Face Recognition with Python

Deepika Rawat*[1], Deepanshu*[2], Vishal Gupta*[3], Megha Attri*[4], Abhishek Upadhyay*[5]

[1,2,3,4,5]*Computer Science and Engineering Department HMR Institute of Technology and Management Delhi, India*

[1]*deepikarawat1221@gmail.com*

[2]*deepanshurohtela10000000000@gmail.com*

[3]*vishalgupta5083@gmail.com*

[4]*meghaattri1208@gmail.com*

[5]*abhisheku722@gmail.com*

***Abstract*—*This paper is based upon the fact that an enormous number of people are using Face Recognition to login into their mobile phones or laptops or in other areas for security reasons. Our aim here is to detect maximizing of the Face Recognition while minimizing the error.*

Keywords*—*Face Recognition, Face Detection, Haar Cascade Classifier, Computer Vision, OpenCV, Python

I. INTRODUCTION

Every time we upload a picture on Facebook, the platform uses facial recognition or detection algorithms to identify the crowd in that picture.

Or police officers around the world use face recognition technology to identify and take down criminals. The use of this computing platform is huge and businesses around the world are already making good use of it. Face recognition is a program(s) designed to detect a person in a photo or image (video).

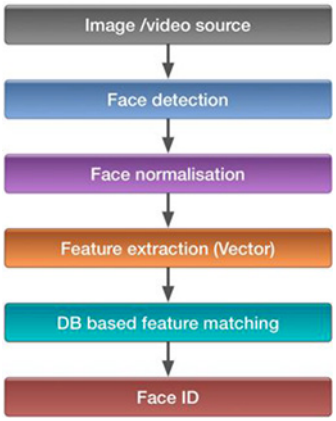
This technology has been in operation for more than 10 years, but its use has become noticeable, and accessible, over the past few years, as it now enables new solutions, such as personal photo applications and second verification of handheld devices. The most widely used algorithm for this is the Haar-like Face Detection Algorithm, and its main advantage is its speed of calculation. Due to the use of integral pictures, it can use any feature in constant time domain.

II. THEORETICAL ANALYSIS

Face recognition has recently gained significant notice as one of the most affluent uses of image or video analysis. It is due to availability of feasible technologies, including mobile phone solutions.

Past 10 years have provided significant improvement in this field due to proceedings in face modeling and analysis methods. Even though, software have been created for face detection and tracking, dependable face recognition still offers a humongous obstacle to computer vision and pattern recognition scientists.

III. BLOCK DIAGRAM*



A. Software Designing Requirements

IDE: Jupyter LAB

Language: Python3

Modules Used: cv2, keras, tensorflow, numpy

B. Hardware Designing Requirements

Recommended:

Processor: Intel Core i3

Modern Operating System: Windows, MacOS, Linux

4GB RAM

5GB free disk space

Internet Connection

Minimum:

Processor: Intel Pentium Gold

Modern Operating System: Windows, MacOS, Linux

2GB RAM

1GB free disk space

Internet Connection

C. Experimental Investigation

We tried and tested different projects and researches already done in the face-recognition domain and found that most of the models only reach the accuracy of 90 percent, which says that, the techniques and algorithms are due for improvements, which is why improving the techniques was fairly clear. We also found that in some devices dot-projectors are utilized for creating 3d model of the face of the person to identify, which is useful but unnecessary in the domain of face recognition.

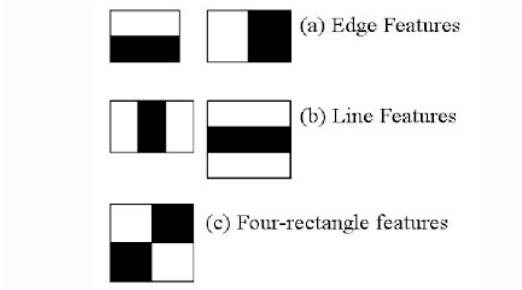
D. Baseline

IV. HAAR- LIKE FACE DETECTION ALGORITHM

1. Haar-like features are digital picture characteristics utilized for Object detection or Face Recognition
2. Haar-like characteristic suppose sad joining rectangle areas at a particular position in a detection box, adds at a Pixels' strength in each area and calculates the difference among these additions.
3. Haar-like feature employs 'Integral Image' which basically allows the features to be calculated rapidly and easy to use.
4. Let us understand how this algorithm works in steps below:

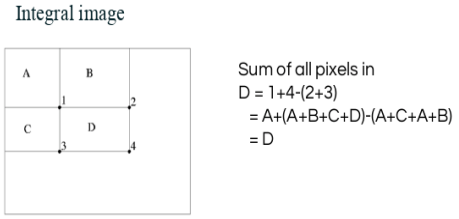
   - 'Haar Feature' Extraction

     First, the information (in the form of pictures) is put into the system, the classifier begins by extracting Haar Features from each and every Image. Haar Features are used to detect whether a suitable feature is present on an image or not. Example of Haar Features are-



   - 'Integral Images' Concept

     This algorithm uses a 24*24 base window size, in which 180,000 characteristics are being computed in this dialog box. The Integral image implies that to find the total of all pixels below any rectangle. All we need is to calculate the Integral Image using the 4 Corner Values.

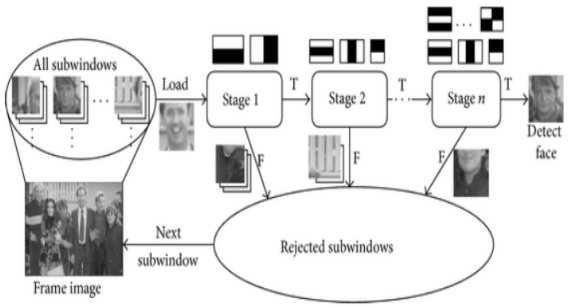


   - 'Adaboost': To Improve Classifier Accuracy

     As we pointed out above there will be above 180,000 characteristic values, so not all the Features are useful for detecting a face. To only sort out the Premium Features, an ML algorithm known as **Adaboost.** This reduces the number of features to around 6000 from around 180,000.

   - Using 'Cascade of Classifiers'

     This algorithm works faster by using a cascade of classifiers, in which each step consists of a powerful classifier. And also, it removes the need to apply all the characteristics at once in a dialog box. Rather, it amasses the features in separate sub-windows and classifiers.



V. K-MEANSCLUSTERINGALGORITHM

Picture segmentation is an issue of importance in today's world. In image processing, the subject of image extraction specifically, facial extract has many applications,

1. There are many facial image extraction methods; Among them, the agglomeration, the method hasn't gotten sufficient attention.
2. Collocation is an important method utilized in many fields of study such as: Facing mining and

discovering knowledge. In the assembly method, a group of objects is divided into subsets in this way.

3. Similar objects are placed inside a block.
4. So, anobject similar to the objects categorized in alike groupwhile they differ from those placed in other groupsIn terms of a pre-defined distance or a measure of similarity.
5. Picture Grouping is a specific grouping method in which objects are used, the photos are to be collaged.

- **What is K- Means Clustering Algorithm?**

The aim of information aggregation, also known as block analysis,detection of the standard assembly of a group of patterns and points, or things. Mass analysis is known as a statistic.

Classification policy used to determine if individuals of the population are divided into different groups through,

Make quantitative comparisons of the branching properties.

The goal is to develop the clustering algorithm that you will find regular groupings in unlabeled object data. Gathering or cluster analyzing is one method for identifying a group of objects.

In groups where all the objects in the block are considered

Being alike based on common features. Collocation is an UL based method for statistical data analyzing, which is used in many areas, not excluding data mining, Image analyzing, pattern recognizing, and image segmenting.

Table 1: The K-means Clustering Algorithm

No.	Description of Algorithm Programming
1	Choose k center of cluster randomly
2	Assign initial values for cluster means c1to ck
3	Repeat (Repeat the following steps until the cluster no longer changes)
4	for i=1 to n do
5	Assign each data point $x_i$to cluster $C_j$where $\|c_j-x_i\|$ is the minimum
6	end for
7	for j=1 to K do
8	Recalculate cluster mean $c_j$of cluster C_j
9	end for
10	until convergence
11	return C

V. VGG16 DEEP LEARNING ALGORITHM FOR IMAGE ANALYSIS

VGG16 (also called OxfordNet) is a convolutional neural network architecture named after the Visual Geometry Group from Oxford, who developed it. It was used to win the ILSVR (ImageNet) competition in 2014. To this day is it still considered to be an excellent vision model, although it has been somewhat outperformed by more recent advances such as Inception and ResNet.

First of all, let's start by defining the VGG16 model in Keras:

```
from Keras import applications

# Build the VGG16 network

model = applications.VGG16(include_top=False,

weights='imagenet')

# get the symbolic outputs of each "key" layer (we gave them unique names).

layer_dict = dict([(layer.name, layer) for layer in model.layers])
```

Note that we only go up to the last convolutional layer --we don't include fully-connected layers. The reason is that adding the fully connected layers forces you to use a fixed input size for the model (224x224, the original ImageNet format). By only keeping the convolutional modules, our model can be adapted to arbitrary input sizes.

