DIGITALIZED ATTENDANCE MARKING USING FACIAL RECOGNITION

T S Mastan Rao[1], C Rohit Kumar[2], U Vamshi Krishna[3], J Ravi[4]
[1]*Associate Professor, CSE, CMR Technical Campus, Hyderabad, India*
[2]*Student, CSE, CMR Technical Campus, Hyderabad, India*
[3]*Student, CSE, CMR Technical Campus, Hyderabad, India*
[4]*Student, CSE, CMR Technical Campus, Hyderabad, India*

***Abstract*: In this paper, a real-time ML-based system was built for Facial Recognition using images that have been captured with the help of a PC camera. The main purpose of this project is to design a model that can capture live feed from a computer web camera, analyse the data, detect and recognize the face. After the recognition is done, the data is then uploaded to the Fire store database (Cloud Database). The students and faculty can then view their status using an android application developed using Fire Store as backend. The novelty of our project is developing a full-stack application by integrating both machine learning and android development, thereby providing an interface to the users. Existing attendance systems use the file as storage devices and thus have a lot of complexities while handling data. Moreover, existing systems won't have the facility of cloud storage and an application interface to interact with. Our model is trained using pre-trained CNN architecture named VGG16. The reason we used VGG16 is it provides higher accuracy while dealing with facial recognition. Our model was trained using 300 images of each student captured using a web camera in different angles and light conditions, in order to increase our accuracy and also to detect and recognize faces in different conditions. Our model scored an accuracy of 96% for the training dataset.**

***Keywords*: Deep Learning VGG16 Architecture, Real Time, HaarCascade Facial Detection Module.**

1. Introduction

As we know each and every organization needs an attendance system for tracking their employees and acting accordingly. Attendance plays a major role in educational institutions.

We are well aware of attendance systems that already exist in our system. Usually, faculty calls out the names of the students and the students respond to the call, then the data is entered in a book or register. The data then needs to be analysed to calculate the percentage of student which makes the task tedious.

To overcome this problem, we developed a centralized model which is capable of both marking attendance as well as storing them in central database.

We achieved facial recognition using deep neural networks. Our model recognize face and then upload it to central database present over internet named Fire Store. Thus, it will reduce the problem of file system existing in prevailing system.

The ML model being discussed in this paper was trained using VGG16 network and tested using HaarCascade Facial recognition module. These are further discussed in detail in following sections.

2. Literature Survey

This section gives a summary on the main face recognition techniques that apply mostly to frontal faces, advantages and drawbacks of every method also are given. The methods considered are eigenfaces, neural networks, dynamic link architecture, hidden Markov model, geometrical feature matching, and template matching. The approaches are analysed in terms of the facial representations they used.

The first approach achieved in relation to facial recognition was by Bledsoe in the year 1964. He began working with Wolf to recognize the faces by using manual markings of various facial markings. Computers were used to find the distances between facial markings and the data was used to recognize the face

Performances of the face detection wasn't satisfactory until Viola and Jones proposed a piece. These Viola and Jones [1][2] are the first who are applying rectangular boxes for the face. But it's lot of drawbacks as its feature size was large. In a 24 × 24 image, the total number of Haar_like features is 160,000[3] and also it is not handled for wild faces and frontal faces.

After the introduction of Neural Networks, facial recognition development process accelerated with the use of Convolutional Neural Networks.

In our model, the dataset we needed to train was created by capturing images of students from the live feed of the PC Camera. In this process of training, we were able to achieve an accuracy of 92%. We also can increase the accuracy of our model by capturing images from a high-resolution camera. This also has its own disadvantages as they are expensive and not every organization can afford them.

The CNN architecture we adopted is VGG16 which was proven to be good in facial recognition applications.

Using CNN has certain limitations as we only need to give equal-sized images to train our model. The reason we chose CNN as our architecture is it yields better results in different light conditions and can also detect from different angles.

There also exist other alternative methods of facial recognition such as using the Facial_Recognition module of python. Every model has some limitations related to them.

