Facial Image Identification Using The Viola-Jones Method

Farkhan
Department of Informatics
Univeristas Pembangunan Nasional
"Veteran" Jawa Timur
Surabaya, Indonesia
farkhanjayadi@gmail.com

Azka Avicenna Rasjid
Department of Informatics
Univeristas Pembangunan Nasional
"Veteran" Jawa Timur
Surabaya, Indonesia
azkablack@gmail.com

Raviy Bayu Setiaji
Department of Informatics
Univeristas Pembangunan Nasional
"Veteran" Jawa Timur
Surabaya, Indonesia
bayu.setiaji709@gmail.com

Revansyah Rachmad Akbar
Department of Informatics
Univeristas Pembangunan Nasional
"Veteran" Jawa Timur
Surabaya, Indonesia
refansyah69@gmail.com

***Abstract*— Identification technology using digital images has developed rapidly. Some things that are identified from digital images include fingerprints, palms, and human faces. Face identification is easier to do because it only requires a digital image with a human face which is easier to obtain compared to getting someone's fingerprints or palms. This research was conducted to explain the process of identifying faces using the Viola-Jones method which has high accuracy with a fast calculation time. The Viola-Jones method uses the haar function as a descriptor and then combines Integral Image and AdaBoost to find and select key values to form a cascade classifier. The face detection system implemented using the Python programming language managed to get 90.9% accuracy and required an average time of 15 seconds for all samples tested from the method used.**

Keywords—viola jones, face detection, haar feature, cascade classifier

I. Introduction

Identification technology using digital images has developed rapidly. Some things that are identified from digital images include fingerprints, palms, and human faces. Face identification is easier to do because it only requires a digital image with a human face which is easier to obtain compared to getting someone's fingerprints or palms. Therefore, in this paper the author will only identify faces using the Viola Jones method.

From a human face, there is some feature information that can be obtained, including the eyes, nose and mouth. With this feature information, it can be determined whether there are faces or not in an image. The case of determining the location of the face aims to detect the position of the previously mentioned facial features using a face detection system [1].

In this research, the author will explain the workings of the Viola Jones method and the form of its application into a simple system using the OpenCV library and the Python programming language. When the system has been completed, it will then explain how face detection works starting from image acquisition, image processing, pattern recognition, and image analysis. Then the last test is done on facial characters that can be detected.

II. Related Works

Previous research has been carried out by Yi-Qing Wang [2] with the title "An Analysis of the Viola-Jones Face Detection Algorithm" which aims to describe the Viola Jones algorithm as the first face detection system in real time. In this study, it was said that there were three ingredients for fast and accurate face detection, namely integral images, adaboost, and attention cascades.

In another study entitled "Face Detection System Based on the Viola-Jones Algorithm", a face detection process was carried out from direct images using the Viola Jones algorithm and the Adaboost algorithm from the haar feature was used to extract facial areas from images. In addition, this research also uses cascading of stages to make the process run faster [3].

In a study entitled "Wider Face: A Face Detection Benchmark" has introduced a WIDER FACE dataset, which is a face dataset that can be used to train and evaluate face detection algorithms and has 32,203 images with 393,703 labeled faces, the number is ten times more larger than the currently available facial dataset. The WIDER FACE dataset has rich annotations, including occlusions, poses, event categories, and face bounding boxes. The facial images in the proposed dataset are very challenging because they have a large amount of variation. This research shows an example of using WIDER FACE using a multi-scale two-stage cascade framework that uses a divide and conquer strategy to deal with large-scale data variations. A set of convolutional networks with various input sizes used in this framework is trained to handle facial images with a certain scale range. Four representative algorithms were compared and evaluated at different settings and analyzed the conditions under which a method failed [4].

Another study entitled "Facial Parts Detection Using Viola Jones Algorithm" performs face detection and searches for facial features in an image by involving the viola jones cascade object detector algorithm which provides various combinations of filters and methods to detect facial expressions. The face detection process is carried out on parts of the face such as the eyes, nose, mouth, and the whole face. In this study, a face database called the Bao database was used and it gave an accuracy of 92%. The Bao database has more variations and a higher level of complexity than the AR-Face and Yale Face databases [5].