The model loads a set of weights pre-trained on ImageNet.

Now let's define a loss function that will seek to maximize the activation of a specific filter (filter_index) in a specific layer (layer_name). We do this via a Keras backend function, which allows our code to run both on top of TensorFlow and Theano.

```
from keras import backend as K

layer_name = 'block5_conv3'

filter_index = 0 # can be any integer from 0 to 511, as there are 512 filters in that layer

# build a loss function that maximizes the activation

# of the nth filter of the layer considered

layer_output = layer_dict[layer_name].output

loss = K.mean(layer_output[:,:,:, filter_index])

# compute the gradient of the input picture wrt this loss

grads = K.gradients(loss, input_img)[0]

# normalization trick: we normalize the gradient

grads /= (K.sqrt(K.mean(K.square(grads))) + 1e-5)

# this function returns the loss and grads given the input picture

iterate = K.function([input_img], [loss, grads])
```

All very simple. The only trick here is to normalize the gradient of the pixels of the input image, which avoids

very small and very large gradients and ensures a smooth gradient ascent process.

Now we can use the Keras function we defined to do gradient ascent in the input space, with regard to our filter activation loss:

import numpy as np

we start from a gray image with some noise

input_img_data = np.random.random((1, 3, img_width, img_height)) * 20 + 128.

run gradient ascent for 20 steps

for i in range(20):

loss_value, grads_value = iterate([input_img_data])

input_img_data += grads_value * step

This operation takes a few seconds on CPU with TensorFlow.

We have used VGG 16 in one more instance of Face Recognition where this algorithm gives identical results to KNN but is performing sluggishly.

Advantage:

- Able to process enormous amounts of data for image analysis.

Disadvantage:

- Needs specialized hardware to perform optimally.

VI. VGG19DEEPLEARNINGALGORITHM FOR IMAGE ANALYSIS

VGG19 is a variant of VGG model which in short consists of 19 layers (16 convolution layers, 3 Fully connected layer, 5 MaxPool layers and 1 SoftMax layer). There are other variants of VGG like VGG11, VGG16 and others. VGG19 has 19.6 billion FLOPs.

Background

AlexNet came out in 2012 and it improved on the traditional Convolutional neural networks, So we can understand VGG as a successor of the AlexNet but it was created by a different group named as Visual Geometry Group at Oxford's and hence the name VGG, It carries and uses some ideas from its predecessors and improves on them and uses deep Convolutional neural layers to improve accuracy.

Let's explore what VGG19 is and compare it with some of other versions of the VGG architecture and also see some useful and practical applications of the VGG architecture.

Method	top-1 val. error (%)	top-5 val. error (%)	top-5 test error (%)
VGG (2 nets, multi-crop & dense eval.)	**23.7**	**6.8**	**6.8**
VGG (1 net, multi-crop & dense eval.)	24.4	7.1	7.0
VGG (ILSVRC submission, 7 nets, dense eval.)	24.7	7.5	7.3
GoogLeNet (Szegedy et al., 2014) (1 net)	-	7.9	
GoogLeNet (Szegedy et al., 2014) (7 nets)	-	**6.7**	
MSRA (He et al., 2014) (11 nets)	-	-	8.1
MSRA (He et al., 2014) (1 net)	27.9	9.1	9.1
Clarifai (Russakovsky et al., 2014) (multiple nets)	-	-	11.7
Clarifai (Russakovsky et al., 2014) (1 net)	-	-	12.5
Zeiler & Fergus (Zeiler & Fergus, 2013) (6 nets)	36.0	14.7	14.8
Zeiler & Fergus (Zeiler & Fergus, 2013) (1 net)	37.5	16.0	16.1
OverFeat (Sermanet et al., 2014) (7 nets)	34.0	13.2	13.6
OverFeat (Sermanet et al., 2014) (1 net)	35.7	14.2	-
Krizhevsky et al. (Krizhevsky et al., 2012) (5 nets)	38.1	16.4	16.4
Krizhevsky et al. (Krizhevsky et al., 2012) (1 net)	40.7	18.2	-

VGG 19 Algorithm basically works same way as the previous VGG16 Algorithm and processes the data in the same way except performance is a bit higher than the previous one.

