Mini Project On

Face Emotion Detection

By

Sudhir Gomase (2021510018)
Sandesh Shivane (2021510063)

Under the guidance of

Internal Supervisor

Prof. Harshil Kanakia

Department of Master Of Computer Application
Sardar Patel Institute of Technology
Autonomous Institute Affiliated to Mumbai University

2020-21

CERTIFICATE OF APPROVAL

This is to certify that the following students

Sudhir Gomase (2021510018)
Sandesh Shivane (2021510063)

Have satisfactorily carried out work on the project entitled

"Face Emotion Detection"

Towards the fulfilment of project, as laid down
by
Sardar Patel Institute of Technology
during year
2021-22.

Project Guide:
Prof. Harshil Kanakia

PROJECT APPROVAL CERTIFICATE

This is to certify that the following students

Sudhir Gomase (2021510018)
Sandesh Shivane (2021510063)

Have successfully completed the Project report on

"Face Emotion Detection",

which is found to be satisfactory and is approved

at

SARDAR PATEL INSTITUTE OF TECHNOLOGY,
ANDHERI (W), MUMBAI

INTERNAL EXAMINER EXTERNAL EXAMINER

HEAD OF DEPARTMENT PRINCIPAL

Contents

Abstract **i**

List Of Figures **ii**

1 Introduction **1**
1.1 Problem Definition 1
1.2 Objectives and Scope 2
1.2.1 Objectives 2
1.2.2 Scope 2

2 Software Requirement Specification (SRS) and Design **3**
2.1 Purpose 3
2.2 Definition 3

3 Motivation **4**

4 CONVOLUTIONAL NEURAL NETWORKS **5**
4.1 The CNN concept 5
4.2 Convolution operation 6
4.3 Pooling operation 6
4.4 Fully connected layer 7
4.5 Dropout 7
4.6 Batch normalization 8
4.7 Activation functions 8

5 Software Requirement **9**
5.1 Hardware Interfaces 9

6 Proposed method MODULES **10**
6.1 Dataset module 10
6.2 Pre-processing module 10
6.3 Training module 10
6.4 Face recognition module 11
6.5 Expression recognition module 11

7 Dataset Used **12**

8 Frontend in Streamlit **13**

9 Result and Output **14**

10 Conclusion **19**

11 Future scope **20**

12 References **21**

13 Code **22**
13.1 training.py 22
13.2 app.py 26

Sudhir Gomase (2021510018)
Sandesh Shivane (2021510063)

Abstract

The Human facial expressions convey a lot of information visually rather than articulately. Facial expression recognition plays a crucial role in the area of human-machine interaction. Automatic facial expression recognition system has many applications including, but not limited to, human behavior understanding, detection of mental disorders, and synthetic human expressions. Recognition of facial expression by computer with high recognition rate is still a challenging task.

The objective of this project is to develop Automatic Facial Expression Recognition System which can take human facial images containing some expression as input and recognize and classify different expression

We applied various deep learning methods (convolutional neural networks) to identify the key five human emotions: anger, happiness, sadness, surprise and neutrality.

List of Figures

1.2.1Emotions 2
2.2.1Steps for FER 3
3.0.1Picture of Mona Lisa 4
3.0.2Picture of deaf and dumb 4
4.0.1A fully connected NN 5
4.1.1The CNN concept 6
4.3.1Max and average pooling outputs for an image 7
4.5.1Dropout in a NN 8
6.1.1Dataset 10
6.3.1Training Model 11
9.0.1Home page 14
9.0.2About Project page 14
9.0.3About Us 15
9.0.4Live Feed 15
9.0.5Neutral face Recognition 16
9.0.6Happy face Recognition 16
9.0.7Sad face Recognition 17
9.0.8Angry face Recognition 17
9.0.9Surprise face Recognition 18

Sudhir Gomase (2021510018)
Sandesh Shivane (2021510063)

1 Introduction

1.1 Problem Definition

Human facial expressions can be easily classified into 5 basic emotions: happy, sad, surprise, anger and neutral. Our facial emotions are expressed through activation of specific sets of facial muscles. These sometimes subtle, yet complex, signals in an expression often contain an abundant amount of information about our state of mind. Through facial emotion recognition, we are able to measure the effects that content and services have on the audience/users through an easy and low-cost procedure. For example, retailers may use these metrics to evaluate customer interest. Healthcare providers can provide better service by using additional information about patients' emotional state during treatment. Entertainment producers can monitor audience engagement in events to consistently create desired content.

Humans are well-trained in reading the emotions of others, in fact, at just 14 months old, babies can already tell the difference between happy and sad. But can computers do a better job than us in accessing emotional states? To answer the question, We designed a deep learning neural network that gives machines the ability to make inferences about our emotional states. In other words, we give them eyes to see what we can see.

