

FACE RECOGNITION

Devi Abhigna 160118737002¹, SLS Harshitha 160118737008²

^{1,2}B.E., Department of Information Technology, CBIT, Hyderabad

Under the guidance of Smt.T.Prathima, Assistant Professor, Department of IT, CBIT, Hyderabad

ABSTRACT

Background

The technology around us keeps advancing every minute and every second. There are many examples where things that were once done manually are now done automatically all thanks to technology. New technologies like Artificial Intelligence, Machine Learning, Internet Of Things, Virtual Reality, Robotics and more are almost ruling the world with new innovations. Face Recognition of Machine Learning is one of those topics that is always improving. Face recognition can be done in many ways, like using CNN(Convolutional Neural Networks) or any other Deep Learning Techniques or using Opencv. The use and importance of face recognition is increasing from day to day and the areas of its use is also increasing widely. Face recognition is used in Unlocking phones, Smarter Advertising, Help the blind, Aid Forensic Investigations and also to Protect Law Enforcement. The evolution of Face recognition Started way back in 1960's and now there are a number of ways to perform Face recognition.

Design

The main objective of this project is to recognize the faces with high accuracy. Data pertaining to the recognition of faces was collected from our friends. We collected 50 photos of 10 people each and made them into a dataset. The photos we collected are at different distances, different angles, both in daylight, dark light and different saturation levels with varying resolution starting from less than 1 megapixel to 13 megapixel. Each photo is a 2D-RGB image, which are collected from mobile phone, digital camera and social media. The data collected is arranged into three different folders test, train and validation. We used 80% of the data for training the model, the remaining 10% was used for testing and the other 10% for validation. The most difficult thing in face recognition is the recognition of faces in different face orientations such as lighting, background, haircut, mustache and beard, head cover, glasses, and differences of expression, e.g. smiling, laughing, being angry, and being sad. The other obstacle in face recognition is the matching imagery. The imagery used is not an image originated from a single device (i.e. mobile phone device and professional digital camera). We defined pre-processing transformations on raw images of training data and generated slightly twisted versions of the original image on the idea that the model can learn on good and bad mix of images. However no transformations are done on testing images. We gave class labels for each face, class indices have the numeric tag for each face. For our model the number of neurons is equal to the number of faces. We made use of the Sequential model. Sequential model API is a way to build deep learning models in which sequential classes and model layers are created and added. The input to a convolutional neural network is an $(n \times m)$

x 3) for colored images, where the number 3 represents the red, green, and blue components of each pixel in the image.

For this model, we first create a 2D convolutional layer with 32 filters of 5 x 5 kernels and a Rectified Linear Unit(ReLU) activation. In the following layers, we perform batch normalization which is used to scale data by a certain factor and pooling we use maximum pooling with a pooling size of two. Next, two blocks of 2D Convolutional layer are created with 64 filters and ReLU activation followed by a pooling layer. Finally we added a dense layer with 64 filters followed by a layer of ReLU activation and pooling. Then we flatten the output from these layers so the data can proceed to fully connected layers. Flatten is used to convert data into a 1-Dimensional form. Finally, we use the softmax activation function to convert the outputs into probability values.

An optimizer is an algorithm or methodology used to reduce losses by modifying the weights and learning rate of a neural network. Optimizers train models faster and work more efficiently. We initialize our optimizer with the learning rate before we start training our model.

As we have a multi class classification problem, we use the Adam optimization technique because it always leads to a smoother way than other optimization techniques. Adam is an optimization algorithm that uses adaptive moment estimation to generate more efficient neural network weights. We determine the loss by using categorical cross entropy. Categorical cross entropy can be used for integer targets instead of categorical vectors.

We train our model over 60 epochs. A higher number of training epochs increases its accuracy along with lowering the loss.

Software Requirements

Software requirements required for this project are Google colab or jupyter notebook, Python libraries, Keras models, Keras layers, Keras Preprocessing, numpy etc.

KeyWords

ImageDataGenerator, categorical, Sequential, Maxpooling2D, Convolution2D, Dense, Fattening, Activation etc.

1. INTRODUCTION

Objective

The main objective of this project is to recognize faces from testing dataset images that are randomly selected after the Neural network is trained from training dataset. The model should be able to recognize the face most accurately even if the image of the person is blurry, or with extra filters, or even if the image of the person is a little old, or if the image is at different angles. The model is developed using CNN. The image processing technique using keras is performed on a training dataset to process the image and obtain pixel values. Since we are working with coloured images each pixel has three different values called RGB(Red Blue Green) values as a combination RGB can produce all possible colour pallets. Various layers are used as part of the Neural Network to train the model to produce accurate predictions.

Scope

The main aim of this project is to identify faces with high accuracy so that we can extend it to the college attendance system where faces are captured with a live camera and the model is connected to the student and attendance is given to them. This saves a lot of time when compared to manually collecting the attendance. So in face recognition there are a lot of problems like lighting, angles, distance and many more that are faced during the creation of any face recognition model. Our attempt is to decrease these effects on the output of the final recognition and to increase the accuracy of the model.

Motivation

Emerging technologies and their applications in the real world are the main source of motivation for this project. As the number of technologies increase the number of challenges also increase and to overcome these challenges is the main aim which leads to successfully creating any model. Faculty face major issues such as increase in the student strength in a class, proxy attendance, handling late comers, shortage of time for syllabus coverage. In an automated system with high accuracy, efforts for taking the attendance will be greatly reduced and there is a possibility of preventing the proxy attendance.

Overview

Face recognition is a recognising model that is developed using CNN and keras to recognize the faces of people that are stored in the dataset. There are two sets, one of them is training dataset and the other one is testing dataset. The images in the training dataset are used for training the model for it to be able to detect the face from the testing dataset. The images are preprocessed in the model by adjusting their pixel sizes to make all the images of the same size and shape for the model to start training. The images are assigned to specific labels by the model to use them later in the recognition. The model is trained with Neural Networks and also layers are applied for it to recognize the face from the testing dataset, where the images of people in the dataset are randomly selected. After adding a convolutional layer, additional hidden layers are added to increase the accuracy of the model and then there is Flattening process and addition of Dense layer. After all

this training, the model then predicts its accuracy through epoch values. Then the model is ready to test where the images from the testing dataset are given to the model and the model recognizes the faces and displays the name of that particular person.