The references are mentioned at the end of the paper.

3. Methodology

A. Dataset Creation

In this project, a real-time Facial Recognition ML model was built with the help of Neural Networks and HaarCascade Object Detection API, using real colouring images.

Initially, we have divided our project into five different phases. The first was to collect photos of students in order to create our dataset. The photos we needed for training are collected using a web camera. The labelling of our photos is done implicitly by our photo collecting code. The dimension we chose was 224*224*3. Our program lets the system take 300 snapshots of users from different angles and label them using numbers from 1 to 300. We have used the "HaarCascadeFrontalFace.xml" file to check whether the live feed has a human face or not. On detecting human faces program takes snapshots of the user.

Our program will capture 300 images of users and store them in our project folder. Later we manually divided our 300 images into two different folders namely "Train" and "Test" in the ratio of 1:0.4. Since we need to work with many faces we have to internally create folders with the names of students.

The below figure shows the process of capturing images using our image capturing program.

Fig. 1. Capturing Images Using Web Camera

The below figure shows hierarchy of our dataset developed using captured images.

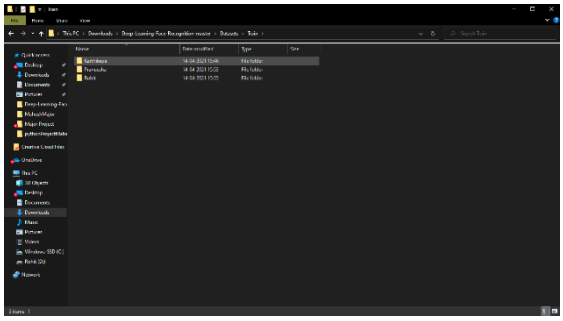

Fig. 2. Hierarchy of Dataset

Our final predicts the class numbers present in our dataset. So, we should not include duplicate folder names in our dataset. If present the trained model won't be able to recognize the correct class of the input image.

B. System Architecture

The second phase after completing dataset preparation is training our model and saving it.

The training of our ML project is done using TensorFlow and Keras. Since our model involves the recognition of image data, we used Convolutional Neural Networks. Convolutional Neural Networks is an extension to Neural Networks which was developed to deal with images. The CNN architecture we chose is VGG16.

Once the dataset is imported and pre-processing of data is completed, we trained our model using TensorFlow and Keras. To avoid the problem of training our dataset every time we want to use our model, we saved our trained model using Keras.

The below figure shows the architecture of our VGG16 network.

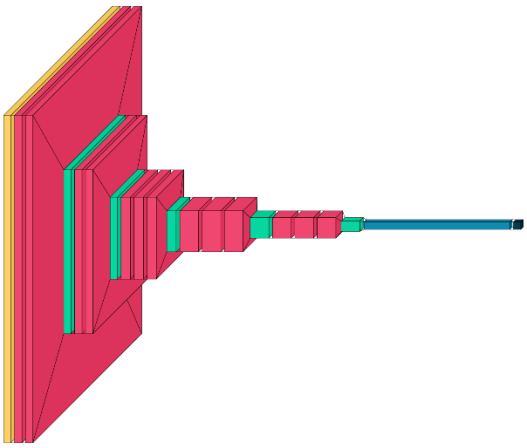

Fig. 3. VGG16 Network Architecture

The below figure depicts the system architecture of our ML project.

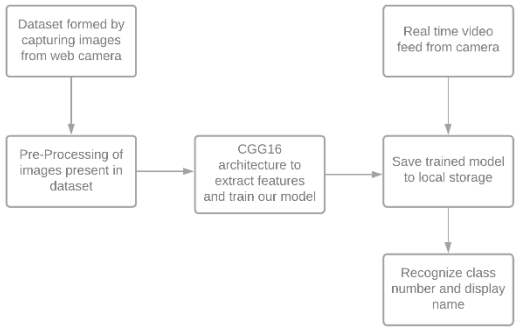

Fig. 4. System Architecture

In order to make our model recognize the faces present in our dataset, we need to first train our model. We use the TensorFlow module to train.

