Facial Expression Recognition

Team Members:

Omar Ismail(Leader)

Vineeth Reddy Sheri

Piyush Narhire

Story

A Human being can easily sense another person's emotion, but it is a very difficult task for the machines to understand. Person emotions consist of Fear, Anger, Happy, Surprise, Sad, Neutral and Disgust. If a machine was able to detect these emotions and take necessary actions it would be a huge success like preventing people from making some rash decisions, enlighten the mood. For instance, in a classroom environment, an emotion detection model will help in generating a report of students' emotions which will help the lecturer to improve the upcoming lecturers to attract more students.

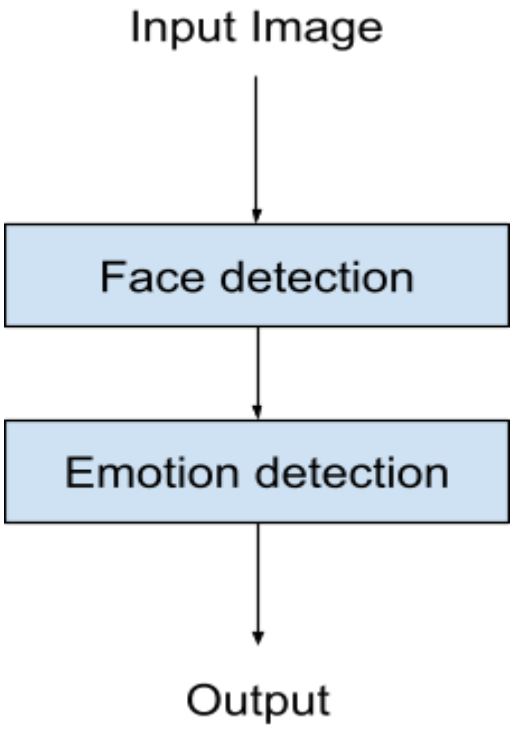
An web-based application where the user can either upload the image from the file system or take a photo using the webcam. Then gonna play music according to the user's mood. For training the emotion module, we are going to make use of an open-source data set called Face Emotion Recognition (FER) from Kaggle. The model consists of two modules one is Face detection and the other is Emotion detector. Face detection will detect the faces in the image and pass these identified faces to the Emotion detector module to identify the person's emotion and will play the music based on mood detected.

Workflow

The data we utilized for the project is the FER2013. The dataset contains many facial Expressions pictures over different Expressions such as Fear, Anger, Happy, Surprised and Neutral. We are making use of Google Colab as the IDE as it provides a free GPU. Setting up the IDE to load the data FER2013 from the Kaggle. Performing preprocessing tasks on the dataset

like normalizing the data and splitting the train dataset in 4:1 ratio for training and validation phase.

The below figure is the dataflow of our application. Where the user provides the image, the provided image is first sent to the Face detection module to identify and get the coordinates of the face which is then sent to the emotion detection module. From the emotion detection module, we are gonna get the emotion of the people in the image as the output.



WorkFlow Diagram

Face detection

MTCNN(Multi-task Cascaded Convolutional Neural Networks) is used to detect the human faces in an image. And passing those identified faces to the emotion detection module. The MTCNN facial detection accuracy is very good when compared to HOG and haar cascade.

Emotion Module

For the emotion model, we came up with two different approaches. One approach is building our own custom model and another approach is by transfer learning on a pre-trained MobileNetV2 model.

Custom Model:

Defined the custom model which contains 11 convolutional layers and 5 fully connected layers. Introduced Batch Normalization layer to deal with covariate shift and dropout layer to avoid overfitting of data. Used Softmax at the output layer since dealing with more than 2 categorical data. The model contains 9,019,719 parameters out of which 9,012,295 trainable parameters. Trained the model for 100 epochs with learning rate as 0.001, Adam optimizer and loss function as categorical cross entropy. Making use of ReduceLROnPlateau callback function to reduce the learning rate by a factor of 0.9 when there is no improvement in the validation loss for 3 epochs, EarlyStopping callback function to stop the training process when the validation loss doesn't decrease for the last 6 epochs and a ModelCheckpoint callback for Storing the models trained parameters when there is a improvement in the validation loss. After training for 100 epochs,the training accuracy of the model is 92%, validation accuracy is 97% and testing accuracy is 67.5%. Lower accuracy may be because the model is overfitted.

