Data Structure Project

(CSD-223)

Implement face recognition algorithm which requires extreme and optimal use of arrays and its management.

Submitted By

Group No - 18
Ritwik Duggal (185528)
Himanshi (185539)
Abhishek kumar (185540)

Submitted To

Er.Nitin Gupta

Department of Computer Science and Engineering

INTRODUCTION

In our project we design a Face recognition model which has basically threee parts, namely are:

1. Facial Detection

2. Object Tracking

3. Facial Recognition

SOFTWARE USED

1. OpenCV

OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel). The library is cross-platform and free for use under the open-source BSD license.

OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. All of the new developments and algorithms appear in the C++ interface. There are bindings in Python, Java and MATLAB/OCTAVE. The API for these interfaces can be found in the online documentation. Wrappers in other languages such as C, Perl, Ch, Haskell, and Ruby have been developed to encourage adoption by a wider audience. Since version 3.4, OpenCV.js is a JavaScript binding for selected subset of OpenCV functions for the web platform.

Figure 1: OpenCV

2. NumPy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays, requiring rewriting some code, mostly inner loops, using NumPy.

Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted,[17] and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars. In comparison, MATLAB boasts a large number of additional toolboxes, notably Simulink, whereas NumPy is intrinsically integrated with Python, a more modern and complete programming language. Moreover, complementary Python packages are available; SciPy is a library that adds more MATLAB-like functionality and Matplotlib is a plotting package that provides MATLAB-like plotting functionality. Internally, both MATLAB and NumPy rely on BLAS and LAPACK for efficient linear algebra computations.

Python bindings of the widely used computer vision library OpenCV utilize NumPy arrays to store and operate on data. Since images with multiple channels are simply represented as three-dimensional arrays, indexing, slicing or masking with other arrays are very efficient ways to access specific pixels of an image. The NumPy array as universal data structure in OpenCV for images, extracted feature points, filter kernels and many more vastly simplifies the programming workflow and debugging.

Figure 2: NumPy

3. Matplotlib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of Matplotlib.

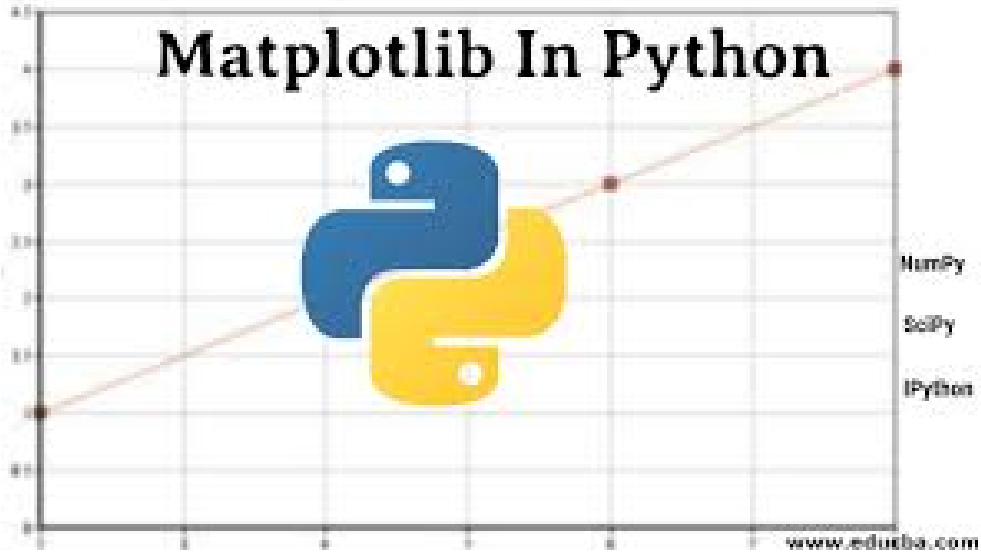

Figure 3: Matplotlib

4. Keras

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.

Features:

Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code. The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel.

In addition to standard neural networks, Keras has support for convolutional and recurrent neural networks. It supports other common utility layers like dropout, batch normalization, and pooling.

Figure 4: Keras

5. TensorFlow

TensorFlow is a free and open-source software library for data flow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks.It is used for both research and production at google.

