FACIAL EXPRESSION EMOTION RECOGNITION

Submission By:
Devangi Bedi (IT, 00215603121)
Divyansh Karakoti (IT, 10515603121)
Aarav Bamba (IT, 05315603121)

DECLARATION

I hereby declare that the project work entitled “Facial Expression Emotion Recognition” submitted is a record of an original work done by me under the guidance of Mr. Himanshu Souda, and this project work is submitted in the partial fulfillment of the requirements for the award of the certification of ‘Artificial Intelligence & Deep Learning’.

The results embodied in this project have not been submitted to any other organization for the award of any certification.

CERTIFICATE

This is to certify that Devangi Bedi, Divyansh Karakoti and Aarav Bamba have successfully completed the project ‘Facial Expression Emotion Recognition’ using Artificial Intelligence and Machine Learning. We would like to thank our trainers and teachers for guiding us throughout this project.

TEACHER’S SIGNATURE DATE:

______________________________ _________________

Mr. Himanshu Souda
IBM Expert

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to all those who contributed to the successful completion of this project. Their support, guidance, and encouragement were invaluable, and we are deeply appreciative of their efforts.

First and foremost, we are indebted to our trainer Mr. Himanshu Souda whose expertise, insightful feedback, and unwavering encouragement guided us throughout this journey. His mentorship was pivotal in shaping the direction and scope of this project.

We are grateful to Dr. Akhilesh Das Gupta Institute of Professional Studies for providing the necessary resources and facilities that supported the development of this project. Their commitment to fostering a conducive learning environment is commendable.

Thank you.

DEVANGI BEDI
AARAV BAMBA
DIVYANSH KARAKOTI

TABLE OF CONTENTS

CHAPTER 1

ABSTRACT

In today's rapidly evolving business landscape, the role of emotion recognition has emerged as a critical factor for success. As businesses strive to enhance customer experiences, streamline operations, and foster meaningful connections, understanding and interpreting human emotions has become imperative. Emotion recognition technologies offer the ability to extract invaluable insights from facial expressions, voice tones, and other non-verbal cues, enabling companies to gauge consumer sentiment, tailor their offerings, and personalize interactions. Moreover, these technologies extend beyond customer-facing aspects, aiding in employee well-being, teamwork, and decision-making by gauging sentiments within the workforce. By integrating emotion recognition into their strategies, businesses can unlock a deeper understanding of human behavior, leading to more empathetic engagement, higher satisfaction levels, and ultimately, sustained growth in today's competitive markets.

INTRODUCTION

This project first detects a face and then based on facial expressions it will detect one out of the seven emotions it has been trained for. It utilizes technologies like Tensorflow, Keras, OpenCV and Python to implement a unique ML model trained using transfer learning.

A real-time face recognition system is a technology that uses computer vision and machine learning techniques to identify and verify individuals by analyzing their facial features in real-time. These systems have a wide range of applications, including security, access control, surveillance, user authentication, and more.

MOTIVATION

Emotion recognition technology offers a unique opportunity to proactively monitor and support employees' emotional states. By accurately detecting signs of stress, anxiety, or burnout through facial expressions and other cues, the project aims to provide timely interventions and resources to individuals in need. This technology can help companies foster a culture of open communication, reduce the stigma around mental health discussions, and offer tailored assistance to employees.

Moreover, solving mental health issues in the business context aligns with broader societal goals of promoting well-being and mental health awareness. As companies embrace their responsibility to care for their employees, they contribute to a more compassionate and sustainable work culture. This project's potential to improve mental health outcomes not only benefits individuals but also enhances overall business performance, making it a compelling and socially impactful endeavor.

STATEMENT OF THE PROBLEM

To develop a machine learning model that detects facial expressions that can be used by businesses and organizations to understand the employee’s mental wellbeing while working.

FEASIBILITY STUDY

The financial feasibility assessment will focus on estimating the initial investment and potential returns. This includes costs associated with acquiring software and hardware, customization, integration, and ongoing maintenance. The study will also explore potential benefits such as improved customer satisfaction, enhanced marketing strategies, and reduced employee turnover. An analysis of the return on investment (ROI) over a defined timeframe will provide insights into the economic viability of the project.

Assessing the organization's readiness for technology integration is crucial. This section will evaluate the alignment of the facial emotion recognition system with the organization's culture, change management processes, and employee training needs. Stakeholder engagement, including management and employees, will be addressed to ensure a smooth adoption process. Considerations of how the technology supports the organization's strategic goals will be explored.

Ethical considerations surrounding data privacy, consent, and potential biases in the algorithm's predictions will be analyzed in this section. The study will identify measures to address these concerns, including implementing robust data protection practices, ensuring transparency in data usage, and mitigating algorithmic biases. Strategies to maintain compliance with relevant regulations and foster public trust will be explored.

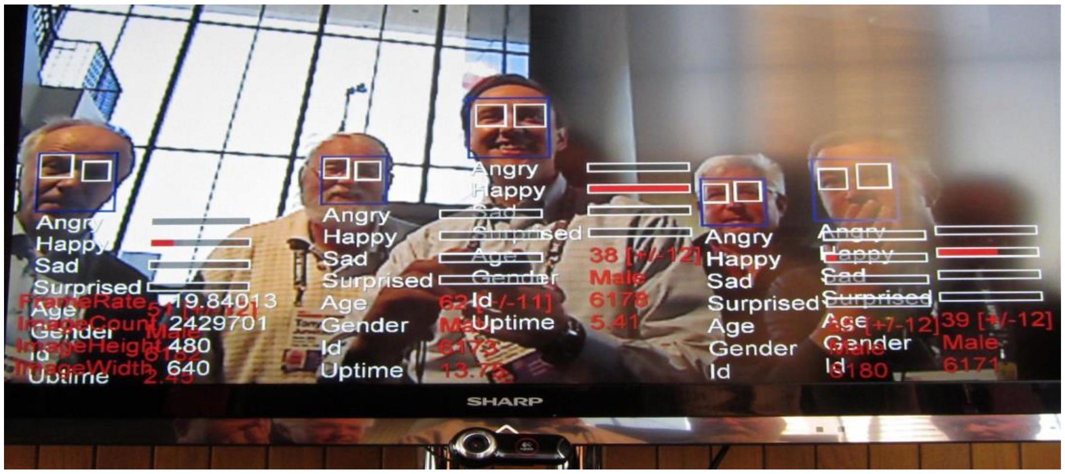
CHAPTER 2: LITERATURE REVIEW

FER2013 Dataset Overview

The FER2013 dataset (Facial Expression Recognition 2013) is a dataset that contains grayscale images of faces labeled with one of seven different facial expressions: anger, disgust, fear, happiness, sadness, surprise, and neutrality. The dataset was collected from the internet and contains images that vary in quality, lighting conditions, and pose.

