed the medians of the face and non-face dataset over each feature in

order to reduce our search space. The calculation was vectorized for the entire feature space and the search was done between the medians of the face and non-face data. The threshold for the subsequent weak classifier search was kept constant keeping in mind the time spent on the optimal threshold calculation. For each iteration, the classifier with minimum error on the training data was selected as the best classifier and the weights were then updated for the subsequent calculations based on the misclassification errors of the best classifiers on both face and non-face images. This was repeated till we extracted 1100 weak classifiers as compared to a set target of false positive rate as computation time was the major limitation we faced for the adaboost process. It took nearly 2 days for us to extract 1100 weak classifiers on cloud computing station.
The first features we got out of the adaboost were relevant and were easily seen as why they were selected over the other features. Some of them were the bridge feature of the nose, the two-horizontal over the eyes and the curve of the chin. Initial error we got was around 0.16 as compared to 0.10 in the paper[1] .

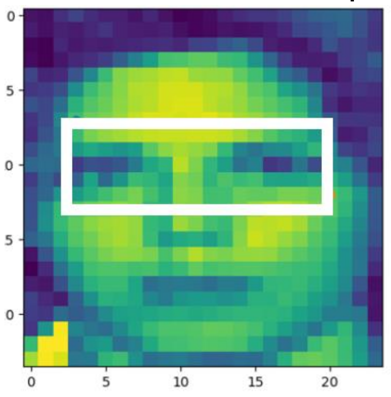


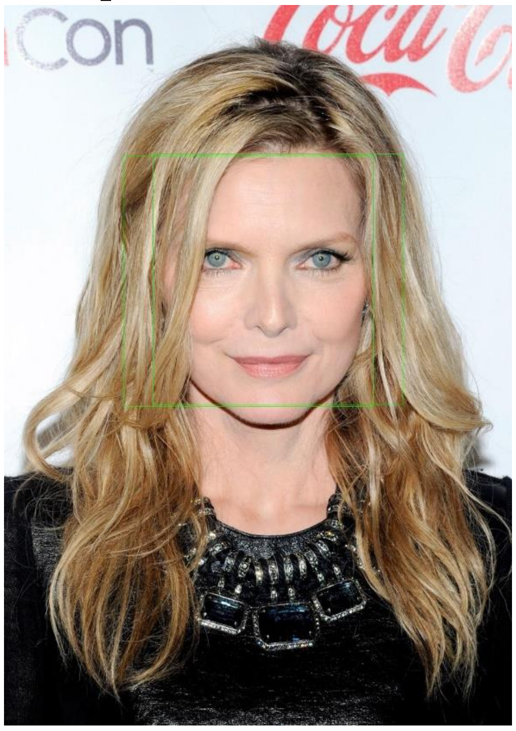
Figure3: Sample weak classifier extracted using adaboost

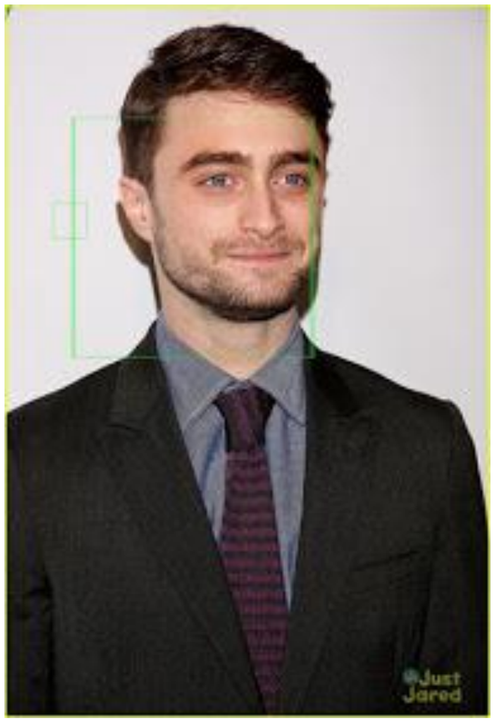
This can be attributed to the fact that we did a limited search between the medians by dividing the space into 25 points and taking the threshold corresponding to minimum error. Subsequently, the error increase to 0.20+ and then 0.30+ in a few hundred iterations. The error finally rose to 0.4+ towards the end.

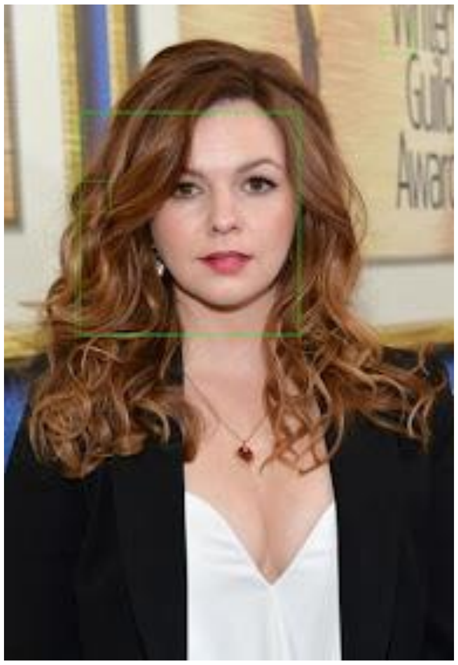
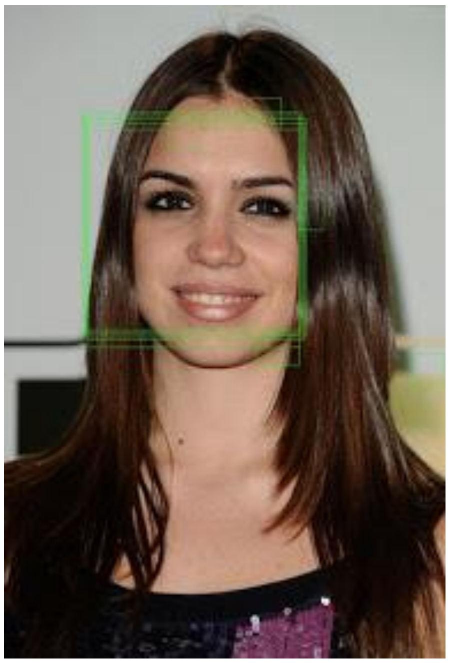
Cascading –

The extracted classifiers were finally cascaded to get multiple detection stages. This was divided **into 20 stages** with each stage having a true positive rate of 99%. The first stage had a false positive rate of 50% but it was not possible to reduce the false positive rate to below 70% for the subsequent stages. **This gave a total true positive rate of 82% and a false positive rate of 6%.** The cascade training was done with the condition to achieve a set false positive and true positive rate for each stage. However, since we had limited weak classifiers and thresholds calculated up-to a certain best value, we could not achieve a very good false positive rate.

Sample Results –





References

1. Rapid Object Detection using a Boosted Cascade of Simple Features - Paul Viola, Michael Jones