III. Research Metodology

A. *Research methods*

In conducting this research, the following are the methods that will be used to carry out the research, namely as follows.

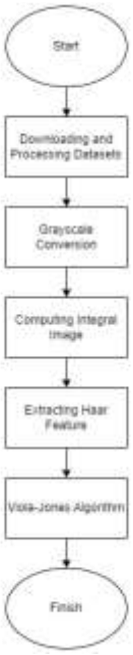


Fig. 1. Research Flow Chart

Figure 3.1 is a flowchart of the research stages that will be carried out by the author. The stage starts from the stage of downloading and processing the required data set to be used in the research being conducted. The next process is to convert the image to gray. If the image used has turned gray, then proceed with calculating the integral value in the image used. After the integral value of the image is obtained, then proceed with haar feature extraction. The last process is building the viola jones architecture to train data to recognize facial images.

B. *Downloading and Processing Datasets*

The earliest step in this research was to find the right data set containing facial and non-face images to use in the study. For the positive class (face images), the researcher used the dataset "A Century of Portraits: A Visual Historical Record of American High School Yearbooks". The dataset contains 37,921 front-facing portrait images. From this dataset, researchers only used 2,000 images. This is due to the limited computational capabilities available. For the negative class (non-face images) the researcher has used the Stanford background image data set from http://dags.stanford.edu/projects/scenedataset.html which contains 715 images of various backgrounds that are not “faces”.

The number of images available from this dataset is not balanced enough to be juxtaposed with the face dataset to be used, so it is necessary to balance the dataset. To do so, the researchers performed data augmentation to increase the pool size of the non-face dataset to 1430 images by randomly cropping regions from the original 715 images that had been used as non-face images.

Next, the researcher resized all the images to a uniform size, namely 22x22 and divided our entire data set into training data which contained 1600 facial images and 1144 non-faced images and test data which contained 400 facial images and 286 non-faced images.

C. *Grayscale Conversion*

This study performs face detection in a given image, in performing face detection in images using the Viola Jones method, color information is not needed. Therefore, the researcher converted the image into a grayscale image. Image conversion is performed on both facial and non-face images. Apart from unnecessary color information, this process is also needed to reduce the computational burden when processing image data.

Fig. 2. Grayscale Face Image

Fig. 3. Grayscale Non-Face Image

D. *Computing Integral Image*

The next step is to perform calculations using integral images. The integral image is used so that each rectangular sum can be calculated in four array references. Rectangle features calculated using integral images can be done more quickly. Integral images at location (x, y) are calculated using the number of pixels above and to the left of x, y, inclusive. The figure 4 illustrates the integral image calculation [6].

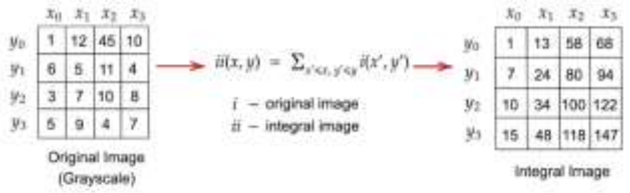


Fig. 4. The Integral Image Calculation

Using the Integral figure, the sum of any rectangle can be calculated using the four array references. Since the process of extracting haar-like features involves calculating the sum of the rectangles from the lighter/darker regions, Integral image recognition greatly speeds up the process. Integral image calculations were performed for our entire data set. We then proceed to build haar-like features.

E. *Extracting Haar Feature*

Viola and Jones in their paper have defined the following 3 types of Haar-like features as follows [6].

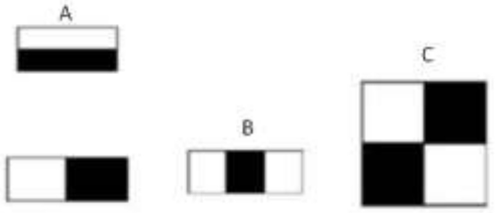


Fig. 5. The Haar-like Features

The way to do calculations on the rectangular feature is to add up the pixels in the white rectangle and then subtract the number of pixels in the black rectangle. Figure (A) shows the edge features. Figure (B) shows a line feature, and (D) a four-sided feature. In addition to the four features above, the researcher also considers the fifth feature of the line type feature in this study which can be seen in the figure 6.