2. LITERATURE SURVEY

1. Face Recognition through Geometrical Features by R. Brunelli and T. Poggio, published in 1992. The Dataset they used for the comparison of the different strategies is composed of 188 images, four for each of 47 people. To make correlation more robust against illumination gradients, each image was preprocessed by dividing each pixel by the average intensity on a suitably large neighborhood. The Modelling approach they used is Face Recognition through Geometrical such as nose width and length, mouth position and chin shape. The Training set and Testing set ratio is 4 photos in total, 2 to train and 2 to test.
2. Face Recognition using Principal Component Analysis and Neural Networks by A. S. Syed Navaz , T. Dhevi Sri & Pratap Mazumder, published in 2013. The dataset they used are photos of few people taken in normal daylight and the preprocessing done is the main features of the face are extracted and eigenvectors are formed. The two methods of approach used are Principal component analysis(PCA) and Neural networks(ANN).
3. LBPH-based Enhanced Real-Time Face Recognition published in 2019. The dataset contains a total 1000 images, 333 face images of each person with 60×60 resolution of each image. Preprocessing is done on the images. Modelling approach they used is a facial recognition system based on the Local Binary Pattern Histogram (LBPH).
4. Deep Face Recognition by Omkar M. Parkhi, Andrea Vedaldi and Andrew Zisserman from Visual Geometry Group, Department of Engineering Science, University of Oxford. The first dataset is Labeled Faces in the Wild dataset (LFW) . It contains 13,233 images with 5,749 identities, and is the standard benchmark for automatic face verification. The second dataset is YouTube Faces (YTF). It contains 3,425 videos of 1,595 people collected from YouTube, with an average of 2 videos per identity, and is a standard benchmark for face verification in video. For both of them standard evaluation protocol defined for the “unrestricted setting” is followed. The implementation is based on the MATLAB toolbox

MatConvNet which is linked against the NVIDIA CuDNN libraries to accelerate training. The model is trained using CNN.

5. Discriminative Covariance Oriented Representation Learning for Face Recognition with Image Sets by Wen Wang, Ruiping Wang, Shiguang Shan and Xilin Chen. The YouTube Celebrities dataset contains 1,910 YouTube videos of 47 subjects. The YouTube Face dataset contains 3,425 videos of 1,595 subjects. Implementation is based on Discriminative Covariance oriented Representation Learning. Their method extracts and organizes the discriminative information implicit in the images and image sets jointly which positively facilitates the image set classification. Efficient representation automatically due to deep learning.

3. METHODOLOGY

So in our proposed methodology, face recognition is achieved using Deep Learning's subfield that is Convolutional Neural Network (CNN). A Convolutional neural network is a neural network that has one or more convolutional layers and is used mainly for image processing, classification, segmentation and also for other auto correlated data. A convolution is essentially sliding a filter over the input. Training the Neural Networks properly to recognize the face of the person correctly with good accuracy is what that is aimed for.

The stages of the proposed system are - Pre-processing the images, Generating datasets and Class indices, Mapping of Faces to their ID's, Training the model(Convolution, Max Pooling, Flattening, Fully Connected Neural Network and Compiling CNN) and finally testing the model.

Pre-processing The Images

Pre-processing of images is important as this process helps in the uniformity of images which further helps in training. Since we are considering different varieties of pictures like, pictures with different lightings, angles, filters and many more, pre-processing really helps to maintain the unity of pictures for the training. The hyper parameters present helps to generate slightly twisted versions of the original image, which leads to a better model, since it learns on the good and bad mix of images. These transformations are only done on a training set and not on a testing dataset.

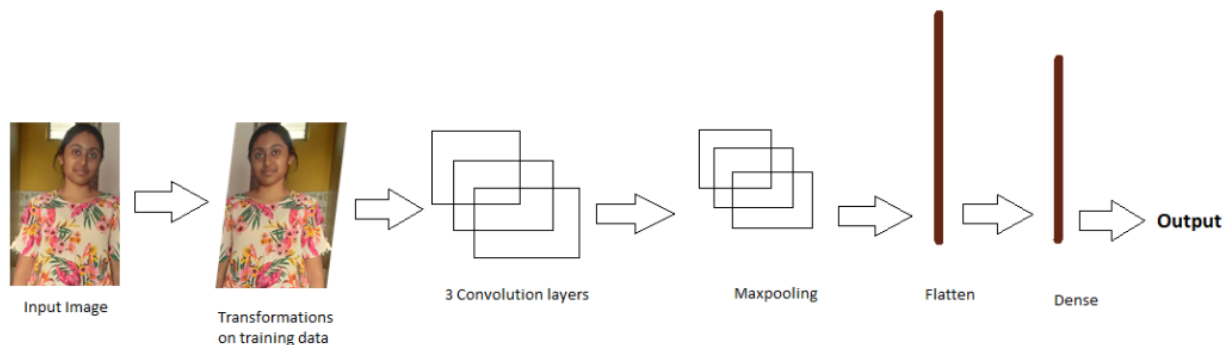
Generating Datasets and Class Indices

Generating the training and testing datasets. During the generation we give parameters like the location of the images that are stored in the folders, the target sizes and the batch size. We also give the class mode where we specify whether it is binary or categorical. Since we are using many images in many classes we defined categorical. We do this process for both the training dataset and testing dataset. After the generation of datasets are done then we print class labels for each face. Each face in the dataset is labeled with a unique number.

Mapping of Faces to their ID's

All the class indices have the numeric tag for each and every face in the dataset. So we store the face and the numeric tag for future references and to do this we map them and store them. The model will give an answer as a numeric tag. This mapping will help to get the corresponding face name for it. The number of neurons for the output layer are the number of faces present. Mapping is really helpful while recognising the faces in the testing phase.

