iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 7 **Issue:** XI **Month of publication:** November 2019

DOI: http://doi.org/10.22214/ijraset.2019.11104

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.177
Volume 7 Issue XI, Nov 2019- Available at www.ijraset.com

Face Detection and Eye Detection in Image and Video using Pre-trained HAAR-Cascade Classifier

Mr. Tejas Phase[1], Prof. Dr. Suhas S. Patil[2]
[1, 2]*Dept. of Electronics Engineering, Karmaveer Bhaurao Patil College of Engineering, Satara, India*

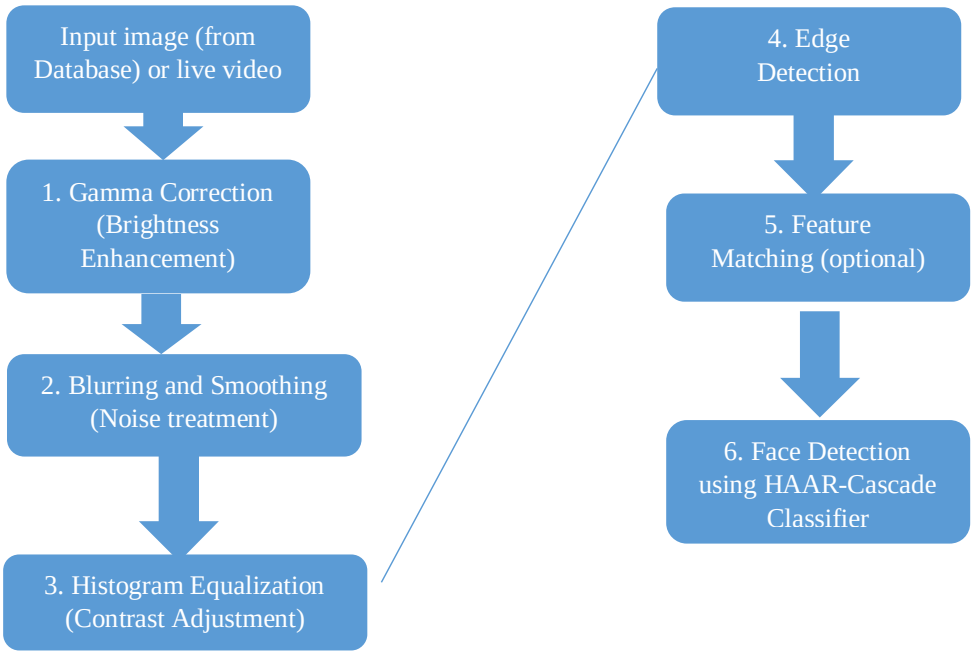
Abstract: ***This paper aims to detect the face in the given image and in live video. For this purpose, we are using pre-trained "Haar Cascade Classifier" to locate face/faces in the image and in live video. But before this, we are doing "Image Processing" for the purpose of Input Pre-processing. And then Image Enhancement to correct Non-uniform illuminations and then use pre-trained Cascade-Classifier to locate face/faces and eyes in the image and in live video.***
Keywords: Face-detection, Image processing, Image Filtering, Histogram Equalization, HAAR-Cascade Classifier.

I. INTRODUCTION

Human Face Detection is an active area of research include several disciplines such as Digital Image Processing, Image Enhancement, Edge detection and Computer Vision. Face detection is the first stage in various applications ranges from person's identification, video supervision, lips tracing, obtaining face expressions, gender categorization and modern human and machine interactions. For detection of the face, different face areas such as eyes, nose, mouth and chicks are detected as a feature components by using template matching technique and then combined them to identify face portion in the given input image [1, 2]. Face detection falls into a study of Object-Class detection in which the goal is to locate and identify all the objects in the given image. Face-detection technique focus on the frontal human faces. It is similar to image recognition system in which the image of an object or person is matched with reference image (in database) bit by bit. This matching of the image is done with reference images that stores in the image/object database. Any feature data change in the database will fails the matching process [3].

In this paper we demonstrate the detection of the face or faces in an image and in live video with different illumination variations and angle of the face. For this purpose, we are using pre-trained "Haar Cascade Classifier" to locate face/faces in the image and in video. But before detection, we perform "Image Processing" for the purpose of Input Pre-processing which include Noise Removal using "Median Filtering" and Contrast adjustment using Gamma correction". Then we perform Histogram Equalization to correct contrast adjustment to get the resultant image suitable for further image analysis. Then perform edge detection and then finally use the pre-trained Haar Cascade Classifier to detect or locate face/faces in the given image.