1.2 Objectives and Scope

1.2.1 Objectives

The objective of this project is to develop Automatic Facial Expression Recognition System which can take human facial images containing some expression as input and recognize and classify it into five different expression class such as :

- Happy
- Sad
- Surprise
- Angry
- Neutral

1.2.2 Scope

We would also like to train more databases into the system to make the model more and more accurate but again resources becomes a hindrance in the path and we also need to improve in several areas in future to resolve the errors and improve the accuracy

1.2.1: Emotions

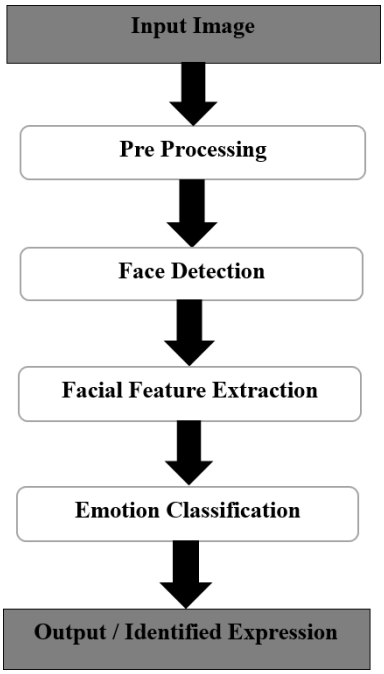
2 Software Requirement Specification (SRS) and Design

2.1 Purpose

The purpose of our project is to develop an UI application that can help user to detect the facial expressions. The facial expression include Happiness, Sadness, Anger, Surprise, Neutral
This system can be used in any other video calling system where by making some changes the user at other side can see the type of expression.

2.2 Definition

To build a System which can help user or any organization to detect the expressions of the people.



2.2.1: Steps for FER

3 Motivation

Significant debate has risen in past regarding the emotions portrayed in the world famous masterpiece of Mona Lisa. British Weekly "New Scientist" has stated that she is in fact a blend of many different emotions, 83 percent happy, 9 percent disgusted, 6 percent fearful,2 percent angry.

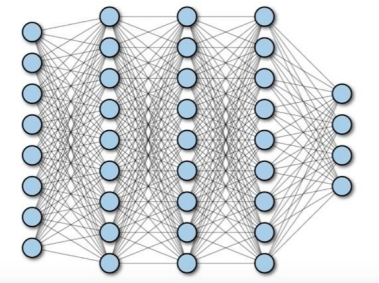
3.0.1: Picture of Mona Lisa

We have also been motivated observing the benefits of physically handicapped people like deaf and dumb. But if any normal human being or an automated system can understand their needs by observing their facial expression then it becomes a lot easier for them to make the fellow human or automated system understand their needs.

3.0.2: Picture of deaf and dumb

4 CONVOLUTIONAL NEURAL NETWORKS

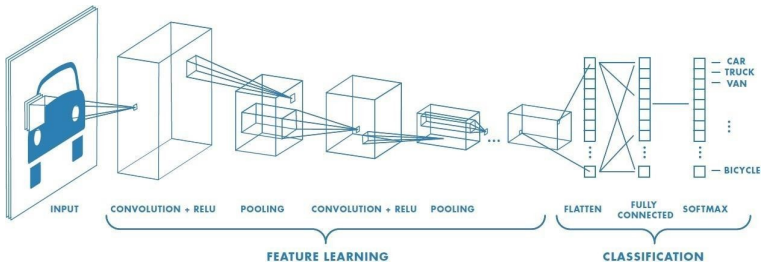
In deep learning, a convolutional neural network (CNN/ConvNet) is a class of deep neural networks, most commonly applied to analyze visual imagery. Now when we think of a neural network we think about matrix multiplications but that is not the case with ConvNet. It uses a special technique called Convolution.Convolutional neural networks are composed of multiple layers of artificial neurons. Artificial neurons, a rough imitation of their biological counterparts, are mathematical functions that calculate the weighted sum of multiple inputs and outputs an activation value. When you input an image in a ConvNet, each layer generates several activation functions that are passed on to the next layer.



4.0.1: A fully connected NN

4.1 The CNN concept

A CNN is a DL algorithm which takes an input image, assigns importance (learnable weights and biases) to various aspects/objects in the image and is able to differentiate between images. The preprocessing required in a CNN is much lower than other classification algorithms. Figure below shows the CNN operations. The architecture of a CNN is analogous to that of the connectivity pattern of neurons in the human brain and was inspired by the organization of the visual cortex. One role of a CNN is to reduce images into a form which is easier to process without losing features that are critical for good prediction. This is important when designing an architecture which is not only good at learning features but also is scalable to massive datasets. The main CNN operations are convolution, pooling, batch normalization and dropout which are described below.