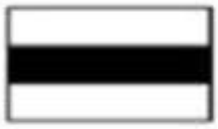
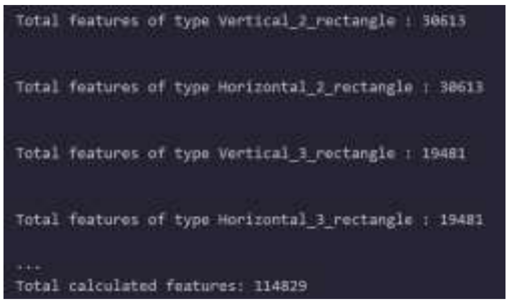


Fig. 6. The Fifth Line Feature

In the code, we have created a library for the above five haar-like feature types which stores tuples denoting our individual Haar features. The total number of features of different types for a 22x22 image is shown in the figure 7.



```
Total features of type Vertical_2_rectangle : 30613

Total features of type Horizontal_2_rectangle : 30613

Total features of type Vertical_3_rectangle : 19481

Total features of type Horizontal_3_rectangle : 19481

...
Total calculated features: 114829
```

Fig. 7. Output Number of Feature of 22x22 Images

When these features are overlaid on an image, the values corresponding to the light regions are added and the values corresponding to the dark regions are subtracted from the above sum. This haar-like feature helps researchers extract useful information such as edges, straight lines and diagonals that can be used to identify objects, such as human faces.

Even for small images, the researcher will get many haar features, in this study more than 110,000 features for a 22 x 22 image. In order to enumerate all the features efficiently, Viola and Jones introduce the integral image which was mentioned in the previous step.

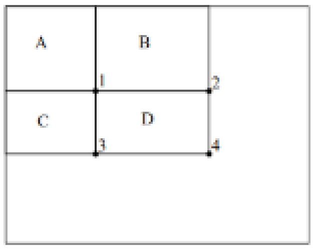


Fig. 8. Four Array References

Using an integral image, the number of pixels in rectangle D in the figure 8 can be calculated with four array references. The number of pixels in rectangle A gives the integral image value at location 1 in the figure 8. A+B gives the value at location 2, A+C gives the value at location 3, and A+B+C+D gives the value at location 4. The sum in D can be calculated as 4+1 - (2 +3). By using this method, we can calculate the value of all haar-like features.

F. *Viola Jones Algorithm*

The Viola-Jones algorithm uses a variant of AdaBoost for training. The following are the steps of the Viola-Jones algorithm.

1. Initialize the weights for each training example

   At the start of the algorithm, we give each training example the same weight. If the positive classes are p and negative classes are n, then we assign the following weights to the training examples:

$$w_i = \begin{cases} \frac{1}{2p} \; if x = 1, \\ \frac{1}{2n} \; if \; x = 0 \end{cases} \tag{1}$$

   Where x is the class of the training example.

2. Building the features

   The training function requires selecting the best weak classifier and every possible feature giving one weak classifier. So, we build all the features before implementing the training and store the type of each feature (out of 5 defined haar-like feature types). The algorithm iterates through all the rectangles in the image and checks if it is possible to generate features.

3. Applying the features

   We implemented the features before we started training the classifier because the value of each feature for the image never changes. Implementing a feature before training also allows us to pre-select features (we use the SelectPercentile from the sklearn.feature_selection library to pre-select features) to speed up training.