II. DESIGN FLOW



©IJRASET: All Rights are Reserved

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.177
Volume 7 Issue XI, Nov 2019- Available at www.ijraset.com

A. Gamma Correction

This method can be applied to an image to make it appear brighter or darker depending on the Gamma value chosen.

Fig. 1. Input Image for Gamma Correction

If we choose gamma value less than 1 then the resulting image is brighter and as we increase the gamma value the resulting image gets darker and darker.

```
#Set Gamma value above 1 and view the image:
gamma = 4
#Taking value of each pixel and raises the power with value of gamma.
Result_image = np.power(I,gamma)
disp_img(result_image)
```

Fig. 2. Output image with Gamma value above 1(less bright)

```
#Set Gamma value less than 1 and view the image:
gamma = 1/4
#Taking value of each pixel and raises the power with value of gamma.
Result_image = np.power(I,gamma)
disp_img(result_image)
```

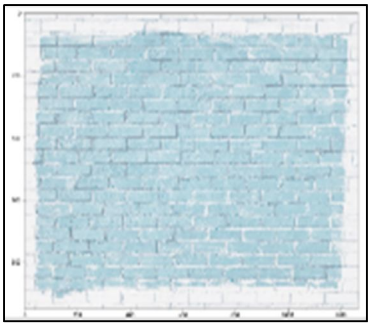


Fig. 3. Output image with Gamma value less than 1(more brighter)

B. Image Blurring (Treatment for Noise Removal)

1) Smoothing an image can help get rid of noise and help a computer vision application to focus on general details.

2) There are many methods of blurring and smoothing. Often blurring and Smoothing combined with edge detection.

3) Edge detection algorithms detect too many edges when shown a high resolution image without any blurring.

©IJRASET: All Rights are Reserved

a) Median Blurring

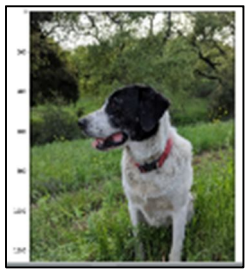


Fig. 4. Input image for inserting noise

b) Injection of Salt and Pepper noise and view the image

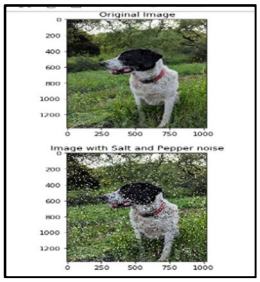


Fig. 5. Image after injecting salt & pepper noise

```
#Apply Median Filter to blur the image for removing the noise:
median_img = cv2.medianBlur(output_image,ksize=5)
plt.title("Image after removing noise")
plt.imshow(median_img)
```

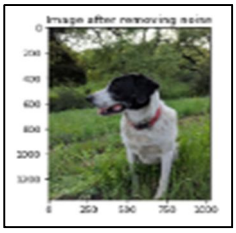


Fig. 6. Output Image after blurring (removed noise)

C. Histogram Equalization

Histogram Equalization is a method of contrast adjustment based on image's histogram.

1) Histogram Equalization on 'Gray-scale' Image

Fig. 7. Input Image for Histogram Equalization

```
#Calculate histogram values:
hist_values = cv2.calcHist ([gorilla], channels= [0], mask=None,histSize=[256],ranges=[0,256])
#Plot the histogram values:
plt.plot(hist_values)
plt.show ()
```

©IJRASET: All Rights are Reserved

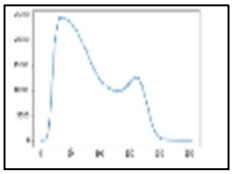


Fig. 8 Histogram of the Original Image

```
#Equalize the histogram:
equalize_gorilla = cv2.equalizeHist (gorilla)
#View the image:
display_img(equalize_gorilla,cmap='gray')
```