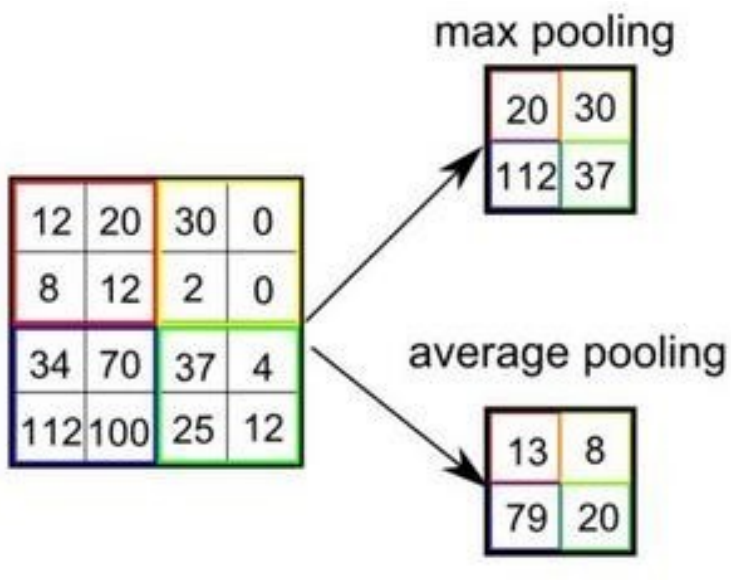
4.1.1: The CNN concept

4.2 Convolution operation

The objective of the convolution operation is to extract high level features such as edges from an input image. The convolution layer functions are as follows.
i. The first convolutional layer(s) learns features such as edges, color, gradient orientation and simple textures.
ii.The next convolutional layer(s) learns features that are more complex textures and patterns.
iii.The last convolutional layer(s) learns features such as objects or parts of objects.
The element involved in carrying out the convolution operation is called the kernel. A kernel filters everything that is not important for the feature map, only focusing on specific information. The filter moves to the right with a certain stride length till it parses the complete width. Then, it goes back to the left of the image with the same stride length and repeats the process until the entire image is traversed.The convolved feature can have the same dimensions as the input or the kernel. This is done by same or valid padding. Same padding is when the convolved feature has the dimensions of the input image and valid padding is when this feature has the dimensions of the kernel. .

4.3 Pooling operation

The pooling layer reduces the spatial size of a convolved feature. This is done to decrease the computations required to process the data and extract dominant features which are rotation and position invariant. There are two types of pooling, namely max pooling and average pooling. Max pooling returns the maximum value from the portion of the image covered by the kernel, while average pooling returns the average of the corresponding values. Figure below shows the outputs obtained by performing max and average pooling on an image.



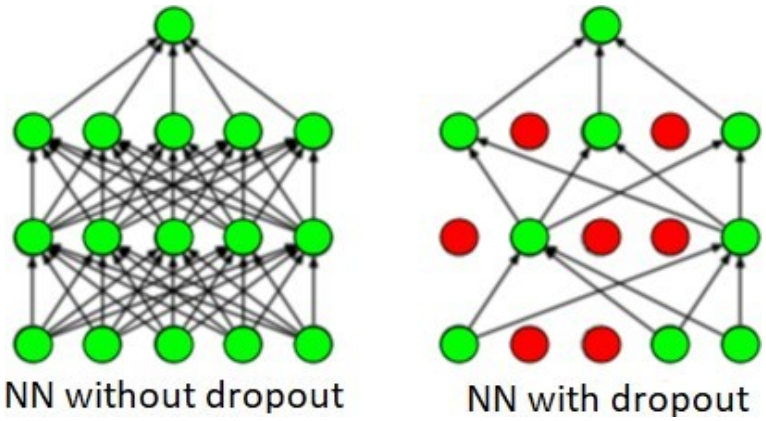
4.3.1: Max and average pooling outputs for an image

4.4 Fully connected layer

Neurons in a fully connected layer have connections to all neurons in the previous layer. This layer is found towards the end of a CNN. In this layer, the input from the previous layer is flattened into a one-dimensional vector and an activation function is applied to obtain the output.

4.5 Dropout

Dropout is used to avoid overfitting. Overfitting in an ML model happens when the training accuracy is much greater than the testing accuracy. Dropout refers to ignoring neurons during training so they are not considered during a particular forward or backward pass leaving a reduced network. These neurons are chosen randomly and an example is shown in Figure below. The dropout rate is the probability of training a given node in a layer, where 1.0 means no dropout and 0.0 means all outputs from the layer are ignored.



4.5.1: Dropout in a NN

4.6 Batch normalization

Training a network is more efficient when the distributions of the layer inputs are the same. Variations in these distributions can make a model biased. Batch normalization is used to normalize the inputs to the layers.

4.7 Activation functions

Activation functions are the most crucial part of any neural network in deep learning. In deep learning, very complicated tasks are image classification, language transformation, object detection, etc which are needed to address with the help of neural networks and activation function.
Activation function defines the output of input or set of inputs or in other terms defines node of the output of node that is given in inputs.

They basically decide to activate or deactivate neurons to get the desired output. It also performs a nonlinear transformation on the input to get better results on a complex neural network.

Activation function also helps to normalize the output of any input in the range between 1 to -1. Activation function must be efficient and it should reduce the computation time because the neural network sometimes trained on millions of data points.

5 Software Requirement

Keras

Keras is an open-source high-level Neural Network library, which is written in Python is capable enough to run on Theano, TensorFlow, or CNTK. It was developed by one of the Google engineers, Francois Chollet. It is made user-friendly, extensible, and modular for facilitating faster experimentation with deep neural networks. It not only supports Convolutional Networks and Recurrent Networks individually but also their combination.

Numpy

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

OpenCV

OpenCV is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human. When it is integrated with various libraries, such as Numpy which is a highly optimized library for numerical operations, then the number of weapons increases in your Arsenal i.e whatever operations one can do in Numpy can be combined with OpenCV.

Streamlit

Streamlit is an open source app framework in Python language. It helps us create web apps for data science and machine learning in a short time. It is compatible with major Python libraries such as scikit-learn, Keras, PyTorch, SymPy(latex), NumPy, pandas, Matplotlib etc.