Currently, the dataset contains 28709 training images & 7178 testing/validation images.

Below are some examples of different emotions a computer ML model can detect:



Our model utilizes the dataset FER2013 which consists of 7 different emotions: angry, sad, surprise, happy, neutral, disgust, and fear. However, many models are also fine-tuned to detect very specific facial expressions as well.

OpenCV Library

OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. It provides a wide range of tools and functions for real-time computer vision applications, image and video processing, object detection and recognition, machine learning, and more. OpenCV is written in C++ and supports multiple programming languages, including Python, Java, and MATLAB.
Some of the key features and functionalities of OpenCV include:

Image Processing: OpenCV offers a comprehensive set of image processing functions such as image filtering, thresholding, morphological operations, color manipulation, and more.

Video Processing: It supports capturing and processing video streams, including features like video file I/O, real-time video capturing, and video recording.

Computer Vision Algorithms: OpenCV provides various computer vision algorithms like feature detection (SIFT, SURF, ORB), object tracking, optical flow, camera calibration, and 3D reconstruction.

Machine Learning: OpenCV includes machine learning tools and models for tasks like image classification, object recognition, face detection, and gesture recognition.

Deep Learning: OpenCV integrates with deep learning frameworks like TensorFlow and PyTorch, allowing you to use pre-trained neural network models for various tasks.

GUI Components: OpenCV includes GUI components for creating graphical interfaces, displaying images and videos, and interacting with user input.

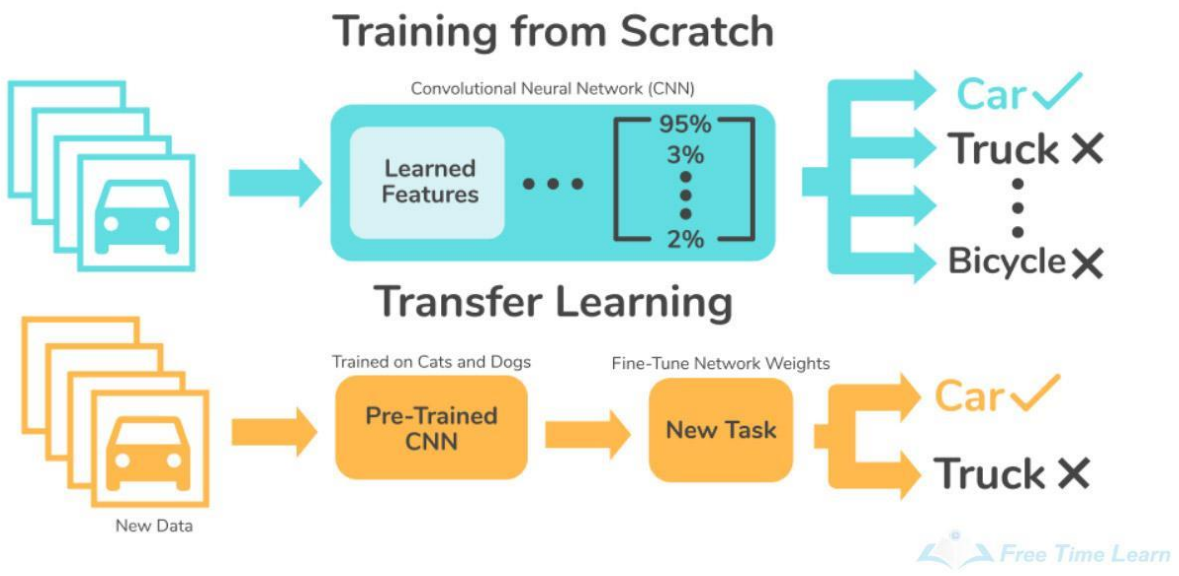
OpenCL and GPU Acceleration: It supports OpenCL and GPU acceleration for faster processing of algorithms on compatible hardware.

Cross-Platform: OpenCV is designed to work on various platforms, including Windows, macOS, Linux, Android, and iOS.

To use OpenCV in your project, you typically need to install the library and its relevant bindings for your preferred programming language. For example, if you're using Python, you can install OpenCV using the following command:

```
pip install opencv-contrib-python
```

TRANSFER LEARNING



Transfer learning is a powerful technique in the field of machine learning and computer vision that involves leveraging the knowledge gained from training a model on one task and applying it to another related task. In the context of face emotion recognition, transfer learning can be particularly beneficial, as it allows you to take advantage of pre-trained models that have already learned useful features from large datasets. Here's how you can apply transfer learning for face emotion recognition:

Select a Pre-trained Model:
Choose a pre-trained deep learning model that has been trained on a large dataset for a related task. Popular choices include Convolutional Neural Networks (CNNs) like VGG, ResNet, Inception, or MobileNet.

Data Preparation:
Prepare your face emotion recognition dataset. Make sure your dataset is properly labeled with the various facial expressions you want to recognize (e.g., happiness, sadness, anger, etc.). Preprocess the images, including resizing them to the input dimensions required by the chosen pre-trained model.

Model Adaptation:
You'll need to adapt the pre-trained model to your specific task. This typically involves modifying the last few layers of the pre-trained network to match the number of classes (emotions) you have in your dataset. The existing layers (feature extraction layers) can often be frozen to retain the knowledge they've learned. You might also need to adjust hyperparameters, such as learning rates, to fine-tune the model to your specific dataset.