4. Training the weak classifier

   Viola-Jones uses a series of weak classifiers and combines them according to their weights to get a final strong classifier. Each weak classifier looks at one feature (f). Each weak classifier has a threshold (θ) and polarity (p) to determine the classification of the training examples according to the following equation:

$$h(x, f, p, \theta) = \begin{cases} 1 \; if \; pf(x) < p\theta \\ 0 \; otherwise \end{cases} \tag{2}$$

   Each feature is the sum of the positive rectangular areas minus the negative rectangular area sums. For each feature, the researcher trained a classifier to find its optimal threshold and polarity using the training sample weights. Each time a new weak classifier is selected as the best, all weak classifiers must be retrained because the training examples are assigned different weights after each round. Since this is a computationally expensive process, optimized methods are used to perform weak classifier training. First sort the weights based on the corresponding feature value. It then iterates through the series of weights and calculates the error if a threshold is chosen to be that feature. The error is calculated using the following equation:

$$\text{Error} = \text{minimum}(P + TN - N, N + TP - P) \tag{3}$$

Where, P = the sum of the weights of all the positive examples seen so far. N = sum of the weights of all the negative examples seen so far. TP = Total weight of all positive examples. TN = Total weight of all positive examples.

By using this equation, we can find the error of each threshold in constant time and the error of all thresholds in linear time. We find thresholds and polarities with minimum errors. The possible values for the thresholds are the feature values in each training example. The threshold is set to the feature value that has the minimum error. Polarity is determined by how many positive and negative examples have feature values (of the particular feature being considered) that are lower or greater than the threshold. Polarity p=1, If there are more positive examples with feature values less than the threshold, otherwise p=-1.

5. Selecting the best weak classifier

   To select the best weak classifier in each round, we iterate through all classifiers and calculate the weighted average error of each classifier on the training dataset, and choose the classifier with the lowest error.

6. Updating the weights

   After selecting the best weak classifier, we update the weights with the errors of the selected weak classifier. Training examples that were classified correctly were given a smaller weight, examples that were classified incorrectly had no change in weight. We update the weights according to the following equation:

$$\epsilon = min_{f,p,\theta} \sum_i^N w_i \, |h(x,f,p,\theta) - y_i| \tag{4}$$

$$\beta = \frac{\epsilon}{1-\epsilon} \tag{5}$$

$$W_i = w_i \beta^{1-e_i} \tag{6}$$

   Wi is the weight of the i-th example, ϵ is given by the error of the weakest classifier, and $\boldsymbol{\beta}$ is obtained using the above equation, giving the factors used to change the weights. The power of $\boldsymbol{\beta}$ is 1-e where e is 0 if the training sample is correctly classified and 1 if it is classified incorrectly.

7. Strong classifier

   The final classifier is defined as follows:

$$\mathrm{C(x)} = \begin{cases} 1 & \text{if } \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t, \\ 0 & \text{otherwise} \end{cases}$$
$$\alpha_t = ln(\frac{1}{\beta_t}) \tag{7}$$

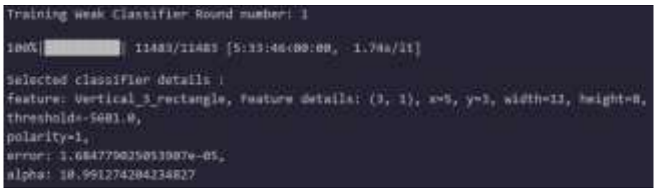
   The final classifier is a weighted linear combination of T (in our case T=10) the number of weak classifiers in which the weights (denoted by the coefficient $\boldsymbol{\alpha}$) depend on the error because they are the natural log of the inverse of $\boldsymbol{\beta}$. If the weighted sum of the weak classifier decisions is greater than or equal to half of the sum of all $\boldsymbol{\alpha}$, then we assign it to class 1 otherwise it is assigned to class 0.

8. Attentional cascade

   The attentional cascade uses a series of Viola-Jones classifiers, each increasing in complexity for classifying images. Classifiers are used in a cascade mode where only the first classifier trains all training examples, and each subsequent classifier is trained on all positive and negative examples that were misclassified by the previous classifier i.e., false positives.