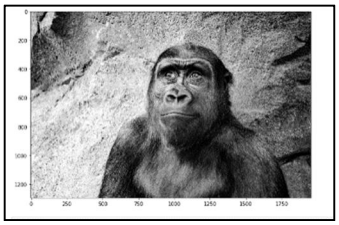


Fig. 9. Histogram Equalized Image (Enhanced Contrast)

```
# Calculate histogram values for equalized image:
eq_hist_values = cv2.calcHist ([equalize_gorilla], channels=[0],mask=None,histSize=[256],ranges=[0,256])
#Plot the values:
plt.plot(eq_hist_values)
```

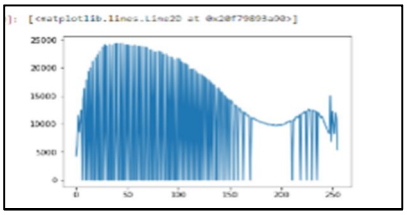


Fig. 10. Histogram of Equalized Image

2) Histogram Equalization on colour Image

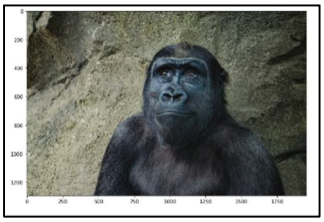


Fig. 11. Input Colour Image

```
#Convert the image into HSV version(Hue-Saturation-Value)
hsv_image = cv2.cvtColor (color_gorilla,cv2.COLOR_BGR2HSV)
#Get the value channel:
hsv_image[:,:,2] = cv2.equalizeHist(hsv_image[:,:,2])
#Convert the HSV image back to RGB image:
eq_color_gorilla = cv2.cvtColor (hsv_image,cv2.COLOR_HSV2RGB)
#Display the image:
display_img (eq_color_gorilla)
```

©IJRASET: All Rights are Reserved

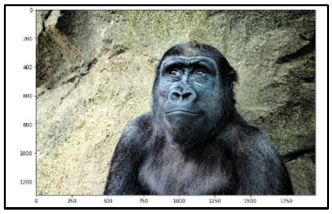


Fig. 12. Equalized colour Image

D. Edge Detection

We are going to perform Edge detection here using "Canny Edge Detector":

1) Canny Edge Detection Process

a) Apply Gaussian filter to smooth the image in order to remove the noise.
b) Find the intensity gradients of the image.
c) Apply non-maximum suppression to get rid of spurious response to edge detection.
d) Apply double threshold to determine potential edges.
e) Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.

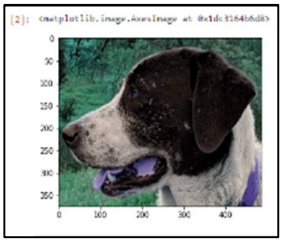


Fig. 13. Input Image for Edge detection

```
#Read the image:
#Let's create an object of canny edge detector:
edges = cv2.Canny(image=puppy_img,threshold1=127,threshold2=127)
#Let's view the edges:
plt.imshow(edges)
```

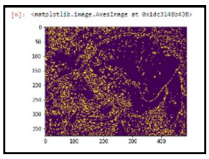


Fig. 14. Output of Edge Detection (Set threshold to value-127)

```
#Let's play around some threshold values:
edges = cv2.Canny(image=puppy_img,threshold1=0,threshold2=255)
plt.imshow(edges)
```

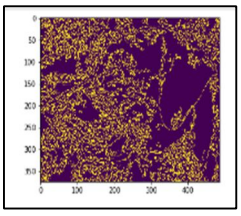


Fig. 15. Output of Edge Detection (Set threshold1=0 and threshold2= to value-255)

©IJRASET: All Rights are Reserved

But to play with this like a small child we can use "blurring of the image" instead first and then perform edge detection:

```
blurred_puppy_img = cv2.blur(puppy_img,ksize=(5,5))
# Perform edge detection again:
edges = cv2.Canny(image=blurred_puppy_img,threshold1=lower,threshold2=upper)
plt.imshow(edges)
```

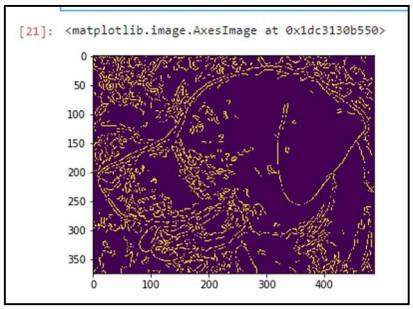


Fig. 17. Edge detection after blurred the input image

From the above output we can clearly see the strong edges of the dog's face. Let's increase the upper limit by some amount and check the result:

```
edges = cv2.Canny(image=blurred_puppy_img,threshold1=lower,threshold2=upper+50)
plt.imshow(edges)
```