Transfer Learning Strategies:
There are a few strategies for using transfer learning:

> **Feature Extraction:** Keep the pre-trained model's convolutional layers frozen and replace the classification layers with your own. Train only the new layers on your dataset.
>
> **Fine-tuning:** In addition to replacing the classification layers, unfreeze some of the final layers in the pre-trained model and train them along with the new layers on your dataset. This can help the model adapt to your specific task better.
>
> **Training and Evaluation:** Train the adapted model on your face emotion recognition dataset. Monitor the loss and accuracy on a validation set to ensure your model is learning effectively. Evaluate the model's performance on a separate test set.

Regularization and Augmentation:
To prevent overfitting, use techniques like dropout, batch normalization, and data augmentation during training. Data augmentation involves applying random transformations (such as rotations, flips, and translations) to the training images, which increases the model's robustness.

Hyperparameter Tuning:
Experiment with hyperparameters like learning rate, batch size, and the number of layers to fine-tune the model's performance on your specific task.

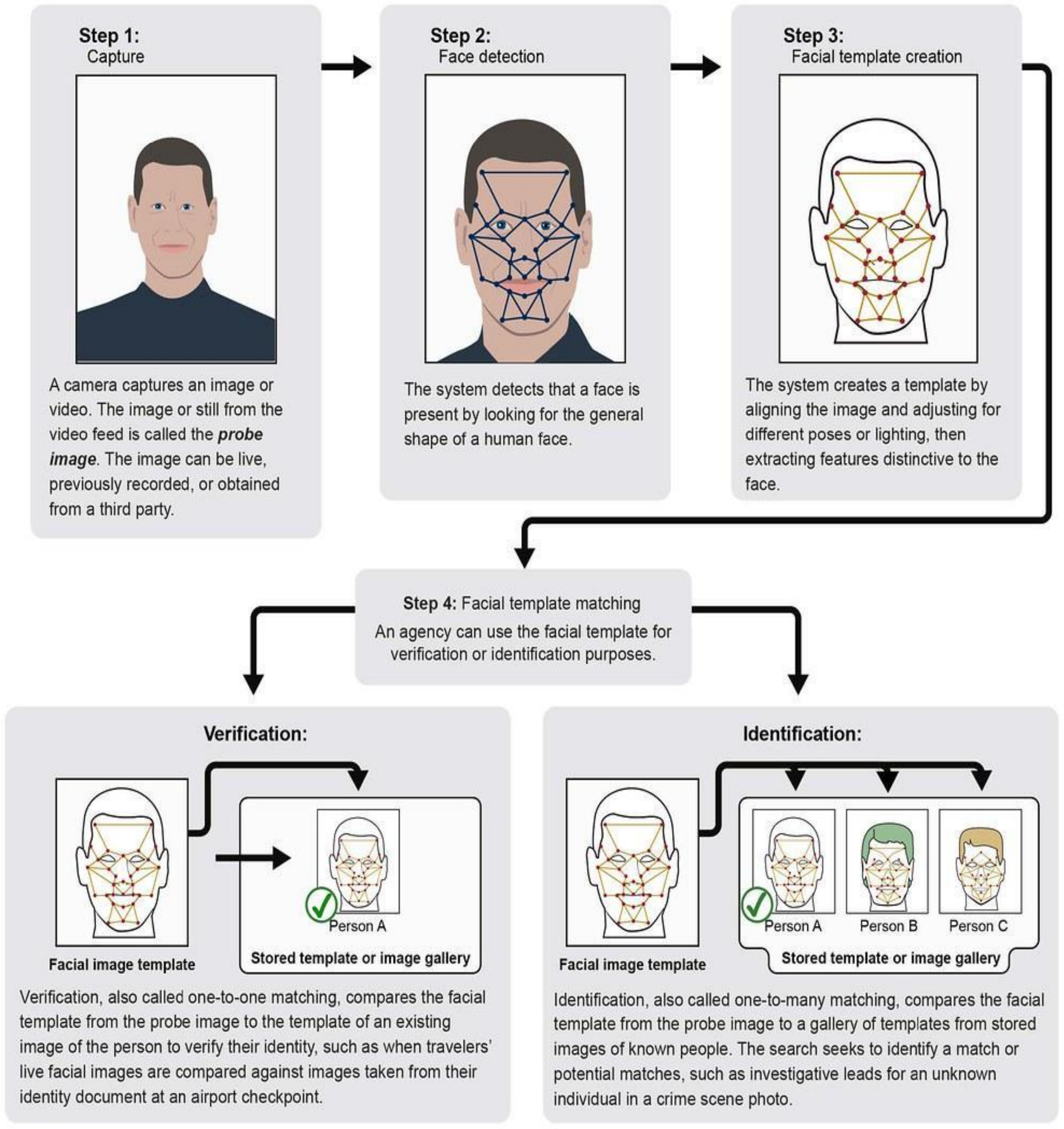
Iterative Refinement:
Based on the evaluation results, iterate and make adjustments to your model, data preprocessing, and training strategy. This process might involve trying different pre-trained models, tweaking architectures, and testing various hyperparameters.

Transfer learning can significantly speed up the development of accurate face emotion recognition models, especially if you have limited labeled data.

WORKING OF MODEL

A face emotion recognition system is a technology that uses computer vision and machine learning techniques to automatically detect and classify human emotions based on facial expressions.

Overall, a face emotion recognition system combines computer vision techniques with machine learning to automatically analyze facial expressions and predict the emotions displayed by individuals. It finds applications in various fields, including human-computer interaction, user experience analysis, and psychological research.



A camera captures an image or video. The image or still from the video feed is called the ***probe image***. The image can be live, previously recorded, or obtained from a third party.

The system detects that a face is present by looking for the general shape of a human face.

The system creates a template by aligning the image and adjusting for different poses or lighting, then extracting features distinctive to the face.