IV. Result and Discussion

The algorithm that has been made, then the researcher runs as many as 10 rounds. Following are the details of the weak classifiers that Adaboost gets in every round that has been made:

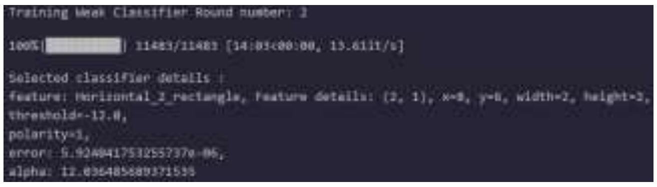


```
Training Weak Classifier Round number: 1

100%|██████████| 11483/11483 [5:33:46<00:00,  1.74s/it]

Selected classifier details :
feature: Vertical_3_rectangle, Feature details: (3, 1), x=5, y=3, width=12, height=8,
threshold=-5081.0,
polarity=1,
error: 1.6847790250539807e-05,
alpha: 10.991274204234827
```

Fig. 9. AdaBoost 1st Round

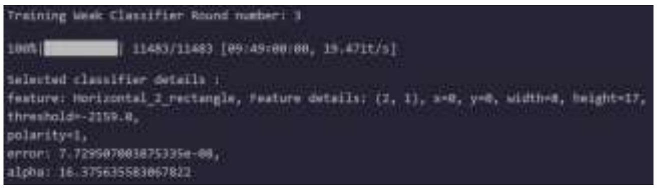


```
Training Weak Classifier Round number: 2

100%|██████████| 11483/11483 [14:03<00:00, 13.61it/s]

Selected classifier details :
feature: Horizontal_2_rectangle, Feature details: (2, 1), x=8, y=6, width=2, height=2,
threshold=-12.0,
polarity=1,
error: 5.924041753255737e-06,
alpha: 12.036485689371535
```

Fig. 10. AdaBoost 2n Round

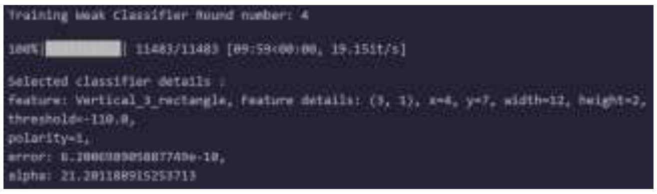


```
Training Weak Classifier Round number: 3

100%|██████████| 11483/11483 [09:49<00:00, 19.47it/s]

Selected classifier details :
feature: Horizontal_2_rectangle, Feature details: (2, 1), x=0, y=0, width=8, height=17,
threshold=-2159.0,
polarity=1,
error: 7.729507003875335e-08,
alpha: 16.375635583067822
```

Fig. 11. AdaBoost 3rd Round

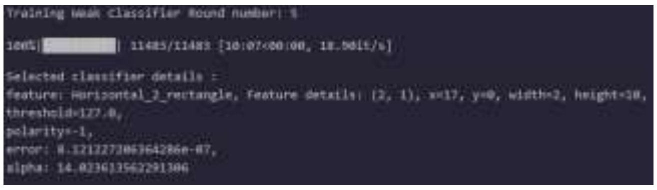


```
Training Weak Classifier Round number: 4

100%|██████████| 11483/11483 [09:59<00:00, 19.15it/s]

Selected classifier details :
feature: Vertical_3_rectangle, Feature details: (3, 1), x=4, y=7, width=12, height=2,
threshold=-110.0,
polarity=1,
error: 6.200038905887749e-10,
alpha: 21.201188915253713
```

Fig. 12. AdaBoost 4th Round

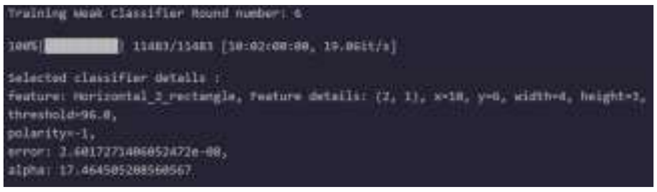


```
Training Weak Classifier Round number: 5

100%|██████████| 11483/11483 [10:07<00:00, 18.90it/s]

Selected classifier details :
feature: Horizontal_2_rectangle, Feature details: (2, 1), x=17, y=0, width=2, height=10,
threshold=127.0,
polarity=-1,
error: 8.121227306364286e-07,
alpha: 14.023613562291306
```

