Microsoft Innovation Lab

PES University

Summer Internship – 2017

(June – July 2017)

Mapping of Textual Description to Facial Image

Team Members:

Varun Ranganathan

Piyush Surana

Mentor:

Shashwath S Bharadwaj

Nishant Hegde

Contents

Abstract

Our idea involves the creation of a software application which helps the user retrieve the face of a person from a database just by inputting a description of his facial features. Our implementation makes use of image processing principles in conjunction with a neural network to extract and classify features from a given database of facial images at its core while, the complete application will also make use of NLP concepts.

This application can be used as an assistant to sketch artists in the criminal investigation procedure, to reduce the time required to develop a sketch by searching for the suspect from a database of images of known criminals. It can also work like a shazam for celebrities and in other similar applications where an image is to be retrieved based on its textual/verbal description.

1 Introduction

1.1 Domain

1.1.1 Machine Learning

Machine Learning can be the field of study that gives computers the ability to learn without being explicitly programmed.This is an older, informal definition. Tom Mitchell provides a more modern definition: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."
Example: playing checkers.
E = the experience of playing many games of checkers T = the task of playing checkers P = the probability that the program will win the next game.

1.1.2 Neural Networks

An artificial neural network is an interconnected group of nodes, akin to the vast network of neurons in a brain. Here, each circular node represents an artificial neuron and an arrow represents a connection from the output of one neuron to the input of another.
In machine learning, a convolutional neural network (CNN, or ConvNet) is a class of deep, feed-forward artificial neural network that have successfully been applied to analyzing visual imagery. CNNs use a variation of multilayer perceptrons designed to require minimal preprocessing.[1] They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics.

1.1.3 Image Processing

Digital image processing is the use of computer algorithms to perform image processing on digital images. As a subcategory or field of digital signal processing, digital image processing has many advantages over analog image processing. It allows a much wider range of algorithms to be applied to the input data and can avoid problems such as the build-up of noise and signal distortion during processing. Since images are defined over two dimensions (perhaps more) digital image processing may be modeled in the form of multidimensional systems.
OpenCV provides a very good library for image processing in Python
HaarCascade Classifiers available in OpenCV can be very helpful in facial recognition and extracting features of the face

1.1.4 Natural Language Processing

Natural language processing (NLP) is a field of computer science, artificial intelligence and computational linguistics concerned with the interactions between computers and human (natural) languages, and, in particular, concerned with programming computers to fruitfully process large natural language corpora. Challenges in natural language processing frequently involve natural language understanding, natural language generation (frequently from formal, machine-readable logical forms), connecting language and machine perception, dialog sys-

tems, or some combination thereof.
Python supports NLP with the help of NLTK library