An agency can use the facial template for verification or identification purposes.

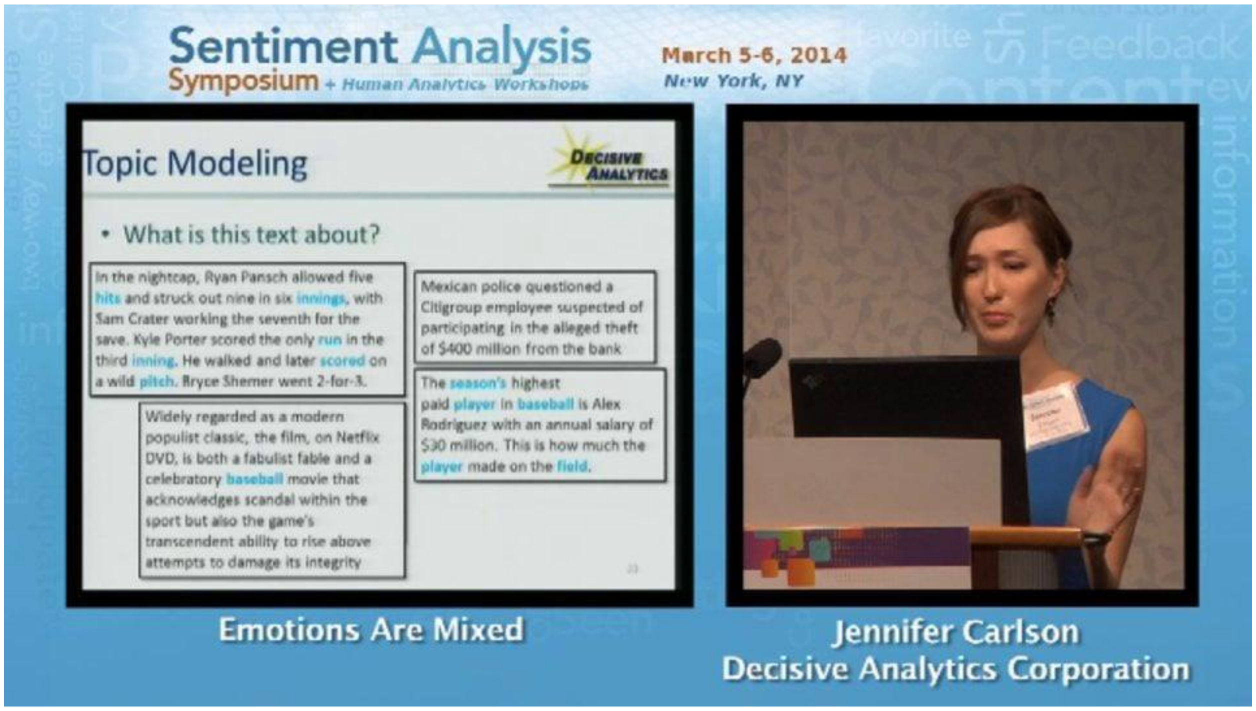
Verification, also called one-to-one matching, compares the facial template from the probe image to the template of an existing image of the person to verify their identity, such as when travelers' live facial images are compared against images taken from their identity document at an airport checkpoint.

Identification, also called one-to-many matching, compares the facial template from the probe image to a gallery of templates from stored images of known people. The search seeks to identify a match or potential matches, such as investigative leads for an unknown individual in a crime scene photo.

Source: GAO analysis. | GAO-21-526

CHAPTER 3

METHODOLOGY OF THE PROJECT



Topic Modeling

- What is this text about?

In the nightcap, Ryan Pansch allowed five hits and struck out nine in six innings, with Sam Crater working the seventh for the save. Kyle Porter scored the only run in the third inning. He walked and later scored on a wild pitch. Bryce Shemer went 2-for-3.

Widely regarded as a modern populist classic, the film, on Netflix DVD, is both a fabulist fable and a celebratory baseball movie that acknowledges scandal within the sport but also the game's transcendent ability to rise above attempts to damage its integrity

Mexican police questioned a Citigroup employee suspected of participating in the alleged theft of $400 million from the bank

The season's highest paid player in baseball is Alex Rodriguez with an annual salary of $30 million. This is how much the player made on the field.

A face emotion recognition system is a technology that uses computer vision and machine learning techniques to automatically detect and classify human emotions based on facial expressions. Here's an overview of how such a system typically works:

Data Collection and Preprocessing:
The system requires a dataset of labeled facial images that represent different emotions (such as happiness, sadness, anger, surprise, etc.). These images are used to train the machine learning model. The images may be collected from various sources, including databases, online sources, or by capturing images with known emotional expressions.

Feature Extraction:
Each face in the images needs to be processed to extract relevant features that represent the facial expressions. These features might include aspects like the position of facial landmarks (such as the eyes, nose, and mouth), facial muscle movements, and texture patterns.

Machine Learning Model:
The system uses a machine learning model to learn the relationships between the extracted features and the corresponding emotions. Commonly used models include Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), or combinations of these.

Training:
The model is trained using the labeled dataset. During training, the model learns to recognize patterns in the extracted features that are indicative of specific emotions. The dataset is typically split into training and validation sets to prevent overfitting and to fine-tune the model's parameters.

Testing and Validation:
After training, the model is evaluated on a separate testing dataset to assess its performance. Metrics like accuracy, precision, recall, and F1-score are often used to measure how well the model predicts emotions on unseen data.

Real-time Emotion Detection:
Once the model is trained and validated, it can be used to detect emotions in real-time by processing live video streams or images. The system captures the face from the input data, extracts relevant features, and feeds them into the trained model.

Visualization and Output:
The recognized emotion can be visualized by overlaying labels or icons on the video feed or image. Depending on the application, the system might also store or display historical data regarding detected emotions.

Optimization and Updates:
The system's performance can be improved over time by continuously collecting new data, retraining the model, and optimizing its parameters. This helps the model adapt to different lighting conditions, diverse facial expressions, and evolving user preferences.