Fig. 13. AdaBoost 5th Round

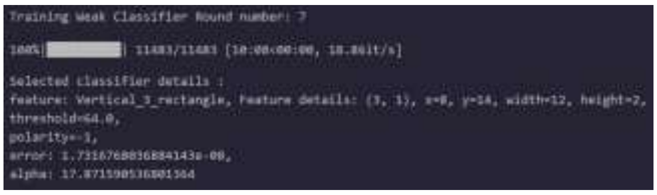


```
Training Weak Classifier Round number: 6

100%|██████████| 11483/11483 [10:02<00:00, 19.06it/s]

Selected classifier details :
feature: Horizontal_2_rectangle, Feature details: (2, 1), x=10, y=0, width=4, height=2,
threshold=96.0,
polarity=-1,
error: 2.6017273406052472e-08,
alpha: 17.464505208560567
```

Fig. 14. AdaBoost 6th Round

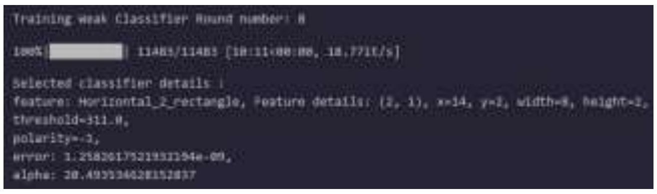


```
Training Weak Classifier Round number: 7

100%|██████████| 11483/11483 [10:08<00:00, 18.86it/s]

Selected classifier details :
feature: Vertical_3_rectangle, Feature details: (3, 1), x=8, y=14, width=12, height=2,
threshold=64.0,
polarity=-1,
error: 1.7316768836884143e-08,
alpha: 17.871590536801364
```

Fig. 15. AdaBoost 7th Round

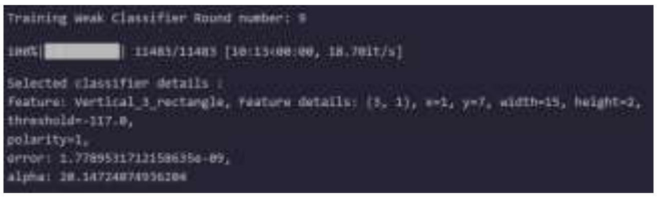


```
Training weak Classifier Round number: 8

100%|██████████| 11483/11483 [10:11<00:00, 18.77it/s]

Selected classifier details :
feature: Horizontal_2_rectangle, Feature details: (2, 1), x=14, y=2, width=8, height=2,
threshold=311.0,
polarity=-1,
error: 1.2582617521932194e-09,
alpha: 20.493534628152837
```

Fig. 16. AdaBoost 8th Round

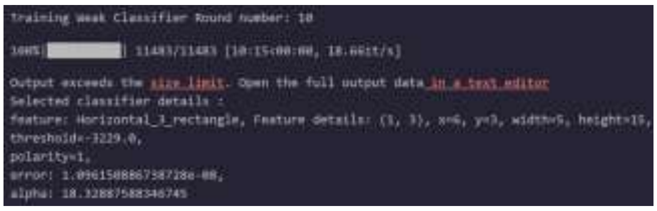


```
Training Weak Classifier Round number: 9

100%|██████████| 11483/11483 [10:13<00:00, 18.70it/s]

Selected classifier details :
Feature: Vertical_3_rectangle, Feature details: (3, 1), x=1, y=7, width=15, height=2,
threshold=-117.0,
polarity=1,
error: 1.7789531712158635e-09,
alpha: 20.147248749562 04
```

Fig. 17. AdaBoost 9th Round



```
Training Weak Classifier Round number: 10

100%|██████████| 11483/11483 [10:15<00:00, 18.66it/s]

Output exceeds the size limit. Open the full output data in a text editor
Selected classifier details :
feature: Horizontal_3_rectangle, Feature details: (1, 3), x=6, y=3, width=5, height=15,
threshold=-3229.0,
polarity=1,
error: 1.096150886738728e-08,
alpha: 18.328875883346745
```

Fig. 18. AdaBoost 10th Round

The classifier evaluation metric on our training dataset obtains an accuracy rate after 10 rounds, namely 99.9271% and the maximum accuracy level obtained is in the 10th round with a value of 99.9271%. Details of the classifier evaluation metrics on the training dataset can be seen in table 1.