5.1 Hardware Interfaces

Hardware Interfaces
1. Processor : Intel CORE i5 processor.
2. RAM : Minimum 4 GB.
3. Hard Disk : Minimum 500 GB

6 Proposed method MODULES

Modules used in our model design are:
i.Dataset module
ii. Pre-processing module
iii.Training module
iv. Face recognition module
v . Expression recognition module

6.1 Dataset module

Using dataset images for all five types of emotion differently and and storing it in different folders. Creating two seprate folders for training purpose and validation purpose

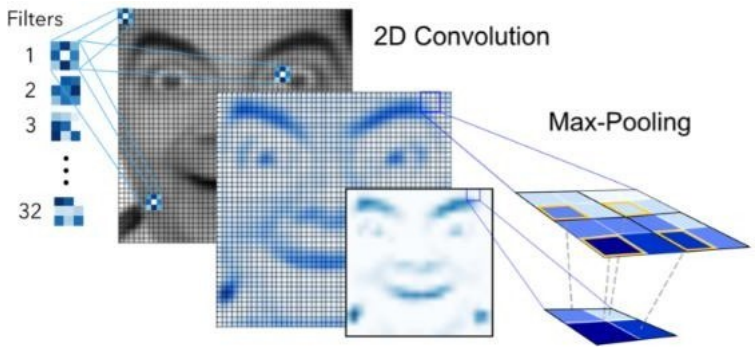
6.1.1: Dataset

6.2 Pre-processing module

Following the capture of photos, we will do image processing on the captured images. The grey scale photos will be created by converting the colour photographs to grey scale.

6.3 Training module

This step will involve the preparation of a dataset, which will consist of a binary array of all the photographs that have been taken. The collected photographs will be saved in a.YML file, which will contain all of the face data that was obtained. The YML file allows us to process the collected photos more quickly because of its compressed nature.



6.3.1: Training Model

6.4 Face recognition module

The first phase in the face recognition process is to train the host system on the facial data that has been collected. The face is photographed using the web camera on the computer system, which captures 60 different photos of the subject's face . In this session, we will learn how to detect people's faces using the LBP algorithm. The abbreviation LBPH stands for local binary pattern histogram. With the face ID and NAME that were previously stored, it will recognise the faces in the database.

6.5 Expression recognition module

Facial expression recognition software is a system that detects emotions in human faces by using biometric indicators . Because it collects and analyses information from images, it is possible to offer an unfiltered, unbiased emotional reaction or data that is unfiltered and impartial.

7 Dataset Used

Several public databases were used in order to assess face expression recognition algorithms: Frontal face dataset from Haarcascade: HaarCascade Classifier is used to recognise faces in pictures utilising characteristics. The frontal face is detected using the haarcascadefrontalface default.xml. It was created by Viola and Jones in response to a proposal made in 1998 by Papa Georgiou et al. To verify that the retrieved faces are all in the same location, we utilised an additional classifier named 'haarcascade eye.xml' from the same OpenCV library. This identifies the region around the eyes and then adjusts the left and right borders of the face window to maintain an equal distance between the eyes and the sides of the face. Thus, superfluous information (such as hair, ears, and background) is removed, and the retrieved faces have their locations adjusted.FER2013 dataset: FER2013 is an open-source dataset generated by Pierre-Luc Carrier and Aaron Courville for an ongoing project and later given publicly for a Kaggle competition. The FER2013 database was launched during the 2013 International Conference on Machine Learning's Challenges in Representation Learning. FER2013 is a massive and unrestricted database that was automatically compiled using the Google image search API. After rejecting incorrectly labelled frames and modifying the cropped region, all photos have been registered and resized to 48*48 pixels. This dataset contains 35,887 grayscale, 48x48-pixel pictures of faces displaying a range of emotions -5 emotions, all labeled-.

Emotion labels in the dataset:

0: -3993 images- Angry

1: -7164 images- Happy

2: -4982 images- Neutral

3: -4938 images- Sad

4: -3205 images- Surprise

8 Frontend in Streamlit

Streamlit is an open source app framework in Python language. It helps us create web apps for data science and machine learning in a short time. It is compatible with major Python libraries such as scikit-learn, Keras, PyTorch, SymPy(latex), NumPy, pandas, Matplotlib etc. With Streamlit, no callbacks are needed since widgets are treated as variables. Data caching simplifies and speeds up computation pipelines. Streamlit watches for changes on updates of the linked Gitrepository and the application will be deployed automatically in the shared link.

Created different modules in drop down menu:

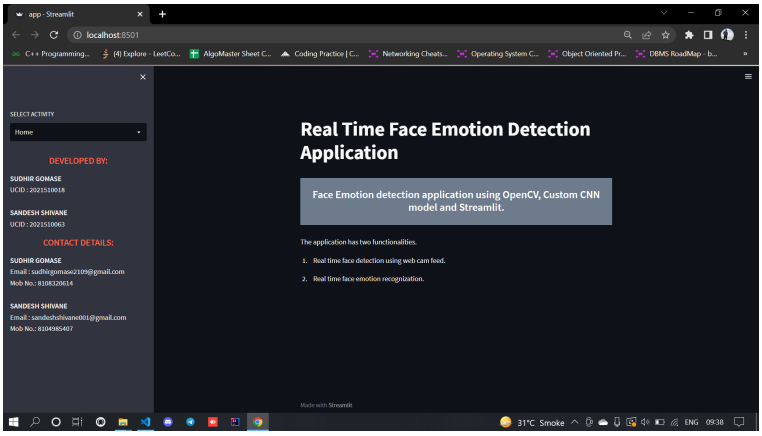
1. Home page: Intro to the application

2. Project Info: Information related to the project and its Implementation

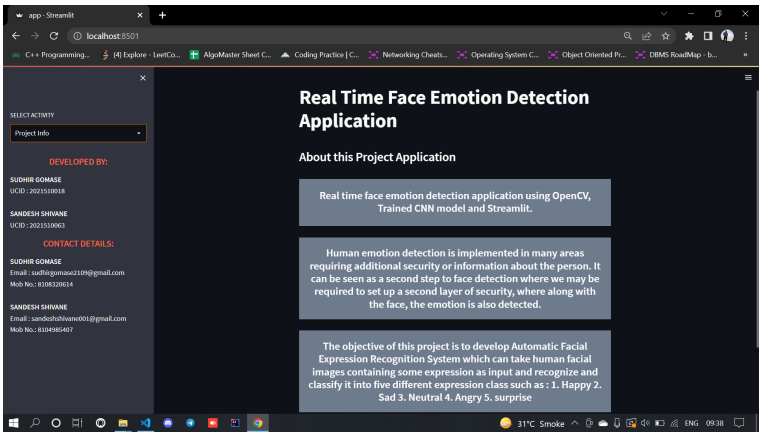
3. Web Live Feed: This module consists of web cam feed where cam detects your face emotion and shows on your screen