Training The Model

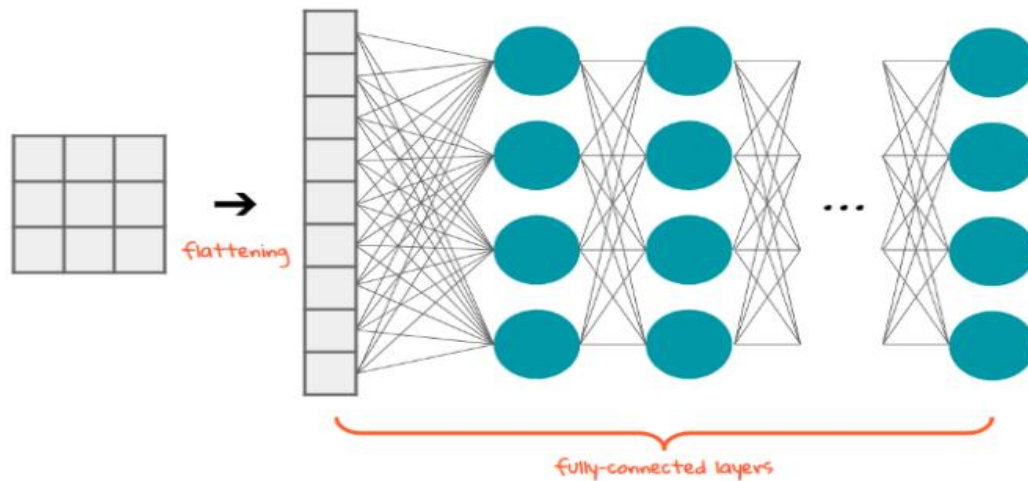


1.Convolution: A convolution converts all the pixels in its receptive field into a single value. If we apply a convolution to an image, we will be decreasing the image size as well as bringing all the information in the field together into a single pixel. The final output of the convolutional layer is a vector. We used the (64,64,3) format because we used the TensorFlow backend. It means 3 matrix of size (64X64) pixels representing Red, Green and Blue components of pixels.

2. Max Pooling: Pooling is the process of merging for the purpose of reducing the size of the data. Max pooling is a pooling operation that selects the maximum element from the region of the feature map covered by the filter. Thus, the output after max-pooling layer would be a feature map containing the most prominent features of the previous feature map.

3. Flattening: Flatten is the function that converts the pooled feature map to a single column that is passed to the fully connected layer. In this we convert the data into a 1-dimensional array for inputting it to the next layer. We flatten the output of the convolutional layers to create a single

long feature vector and then it is connected to the final classification model, which is called a fully-connected layer.



4.Fully Connected Neural Network: Dense adds the fully connected layer to the neural network. Fully Connected Layer is feed forward neural networks. Fully Connected Layers form the last few layers in the network. The input to the fully connected layer is the output from the final Pooling or Convolutional Layer, which is flattened and then fed into the fully connected layer.

5.Compiling the CNN: Compiling the CNN using the compile function. This function expects three parameters: the optimizer, the loss function, and the metrics of performance. The optimizer is the gradient descent algorithm. We used Adam optimizer. Adam optimization is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments. For loss function we used categorical cross entropy. It is a loss function that is used in multi-class classification tasks. And for metrics performance we used Accuracy, it is one metric for evaluating classification models.

Some of the hyperparameters used are:

Filters=32: This number indicates how many filters we are using to look at the image pixels during the convolution step. Some filters may catch sharp edges, some filters may catch color variations, some filters may catch outlines, etc. In the end, we get important information from the images. In the first layer the number of filters=32 is commonly used, then increasing the power of 2. Like in the next layer it is 64, in the next layer, it is 128 so on and so forth.

kernel_size=(5,5): This indicates the size of the sliding window during convolution, in this case study we are using a 5X5 pixels sliding window.

strides=(1, 1): How fast or slow should the sliding window move during convolution. We are using the lowest setting of 1X1 pixels. Means slide the convolution window of 5X5 (kernel_size) by 1 pixel in the x-axis and 1 pixel in the y-axis until the whole image is scanned.

input_shape=(64,64,3): Images are nothing but matrix of RGB color codes. during our data pre-processing we have compressed the images to 64X64, hence the expected shape is 64X64X3. Means 3 arrays of 64X64, one for RGB colors each.

kernel_initializer='uniform': When the Neurons start their computation, some algorithm has to decide the value for each weight. This parameter specifies that. You can choose different values for it like 'normal' or 'glorot_uniform'.

activation='relu': This specifies the activation function for the calculations inside each neuron. You can choose values like 'relu', 'tanh', 'sigmoid', etc.

optimizer='adam': This parameter helps to find the optimum values of each weight in the neural network. 'adam' is one of the most useful optimizers, another one is 'rmsprop'