CHAPTER 4

IMPLEMENTATION

Implementing a real-time face emotion recognition system involves several steps, including setting up the environment, collecting data, building and training a deep learning model, and integrating the model into a real-time application. Below is the code for implementation of real-time emotion recognition:

```
#Importing FER 2013 dataset from Google Drive

from google.colab import drive
        drive.mount('/content/drive')

from zipfile import ZipFile
        file_name = "/content/drive/My Drive/archive.zip"

with ZipFile(file_name, 'r') as zip:
zip.extractall()
        print("Done")
```

```
Mounted at /content/drive
Done
```

```
#Libraries

#For modifying images & training model
import numpy as np #convert img to high speed arrays
import cv2 #capturing & displaying images
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D
from keras.optimizers import Adam
from keras.layers import MaxPooling2D
from keras.preprocessing.image import ImageDataGenerator
```

```
import keras.utils as image
import matplotlib.pyplot as plt #Display image through graph plots

#For saving and loading pretrained models
import tensorflow as tf
from keras.models import load_model

#For capturing webcam content & displaying it on Google Colab
from IPython.display import display, Javascript
from google.colab.output import eval_js
from base64 import b64decode, b64encode
import PIL
import io
import html
import time
```

```
# Separating training & testing data from FER 2013 dataset
train_dir = 'train'
val_dir = 'test'
train_datagen = ImageDataGenerator(rescale=1./255) #normalizing dataset
val_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(48,48),
        batch_size=64,
        color_mode="grayscale",
        class_mode='categorical')

validation_generator = val_datagen.flow_from_directory(
        val_dir,
        target_size=(48,48),
        batch_size=64,
        color_mode="grayscale",
        class_mode='categorical')
```

```
Found 28709 images belonging to 7 classes.
Found 7178 images belonging to 7 classes.
```

```
#Configuring model by adding convolutional layers and max pooling data

emotion_model = Sequential()
emotion_model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
input_shape=(48,48,1)))
emotion_model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Dropout(0.25))
emotion_model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Dropout(0.25))
emotion_model.add(Flatten())
emotion_model.add(Dense(1024, activation='relu'))
emotion_model.add(Dropout(0.5))
emotion_model.add(Dense(7, activation='softmax'))
```

```
#Transfer Learning by model fitting

emotion_model.compile(loss='categorical_crossentropy',optimizer=Adam(lr=0.0001,
decay=1e-6),metrics=['accuracy'])
emotion_model_info = emotion_model.fit_generator(
        train_generator,
        steps_per_epoch=28709 // 64,
        epochs=150, #Number of cycles model training runs
        validation_data=validation_generator,
        validation_steps=7178 // 64)
```

```
…
…
Epoch 125/150
448/448 [==============================] - 14s 32ms/step - loss: 0.1196 -
accuracy: 0.9573 - val_loss: 1.5670 - val_accuracy: 0.6223
```

```
Epoch 126/150
448/448 [==============================] - 14s 32ms/step - loss: 0.1185 -
accuracy: 0.9574 - val_loss: 1.5637 - val_accuracy: 0.6200
…
…
Epoch 146/150
448/448 [==============================] - 14s 32ms/step - loss: 0.0976 -
accuracy: 0.9671 - val_loss: 1.6894 - val_accuracy: 0.6228
Epoch 147/150
448/448 [==============================] - 14s 32ms/step - loss: 0.1031 -
accuracy: 0.9641 - val_loss: 1.6696 - val_accuracy: 0.6219
Epoch 148/150
448/448 [==============================] - 15s 33ms/step - loss: 0.1025 -
accuracy: 0.9643 - val_loss: 1.6391 - val_accuracy: 0.6225
Epoch 149/150
448/448 [==============================] - 17s 39ms/step - loss: 0.1014 -
accuracy: 0.9653 - val_loss: 1.6756 - val_accuracy: 0.6243
Epoch 150/150
448/448 [==============================] - 14s 32ms/step - loss: 0.1029 -
accuracy: 0.9644 - val_loss: 1.6969 - val_accuracy: 0.6293
```

```
# Saving the model

emotion_model.save('saved_model/model.h5')
new_model = tf.keras.models.load_model('saved_model/model.h5')
```

```
# Return model prediction

def emotion_analysis(emotions):
        objects = {0:'sad', 1:'disgust', 2:'fear',
                3:'angry', 4:'neutral', 5:'happy',
                6: 'surprise'}
        return objects[(np.argmax(emotions))]
```

```
# Capturing image on Google Colab

def take_photo(filename='photo.jpg', quality=0.8):
    js = Javascript('''
    async function takePhoto(quality) {
        const div = document.createElement('div');
        const capture = document.createElement('button');
        capture.textContent = 'Capture';
        div.appendChild(capture);

        const video = document.createElement('video');
        video.style.display = 'block';
        const stream = await navigator.mediaDevices.getUserMedia({video:
        true});

        document.body.appendChild(div);
        div.appendChild(video);
        video.srcObject = stream;
        await video.play();

        // Resize the output to fit the video element.
        google.colab.output.setIframeHeight(document.documentElement.scrollH
        eight, true);

        // Wait for Capture to be clicked.
        await new Promise((resolve) => capture.onclick = resolve);

        const canvas = document.createElement('canvas');
        canvas.width = video.videoWidth;
        canvas.height = video.videoHeight;
        canvas.getContext('2d').drawImage(video, 0, 0);
        stream.getVideoTracks()[0].stop();
        div.remove();
        return canvas.toDataURL('image/jpeg', quality);
    }
    ''')
```