4. About Us: Info about us and the project.

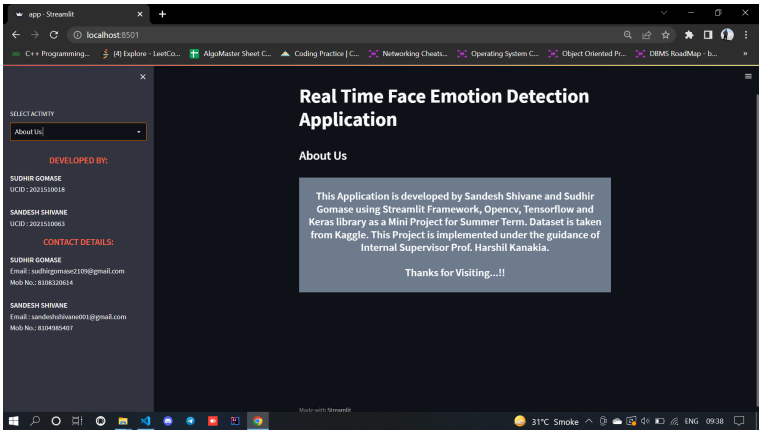
9 Result and Output



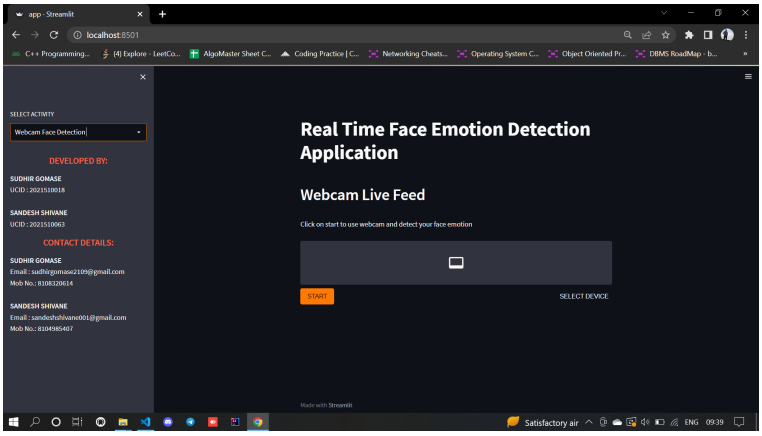
9.0.1: Home page



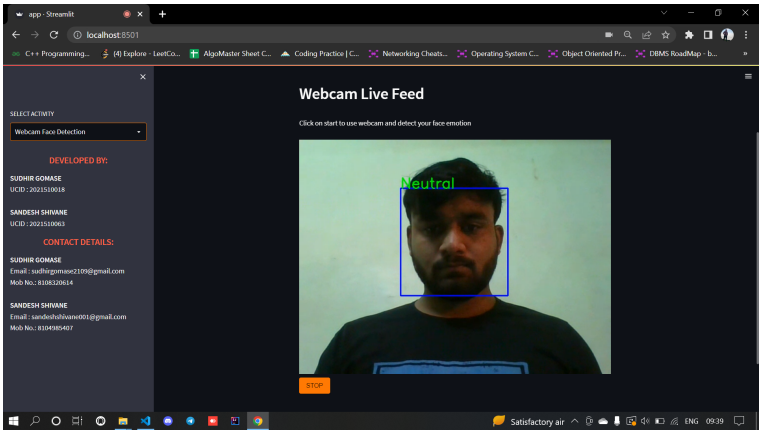
9.0.2: About Project page



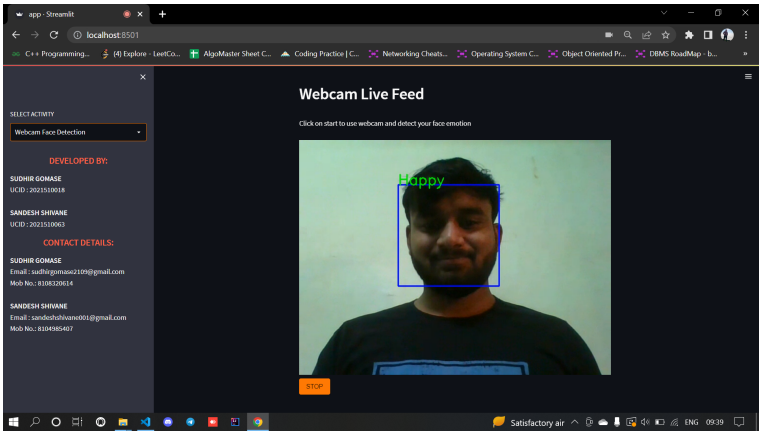
9.0.3: About Us



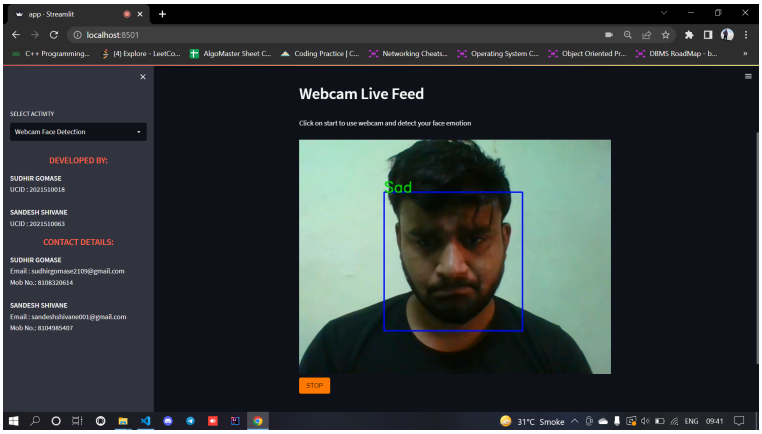
9.0.4: Live Feed



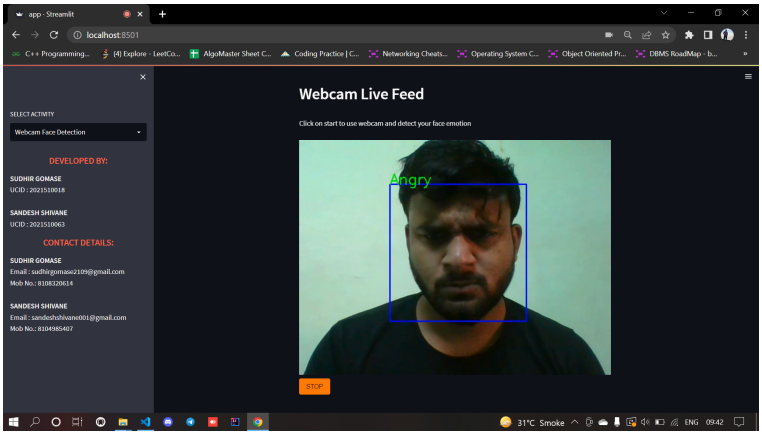
9.0.5: Neutral face Recognition



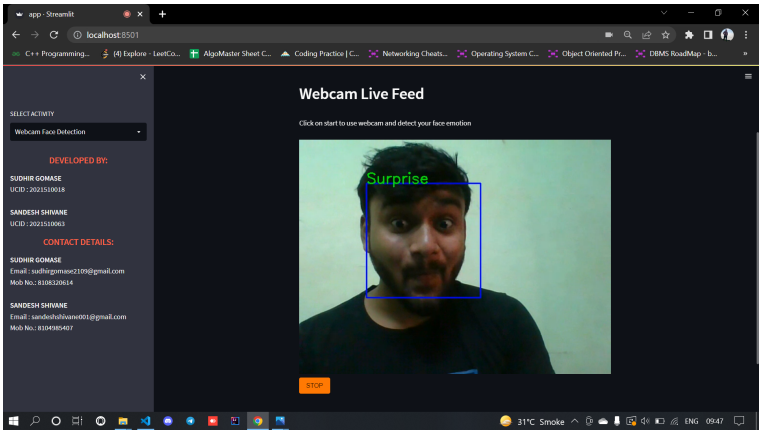
9.0.6: Happy face Recognition



9.0.7: Sad face Recognition



9.0.8: Angry face Recognition



9.0.9: Surprise face Recognition

10 Conclusion

In this case, when the model predicts incorrectly, the correct label is often the second most likely emotion.

The facial expression recognition system presented in this research work contributes a resilient face recognition model based on the mapping of behavioral characteristics with the physiological biometric characteristics. The physiological characteristics of the human face with relevance to various expressions such as happiness, sadness, fear, anger, surprise and disgust are associated with geometrical structures which restored as base matching template for the recognition system.

The behavioral aspect of this system relates the attitude behind different expressions as property base. The property bases are alienated as exposed and hidden category in genetic algorithmic genes. The gene training set evaluates the expressional uniqueness of individual faces and provide a resilient expressional recognition model in the field of biometric security.

The design of a novel asymmetric cryptosystem based on biometrics having features like hierarchical group security eliminates the use of passwords and smart cards as opposed to earlier cryptosystems. It requires a special hardware support like all other biometrics system. This research work promises a new direction of research in the field of asymmetric biometric cryptosystems which is highly desirable in order to get rid of passwords and smart cards completely. Experimental analysis and study show that the hierarchical security structures are effective in geometric shape identification for physiological traits.

11 Future scope

It is important to note that there is no specific formula to build a neural network that would guarantee to work well. Different problems would require different network architecture and a lot of trail and errors to produce desirable validation accuracy. This is the reason why neural nets are often perceived as "black box algorithms.".