Figure 5: TensorFlow

6. SciPy

SciPy is a free and open-source Python library used for scientific computing and technical computing.

SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

SciPy builds on the NumPy array object and is part of the NumPy stack which includes tools like Matplotlib, pandas and SymPy, and an expanding set of scientific computing libraries. This NumPy stack has similar users to other applications such as MATLAB, GNU Octave, and Scilab. The NumPy stack is also sometimes referred to as the SciPy stack.

7. Object Tracking Software

OpenCV 3 comes with a new tracking API that contains implementations of many single object tracking algorithms. There are 8 different trackers available in OpenCV 3.4.1

a. BOOSTING Tracker: Based on the same algorithm used to power the machine learning behind Haar cascades (AdaBoost), but like Haar cascades, is over a decade old. This tracker is slow and doesn't work very well. Interesting only for legacy reasons and comparing other algorithms.

b. MIL Tracker: Better accuracy than BOOSTING tracker but does a poor job of reporting failure.

c. KCF Tracker: Kernelized Correlation Filters. Faster than BOOSTING and MIL. Similar to MIL and KCF, does not handle full occlusion well.

d. TLD Tracker: TLD tracker was incredibly prone to false-positives.

e. MedianFlow Tracker: Does a nice job reporting failures; however, if there is too large of a jump in motion, such as fast moving objects, or objects that

change quickly in their appearance, the model will fail.

f. GOTURN Tracker: The only deep learning-based object detector included in OpenCV. It requires additional model files to run.

g. MOSSE Tracker: Very, very fast. Not as accurate as CSRT or KCF but a good choice if you need pure speed.

h. CSRT Tracker: Discriminative Correlation Filter (with Channel and Spatial Reliability). Tends to be more accurate than KCF but slightly slower.

8. Image Classifier

Object Detection using Haar feature-based cascade classifiers is an effective method proposed by Paul Viola and Michael Jones in the 2001 paper, "Rapid Object Detection using a Boosted Cascade of Simple Features". It is a machine learning based approach in which a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

FACE DETECTION

For object/face detection we find features in the provided image. There are various features but the main features of facial detection are :

1. Edge Features
2. Line Features
3. Four Rectangle Features

Generally there are many other features that we detect such as :

1. Edge Detection Using

a. Threshold
b. Gradient Change
c. Canny Age Detection

2. Corner Detection

a. Harris Corner Detection
b. Shi-Tomasi Corner Detection

3. Grid Detection

4. Contour Detection

For feature matching some common methods are:

1. Brute Force with ORB Descriptors
2. Brute Force with SIFT Descriptors
3. Ratio Test
4. FLANN Based Matcher

And in some cases where we have to detect internal edges we use the **Watershed Algorithm**.

For Facial Detection there are also many methods already made such as **Viola Jones Algorithm** with **Haar Cascade** that we have in our Facial Detection.

Working

In our project we use **Haar Cascade** image classifier.