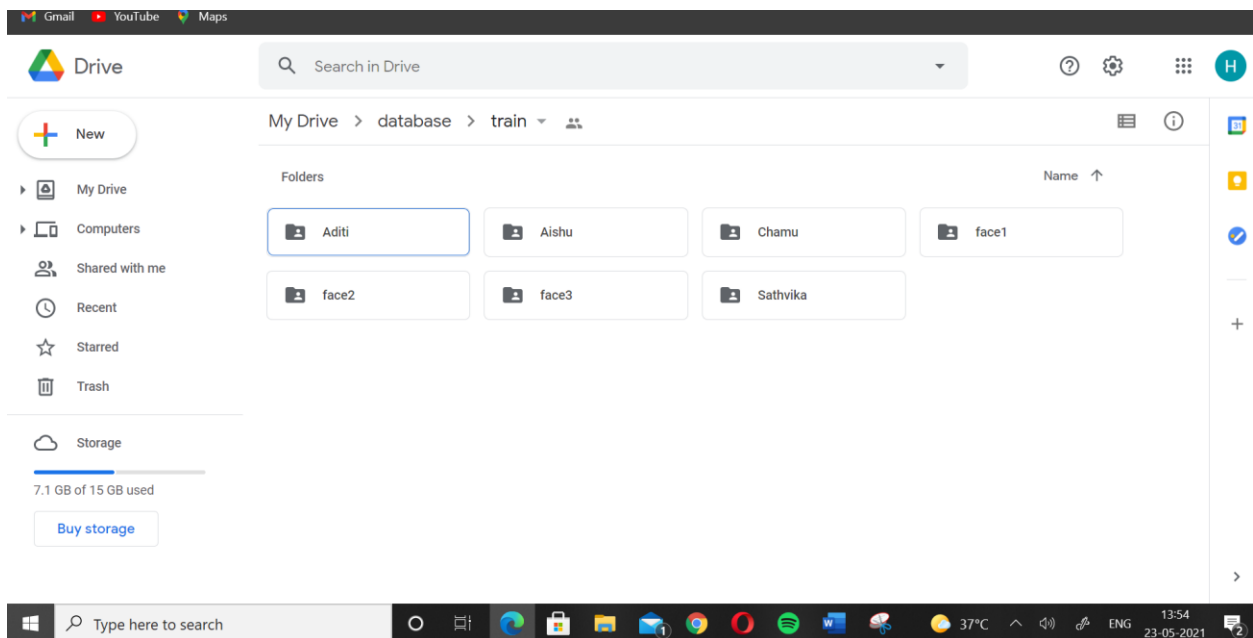
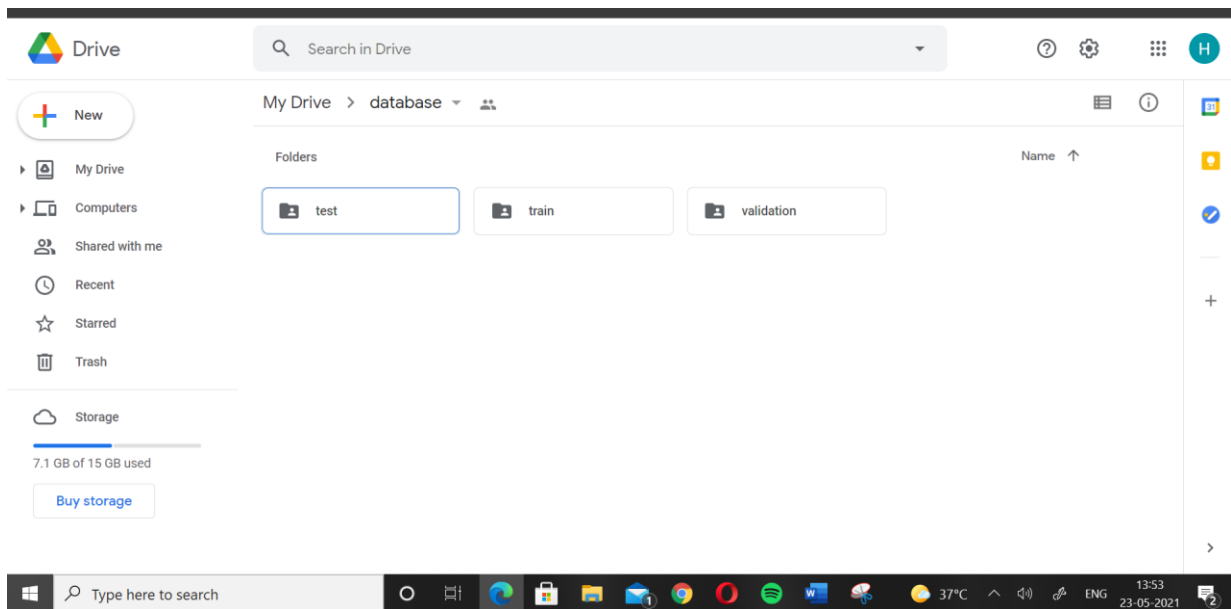
batch_size=10: This specifies how many rows will be passed to the Network in one go after which the SSE calculation will begin and the neural network will start adjusting its weights based on the errors.


When all the rows are passed in the batches of 10 rows each as specified in this parameter, then we call that 1-epoch. Or one full data cycle. This is also known as mini-batch gradient descent. A small value of batch_size will make the LSTM look at the data slowly, like 2 rows at a time or 4 rows at a time which could lead to overfitting, as compared to a large value like 20 or 50 rows at a time, which will make the LSTM look at the data fast which could lead to underfitting. Hence a proper value must be chosen using hyperparameter tuning.

Epochs=40: The same activity of adjusting weights continues for 40 times, as specified by this parameter. In simple terms, the LSTM looks at the full training data 10 times and adjusts its weights.

4. RESULTS

Datasets:





Drive

?

⚙

⋮

H

New

My Drive

Computers

Shared with me

Recent

Starred

Trash

Storage


7.1 GB of 15 GB used

Buy storage


My Drive > database > train > Aditi

Files


Name ↑




Adi_1.jpg




Adi_2.jpg



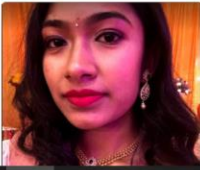
Adi_3.jpg



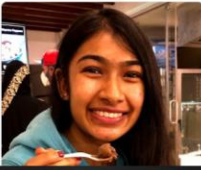
Adi_4.jpg



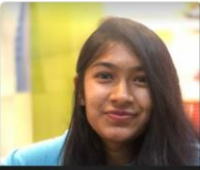
Adi_5.jpg



Adi_6.jpg




Adi_7.jpg




Adi_8.jpg

Type here to search



33°C 21:59 23-05-2021



Drive

?