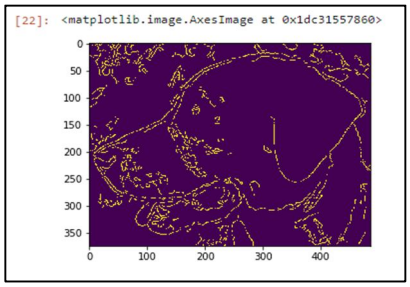


Fig.18. Edge detection after blurred the image as well as set upper threshold value to higher value

E. Face Detection using HAAR-Cascade

```
#DETECTION OF THE FACE
#Create an Object of Haar-Cascade Classifier:
cascade_class = cv2.CascadeClassifier ("C:/Users/Phase/Downloads/Computer-Vision-with-
Python/DATA/haarcascades/haarcascade_frontalface_default.xml")
#Let's write one function for image input and detect face:
def detect_face(img):
    #Create a copy of the original image:
        face_img = img.copy()
    #Call MultiScale method of classifier:
        face_rects = cascade_class.detectMultiScale(face_img)
    #Draw rectangle around detected face:
        for (x,y,w,h) in face_rects:
        cv2.rectangle(face_img,(x,y),(x+w,y+h),(255,255,255),10)
return face_img
#Let's run our function on any image:
result_img = detect_face(nadia)
#View the image:
plt.imshow (result_img,cmap='gray')
plt.show ()
```

©IJRASET: All Rights are Reserved

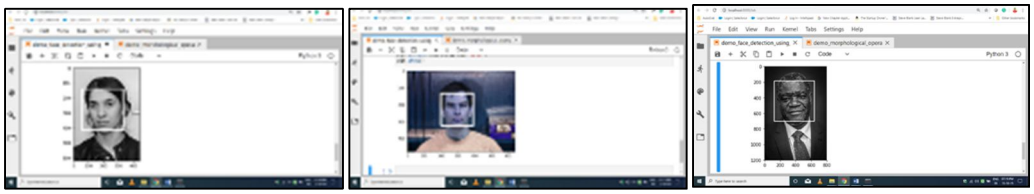


Fig. 19. Detection of the Face (in single face image)