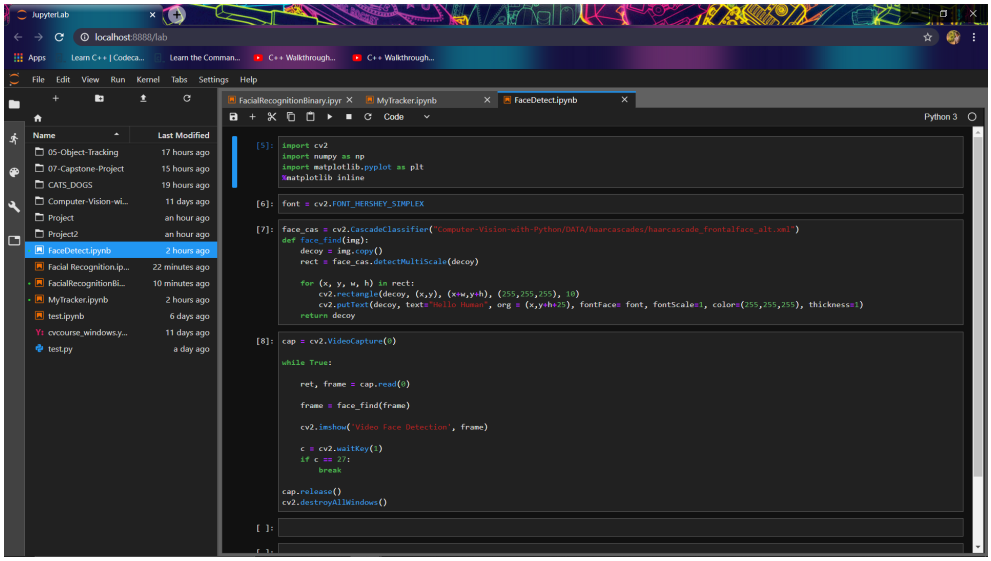

Figure 6: FaceDetect

OBJECT TRACKING

All the features we used in Facial Detection are also used in Object Tracking and for tracking the object we use following methods:

1. Lukas Kanade Algorithm
2. Gunner Farenback Algorithm
3. Mean Shift Algorithm

And ther are eight API's :

1. BOOSTING
2. MIL
3. KCF
4. TLD
5. MedianFlow
6. GOTURN
7. MOSSE
8. CSRT

For information about these please refer page no 7 under **object tracking software**.

Working

Below is the working model of our project.