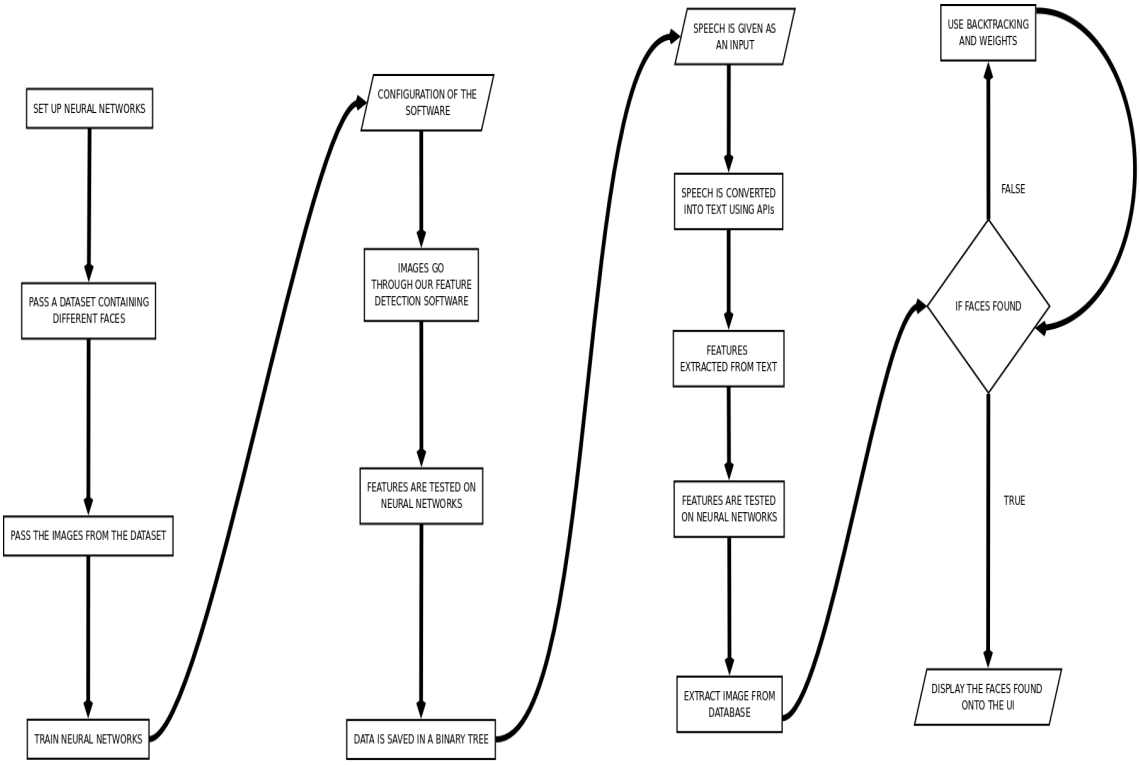
1.2 Project

Creation of a application which helps the user retrieve information about a person just by inputting a description of his/her facial features. This project is inspired by the work of a sketch artist, where his work is automated, and the described person is directly retrieved from the database. We go through the set of frontal facial images of different people, extract their features and classify the features. Based on the textual description, we search our database for the person who matches closest with the description.

1.3 Problem Statement

- To extract all the facial features of the image and store in the database.
- To retrieve an image from the database, given the description as text.

1.4 Block Diagram



2 Project Objectives

1. Design a program which will be able to extract all the features from the face.
2. Classification of features.
3. Identify a person based on the textual description

3 Hardwares and Softwares Used

- Hardware
  - CPU 16 Cores
  - RAM 64 GB
  - GPU Testa K80 x 2
- Software
  - Language
    - Python3
  - Modules and Packages
    - OpenCV
    - Dlib
    - TFLearn
    - Tensorflow-gpu
    - Scikit-learn
    - h5py
    - NLTK
    - Glob
    - Pickle
    - TKinter
    - Numpy
    - CUDA 8.0

4 Development Steps of the Project

4.1 Steps Involved

4.1.1 Feature Extraction

1. Found a suitable database i.e. the Color Feret database - a pre-classified database for gender, moustache, beard and glasses.

2. Extracted features such as face, eyes, nose, mouth and ears using HaarCascade classifiers available in OpenCV.

3. Extracted face shape using Dlib library by extracting the 68 points on a face and connecting them together.

4. Extracted skin colour by Image Processing using OpenCV and K-Nearest Neighbours.

4.1.2 Neural Networks

1. Learnt about Machine Learning with Neural Networks.

2. Studied the behaviour of Artificial Neural Networks, and concluded that Convolution Neural Networks are the best for image classification.

3. Implemented the Inception v3 Model of CNN but due to computational limitations, the model was too big to fit on the RAM, and therefore couldn't train the model.

4. Implemented the AlexNet Model of CNN used for classification of the facial features.

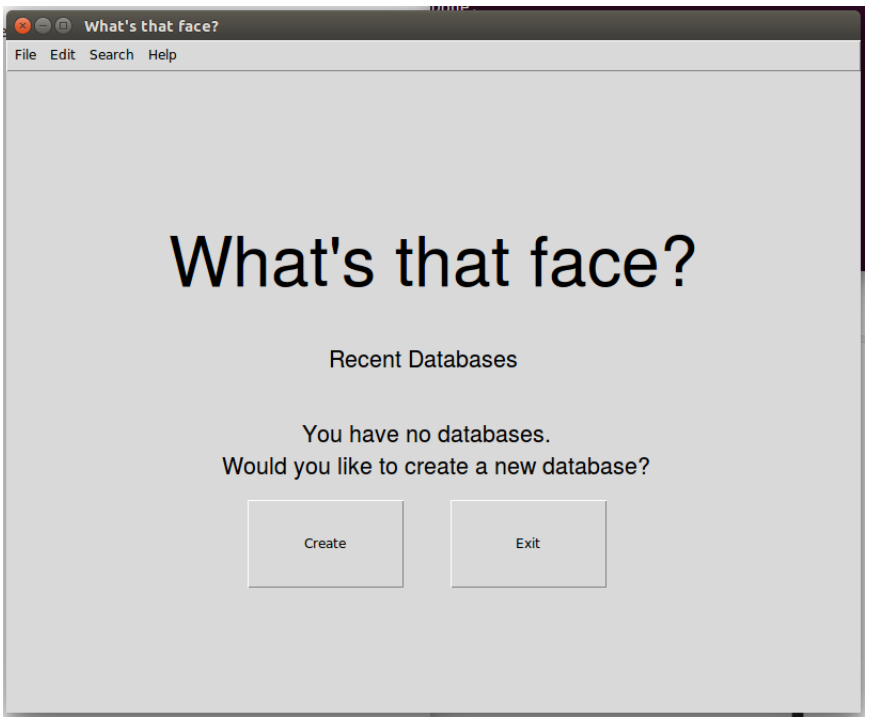
5. The models were trained on Google Cloud Platform.