Custom Model performance

Transfer learning MobileNetV2:

Pre-trained MobileNetV2 model for transfer learning with FER2013. MobileNetV2 model is pretrained on the ImageNet dataset, these pre-trained weights act as a starting point to train our model which enables the model to train much faster and reach the desired point very quickly. Stripping the output layer of the MobileNetV2 and using custom Softmax layers since dealing with only 7 categorical data. Trained the model for 60 epochs with learning rate as 0.001, Adam optimizer and loss function as categorical cross entropy. Making use of ReduceLROnPlateau callback function to reduce the learning rate by a factor of 0.9 when there is no improvement in the validation loss for 3 epochs, EarlyStopping callback function to stop the training process when the validation loss doesn't decrease for the last 6 epochs and a ModelCheckpoint callback for Storing the models trained parameters when there is a improvement in the validation loss. After training for 60 epochs, the validation accuracy of the model is 86% and testing accuracy is 75%.

```
[ ] mobileNet = tf.keras.applications.MobileNetV2(input_shape=(96,96,3))

    Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_96.h5
    14540800/14536120 [==============================] - 0s 0us/step
```

```
[ ] base_input = mobileNet.layers[0].input
    base_output = mobileNet.layers[-2].output
    final_output = layers.Dense(128)(base_output)
    final_output = layers.Activation('relu')(final_output)
    final_output = layers.Dense(64)(final_output)
    final_output = layers.Activation('relu')(final_output)
    final_output =layers.Dense(7,activation='softmax')(final_output)
```

```
[ ] model = keras.Model(inputs = base_input, outputs = final_output)

    learning_rate = 0.001
    adam_optimizer = optimizers.Adam(lr = learning_rate)
    epochs = 50
    model.compile(optimizer = adam_optimizer, metrics = ['accuracy'], loss = 'categorical_crossentropy')
```

```
[ ] learningRate_reducer = ReduceLROnPlateau(monitor='val_loss', patience=3, factor=0.8)
    earlyStopping = EarlyStopping(monitor='val_acc', patience=6)
    checkpointer = ModelCheckpoint('/content/gdrive/MyDrive/Project/Dataset/weights2.hd5', monitor='val_loss', verbose=1, save_best_only=True)
```

```
[ ]
```

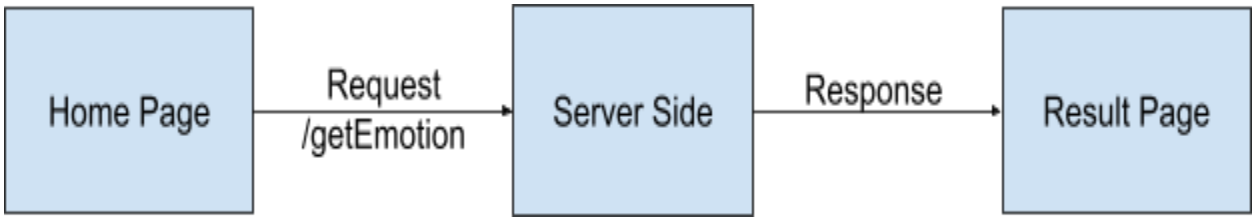
```
[ ] model.fit(
            training_set,
            epochs = epochs,
            validation_data = valid_dataset,
            callbacks=[learningRate_reducer, checkpointer, earlyStopping]
            )
```