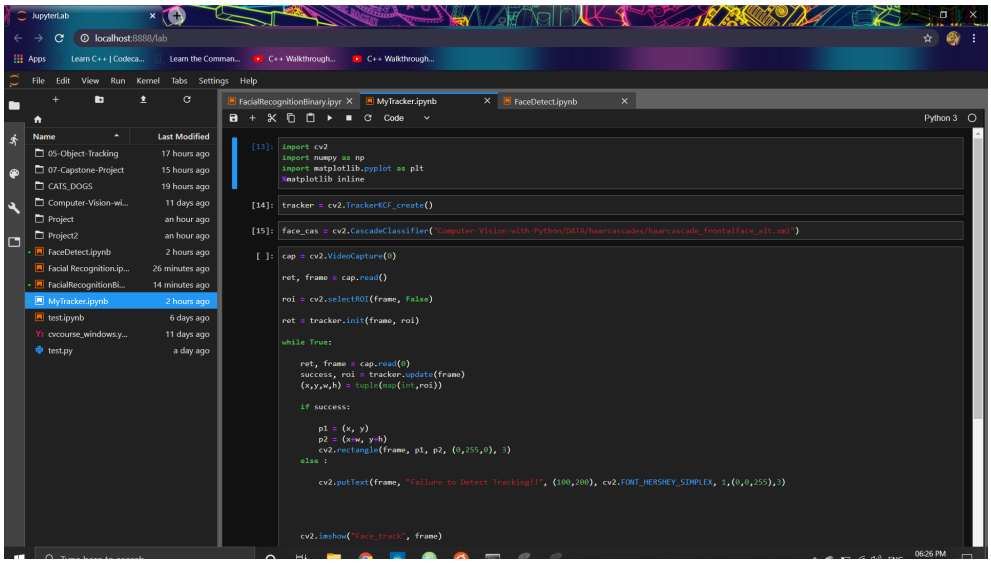

Figure 7: MyTracker1

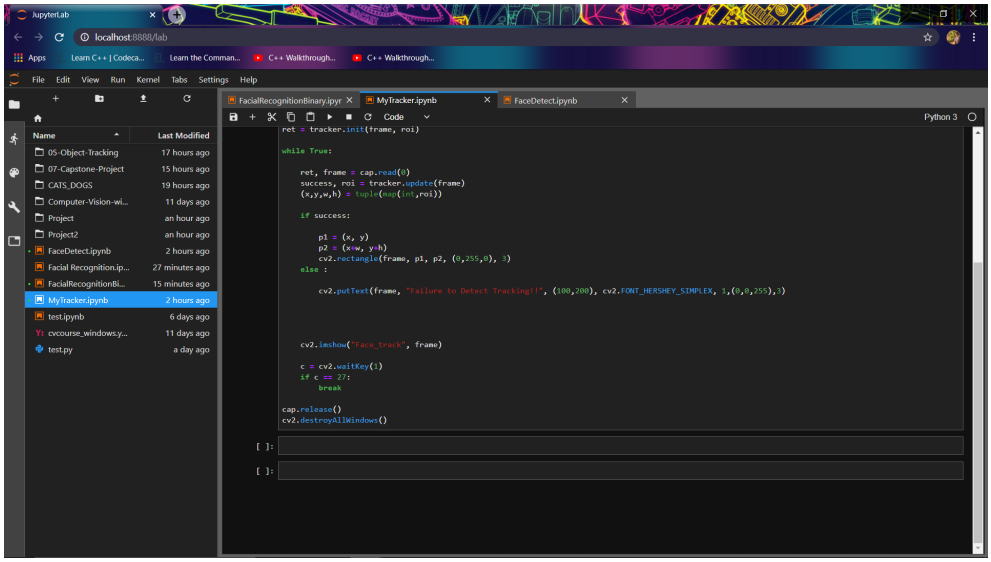

Figure 8: MyTracker2

FACIAL RECOGNITION

In our model of Facial Recognition we used a lot of processes such as Machine Learning along with Deep Learning and Neural Networks.W all used TensorFlow for backend.

Now various tools needed for this are given below:

1. ML Process : It includes 7 steps

a. Data Collection
b. Data Preparation
c. Choose a Model
d. Train the Model
e. Evaluate the Model
f. Parameter Tuning
g. Make Predictions

2. Types of Learning in ML

a. Supervised Learning

Supervised learning is one of the most basic types of machine learning. In this type, the machine learning algorithm is trained on labeled data. Even though the data needs to be labeled accurately for this method to work, supervised learning is extremely powerful when used in the right circumstances.

The algorithm then finds relationships between the parameters given, essentially establishing a cause and effect relationship between the variables in the dataset. At the end of the training, the algorithm has an idea of how the data works and the relationship between the input and the output.

This solution is then deployed for use with the final dataset, which it learns from in the same way as the training dataset. This means that supervised machine learning algorithms will continue to improve even after being de-

ployed, discovering new patterns and relationships as it trains itself on new data.

b. Unsupervised Learning

Unsupervised machine learning holds the advantage of being able to work with unlabeled data. This means that human labor is not required to make the dataset machine-readable, allowing much larger datasets to be worked on by the program.

In supervised learning, the labels allow the algorithm to find the exact nature of the relationship between any two data points. However, unsupervised learning does not have labels to work off of, resulting in the creation of hidden structures. Relationships between data points are perceived by the algorithm in an abstract manner, with no input required from human beings.

The creation of these hidden structures is what makes unsupervised learning algorithms versatile. Instead of a defined and set problem statement, unsupervised learning algorithms can adapt to the data by dynamically changing hidden structures. This offers more post-deployment development than supervised learning algorithms.

c. Reinforcement Learning

Reinforcement learning directly takes inspiration from how human beings learn from data in their lives. It features an algorithm that improves upon itself and learns from new situations using a trial-and-error method. Favorable outputs are encouraged or 'reinforced', and non-favorable outputs are discouraged or 'punished'.

Based on the psychological concept of conditioning, reinforcement learning works by putting the algorithm in a work environment with an interpreter and a reward system. In every iteration of the algorithm, the output result is given to the interpreter, which decides whether the outcome is favorable or not.

In case of the program finding the correct solution, the interpreter reinforces the solution by providing a reward to the algorithm. If the outcome is not favorable, the algorithm is forced to reiterate until it finds a better result.

In most cases, the reward system is directly tied to the effectiveness of the result.

3. Classification Matrix

The classification matrix is a standard tool for evaluation of statistical models and is sometimes referred to as a confusion matrix.
classification matrix is an important tool for assessing the results of prediction because it makes it easy to understand and account for the effects of wrong predictions. By viewing the amount and percentages in each cell of this matrix, you can quickly see how often the model predicted accurately.

a. Accuracy
b. Recoil
c. Precision
d. F1Score

4. Confusion Matrix

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing.

5. Neural Network

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature.

A neural network contains layers of interconnected nodes. Each node is a perceptron and is similar to a multiple linear regression. The perceptron feeds the signal produced by a multiple linear regression into an activation function that may be nonlinear.