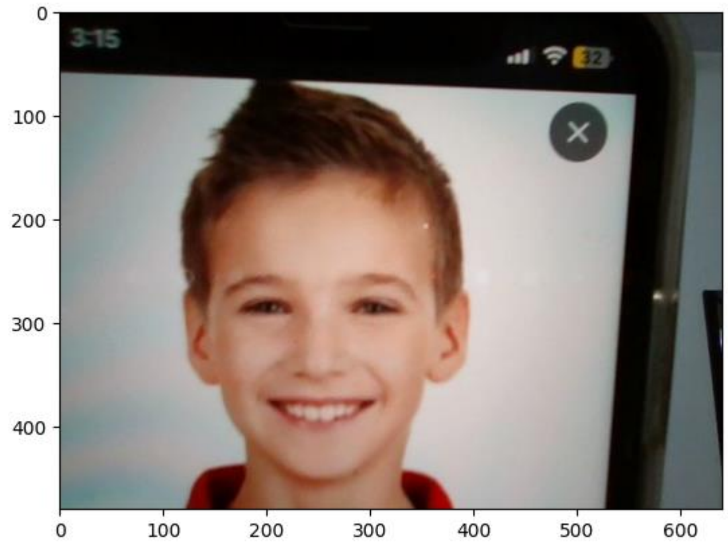
```
display(js)
data = eval_js('takePhoto({})'.format(quality))
binary = b64decode(data.split(',')[1])
with open(filename, 'wb') as f:
    f.write(binary)
return filename

take_photo()
```

'photo.jpg'

```
#Cropping face for model prediction

def facecrop(image):
    facedata = '/content/haarcascade_frontalface_alt.xml'
    cascade = cv2.CascadeClassifier(facedata)
    img = cv2.imread(image)
    try:
        minisize = (img.shape[1],img.shape[0])
        miniframe = cv2.resize(img, minisize)
        faces = cascade.detectMultiScale(miniframe)
        for f in faces:
            x, y, w, h = [ v for v in f ]
            cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)
            sub_face = img[y:y+h, x:x+w]
            cv2.imwrite('capture.jpg', sub_face)
        except Exception as e:
            print (e)

if __name__ == '__main__':
    facecrop('/content/photo.jpg')

file = '/content/photo.jpg'
true_image = image.load_img(file)
img = image.load_img(file, color_mode="grayscale", target_size=(48, 48))
```

```
x = image.img_to_array(img)
x = np.expand_dims(x, axis = 0)

x /= 255 #normalizing image

custom = emotion_model.predict(x,verbose=0)
print(emotion_analysis(custom))

x = np.array(x, 'float32')
x = x.reshape([48, 48]);

plt.imshow(true_image)
plt.show()
```

```
OpenCV(4.8.0) /io/opencv/modules/objdetect/src/cascadedetect.cpp:1689: error: (-215:Assertion failed)

happy
```



```
# function to convert JavaScript data to image
def js_to_image(js_reply):
        image_bytes = b64decode(js_reply.split(',')[1]) # decode base64 image
        jpg_as_np = np.frombuffer(image_bytes, dtype=np.uint8) # convert bytes to
        numpy array
        img = cv2.imdecode(jpg_as_np, flags=1) # decode numpy array into OpenCV
        BGR image
        return img

# function to convert OpenCV Rectangle bounding box image into base64 byte string
to be overlayed on video stream
def bbox_to_bytes(bbox_array):
        bbox_PIL = PIL.Image.fromarray(bbox_array, 'RGBA') # convert array into PIL
        image
        iobuf = io.BytesIO()
        bbox_PIL.save(iobuf, format='png') # format bbox into png for return
        bbox_bytes =
        'data:image/png;base64,{}'.format((str(b64encode(iobuf.getvalue()), 'utf-8'))) #
        format return string

        return bbox_bytes

face_cascade = cv2.CascadeClassifier(cv2.samples.findFile(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')) #detect face
```

```
# Video capture for Google Colab & display UI

def video_stream():
        js = Javascript('''
                var video;
                var div = null;
                var stream;
                var captureCanvas;
                var imgElement;
                var labelElement;

                var pendingResolve = null;
                var shutdown = false;
                function removeDom() {
```