Transfer Learning MobileNetV2 script

MobileNetV2 performance

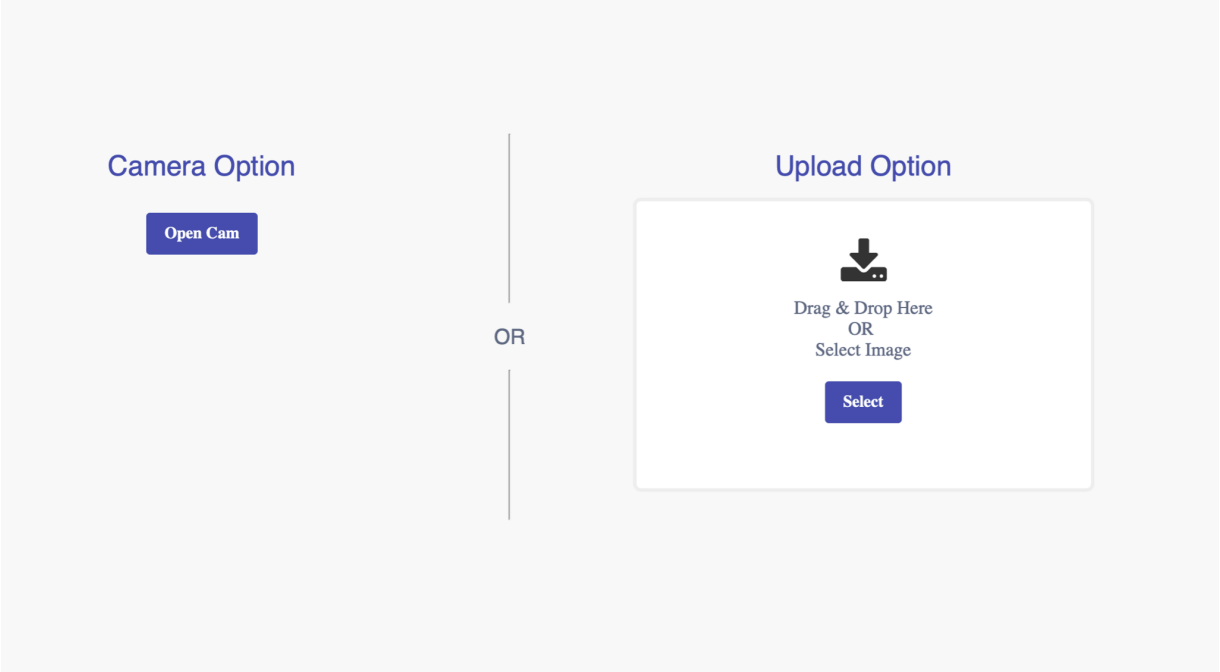
Web Application

Developed a web application using the Flask, html, css and javascript. The user will be given two options for providing the image. One option is through webcam and other through uploading the image through filesystem.



Web Application workflow

At the server side, we are gonna do previously discussed face detection and emotion identification tasks in the getEmotion api call. And display these processed images in the Result page. The below figure is the homepage of the application.



Home Page

Webcam Option:

On the click of the Open Cam button under Camera Option, the webcam will be opened. When the user is within the frame and ready to take the photo then will click on the Take Photo button. If the user is not satisfied with the previous taken image then he/she can take the photo again. Once the user is satisfied with the photo taken, then will click on the Continue Button to make an api call for detecting the emotion.

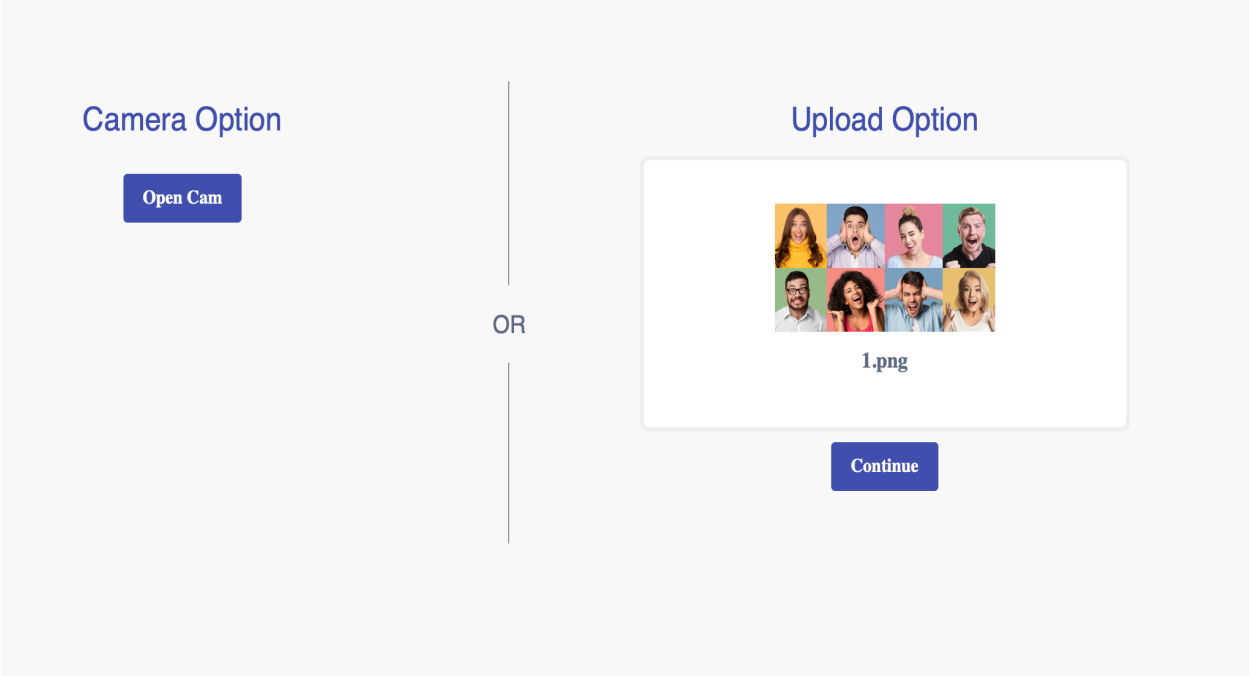
```
// To Take the photo
function captureSnapshot() {
    if(cameraStream) {
        var ctx = capture.getContext( '2d' );
        var img = new Image();
        ctx.drawImage( stream, 0, 0, capture.width, capture.height );
        img.src = capture.toDataURL( "image/png" );
        img.width = 240;
        snapshot = capture.toDataURL( "image/png" );
        closeWebcam();
        document.getElementById('cam').classList.add("hidden");
        document.getElementById('photo').classList.remove("hidden");
        document.getElementById('takePhotoBtn').classList.add("hidden");
        document.getElementById('openCamBtn').classList.remove("hidden");
        document.getElementById('submitCaptureImg').classList.remove("hidden");
    }
}
```