4.1.3 Natural Language Processing and GUI

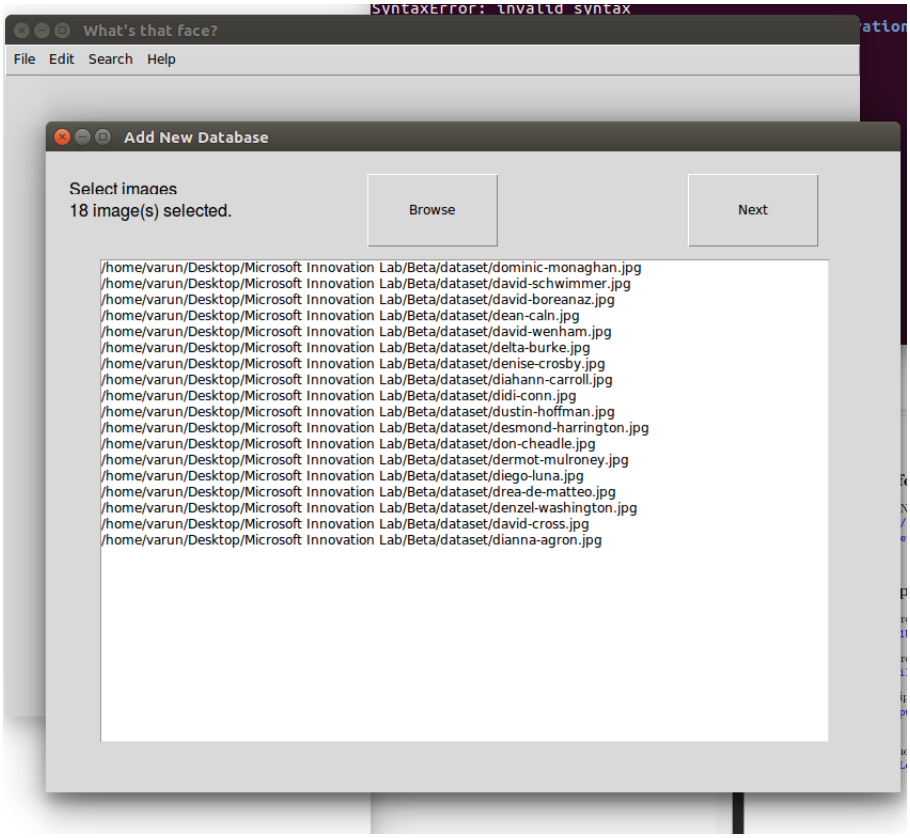
1. Studied and Implemented NLP techniques on text using NLTK on Python.