As Example Neural Functions:

1. Sigmoid Function

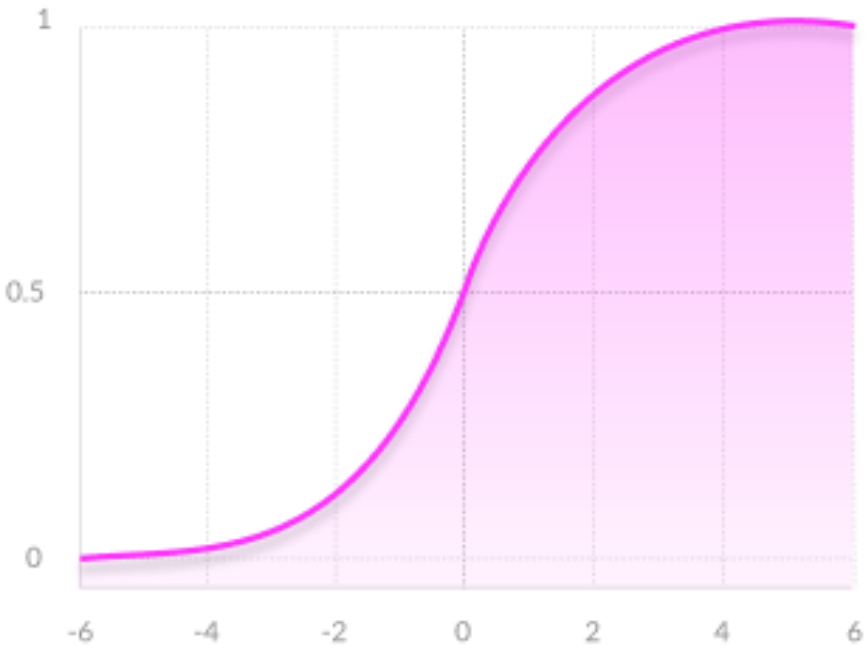

Figure 9: Sigmoid Function

Advantages

a. **Smooth gradient**, preventing "jumps" in output values.
b. **Output values bound** between 0 and 1, normalizing the output of each neuron.
c. **Clear predictions**—For X above 2 or below -2, tends to bring the Y value (the prediction) to the edge of the curve, very close to 1 or 0. This enables clear predictions.

Disadvantages

a. **Vanishing gradient**—for very high or very low values of X, there is almost no change to the prediction, causing a vanishing gradient problem. This can result in the network refusing to learn further, or being too slow to reach an accurate prediction.
b. **Outputs not zero centered.**
c. **Computationally expensive.**

2. ReLU (Rectified Linear Unit)

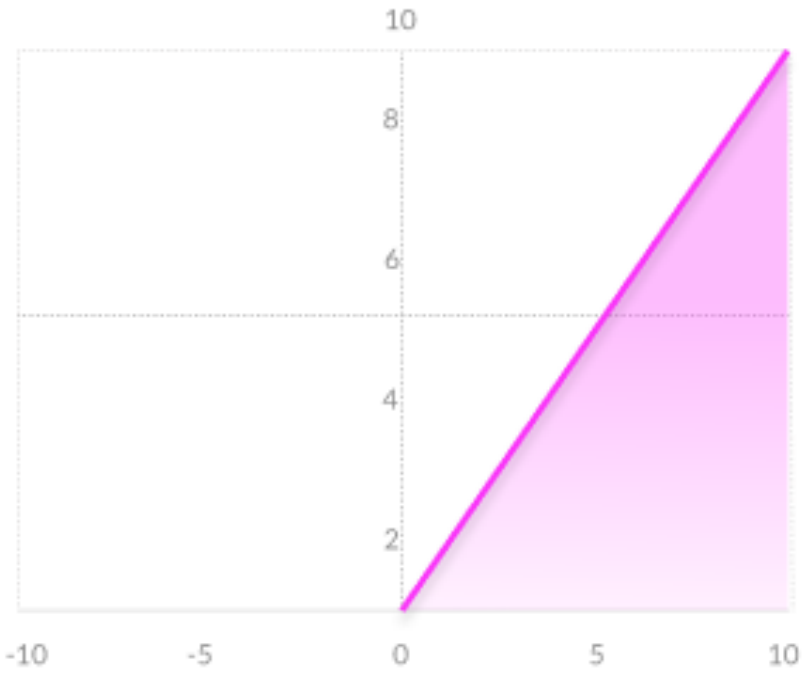

Figure 10: ReLU Function

Advantages

a. **Computationally efficient**—allows the network to converge very quickly.
b. **Non-linear**—although it looks like a linear function, ReLU has a derivative function and allows for backpropagation.