TensorFlow is an open-source library for numerical computation and large-scale machine learning that ease Google Brain TensorFlow, the method of acquiring data, training models, serving predictions, and refining future results.

TensorFlow bundles together Machine Learning and Deep Learning models and algorithms. It uses Python as a convenient front-end and runs it efficiently in optimized C++. TensorFlow allows developers to make a graph of computations to perform. Each node within the graph represents a mathematical process and every connection represents data. Hence, rather than handling low details like deciding proper ways to hitch the output of 1 function to the input of another, the developer can specialize in the overall logic of the application.

A loss function is used to optimize the machine learning algorithm. The loss is calculated on training and testing, and its interpretation is based on how well the model is doing in these two sets. It is the sum of errors made for each example in training or testing sets. Loss value implies how poorly or well a model behaves after each iteration of optimization.

The loss of our model is shown in the below figure.

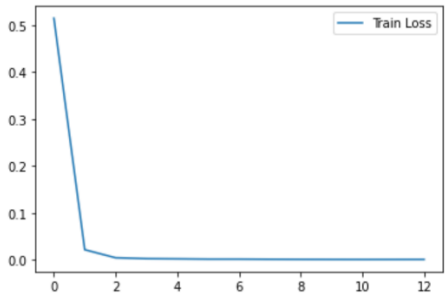

Fig. 5. Loss Graph of our model

The loss function we used is "Categorical_Crossentropy". This loss function is used when there are more than two label classes in our dataset. The formula related to it is as follows

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = -\frac{1}{N}\sum_{i}^{N}\left[y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)\right]$$

The below figure shows the loss after each epoch in our training process.

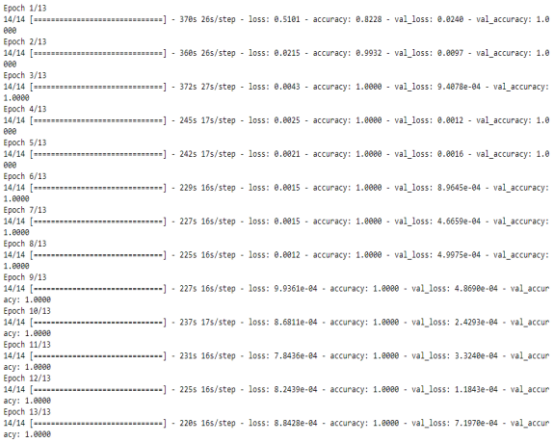

Fig. 6. Loss after each epoch of our model

Since our model is a predictive algorithm as it predicts the names of human faces, we need to evaluate the model before using. Evaluation is nothing but checking the extent up to which we can use our developed model. There are many evaluation criteria present in Keras. We chose our evaluation metric as 'Accuracy'.

The reason we chose accuracy is that it will give accurate results when all the classes in our dataset has same number of images. Since our dataset contains equal number of images in each and every class, we proceeded using this as our evaluation metric. The accuracy graph of our model is as shown in below figure.

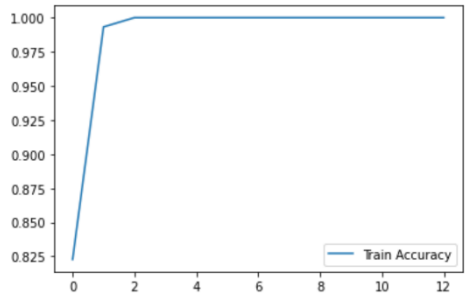

Fig. 7. Accuracy Graph of our model

The mathematical formula behind 'Accuracy metric' is as follows.

$$\text{Accuracy} = \frac{(TP + TN)}{(TP + FP + TN + FN)}$$

The activation functions we used in our hidden layers is 'ReLU'. ReLU stands for rectified linear unit, and is a type of activation function. Mathematically, it is defined as $y = max\ (0, x)$. Visually, it looks like the following:

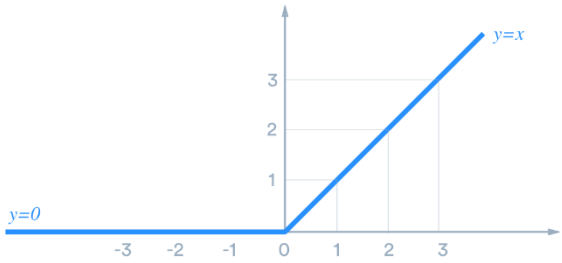

Fig. 8. ReLU Activation Function Graph

The activation function we used in our last layer i.e.; Dense layer is 'SoftMax'. SoftMax activation function is used when there are more than two classes present in our dataset or when our model should predict more than two class labels. The mathematical formula related to 'SoftMax' is as follows.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

C. Facial Recognition

The third phase after training our model is recognizing the faces using real time video feed from web camera.

Facial recognition from real time data is done by importing the trained model using Keras. We need to access the web camera of our laptop/computer with the use of OpenCV. The live video feed captured using OpenCV module is sent for pre-processing frame by frame. Our system web camera is capable of recording video at a rate of 30 frames per second. So, our algorithm sends around 30 frames a second for pre-processing. Usually, the video extracted from web camera is of size 512*512 with 3 channels. Initially while training our model, we set the size of our image to 224*224. So, we need to resize the raw feed in order to sent it to recognition algorithm.

After pre-processing of frame is completed, we need to check whether the frame contains a human face or not. To do this we need to import "HaarCascadeFrontalFace.xml" file which contains the details regarding the facial markings of human face.

The below figure shows the details of "HaarCascadeFrontalFace.xml" file.


```
<cascade type_id="opencv-cascade-classifier"><stageType>BOOST</stageType>
  <featureType>HAAR</featureType>
  <height>24</height>
  <width>24</width>
  <stageParams>
    <maxWeakCount>211</maxWeakCount></stageParams>
  <featureParams>
    <maxCatCount>0</maxCatCount></featureParams>
  <stageNum>25</stageNum>
  <stages>
    <_>
      <maxWeakCount>9</maxWeakCount>
      <stageThreshold>-5.0425500869750977e+00</stageThreshold>
      <weakClassifiers>
        <_>
          <internalNodes>
            0 -1 0 -3.1511999666690826e-02</internalNodes>
          <leafValues>
            2.0875380039215088e+00 -2.2172100543975830e+00</leafValues></_>
        <_>
          <internalNodes>
            0 -1 1 1.2396000325679779e-02</internalNodes>
          <leafValues>
            -1.8633940219879150e+00 1.3272049427032471e+00</leafValues></_>
        <_>
          <internalNodes>
            0 -1 2 2.1927999332547188e-02</internalNodes>
          <leafValues>
            -1.5105249881744385e+00 1.0625729560852051e+00</leafValues></_>
        <_>
          <internalNodes>
            0 -1 3 5.7529998011887074e-03</internalNodes>
          <leafValues>
            -8.7463897466659546e-01 1.1760339736938477e+00</leafValues></_>
        <_>
          <internalNodes>
            0 -1 4 1.5014000236988068e-02</internalNodes>
          <leafValues>
            -7.7945697307586670e-01 1.2608419656753540e+00</leafValues></_>
        <_>
          <internalNodes>
            0 -1 5 9.9371001124382019e-02</internalNodes>
          <leafValues>
            5.5751299858093262e-01 -1.8743000030517578e+00</leafValues></_>
        <_>
          <internalNodes>
            0 -1 6 2.7340000960975885e-03</internalNodes>
          <leafValues>
            -1.6911929845809937e+00 4.4009700417518616e-01</leafValues></_>
        <_>
```


Fig. 9. HaarCascadeFrontalFace.xml file

To check whether the frame has human face or not we have to cross reference our frame with this XML file. It returns the cropped markings of face from frame if human face exists in it, else it returns none.