2. Used TKinter to create the Graphical User Interface.

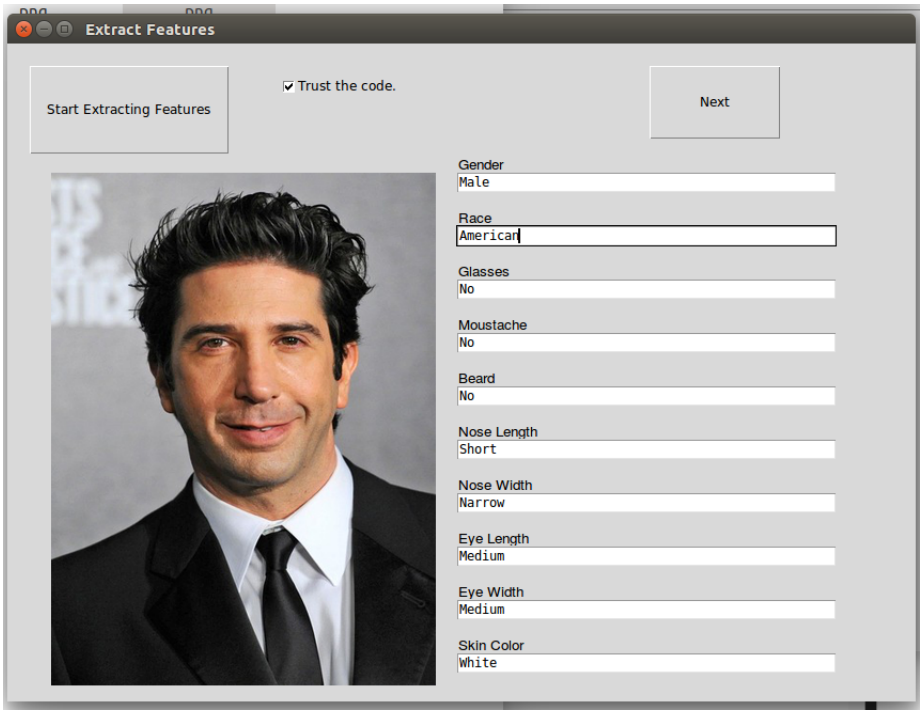
Opening Screen



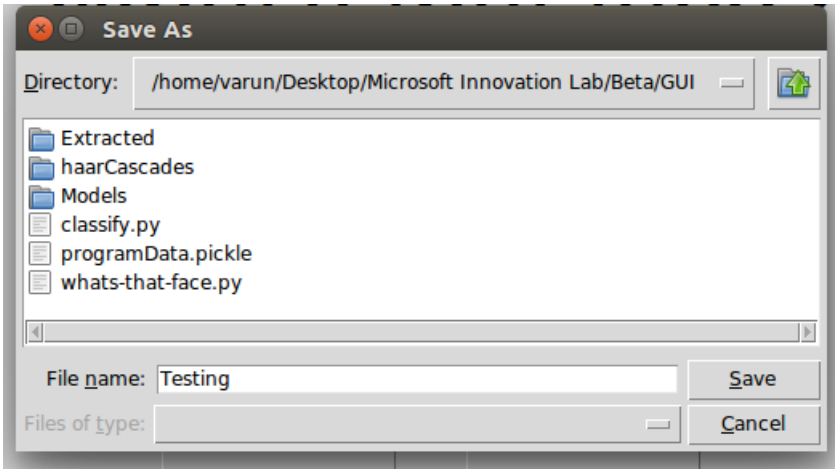
Selecting Images for New Database



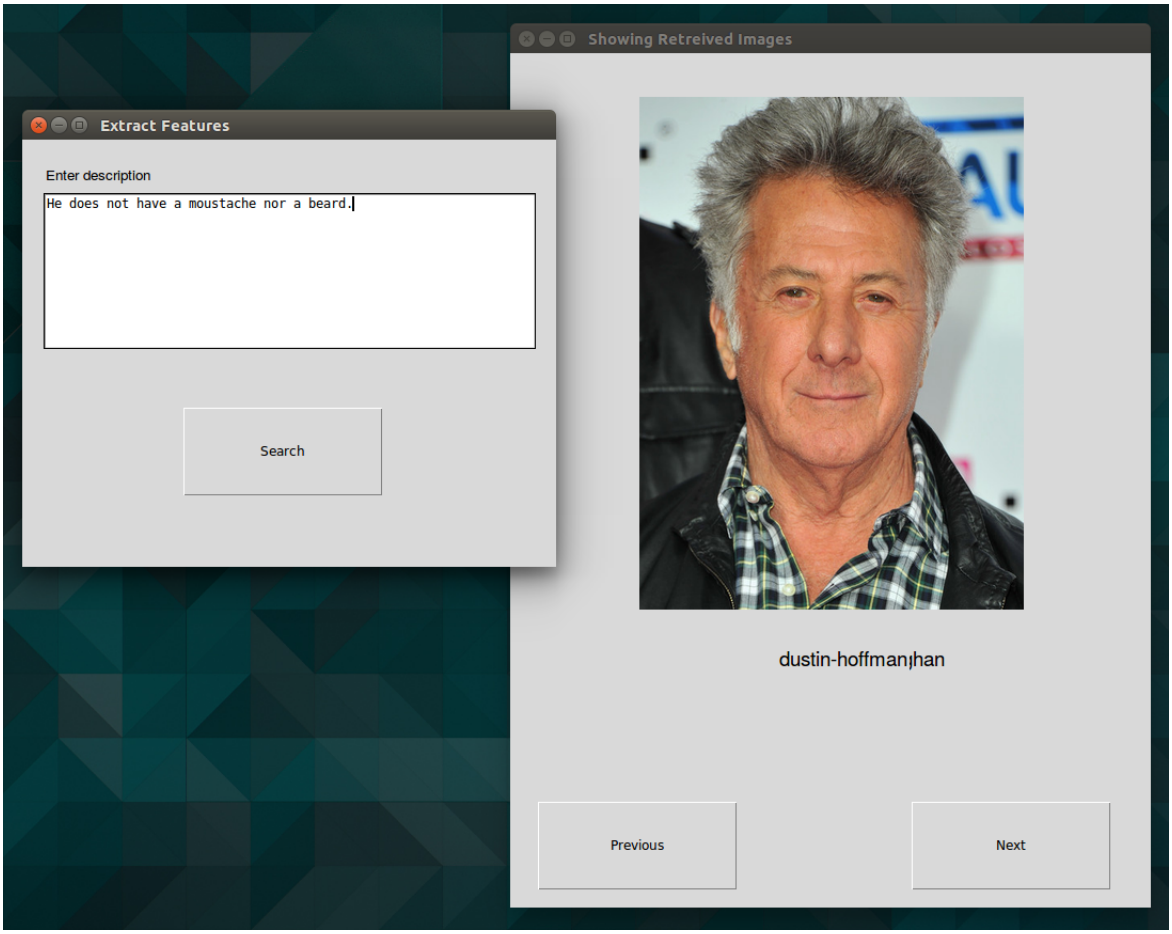
Testing and Classification



Saving the database



Searching via a Text Input Box



5 Challenges Faced

5.1 Read Up

Learning about Machine Learning and Neural Networks in a short amount of time was a very tedious task which required a lot of effort.