Disadvantages

a. **The Dying ReLU problem**—when inputs approach zero, or are negative, the gradient of the function becomes zero, the network cannot perform backpropagation and cannot learn.

In our project we use these two function.

6. Convolutional neural network

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analysing visual imagery. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics. They have applications in image and video recognition, recommender systems, image classification, medical image analysis, natural language processing, and financial time series.

We have used mixture of CNN and DNN.

7. Pooling Layers

A pooling layer is another building block of a CNN. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network. Pooling layer operates on each feature map independently.

8. Models

There are many models for creating Deep learning models in the Keras Python library fo example:

a. Keras Sequential Models

The Sequential model API is a way of creating deep learning models where an instance of the Sequential class is created and model layers are created and added to it.

b. Keras Functional Models

The Keras functional API provides a more flexible way for defining models.

It specifically allows you to define multiple input or output models as well as models that share layers. More than that, it allows you to define ad hoc acyclic network graphs.

Models are defined by creating instances of layers and connecting them directly to each other in pairs, then defining a Model that specifies the layers to act as the input and output to the model.

c. Standard Network Models

When getting started with the functional API, it is a good idea to see how some standard neural network models are defined.

We can define a simple multilayer Perceptron, convolutional neural network, and recurrent neural network.

These will provide a foundation for understanding the more elaborate examples.

d. Shared Layers Model

Multiple layers can share the output from one layer.

For example, there may be multiple different feature extraction layers from

an input, or multiple layers used to interpret the output from a feature extraction layer.

e. Multiple Input and Output Models

The functional API can also be used to develop more complex models with multiple inputs, possibly with different modalities. It can also be used to develop models that produce multiple outputs.

In our we used Keras Sequential Models.

Here is the Face Recognition Screenshots....