Capture Image script

The above script captures the images through webcam and also modifies the css classes to give the user the option to either take the photo again if he/she wishes to do so or continue with the prediction.

Upload Option:

If the user has the image then he/she can upload the image by either drag and drop or by selecting the image from the file system. The selected image will be displayed on the screen. If the user wants to change the image then he/she can do so. Once the user is ready then can click on the Continue Button for detecting the emotion.



User selected Upload Option

On the click of Continue button, we gonna make an getEmotion api call to get the person's emotion using the custom trained model. In the getEmotion method, we are initializing the MTCNN for facial detection and loading the trained custom model for emotion detection. Each identified human face by the MTCNN in the given image is passed to the emotion model. after preprocessing task like normalization and resizing to 48*48*3 to match the model's input layer dimension. After predicting the cropped facial image mood, will draw a rectangle box around the face along with the predicted label. Once every identified person's face by MTCNN is done with emotion detection, then will save the modified image that contains all the facial

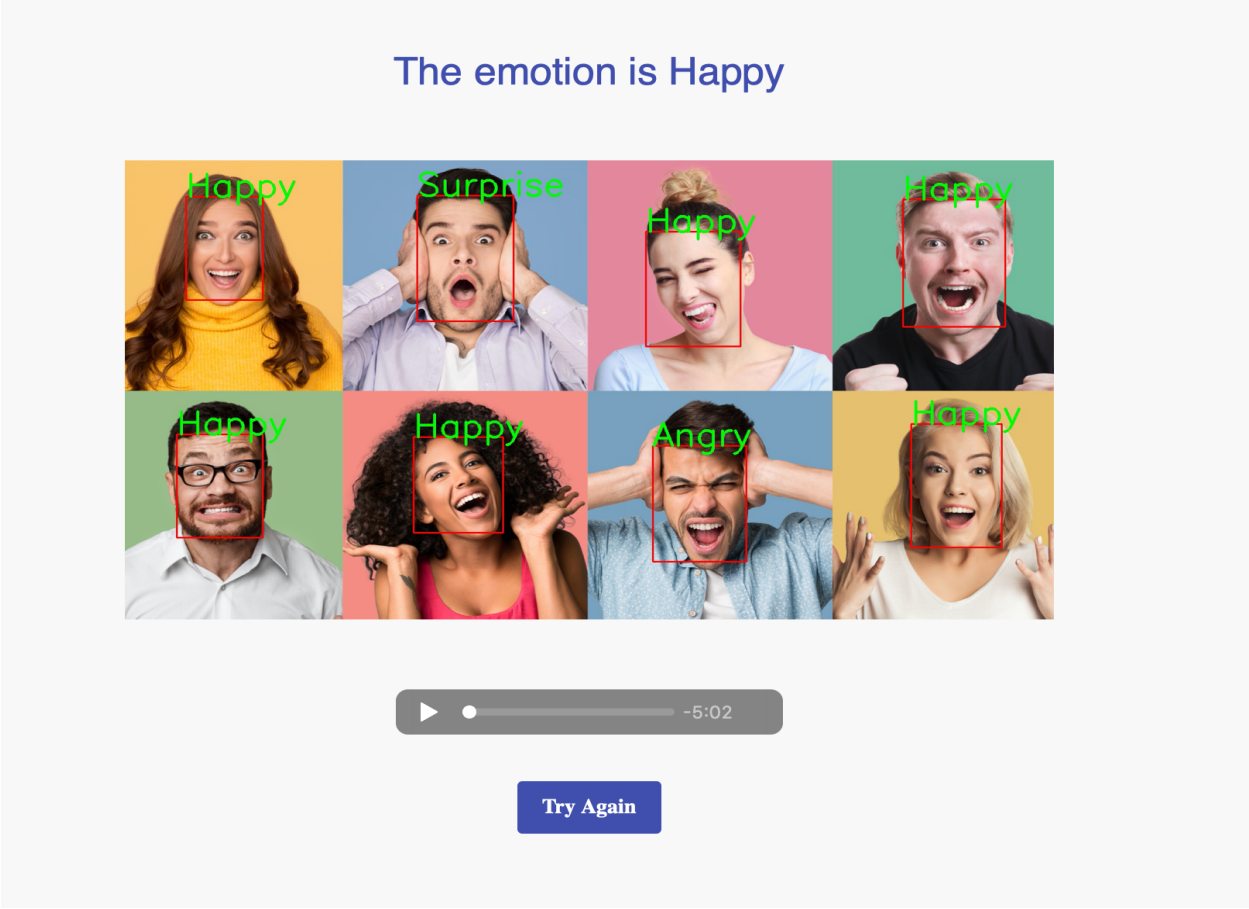
emotion along with the boundary box of every person in .jpg format. We then route to the result page.

```
@app.route("/getEmotion", methods=['GET', 'POST'])
def getEmotion():
    if request.method == "POST":
        img = request.files["fileUpload"]
        img.save('static/file.jpg')
        label_map = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']
        emotionCount = [0, 0, 0, 0, 0, 0, 0]
        global emotion, fileName
        MTCNNDetector = MTCNN()
        model = load_model('/EmotionMusic/model1.h5')
        image = cv2.imread('static/file.jpg')
        image_copy = image
        faces = MTCNNDetector.detect_faces(image_copy)
        for f in faces:
            x, y, w, h = f['box']
            x1, y1 = x + w, y + h
            cv2.rectangle(image, (x, y), (x1, y1), (0, 0, 255), 2)
            roi = image_copy[y:y + h, x:x + w]
            roi = cv2.resize(roi, (48, 48), interpolation=cv2.INTER_AREA)
            roi = roi.astype('float') / 255.0