5.2 Software Problems

Image Processing on OpenCV was a hassle because some functions worked only on Windows and some worked only on Linux, and there were many conflict errors between OpenCV 2.4 and OpenCV 3.1.
Installation of Tensorflow-gpu was the biggest problem we faced. Encountered an infinite login loop error on Ubuntu because the display stack had been corrupt, therefore had to reinstall the whole OS. Installation of GPU Drivers on the cloud was even more tedious.
Dlib did not work on Windows Platform.

5.3 Dataset

Collection of dataset from the NIST (National Institute of Standards and Technology) Color FERET and segregating data based on our classes was time-consuming.
Creation of datasets for Face Shape through Web-Scraping methods wwas time-consuming and eventually proved futile.
Extraction of skin color was difficult because there was a lot of noise in the images due to external illumination. Developing a method to extract the class of skin color was completely different.

5.4 Model of Neural Network

Implementing the Inception v3 model was not possible due to computational limitations. The model required very high specifications and took up more than 420 GB of RAM (without running on a GPU).
The trained neural network for face shape classification gave a very low accuracy of 25% due to the lack of a legitimate dataset.

6 Result

6.1 Neural Networks

6.1.1 About the Neural Network Model

AlexNet is the name of a convolutional neural network running on GPUs implemented in CUDA, which competed in the ImageNet Large Scale Visual Recognition Challenge in 2012. AlexNet was designed by Alex Krizhevsky.
The network is modeled as follows:

- Input Layer
- Convolution Layer
- Max Pooling 2D Layer
- Local Response Normalization Layer
- Convolution Layer
- Max Pooling 2D Layer
- Local Response Normalization Layer
- Convolution Layer
- Convolution Layer
- Convolution Layer
- Max Pooling 2D Layer
- Local Response Normalization Layer
- Fully Connected Layer
- Dropout Layer

We used the AlexNet model to classify the features of the face.

Gender

The cropped and reshaped images of the face (100x100) was used to train the network. The class was also sent along with the images. The data of the class was present in the metadata of the Color FERET database. The network ran for 500 epochs with 1205 training images, and 110 cross-validation images. The cross-validation accuracy at the end of the training process reached upto 95%. The testing accuracy on the data is 92%.

Beard

The cropped and reshaped images of the face (100x100) was used to train the network. The class was also sent along with the images. The data of the class was present in the metadata of the Color FERET database. The network ran for 500 epochs with 1205 training images, and 110 cross-validation images. The cross-validation accuracy at the end of the training process reached upto 95%. The testing accuracy on the data is 90%.

Glasses

Initially, the idea was to send images extracted by the Eye haarCascade Classifier (resized to 70x40) to the neural network along with the classes for training purposes. But haarCascades did not have a very high accuracy, and this inhibited us to achieve a very high accuracy from our neural network. We were able to achieve an accuracy of 70-80% only. Therefore we switched to using the whole face for training. This data of the class, again, was present in the metadata of the Color FERET database. The network ran for 500 epochs with 1205 training images, and 110 cross-validation images. The cross-validation accuracy at the end of the training process reached upto 95%. The testing accuracy on the data is 94%.

Moustache

Even here, initially, the idea was to send images extracted by the Mouth haarCascade Classifier extended above to show the philtrum (resized to 70x40) to the neural network along with the classes for training purposes. But haarCascades did not have a very high accuracy, and this inhibited us to achieve a very high accuracy from our neural network. We were able to achieve an accuracy of 70-80% here also. Therefore we switched to using the whole face for training. This data of the class, again, was present in the metadata of the Color FERET database. The network ran for 500 epochs with 1205 training images, and 110 cross-validation images. The cross-validation accuracy at the end of the training process reached upto 90%. The testing accuracy on the data is 85%.