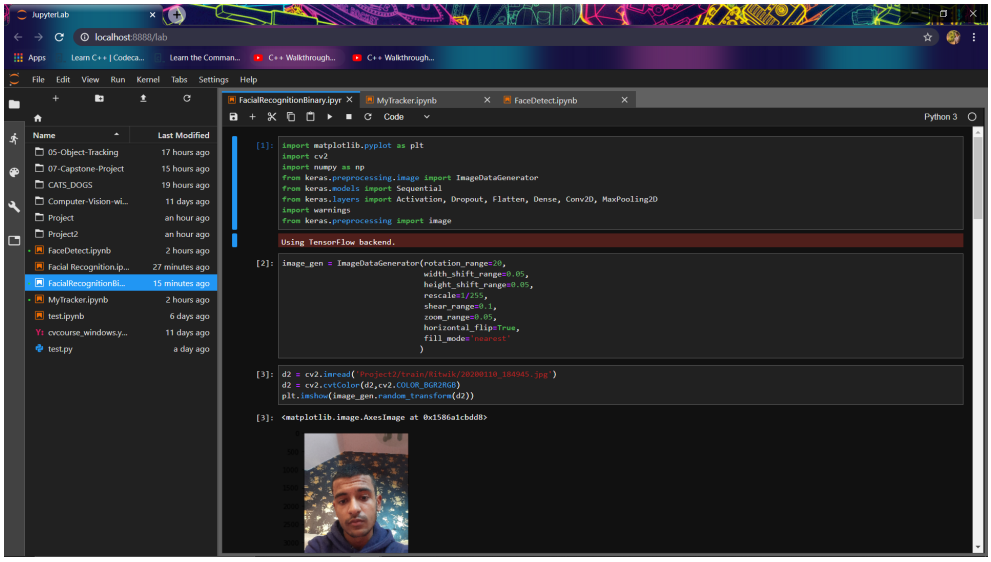

Figure 11: FacialRecognitionBinary1

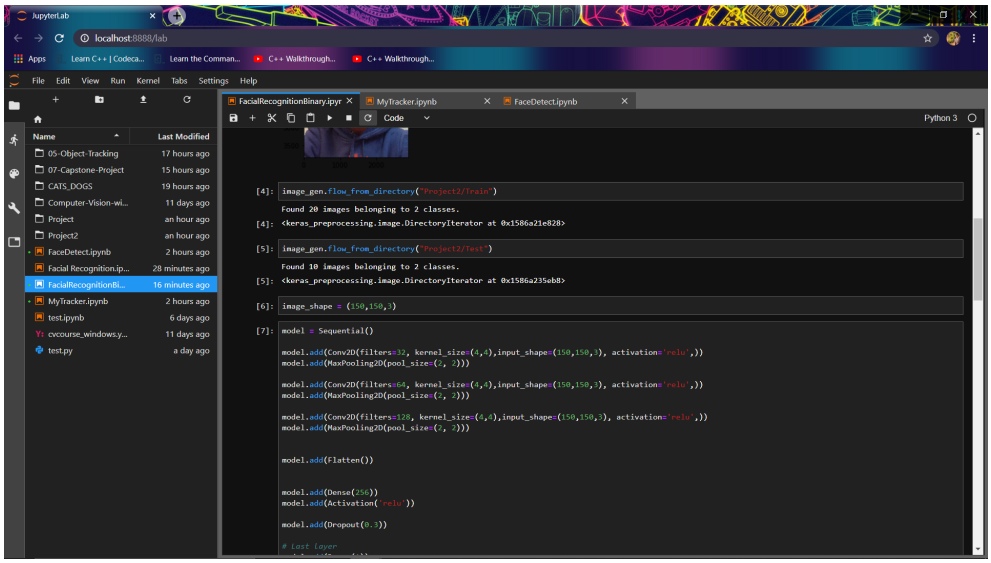

Figure 12: FacialRecognitionBinary2

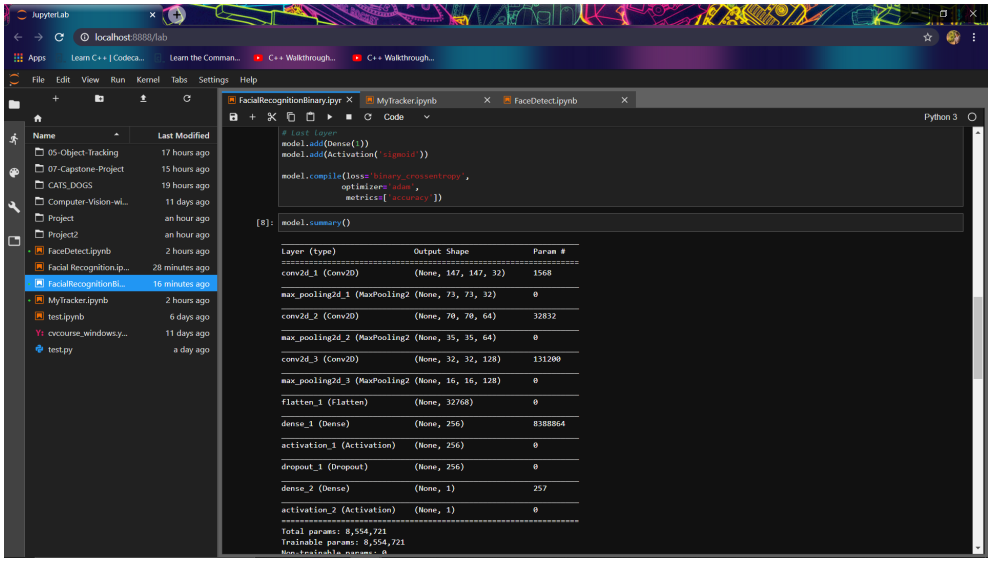

Figure 13: FacialRecognitionBinary3

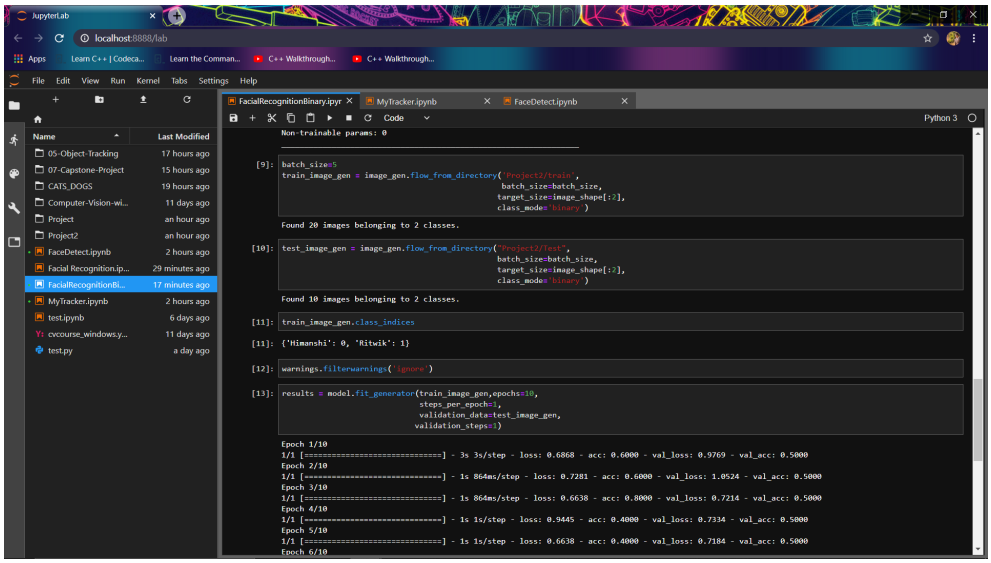

Figure 14: FacialRecognitionBinary4

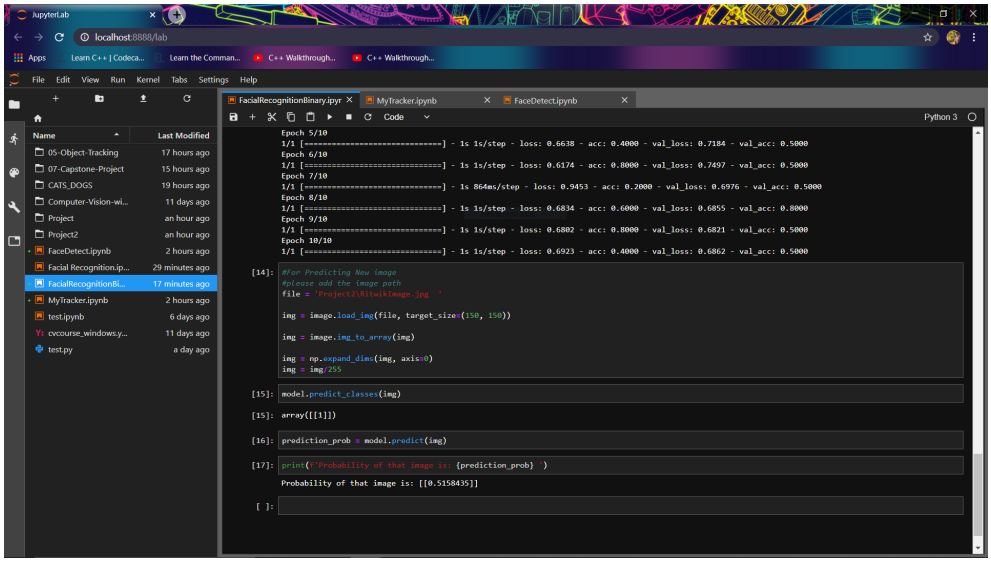

Figure 15: FacialRecognitionBinary5

GROUP-18

1. Ritwik Duggal
Email ID : 185539@nith.ac.in
Contact No : +91 9805334346

2. Himanshi
Email ID : 185539@nith.ac.in
Contact No : +91 9210517519

3. Abhishek Kumar
Email ID : 185540@nith.ac.in
Contact No : +91 6201397289

4. Github Link :**https://github.com/ritwikduggal/Face-Detection-and-recognition-using-numpy-opencv-tensorflow-keras.**