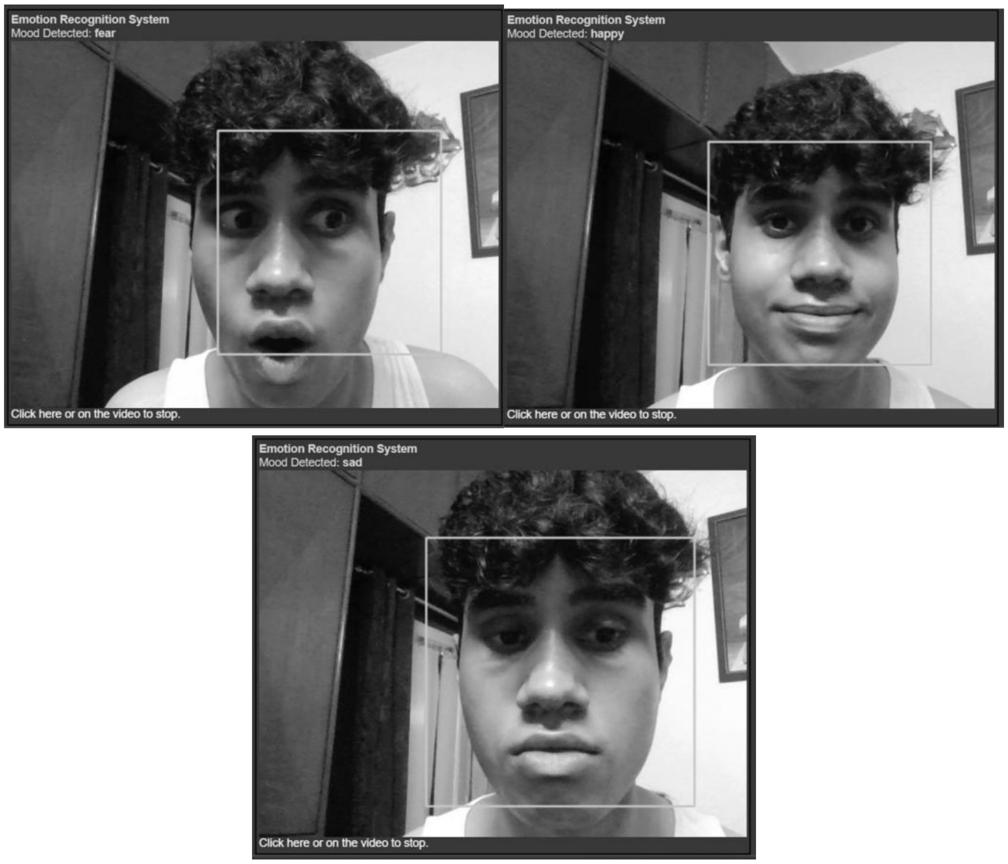
```
            stream.getVideoTracks()[0].stop();
            video.remove();
            div.remove();
            video = null;
            div = null;
            stream = null;
            imgElement = null;
            captureCanvas = null;
            labelElement = null;
        }

        function onAnimationFrame() {
            if (!shutdown) {
                window.requestAnimationFrame(onAnimationFrame);
            }
            if (pendingResolve) {
                var result = "";
                if (!shutdown) {
                    captureCanvas.getContext('2d').drawImage(video, 0,
                    0, 640, 480);
                    result = captureCanvas.toDataURL('image/jpeg', 0.8)
                }
                var lp = pendingResolve;
                pendingResolve = null;
                lp(result);
            }
        }

        async function createDom() {
            if (div !== null) {
                return stream;
            }

            div = document.createElement('div');
            div.style.border = '2px solid black';
            div.style.padding = '3px';
```

```
div.style.width = '100%';
div.style.maxWidth = '600px';
document.body.appendChild(div);

const winTitle = document.createElement('div');
winTitle.innerHTML = "<span>Emotion Recognition
System</span>";
winTitle.style.fontWeight = 'bold';
const modelOut = document.createElement('div');
modelOut.innerHTML = "<span>Mood Detected: </span>";
labelElement = document.createElement('span');
labelElement.innerText = 'No data';
labelElement.style.fontWeight = 'bold';
modelOut.appendChild(labelElement);
div.appendChild(winTitle);
div.appendChild(modelOut);

video = document.createElement('video');
video.style.display = 'block';
video.width = div.clientWidth - 6;
video.setAttribute('playsinline', '');
video.onclick = () => { shutdown = true; };
stream = await navigator.mediaDevices.getUserMedia(
        {video: { facingMode: "environment"}});
div.appendChild(video);

imgElement = document.createElement('img');
imgElement.style.position = 'absolute';
imgElement.style.zIndex = 1;
imgElement.onclick = () => { shutdown = true; };
div.appendChild(imgElement);

const instruction = document.createElement('div');
instruction.innerHTML =
        '<span style="color: ghostwhite;">' +
        'Click here or on the video to stop.</span>';
        div.appendChild(instruction);
        instruction.onclick = () => { shutdown = true; };

        video.srcObject = stream;
        await video.play();

        captureCanvas = document.createElement('canvas');
        captureCanvas.width = 640; //video.videoWidth;
        captureCanvas.height = 480; //video.videoHeight;
        window.requestAnimationFrame(onAnimationFrame);

        return stream;
}
async function stream_frame(label, imgData) {
        if (shutdown) {
                removeDom();
                shutdown = false;
                return '';
        }

        var preCreate = Date.now();
        stream = await createDom();

        var preShow = Date.now();
        if (label != "") {
                labelElement.innerHTML = label;
        }
        if (imgData != "") {
                var videoRect = video.getClientRects()[0];
                imgElement.style.top = videoRect.top + "px";
                imgElement.style.left = videoRect.left + "px";
                imgElement.style.width = videoRect.width + "px";
                imgElement.style.height = videoRect.height + "px";
                imgElement.src = imgData;
        }
```

```
                var preCapture = Date.now();
                var result = await new Promise(function(resolve, reject) {
                    pendingResolve = resolve;
                });
                shutdown = false;

                return {'create': preShow - preCreate,
                    'show': preCapture - preShow,
                    'capture': Date.now() - preCapture,
                    'img': result};
            }
            ''')

            display(js)

def video_frame(label, bbox):
            data = eval_js('stream_frame("{}", "{}")'.format(label, bbox))
            return data
```

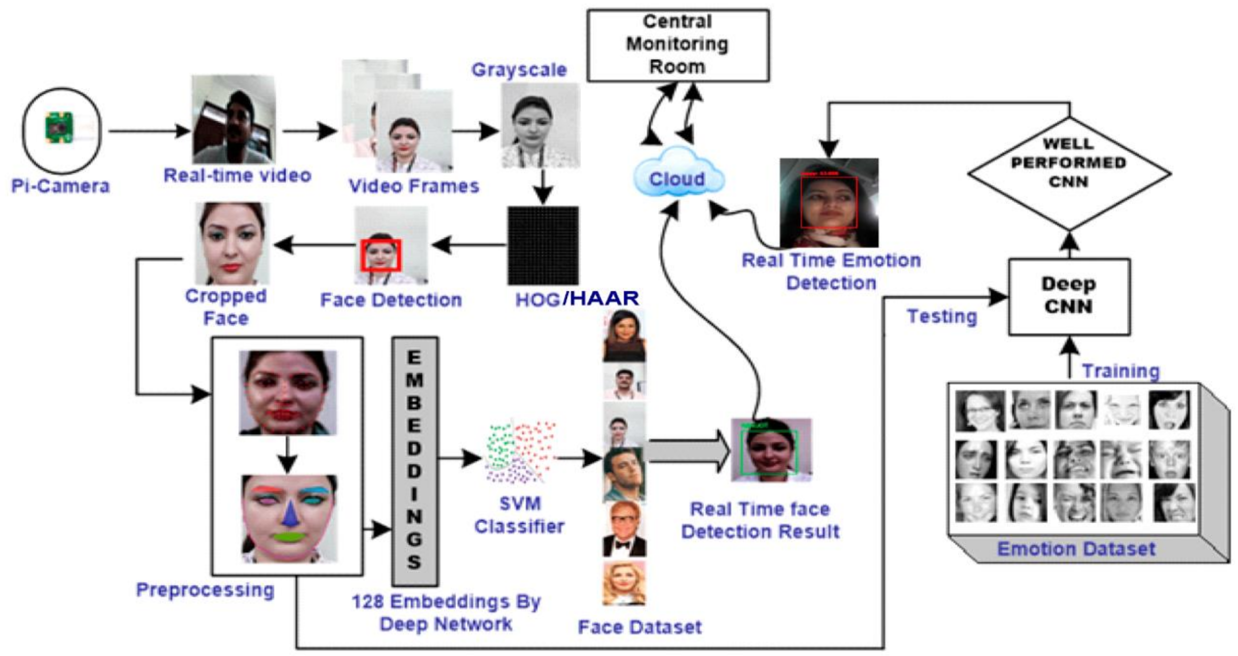
```
#Start video capture & check model prediction

video_stream()

label_html = 'Not detected'
bbox = ''
count = 0
while True:
    js_reply = video_frame(label_html, bbox)
    if not js_reply:
        break
    img = js_to_image(js_reply["img"])
    bbox_array = np.zeros([480,640,4], dtype=np.uint8)
    gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY) # grayscale image for
    face detection
    faces = face_cascade.detectMultiScale(gray)
    for (x,y,w,h) in faces:
```

```
    bbox_array = cv2.rectangle(bbox_array,(x,y),(x+w,y+h),(0,255,255),2)
    img=cv2.resize(img,(48,48))
    try:
        img=cv2.cvtColor(img,cv2.COLOR_RGB2GRAY)
    except Exception as e:
        pass
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis = 0)
    x /= 255 #normalizing data

    custom = emotion_model.predict(x,verbose=0)
    label_html=emotion_analysis(custom)
bbox_array[:,:,3] = (bbox_array.max(axis = 2) > 0 ).astype(int) * 255
bbox_bytes = bbox_to_bytes(bbox_array)
bbox = bbox_bytes
```