6.1.2 Clustering

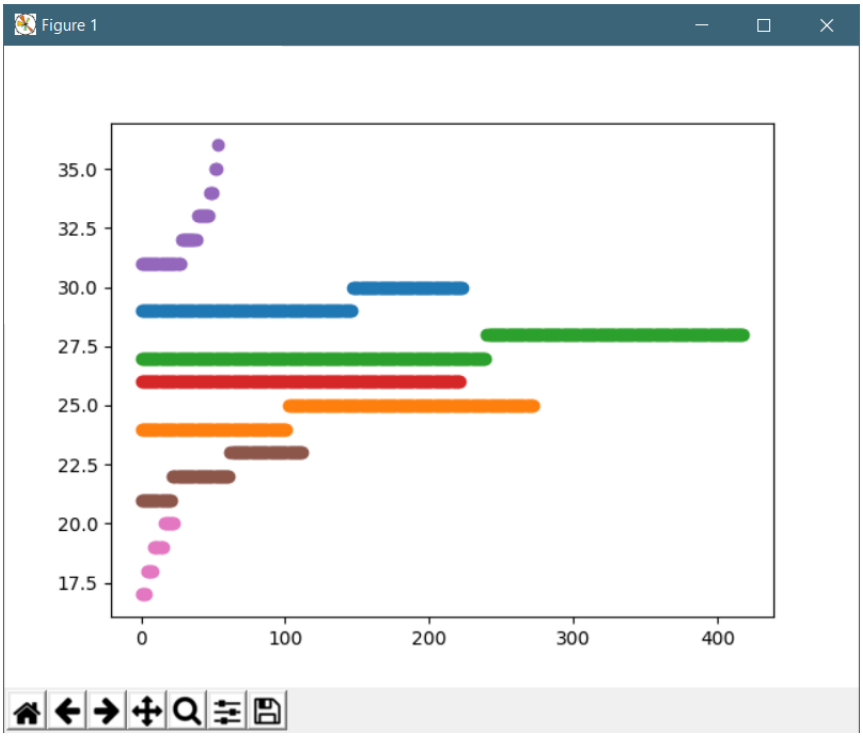
We used the KMeans clustering algorithm for Unsupervised Machine Learning to group dimensional features such as Nose Length, Nose Width, Eye Length, Eye Width. We choose this method because it is the intuitive way to go about the problem of classification of these features. It is matter of perspective from the user to describe these features as "Long", "Medium", "Short" (Nose Length); "Narrow", "Medium", "Wide" (Nose Width); "Big", "Medium", "Small" (Eye Length/Height); "Narrow", "Medium", "Width" (Eye Width).
We are using the clustering algorithm to detect the skin color. We extract the skin from the face using the following code:

```
def getOnlySkin(image):

    # Lower end of the HSV value for skin.
    low = (3,50,50)

    # Higher end of the HSV value for skin.
    up = (33,255,255)

    # Convert OpenCV Matrix to HSV values of the image.
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

    # Get only those pixels within the specified range.
    skinMask=cv2.inRange(hsv, low, up)

    # Helps in forming a custom shape of the face.
```

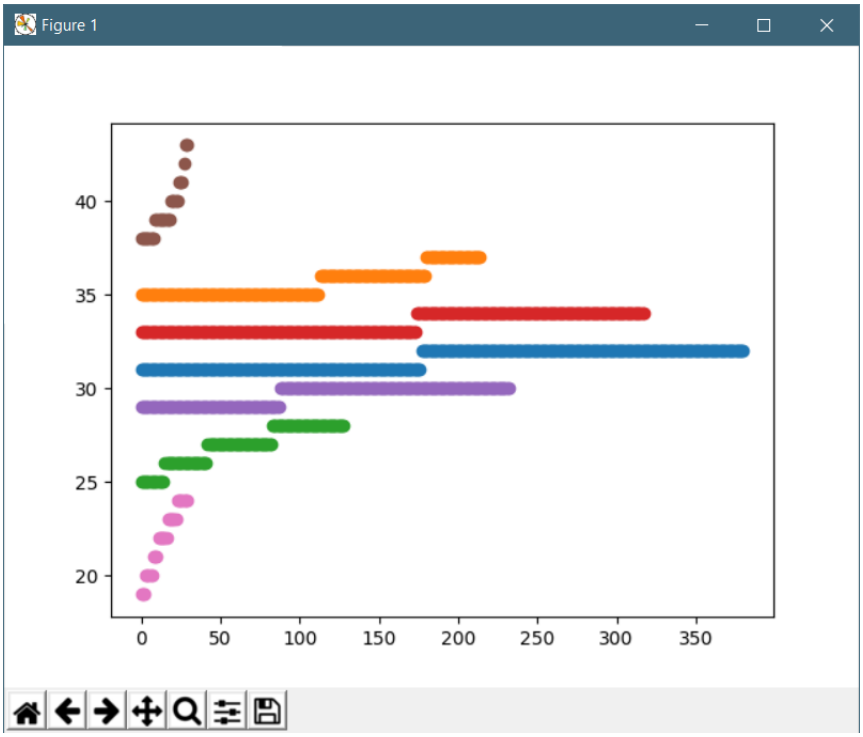
```
        kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (11,
            11))

        # 'Erode away' the boundaries of the foreground object.
        skinMask = cv2.erode(skinMask, kernel, iterations = 2)

        # Used after erosion to because erosion removes away some
            pixels and shrink the object.
        #   This will add some white pixels, joining the broken
            pieces of the image.
        skinMask = cv2.dilate(skinMask, kernel, iterations = 2)

        # Replaces the center pixel with the average of the
            neighbourhood pixel values
        skinMask = cv2.GaussianBlur(skinMask, (3, 3), 0)
        skin = cv2.bitwise_and(image, image, mask = skinMask)

        # Convert the pixels back to standard BGR values.
        skin = cv2.cvtColor(skin, cv2.COLOR_HSV2BGR)

        return skin
```