⚙

⋮

H

New

My Drive

Computers

Shared with me

Recent

Starred

Trash

Storage

7.1 GB of 15 GB used

Buy storage

My Drive > database > train > Sathvika

Sat_5.jpg

Sat_6.jpg

Sat_7.jpg

Sat_8.jpg


Sat_9.jpg

Sat_10.jpg

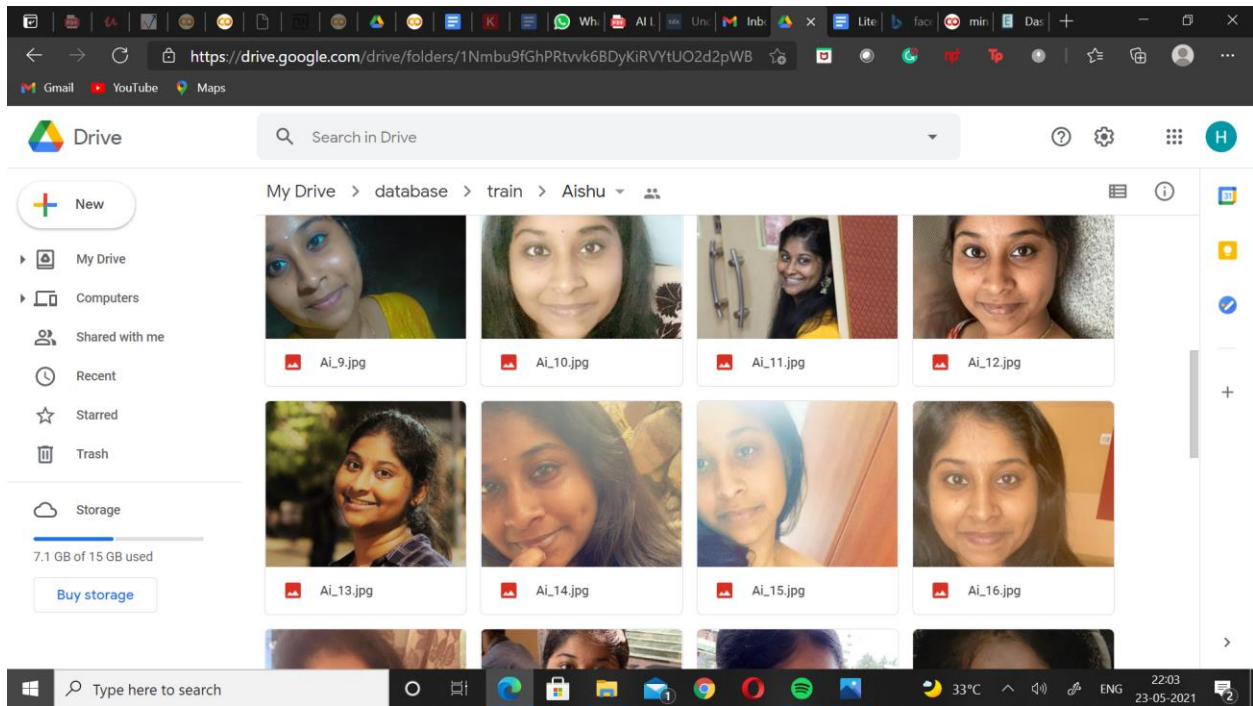
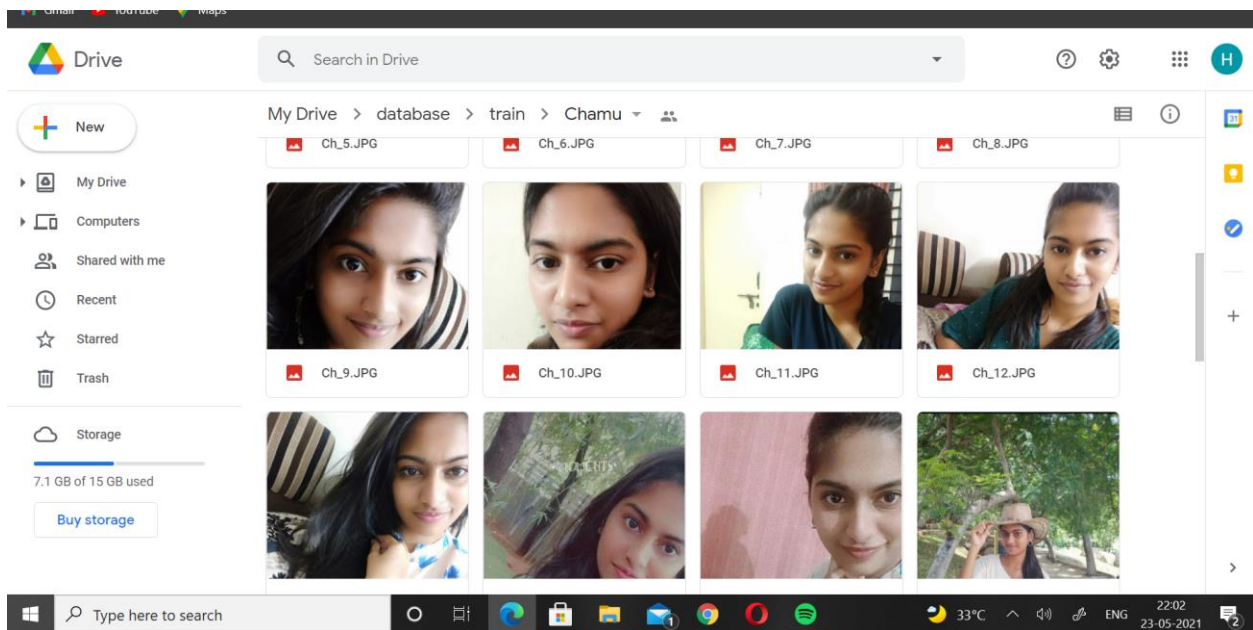
Sat_11.jpg