1) Detecting Faces In The Group Image

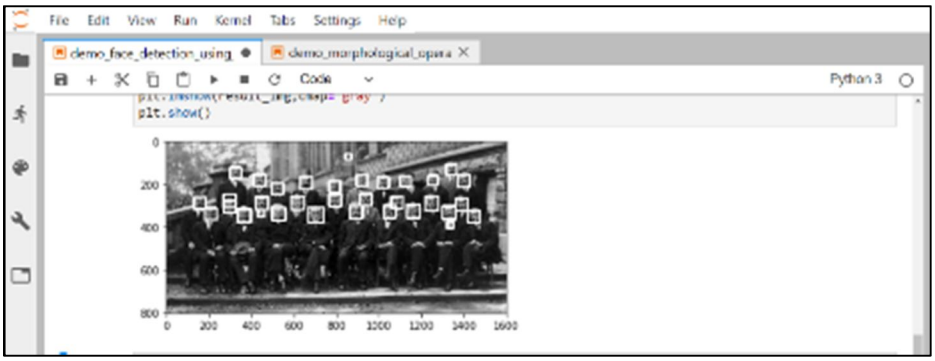


Fig. 20. Detection of the face in multiple faces

2) Detection Of The Face In Live Video Mode

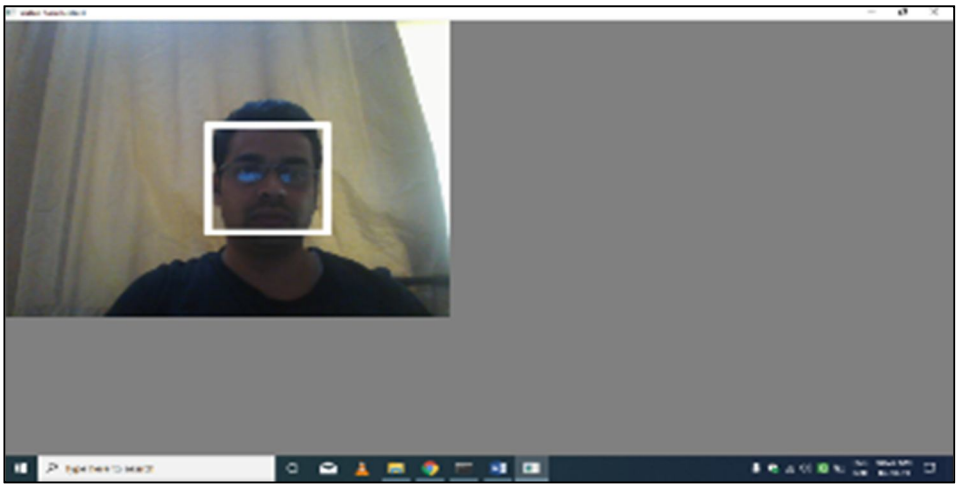


Fig. 21. Detection of the face in live video

3) Detection Of The Eyes In Live Video Mode

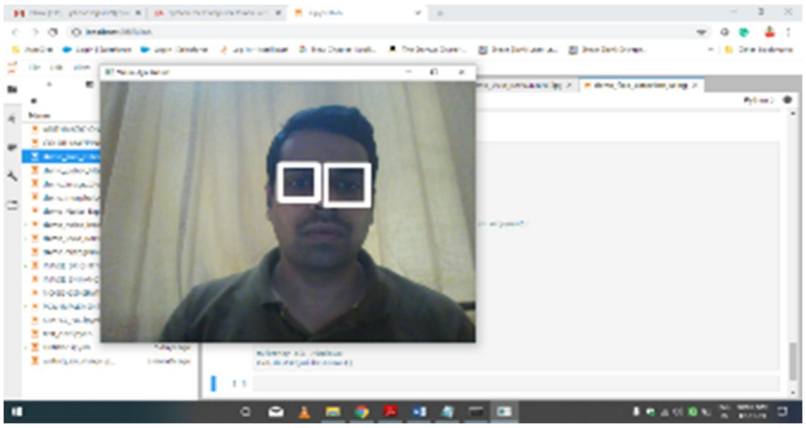


Fig. 22. Detection of the eyes in live video

©IJRASET: All Rights are Reserved

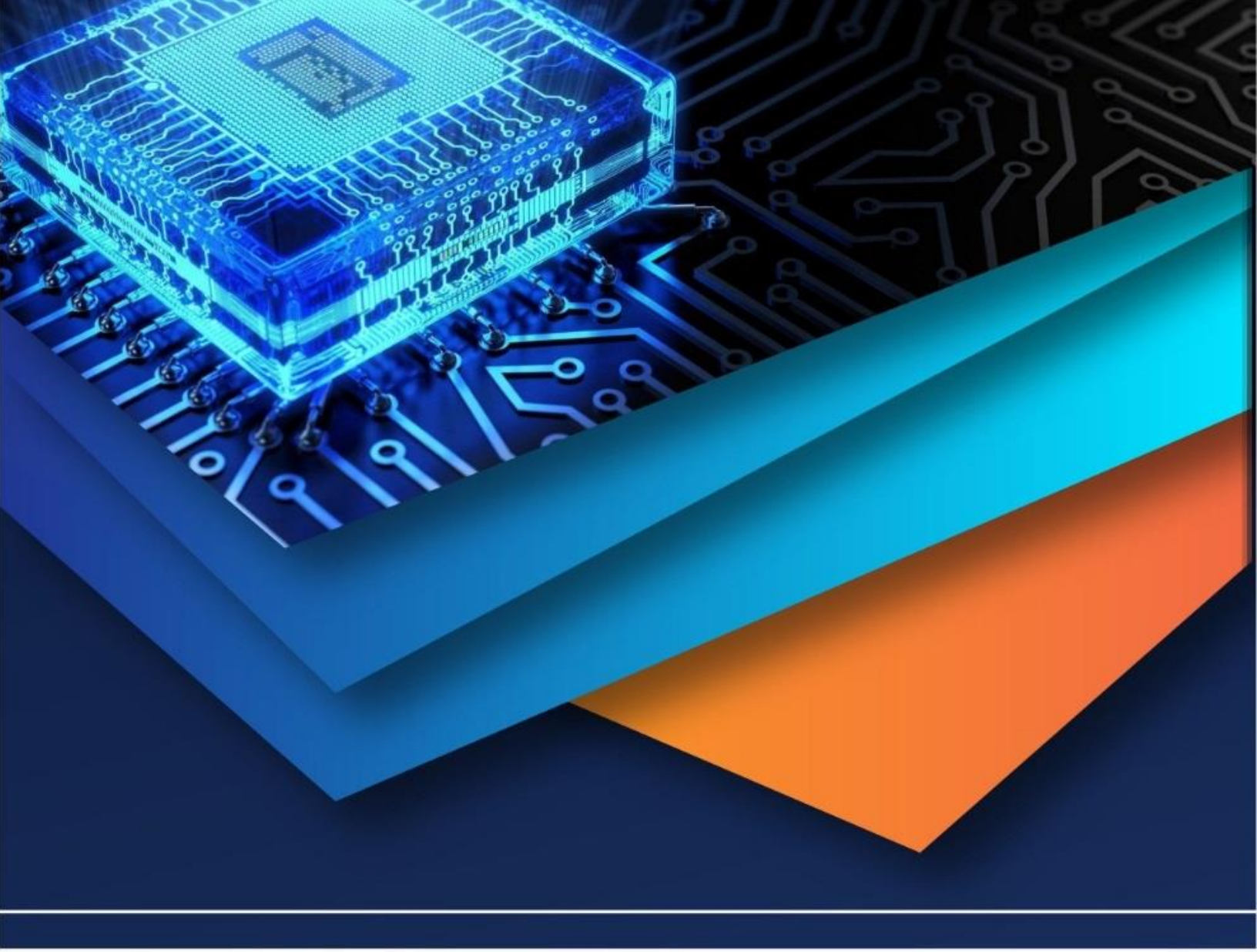
III. CONCLUSION

The downside of the face-detection using HAAR-Cascade classifier is that it requires large data sets to create own features. To minimize this problem we can use deep-learning approach to create our own classification algorithm for distinct group of images. As opposed to traditional Machine Learning algorithms which typically define sets of pre-programmed features and data and they work to extract these pre-defined features as a part of their pipeline while in deep learning the deep learning models tries to learn these features directly from the data as opposed to hand-engineered features by the human. In short, deep learning approach represents the hierarchical model which is capable of representing different levels of abstraction in the data. This emerging field has totally changed the research landscape of face recognition (FR) systems, which launched by the breakthroughs of Deep Face method.

REFERENCE