In this project we got an accuracy of almost 70percent which is not bad at all comparing all the previous models. But we need to improve in specific areas like-
number and configuration of convolutional layers
number and configuration of dense layers
dropout percentage in dense layers

But due to lack of highly configured system we could not go deeper into dense neural network as the system gets very slow and we will try to improve in these areas in future.We would also like to train more databases into the system to make the model more and more accurate but again resources becomes a hindrance in the path and we also need to improve in several areas in future to resolve the errors and improve the accuracy.

Having examined techniques to cope with expression variation, in future it may be investigated in more depth about the face classification problem and optimal fusion of color and depth information. Further study can be laid down in the direction of allele of gene matching to the geometric factors of the facial expressions. The genetic property evolution framework for facial expressional system can be studied to suit the requirement of different security models such as criminal detection, governmental confidential security breaches etc.

12 References

1. https://towardsdatascience.com/face-detection-for-beginners-e58e8f21aad9

2. https://thinkingneuron.com/face-recognition-using-deep-learning-cnn-in-python/

3. https://pysource.com/2021/08/16/face-recognition-in-real-time-with-opencv-and-python/

4. https://www.loginradius.com/blog/engineering/guest-post/opencv-web-app-with-streamlit/

5. https://www.r-bloggers.com/2022/09/training-and-testing-data-in-machine-learning/

6. https://towardsdatascience.com/cnn-based-face-detector-from-dlib-c3696195e01c

7. https://towardsdatascience.com/understand-the-architecture-of-cnn-90a25e244c7

8. https://www.analyticsvidhya.com/blog/2021/06/build-web-app-instantly-for-machine-learning-using-streamlit/

13 Code

13.1 training.py

```
    from __future__ import print_function
import keras
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten, BatchNormalization
from keras.layers import Conv2D, MaxPooling2D
import os

num_classes = 5
img_rows, img_cols = 48,48
batch_size = 32

train_data_dir = r'C:\Users\Sandesh\Desktop\face\train'
validation_data_dir = r'C:\Users\Sandesh\Desktop\face\validation'

train_datagen = ImageDataGenerator(
                        rescale=1./255,
                        rotation_range=30,
                        shear_range=0.3,
                        zoom_range=0.3,
                        width_shift_range=0.4,
                        height_shift_range=0.4,
                        horizontal_flip=True,
                        fill_mode='nearest')

validation_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
                        train_data_dir,
                        color_mode='grayscale',
                        target_size=(img_rows, img_cols),
                        batch_size=batch_size,
                        class_mode='categorical',
                        shuffle=True)

validation_generator = validation_datagen.flow_from_directory(
                                validation_data_dir,
                                color_mode='grayscale',
                                target_size=(img_rows, img_cols),
                                batch_size=batch_size,
                                class_mode='categorical',
                                shuffle=True)
```

```
model = Sequential()

# Block-1

model.add(Conv2D(32,(3,3),padding='same',kernel_initializer='he_normal',
input_shape=(img_rows,img_cols,1)))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(Conv2D(32,(3,3),padding='same',kernel_initializer='he_normal',
input_shape=(img_rows,img_cols,1)))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2))

# Block-2

model.add(Conv2D(64,(3,3),padding='same',kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(Conv2D(64,(3,3),padding='same',kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2))

# Block-3

model.add(Conv2D(128,(3,3),padding='same',kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(Conv2D(128,(3,3),padding='same',kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2))

# Block-4

model.add(Conv2D(256,(3,3),padding='same',kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(Conv2D(256,(3,3),padding='same',kernel_initializer='he_normal'))
model.add(Activation('elu'))
```

```
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2))

# Block-5

model.add(Flatten())
model.add(Dense(64,kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

# Block-6

model.add(Dense(64,kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

# Block-7

model.add(Dense(num_classes,kernel_initializer='he_normal'))
model.add(Activation('softmax'))

print(model.summary())

from keras.optimizers import RMSprop,SGD,Adam
from keras.callbacks import ModelCheckpoint,
EarlyStopping, ReduceLROnPlateau

checkpoint = ModelCheckpoint(r'C:\Users\Sandesh\Desktop\face\pretraining.h5',
                             monitor='val_loss',
                             mode='min',
                             save_best_only=True,
                             verbose=1)

earlystop = EarlyStopping(monitor='val_loss',
                          min_delta=0,
                          patience=3,
                          verbose=1,
                          restore_best_weights=True
                          )

reduce_lr = ReduceLROnPlateau(monitor='val_loss',
                              factor=0.2,
                              patience=3,
```

```
                                verbose=1,
                                min_delta=0.0001)

callbacks = [earlystop,checkpoint,reduce_lr]

model.compile(loss='categorical_crossentropy',
              optimizer = Adam(lr=0.001),
              metrics=['accuracy'])

train_samples = 24176
validation_samples = 3006
epochs=25

history=model.fit_generator(
                train_generator,
                steps_per_epoch=train_samples//batch_size,
                epochs=epochs,
                callbacks=callbacks,
                validation_data=validation_generator,
                validation_steps=validation_samples//batch_size)
```