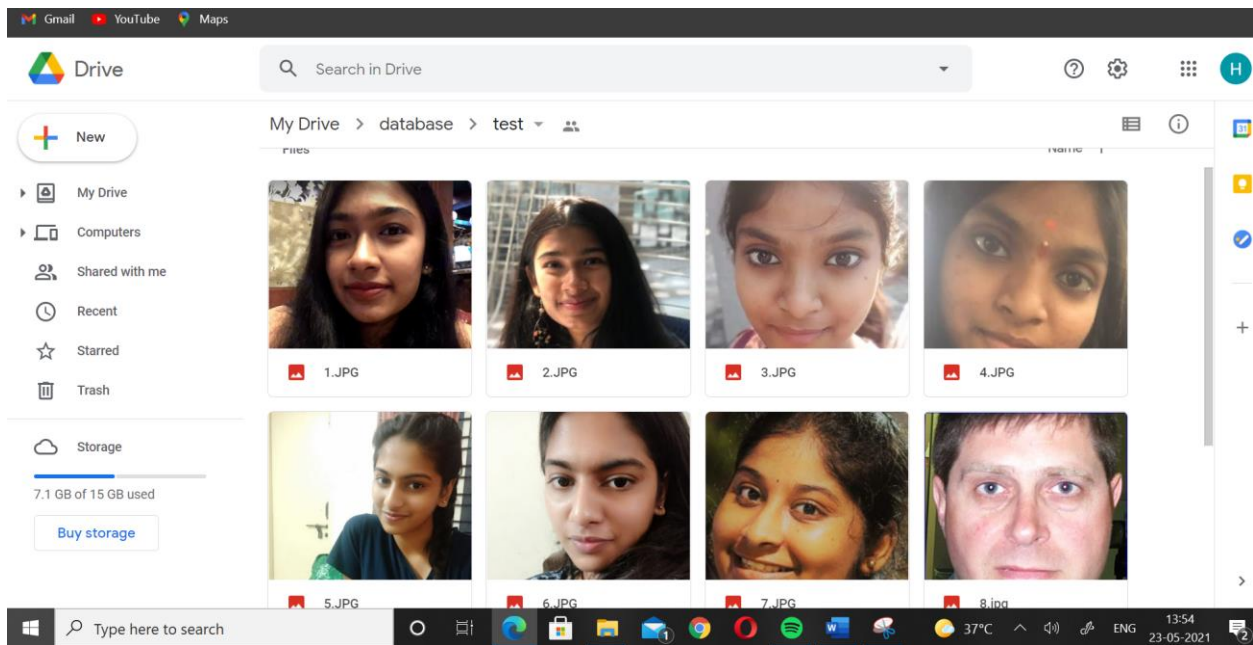
Sat_12.jpg

Type here to search

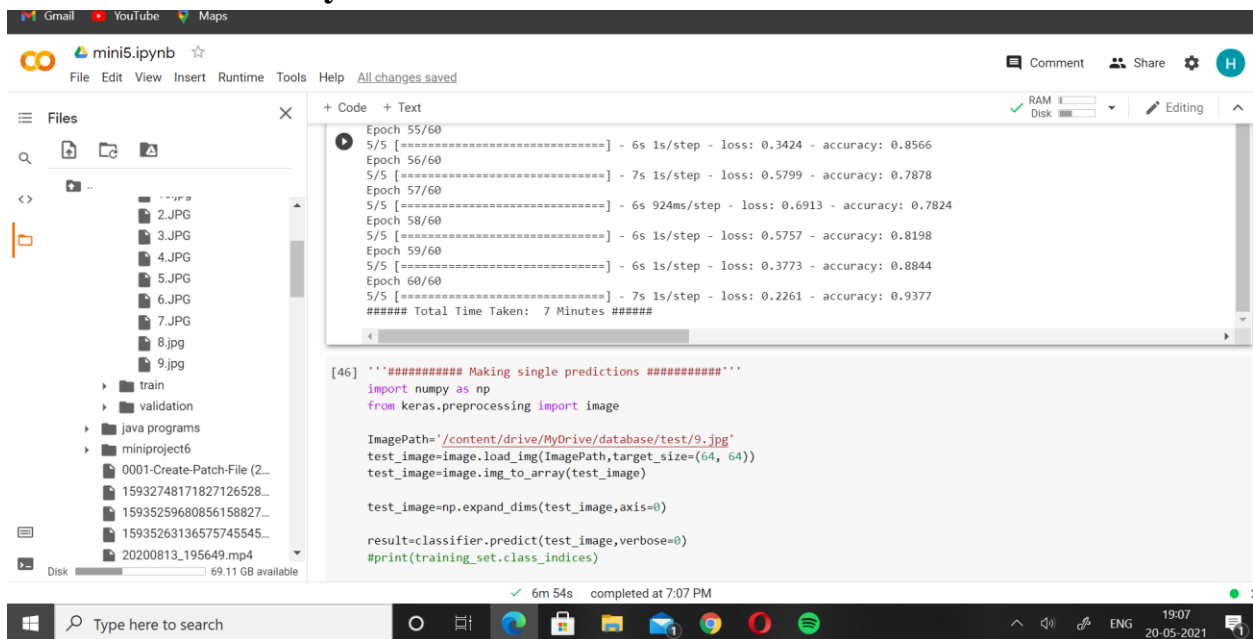


33°C 22:00 23-05-2021





Increase in Accuracy



mini6.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share H

Files

- 11.JPG
- 12.jpg
- 2.JPG
- 3.JPG
- 4.JPG
- 5.JPG
- 6.JPG
- 7.JPG
- 8.jpg
- 9.jpg
- train
- validation
- java programs
- miniproject6
- 0001-Create-Patch-File (2...
- 15932748171827126528...
- 15935259680856158827...