[1] Md. Iqbal Quraishi, J.P. Choudhury, Mallika De, "Image Recognition and Processing Using Artificial Neural Network", 1st Int'l Conf. on Recent Advances in Information and Technology, (2012) , 978-1-4577-0697-4/12.
[2] AMER AL-RAHAYFEH AND MIAD FAEZIPOUR, Member, IEEE, "Eye Tracking and Head Movement Detection: A State-of-Art Survey", IEEE Trans. In Transactional Engineering in Health and Medicine, VOLUME 1, pp. 2168-2372, 14 November 2013.
[3] Ming-Hsuan Yang, "Face Detection", University of California, Merced, CA 95344.
[4] Kurosh Madani, "Artificial Neural Networks Based Image Processing & Pattern Recognition: From Concept to Real-World Applications", Image, Signal and Intelligent Systems Laboratory (LISSI / EA 3956), (2008) 978-1-4244-3322-3/08.
[5] "Image noise", Wikipedia, May 28, 2019
[6] Meftah Ur Rahman, "A comparative study on face recognition techniques and neural network", Department of Computer Science, George Mason University.
[7] "Feature selection", Wikipedia, October 13, 2019.
[8] Mehul Ved, Jul 19(2018), "Feature Selection and Feature Extraction in Machine Learning: An Overview". Retrieved from: https://bit.ly/2NrHKww
[9] "Dimensionality reduction", Wikipedia, October 02, 2019.
[10] Jian Yang, David Zhang, Senior Member, IEEE, Alejandro F. Frangi, and Jing-yu Yang, "Two-Dimensional PCA: A New Approach to Appearance-Based Face Representation and Recognition", IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 26, NO. 1, JANUARY 2004.

©IJRASET: All Rights are Reserved



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)