TABLE I. CLASSIFIER EVALUATION METRICS ON THE TRAINING DATASET

No.	Threshold	Accuracy	False Positive	False Negative
1	0.5	95.2624%	1.6399%	3.0977%
2	0.5	88.7755%	10.2405%	0.9840%
3	0.5	97.9227%	0.2551%	1.8222%
4	0.5	97.9592%	2.0044%	0.0364%
5	0.5	98.8703%	0.2187%	0.9111%
6	0.5	99.0889%	0.5831%	0.3280%
7	0.5	99.5991%	0.3644%	0.0364%
8	0.5	99.3805%	0.1822%	0.4373%
9	0.5	99.6720%	0.2915%	0.0364%
10	0.5	99.9271%	0.0364%	0.364%

The classifier evaluation metric on our testing dataset obtains an accuracy rate after 10 rounds, namely 99.4196% and the maximum accuracy level obtained is in the 7th round with a value of 99.4196%. Details of the classifier evaluation metrics on the testing dataset can be seen in table 2.

TABLE II. CLASSIFIER EVALUATION METRICS ON THE TESTING DATASET

No.	Threshold	Accuracy	False Positive	False Negative
1	0.5	96.9388%	2.1866%	0.8746%
2	0.5	88.4840%	10.7872%	0.7289%
3	0.5	98.1050%	0.8746%	1.0204%
4	0.5	97.8134%	1.7493%	0.4373%
5	0.5	98.3965%	0.5831%	1.0204%
6	0.5	98.2507%	0.7289%	1.0204%
7	0.5	99.4169%	0.4373%	0.1458%
8	0.5	99.2711%	0.4373%	0.2915%
9	0.5	99.4169%	0.2915%	0.2915%
10	0.5	99.4169%	0.2915%	0.2915%

A receiver operating characteristic curve, or ROC curve, is a graphical plot that illustrates the diagnostic capability of a binary classifier system as its discrimination threshold varies [7].

$$C(x) = \begin{cases} 1 \; if \; \sum_{t=1}^{T} a_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} a_t \\ 0 \; otherwise \end{cases}$$

$$a_t = ln\left(\frac{1}{\beta_t}\right) \tag{8}$$

From equation 8, we can see that the threshold of the combined classifier is given by ½ * sum(**α**). We vary the thresholds of these combined classifiers to calculate the False Positive Rate and True Positive Rate for each of the selected thresholds, which we then plot to obtain the ROC curve as shown in figure 19.

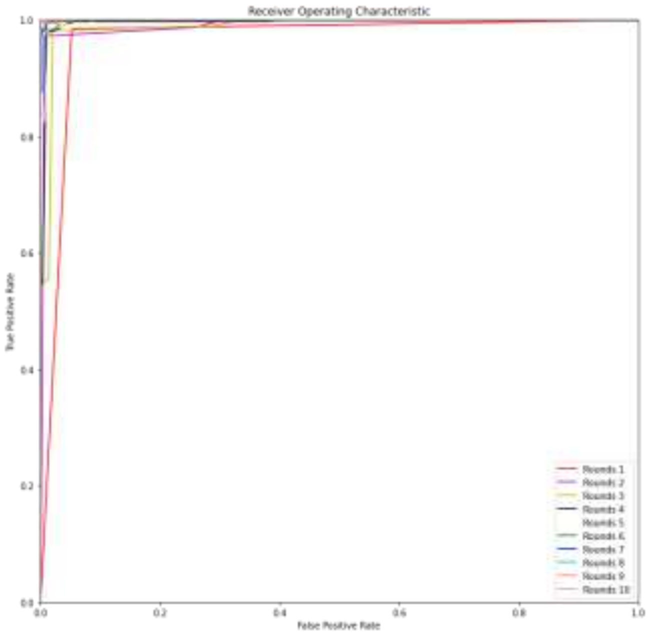


Fig. 19. ROC Curve Results

From the ROC curve shown in Figure 19, we can make observations on (1) how good the classifier is at each round and (2) what are the reasons the classifier gets better as the boosting round increases.