Disk 69.12 GB available

```
[7] 5/5 [=====] - 6s 1s/step - loss: 0.0401 - accuracy: 1.0000
Epoch 47/60
5/5 [=====] - 5s 794ms/step - loss: 0.0231 - accuracy: 1.0000
Epoch 48/60
5/5 [=====] - 5s 760ms/step - loss: 0.0127 - accuracy: 1.0000
Epoch 49/60
5/5 [=====] - 5s 866ms/step - loss: 0.0182 - accuracy: 1.0000
Epoch 50/60
5/5 [=====] - 6s 1s/step - loss: 0.0129 - accuracy: 1.0000
Epoch 51/60
5/5 [=====] - 5s 1s/step - loss: 0.0085 - accuracy: 1.0000
Epoch 52/60
5/5 [=====] - 5s 1s/step - loss: 0.0058 - accuracy: 1.0000
Epoch 53/60
5/5 [=====] - 6s 1s/step - loss: 0.0070 - accuracy: 1.0000
Epoch 54/60
5/5 [=====] - 6s 1s/step - loss: 0.0082 - accuracy: 1.0000
Epoch 55/60
5/5 [=====] - 5s 970ms/step - loss: 0.0038 - accuracy: 1.0000
Epoch 56/60
5/5 [=====] - 5s 1s/step - loss: 0.0045 - accuracy: 1.0000
Epoch 57/60
5/5 [=====] - 5s 922ms/step - loss: 0.0023 - accuracy: 1.0000
Epoch 58/60
5/5 [=====] - 6s 1s/step - loss: 0.0024 - accuracy: 1.0000
Epoch 59/60
5/5 [=====] - 6s 1s/step - loss: 0.0038 - accuracy: 1.0000
Epoch 60/60
5/5 [=====] - 5s 1s/step - loss: 0.0039 - accuracy: 1.0000
##### Total Time Taken: 7 Minutes #####
```

0s completed at 9:17 PM

Type here to search

31°C 21-05-2021

Predictions

mini5.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share H

Files

- drive
- MyDrive
- Classroom
- Colab Notebooks
- Resume
- database
- test
- 1.JPG
- 10.jpg
- 2.JPG
- 3.JPG
- 4.JPG
- 5.JPG
- 6.JPG
- 7.JPG
- 8.jpg
- 9.jpg

Disk 69.12 GB available

```
##### Making single predictions #####
import numpy as np
from keras.preprocessing import image

ImagePath='/content/drive/MyDrive/database/test/7.JPG'
test_image=image.load_img(ImagePath,target_size=(64, 64))
test_image=image.img_to_array(test_image)


test_image=np.expand_dims(test_image,axis=0)

result=classifier.predict(test_image,verbose=0)
#print(training_set.class_indices)

print('####*10)
print('Prediction is: ',ResultMap[np.argmax(result)])

#####
Prediction is: Aishu
```

7.JPG



0s completed at 5:42 PM

Type here to search

17:42 20-05-2021

Gmail YouTube Maps

mini5.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Files

- drive
 - MyDrive
 - Classroom
 - Colab Notebooks
 - Resume
 - database
 - test
 - 1.JPG
 - 10.jpg
 - 2.JPG
 - 3.JPG
 - 4.JPG
 - 5.JPG
 - 6.JPG
 - 7.JPG
 - 8.jpg
 - 9.jpg

69.12 GB available

0s completed at 5:46 PM

```
##### Total Time Taken: 8 Minutes #####

##### Making single predictions #####
import numpy as np
from keras.preprocessing import image

ImagePath="/content/drive/MyDrive/database/test/2.JPG"
test_image=image.load_img(ImagePath,target_size=(64, 64))
test_image=image.img_to_array(test_image)


test_image=np.expand_dims(test_image,axis=0)

result=classifier.predict(test_image,verbose=0)
#print(training_set.class_indices)

print('####*10')
print('Prediction is: ',ResultMap[np.argmax(result)])

#####
Prediction is: Aditi
```

2.JPG X



Gmail YouTube Maps

mini5.ipynb

File Edit View Insert Runtime Tools Help

Files

- drive
 - MyDrive
 - Classroom
 - Colab Notebooks
 - Resume
 - database
 - test
 - 1.JPG
 - 10.jpg
 - 2.JPG
 - 3.JPG
 - 4.JPG
 - 5.JPG
 - 6.JPG
 - 7.JPG
 - 8.jpg
 - 9.jpg

69.12 GB available

0s completed at 5:51 PM

```
##### Total Time Taken: 8 Minutes #####

##### Making single predictions #####
import numpy as np
from keras.preprocessing import image

ImagePath="/content/drive/MyDrive/database/test/5.JPG"
test_image=image.load_img(ImagePath,target_size=(64, 64))
test_image=image.img_to_array(test_image)


test_image=np.expand_dims(test_image,axis=0)

result=classifier.predict(test_image,verbose=0)
#print(training_set.class_indices)

print('####*10')
print('Prediction is: ',ResultMap[np.argmax(result)])

#####
Prediction is: Chamu
```

6.JPG X 5.JPG X



Gmail YouTube Maps

mini5.ipynb

File Edit View Insert Runtime Tools Help All changes saved


Files

- 11.JPG
- 2.JPG
- 3.JPG
- 4.JPG
- 5.JPG
- 6.JPG
- 7.JPG
- 8.jpg
- 9.jpg
- train
- validation
- java programs
- miniproject6
- 0001-Create-Patch-File (2...
- 15932748171827126528...
- 15935259680856158827...
- 15935263136575745545...

Disk 69.11 GB available

```
##### Making single predictions #####  
import numpy as np  
from keras.preprocessing import image  
  
ImagePath='/content/drive/MyDrive/database/test/11.JPG'  
test_image=image.load_img(ImagePath,target_size=(64, 64))  
test_image=image.img_to_array(test_image)  
  
test_image=np.expand_dims(test_image,axis=0)  
  
result=classifier.predict(test_image,verbose=0)  
#print(training_set.class_indices)  
  
print('###*10)  
print('Prediction is: ',ResultMap[np.argmax(result)])  
  
#####  
Prediction is: Chamu
```

11.JPG



0s completed at 8:00 PM

Type here to search

20-05-2021

Gmail YouTube Maps

mini5.ipynb

File Edit View Insert Runtime Tools Help All changes saved

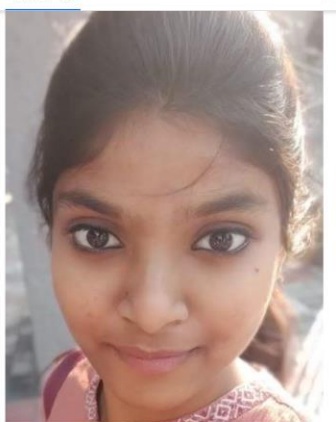
Files

- Resume
- database
 - test
 - 1.JPG
 - 10.jpg
 - 2.JPG
 - 3.JPG
 - 4.JPG
 - 5.JPG
 - 6.JPG
 - 7.JPG
 - 8.jpg
 - 9.jpg
 - train
 - validation
 - java programs
 - miniproject6
 - 0001-Create-Patch-File (2...