A bit more info on how it works:

Arguments

include_top: whether to include the 3 fully-connected layers at the top of the network.

weights: one of None (random initialization), 'imagenet' (pre-training on ImageNet), or the path to the weights file to be loaded.

input_tensor: optional Keras tensor (i.e. output of layers. Input()) to use as image input for the model.

input_shape: optional shape tuple, only to be specified if include_top is False (otherwise the input shape has to be (224, 224, 3) (with channels_last data format) or (3, 224, 224) (with channels_first data format). It should have exactly 3 inputs channels, and width and height should be no smaller than 32. E.g. (200, 200, 3) would be one valid value.

pooling: Optional pooling mode for feature extraction when include_top is False.

None means that the output of the model will be the 4D tensor output of the last convolutional block.

avg means that global average pooling will be applied to the output of the last convolutional block, and thus the output of the model will be a 2D tensor.

max means that global max pooling will be applied.

classes: optional number of classes to classify images into, only to be specified if include_top is True, and if no weights argument is specified.

classifier_activation: A str or callable. The activation function to use on the "top" layer. Ignored unless include_top=True. Set classifier_activation=None to return the logits of the "top" layer. When loading pretrained weights, classifier_activation can only be None or "softmax".

Finally, we found that Accuracy on the data we used in KNN , VGG16 and VGG19 was 100 percent and the loss was 0 percent as the data used here is only meant

to recognize the person in front of it and the person needs to be registered with the system prior to that. So we compared the performance, In KNN using.npy array for storing upto 30000 frames of classified face data, we didn't need to train a model, hence we directly used the. npy array to recognize the face in front of the camera i.e. live video feed.

And in the other two programs we collected the data in similar fashion but used only 100 frames as it reduces the training time of the model significantly and gets us the model very soon for testing.

The performance was sluggish while reading and matching the model.h5.

VIII. THREATS AND BENEFITS OF FACIAL RECOGNITION

Benefits of Facial Recognition

1. This technology of face recognition isn'trestricted to taking down criminals only.
2. For instance, it can also seem easier to find lost children and senior citizen also with Facial Recognition.
3. Face recognition could make security checkpoints at airports less intrusive to passengers.
4. The Process of recognizing a face takes a less time and it is faster, this is incredibly beneficial for companies and business environments.
5. There are many security measures taken by financial institutions, still banking fraud is a recurring problem that keep coming again and again. Face Recognition Online could help to solve it.
6. Face Recognition software are adept at analysing crores of pictures and video(s) from numerous sources.

Limitations of Facial Recognition

1. Poor image quality or poor camera quality limits facial recognition's effectiveness.
2. Small image sizes make facial recognition more difficult and harder to use.
3. Different face angles or view angles can throw off facial recognition's reliability.
4. Facial recognition systems are capable of analyzing millions of images and videos from many sources.
5. Data Processing and storage can limit facial recognition tech and slow down the recognition process.
6. Change in faceexpression is a big limitation in face recognition.
7. Change of illumination conditions.
8. Obstructions (to a lesser extent) is a big problem in face recognition.

CONCLUSION

After training models with Haar-like face detection algorithm and KNN(K Nearest Neighbours) algorithm we got 100 percent accuracy, and the performance was not sluggish in comparison with other two DL algorithms which proves that this pair of algorithms is the best for the face recognition without requiring any other specialized hardware other than the integrated camera in devices.

FUTURE WORK

We would like to enhance this process with the help of Deep Learning in future and create models which will be able to train and decide on their own to classify the faces into Fraud and Genuine.

ACKNOWLEDGMENT

This paper has been possible by the unending support of Ms. Deepika Rawat, Assistant Professor, Department of Computer Science and Engineering, HMR Institute of Technology and Management, Delhi, India and guiding us in every possible way, while we were conducting this research.

REFERENCES

[1] Face Recognition: Issues, Methods and Alternative Applications By Waldemar Wójcik, Konrad Gromaszek and Muhtar Junisbekov Submitted: October 28th 2015Reviewed: March 9th 2016Published: July 6th 2016 DOI: 10.5772/62950

[2] D. Peleshko and K. Soroka, "Research of usage of Haar-like features and AdaBoost algorithm in Viola-Jones method of object detection," 2013 12th International Conference on the Experience of Designing and Application of CAD Systems in Microelectronics (CADSM), 2013, pp. 284-286.

[3] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman and A. Y. Wu, "An efficient k-means clustering algorithm: analysis and implementation," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 7, pp. 881-892, July 2002, doi: 10.1109/TPAMI.2002.1017616.