The reason the Classifier in each round is a good classifier can be found in the following:

- The ROC classifier curve has a larger Area under the curve (AUC) than the line y=x which is the line that passes through the diagonal of the graph. The diagonal line y=x represents a random classifier that randomly classifies images as faces and non-faces without any ability to distinguish between the 2 classes. We know that the better the classifier, the bigger the AUC. So, our classifier works better than random classifier.

- Since the ROC curve of the classifier is far from the diagonal line, we can say that we have a good classifier.
- We get a high True Positive Rate (TPR) with a low False Positive rate (FPR) for our classifier indicating that we have a good classifier.

The reasons that make the classifier better as the upgrade round increases, can be found in the following:

- The AUC of the ROC curve increases with increase in amplifier rotation indicating that we have a better classifier.
- We obtain a higher True Positive Rate (TPR) with a lower False Positive Rate (FPR) with increased amplifier rotation indicating that we have a better classifier.
- The ROC curve is further away from the diagonal line with each increasing round.
- From the graphs, we can observe that the ROC curves for Rounds 6, 7, 8, 9, 10 are almost indistinguishable from one another indicating that there is not much increase in classifier performance after a certain number of rounds. These observations can help us figure out the optimal number of boost rounds needed to train the classifier.

V. Conclusion

In this study, the Viola Jones method was successfully implemented to detect faces in an image using the Python programming language. The evaluation results of the test matrix provide the highest accuracy value of 99.4169% in 10 rounds with an input image resolution of 22x22. In addition to a fairly high level of accuracy, the lowest false negative and false positive values were obtained, respectively 0.2915%.

References

[1] R. E. Putri, T. Matulatan, and N. Hayaty, “Sistem Deteksi Wajah Pada Kamera Realtime dengan menggunakan Metode Viola Jones,” *J. Sustain. J. Has. Penelit. dan Ind. Terap.*, vol. 8, no. 1, pp. 30–37, 2019, doi: 10.31629/sustainable.v8i1.526.

[2] Y.-Q. Wang, “An Analysis of the Viola-Jones Face Detection Algorithm,” *Image Process. Line*, vol. 4, pp. 128–148, 2014, doi: 10.5201/ipol.2014.104.

[3] L. Shi and J. H. Lv, “Face detection system based on AdaBoost algorithm,” *Appl. Mech. Mater.*, vol. 380–384, no. 4, pp. 3917–3920, 2013, doi: 10.4028/www.scientific.net/AMM.380-384.3917.

[4] S. Yang, P. Luo, C. C. Loy, and X. Tang, “WIDER FACE: A face detection benchmark,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 5525–5533, 2016, doi: 10.1109/CVPR.2016.596.

[5] V. K and Dr.S.Padmavathi, “Facial Parts Detection Using Viola,” *Int. Conf. Adv. Comput. Commun. Syst. (ICACCS -2015)*, pp. 1–4, 2017.

[6] P. Viola and M. Jones, “Rapid Object Detection using a Boosted Cascade of Simple Features,” *Accept. Conf. Comput. Vis. PATTERN Recognit. 2001*, pp. 1–9, 2001, doi: 10.1109/CVPR.2001.990517.

[7] S. H. Park, J. M. Goo, and C. H. Jo, “Receiver operating characteristic (ROC) curve: Practical review for radiologists,” *Korean J. Radiol.*, vol. 5, no. 1, pp. 11–18, 2004, doi: 10.3348/kjr.2004.5.1.11.