Disk 69.12 GB available

```
##### Making single predictions #####  
import numpy as np  
from keras.preprocessing import image  
  
ImagePath='/content/drive/MyDrive/database/test/3.JPG'  
test_image=image.load_img(ImagePath,target_size=(64, 64))  
test_image=image.img_to_array(test_image)  
  
test_image=np.expand_dims(test_image,axis=0)  
  
result=classifier.predict(test_image,verbose=0)  
#print(training_set.class_indices)  
  
print('###*10)  
print('Prediction is: ',ResultMap[np.argmax(result)])  
  
#####  
Prediction is: Sathvika
```

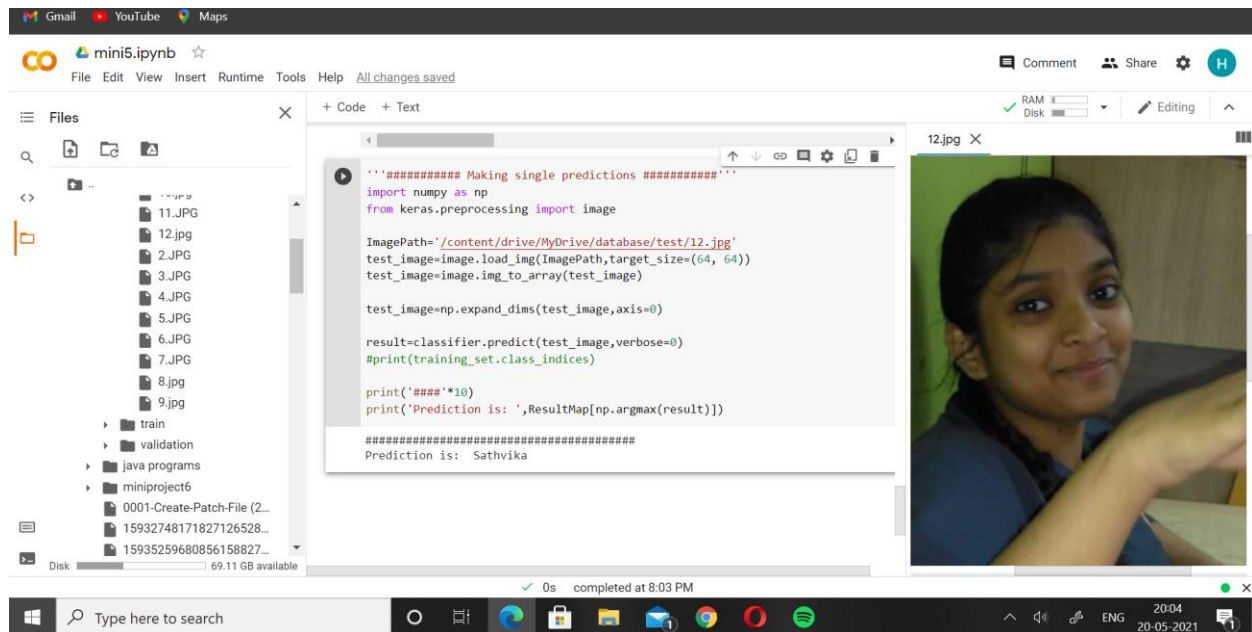
3.JPG



0s completed at 6:00 PM

Type here to search

18:00 20-05-2021



5. CONCLUSION

Face recognition is undoubtedly a massive contribution to the latest technology and its applications keep increasing. There are a number of ways face recognition can be performed. They are mainly divided into two types, image processing or by live capturing. Both the methods have their own advantages and disadvantages. The method we used in our project is image processing where we collected images from known people and some from the internet. Since our main idea of the project is to recognise the faces from the images that have effects like light, filters and many more, we trained the model with maximum photos and in a number of ways.

We applied a number of layers and increased the number of epochs to increase the accuracy of the model. We used a classifier in the model which can be used for classification of labeled data. Since labeling is automatically generated with the help of code, it is used by the classifier for classification of the data.

6. FUTURE WORK

In the future we plan to increase the range of the dataset more and also increase the accuracy of the model even more. We would also improve the model to the point where it would be able to detect even very old pictures of the people. We also want to incorporate this face recognition in automatic attendance. This would be a great way to eliminate manual attendance and would help teachers to have perfect attendance even without taking it manually. This would also help reduce proxy in attendance.

7. REFERENCES

1. https://link.springer.com/content/pdf/10.1007/3-540-55426-2_90.pdf
2. https://www.academia.edu/download/44539045/FACE_RECOGNITION_USING_PRINCIPAL_COMPONENTS20160408-24687-1cbupeu.pdf
3. <https://pdfs.semanticscholar.org/3255/0898eec9c4424932e70e5e32c98b0220a747.pdf>
4. https://ora.ox.ac.uk/objects/uuid:a5f2e93f-2768-45bb-8508-74747f85cad1/download_file?file_format=pdf&safe_filename=parkhi15.pdf&type_of_work=Confer
5. http://openaccess.thecvf.com/content_cvpr_2017/papers/Wang_Discriminative_Covariance_Oriented_CVPR_2017_paper.pdf
6. Convolutional Neural Networks (CNNs): An Illustrated Explanation - XRDSXRDS (acm.org)
7. Anyone can use this powerful facial-recognition tool — and that's a problem - CNN