If our recognition algorithm receives none, it displays none on the output screen. If it receives a cropped human face, it is first converted into an NumPy array. The reason for converting it into NumPy array is Keras only allows detection of images using NumPy array. As we have already imported trained model as 'h5' file we just let our model to predict the class number. Depending on the value it predicts the name associated with it is displayed on the screen.

D. Cloud Data Storage

The fourth phase in our model is uploading the data to a cloud database. The reason we chose cloud database is that it can be accessed from anywhere and is free of cost up to certain limit.

The cloud database we chose is Fire Store provided by google. Fire Store is a popular cloud database while integrating different platforms. As soon as our model recognizes a student, his attendance is then uploaded to fire store. Uploading of data is done using a module named 'firebase_admin'.

The below figure shows the schema of our cloud database.

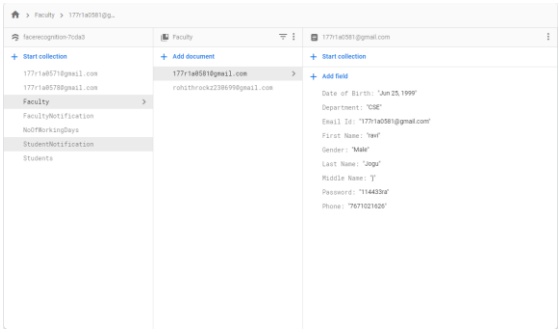

Fig. 10. Cloud Database Schema

Fire Store database provides an easy and efficient way of retrieving data to an android application. Firebase also provides different authentication methods which we can use in our application.

E. Android Application

The last phase in our model is developing an android application. The reason we chose android application as our interface is that it is possible for every student to check their status.

We used cloud database (Fire Store) as our backend for building our application. The details in our cloud database are fetched and displayed to the user. Moreover, our application also provide facility for faculty to easily check the attendance percentage of students just with their application.

Thus, a full stack application was developed by integrating Machine Learning and android development to ease the process of attendance.

The below figures show the activities that can be performed by student using our application.

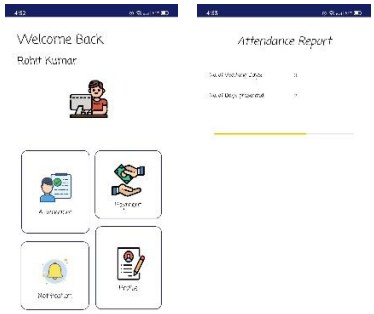

Fig. 11. Student Activities in our Application

The below figures show the activities performed by faculty in our application.

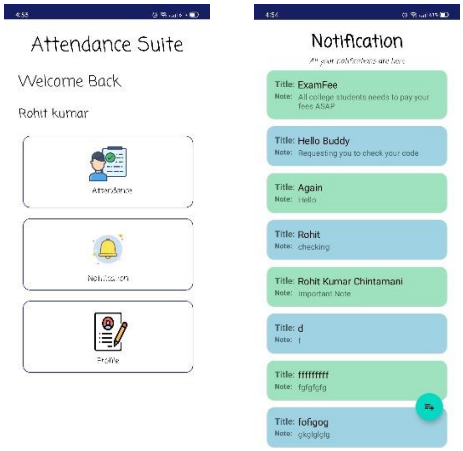

Fig. 12. Faculty Activities in our application

4. Results and Discussions

A real-time Facial Recognition using Neural Networks named VGG16 is introduced. In this paper, facial recognition is done to help ease the process of attendance marking and also integrating ML model with application thus creating a full stack application. This system showed good results in recognizing faces as well as checking their status in our application. With the help of Neural Networks, in particular Convolutional Neural Networks we were able to develop a hybrid model and thereby easing the work of attendance marking.

Neural Networks results in more accurate recognition predictions compared to other methods of facial recognition. I strongly believe neural networks as an ideal way to solve this problem.