After this, we average the values of HLS values and use the Luminosity value to cluster and to determine whether the person is "Dark", "Brown", "Fair".

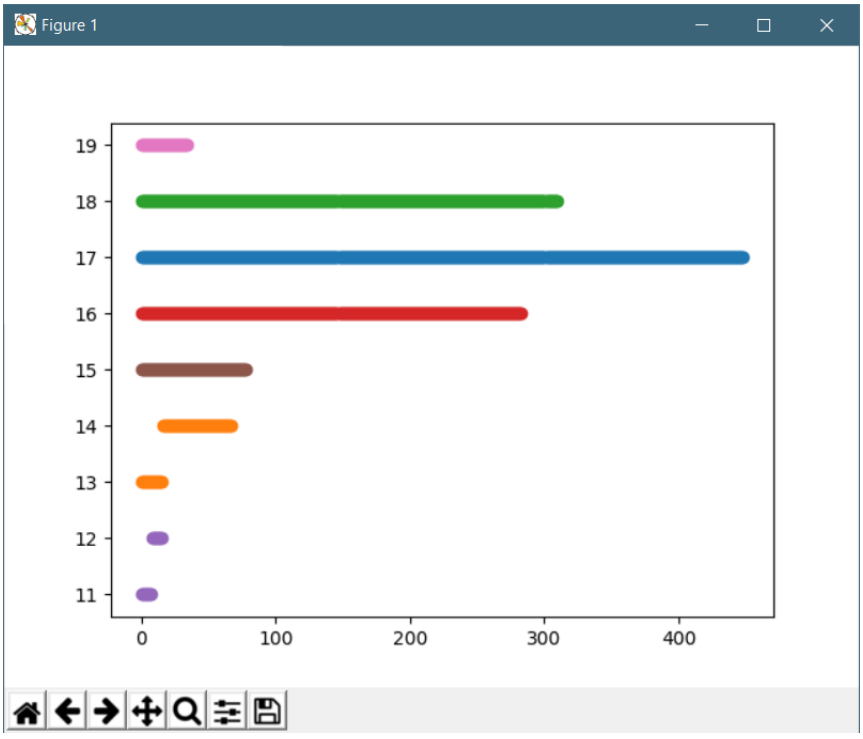
Clusters for Nose Length



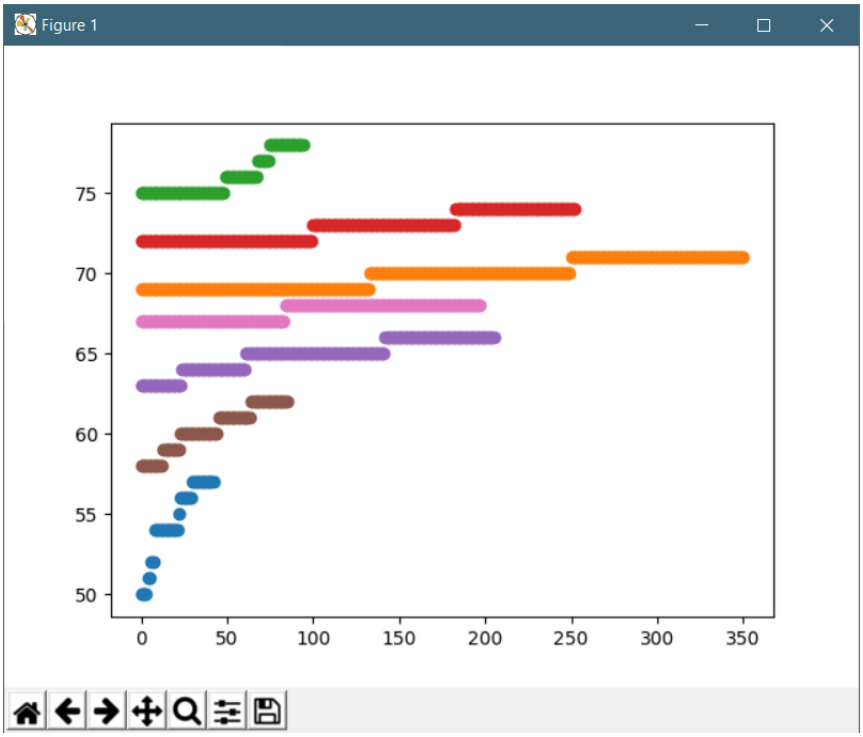
Clusters for Nose Width



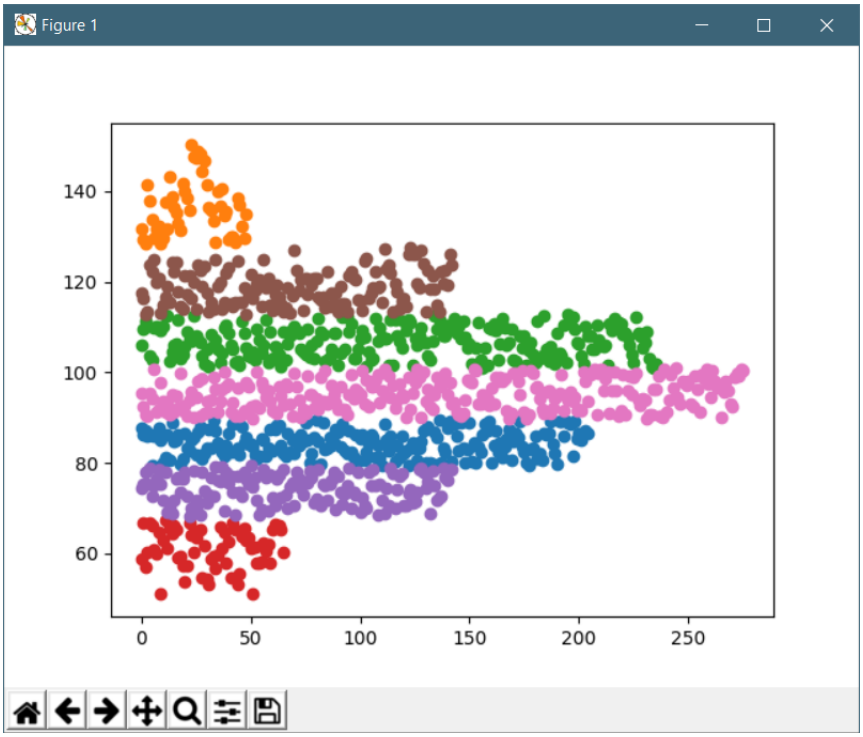
Clusters for Eye Length



Clusters for Eye Width



Clusters for Luminosity value of skin



7 Future Work and Scope

7.1 Future Work

There is a lot more that can be implemented in our software. The classification of race and face shape could be implemented after obtaining a legitimate and non-biased database. Features related to lips, eye brows, forehead, etc could incorporated if possible.

To increase the accuracy, the use of machine learning techniques for identifying the features of the face could be used. The model is going to be as good as the dataset.

In our program, we could use Generative Adversarial Networks to be able to generate a sketch (or animation) of the face, for those cases where a face is not found.

The program could include different types of input, such as form based inputs and speech inputs. It could include other database manipulation tools.

Overall, with a better dataset, more training examples and more accurate preprocessing of the training examples, the neural network will perform better to give a higher accuracy.

7.2 Scope