            roi = np.expand_dims(roi, axis=0)
            predictionProb = model.predict(roi)
            prediction = np.argmax(predictionProb)
            emotionCount[prediction] += 1
            final_prediction = label_map[prediction]
            label_position = (x, y)
            cv2.putText(image, final_prediction, label_position, cv2.FONT_HERSHEY_SIMPLEX, 2, (0, 255, 0), 3)

        emotion = label_map[np.argmax(emotionCount)]
        fileName = "images/outputFile_" + str(time.time()) + ".jpg"
        cv2.imwrite("static/" + fileName, image)
        return redirect("/result")
    return redirect("/")
```

/getEmotion api call script

In the Result page, we're gonna show the previously saved image and play music corresponding to emotion. If the image that is provided by the user contains more than one person then the music that is played will be based on the majority mood. If the user likes to try again, then he/she can do so by clicking on the Try Again button which will route to the homepage.



Result Page

The application has been deployed using Heroku which provides a platform as a service (PaaS). The requirement.txt file contains all the required python libraries. Procfile file contains the command for running the web application.

pip freeze > requirement.txt command will write down all the python libraries into the requirement.txt file which the current environment is using for running the application.

Work Sharing

Omar Ismail - Used transfer learning to train model, backend development, PPT.

Vineeth Reddy Sheri - Training custom model, configuring colab to load FER2013 dataset and Web application, Report.

Piyush Narhire - Face detection module, Web deployment.

Web Page URL: https://emotion-based-music-system.herokuapp.com

Github link: https://github.com/OmarIsmail7980/Facial-Expression-Recognition

References

Kartika Candra Kirana, Facial Emotion Recognition Based on Viola-Jones Algorithm in the Learning Environment.

Ning Zhang, Research on Face Detection Technology Based on MTCNN, IEEE Conference Publication

Mehmet Akif Özdemir, Deep Learning Based Facial Emotion Recognition System, IEEE Conference Publication.

Anne Bonner, Setting Up Kaggle in Google Colab.

Ali Ghofrani, Real Time Face-Detection and Emotion Recognition Using MTCNN and miniShuffleNet V2