Below figure shows the execution of our facial recognition model.

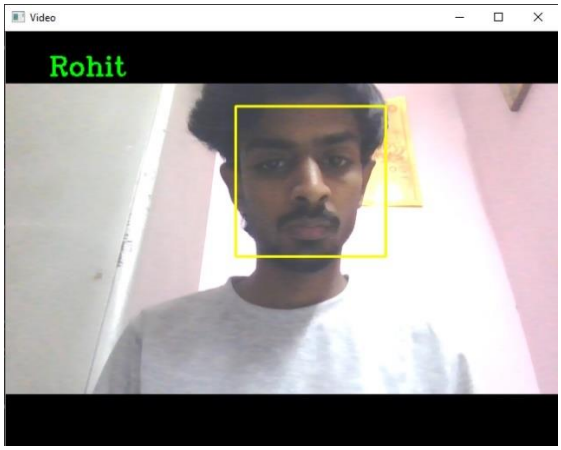

Fig. 13. Facial Recognition of 'Rohit'

As soon as the recognition of face is done, the data will be uploaded to Fire Store and the user will get an alert saying 'Your attendance is marked'. The user can instantly check his status using their mobile application.

5. Conclusion

In this paper, a real-time ML based Facial Recognition system was built using coloured images from PC camera. The use of Convolutional Neural Networks helps our model to recognize in times of dark lightening conditions.

The system achieved a maximal accuracy of about 96% for training and 97% for the validation set. Our model will also recognize faces that aren't trained by our model.

I believe our full stack model will ease the burden of faculty in maintaining attendance and provide easy accessibility to the end users. This way we were able to extend our ML model to a full stack application.

6. References

[1] Nirmalya Kar, Mrinal Kanti Debbarma, Ashim Saha, and Dwijen Rudra Pal," Study of Implementing Automated Attendance System Using Face Recognition Technique",in proc. International Journal of Computer and Communication Engineering, Vol. 1, No. 2, July 2012.

[2] Ajinkya Patil and Mrudang Shukla," Implementation of Classroom Attendance System Based on Face Recognition in Class", in proc . International Journal of Advances in Engineering & Technology, July, 2014.

[3] Jomon Joseph and K. P. Zacharia, "Automatic Attendance Management System Using Face Recognition", in proc. International Journal of Science and Research (IJSR). [4] D. Mart, Sign Language Translator Using Microsoft Kinect XBOX 360 TM, 2012, pp. 1-76.

[5] Rohit Chavan,Baburao phad,Sankalp Sawant and Vinayak Futak,"Attendance Management System using face recognition",in International journal for innovative Research in science and technology,Vol. 1,issue 11,April 2015.

[6] Naveed Khan Balcoh, M. Haroon Yousaf,Waqar Ahmad and M. Iraam Balg,"Algorithm for efficient attendance management:face recognition based approach",in IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 4, No 1, July 2012.

[7] R. Tharanga, Samarakoon, Karunarathne, Liyanage, and D. Parer, "Smart Attendance using real time face recognition (smart-fr)," Semantic Scholar, 2013. 8. K. Selvi, P.Chitrakala, and A. Jenitha, "Face recognition based attendance marking system," IJCSMC, no. 3, p. 337 –342, 2014. 9. S. Chintalapati and M. Raghunad, "Automated attendance management system based on face recognition algorithms," International Conference on Computational Intelligence and Computing Research, 2013

[8] A.F. Abate, M. Nappi, D. Riccio, and G. Sabatino, "2D and 3D face recognition: A survey", Pattern Recognition Letters,vol.28, issue 15, pp.1885-1906, Oct 2007.

[9] F. Ibikunle, Agbetuvi F. and Ukpere G. "Face Recognition Using Line Edge Mapping Approach." American Journal of Electrical and Electronic Engineering 1.3(2013): 52-59

[10] K. Wong, H. Law, and P. Tsang, "A System for Recognising Human Faces," Proc. ICASSP, pp. 1,6381,642, 1989.