This program can be used as an assistant to sketch artists in the criminal investigation procedure, to reduce the time required to develop a sketch by searching for the suspect from a database of images of known criminals. It can also work like a shazam for celebrities and in other similar applications where an image is to be retrieved based on its textual/verbal description.

We could also upscale the project to be able to identify objects, i.e, the user should be able to describe an object (other than just the facial features) and the program should be able to return an image from the database.

8 Conclusion

The project at this current stage can be used by government organizations and other companies to identify people from their databases without the actually needing to sit and classify the facial features. At the current state, it can give a pretty good estimate to the person being described.

The limitations of this method of identifying a person is that the return type will most likely be a set of images, and it won't be able to classify many people to different classes. Accuracy of the neural networks can definitely be improved.

9 References

[1] ImageNet Classification with Deep Convolutional Neural Networks http://vision.stanford.edu/teaching/cs231b_spring1415/slides/alexnet_tugce_kyunghee.pdf.

[2] Dataset - Color FERET http://vision.stanford.edu/teaching/cs231b_spring1415/slides/alexnet_tugce_kyunghee.pdf.

[3] Dataset - SFA Image Database http://www.sel.eesc.usp.br/sfa/.

[4] Dataset - FaceScrub http://vintage.winklerbros.net/facescrub.html.

10 Appendix

- Skin Extraction
  http://www.pyimagesearch.com/2014/08/18/skin-detection-step-step-example-usi
- AlexNet TFLearn https://github.com/tflearn/tflearn/blob/master/examples/images/alexnet.py.