13.2 app.py

```
import numpy as np
import cv2
import streamlit as st
from tensorflow import keras
from keras.models import load_model
from tensorflow.keras.utils import img_to_array
from streamlit_webrtc import webrtc_streamer, VideoTransformerBase

# load model

emotion_dict = ['Angry','Happy','Neutral','Sad','Surprise']



classifier =load_model('pretraining.h5')

classifier.load_weights("pretraining.h5")

#load face
try:
    face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
except Exception:
    st.write("Error loading cascade classifiers")

class VideoTransformer(VideoTransformerBase):
    def transform(self, frame):
        img = frame.to_ndarray(format="bgr24")

        #image gray
        img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(
            image=img_gray, scaleFactor=1.3, minNeighbors=5)
        for (x, y, w, h) in faces:
            cv2.rectangle(img=img, pt1=(x, y), pt2=(
                x + w, y + h), color=(255, 0, 0), thickness=2)
            roi_gray = img_gray[y:y + h, x:x + w]
            roi_gray = cv2.resize(roi_gray, (48, 48),
            interpolation=cv2.INTER_AREA)
            if np.sum([roi_gray]) != 0:
                roi = roi_gray.astype('float') / 255.0
                roi = img_to_array(roi)
                roi = np.expand_dims(roi, axis=0)
                prediction = classifier.predict(roi)[0]
```

```
                maxindex = int(np.argmax(prediction))
                finalout = emotion_dict[maxindex]
                output = str(finalout)
            label_position = (x, y)
            cv2.putText(img, output, label_position,
            cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)

        return img

def main():
    # Face Analysis Application #
    st.title("Real Time Face Emotion Detection Application")
    activiteis = [ "Home", "Project Info", "Webcam Face Detection", "About Us"]
    choice = st.sidebar.selectbox("SELECT ACTIVITY", activiteis)
    info="""<div><h2 style="text-align:center; color:#FF6347;">
    DEVELOPED BY:</h2>
            <b>SUDHIR GOMASE </b><br>
            UCID : 2021510018 <br><br>
            <b>SANDESH SHIVANE </b><br>
            UCID : 2021510063 </div>"""
    st.sidebar.markdown(info, unsafe_allow_html=True)

    contact="""<div><h2 style="text-align:center; color:#FF6347;">
    CONTACT DETAILS:</h2>
            <b>SUDHIR GOMASE </b><br>
            Email : sudhirgomase2109@gmail.com <br>
            Mob No.: 8108320614<br><br>
            <b>SANDESH SHIVANE </b><br>
            Email : sandeshshivane001@gmail.com <br>
            Mob No.: 8104985407</div>"""
    st.sidebar.markdown(contact, unsafe_allow_html=True)

    if choice == "Home":
        html_temp_home1 = """<div style="background-color:#6D7B8D;padding:10px">
                <h4 style="color:white;text-align:center;">
                Face Emotion detection application using OpenCV,
                Custom CNN model and Streamlit.</h4>
                    </div>
                    </br>"""
        st.markdown(html_temp_home1, unsafe_allow_html=True)
        st.write("""
                The application has two functionalities.

                1. Real time face detection using web cam feed.

                2. Real time face emotion recognization.
```

```
            """)
    elif choice == "Webcam Face Detection":
        st.header("Webcam Live Feed")
        st.write("Click on start to use webcam and detect your face emotion")
        webrtc_streamer(key="example",
        video_transformer_factory=VideoTransformer)


    elif choice == "Project Info":
        st.subheader("About this Project Application")


        # from PIL import Image
        # image = Image.open('lisa.jpg')

        # st.image(image, caption='Sunrise by the mountains', width=400)

        pro= """<div style="background-color:#6D7B8D;padding:10px">
                  <h4 style="color:white;text-align:center;">
                  Real time face emotion detection application using OpenCV,
                  Trained CNN model and Streamlit.</h4>
                 </div>
                 </br>"""
        st.markdown(pro, unsafe_allow_html=True)

        proinfo1 = """
            <div style="background-color:#6D7B8D;padding:10px">
            <h4 style="color:white;text-align:center;padding:10px"">Human
            emotion detection is implemented in many areas requiring additional
            security or information about the person. It can be seen as a
            second step to face detection where we may be required to
            set up a second layer of security, where along with the
            face, the emotion is also  detected. </div>
            <br>"""

        st.markdown(proinfo1, unsafe_allow_html=True)

        proinfo2 = """
            <div style="background-color:#6D7B8D;padding:10px">
            <h4 style="color:white;text-align:center;padding:10px"">The
            objective of this project is to develop Automatic Facial
            Expression Recognition System which can take human facial images
            containing some expression as input and recognize and
            classify it into five different expression class such as :
            1. Happy 2. Sad 3. Neutral 4. Angry 5. surprise </div> <br>"""
```

```
        st.markdown(proinfo2, unsafe_allow_html=True)


    elif choice == "About Us":

        st.subheader("About Us")


        ab1 = """
            <div style="background-color:#6D7B8D; padding:15px">
            <h4 style="color:white;text-align:center;">This Application
            is developed by Sandesh Shivane and Sudhir Gomase using
            Streamlit Framework, Opencv, Tensorflow and Keras library
            as a Mini Project for Summer Term. Dataset is taken from
            Kaggle. This Project is implemented under the guidance of
            Internal Supervisor Prof. Harshil Kanakia.</h4>

            <h4 style="color:white;text-align:center;">
            Thanks for Visiting...!!</h4>
            </div>
            <br></br>
            <br></br>"""

        st.markdown(ab1, unsafe_allow_html=True)

    else:
        pass


if __name__ == "__main__":
    main()
```