Emotion Recognition System
Mood Detected: fear
Click here or on the video to stop.
Emotion Recognition System
Mood Detected: happy
Click here or on the video to stop.
Emotion Recognition System
Mood Detected: sad
Click here or on the video to stop.

Screenshots of Real Time Emotion Recognition

CHAPTER 5: RESULTS AND ANALYSIS



The results and analysis of a real-time face emotion recognition system can provide insights into its performance, accuracy, and potential areas for improvement. Here's how you might approach analyzing the results:

1. Performance Metrics:
Start by calculating various performance metrics to quantify the accuracy and effectiveness of your system. Common metrics include:

Accuracy: The overall percentage of correctly predicted emotions.
Precision: The ratio of true positive predictions to the total positive predictions. It measures how many predicted positive instances are actually correct.
Recall (Sensitivity): The ratio of true positive predictions to the total actual positive instances. It measures the system's ability to detect all positive instances.
F1-score: The harmonic mean of precision and recall, providing a balance between the two metrics.
Confusion Matrix: A table that shows the counts of true positive, true negative, false positive, and false negative predictions for each emotion class.

2. Emotion Recognition Accuracy:
Analyze the accuracy of the system's emotion recognition for different emotions. Some emotions might be more challenging to predict accurately than others due to variations in facial expressions or cultural differences. Determine which emotions your system performs well on and where it might struggle.

3. Challenges and Limitations:
Document challenges or limitations your system faces. For example, the system might struggle with recognizing subtle emotions or emotions expressed by people with atypical facial expressions. Identifying these limitations can guide future enhancements.

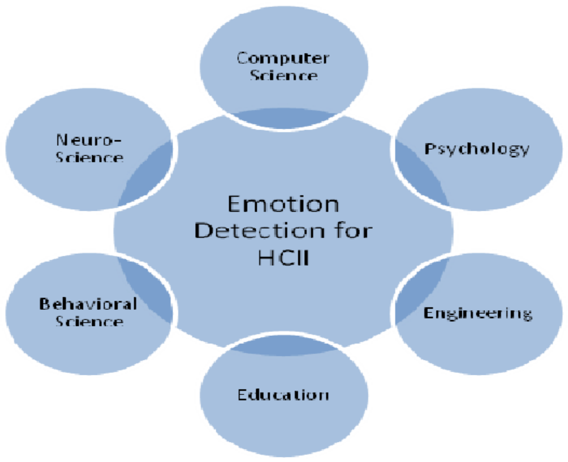
4. False Positives and Negatives:
Analyze cases where the system made false positive or false negative predictions. This analysis can provide insights into specific scenarios or facial expressions that the model might struggle with. It could also indicate the need for more diverse training data.

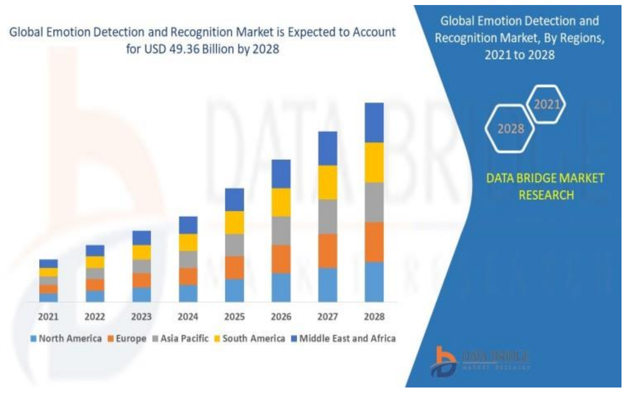
5. Model Optimization:
If you notice specific patterns in the errors your model is making, consider fine-tuning the model or collecting more data to address these issues. This might involve adjusting hyperparameters, modifying the architecture, or expanding the training dataset.

6. Ethical Considerations:
Reflect on ethical considerations related to face emotion recognition, such as potential biases in the training data, privacy concerns, and the implications of using such technology in various contexts. Consider how your system addresses these issues.

7. Benchmarking:
Compare your system's performance with existing state-of-the-art models or other emotion recognition systems. This can give you a broader context for evaluating your system's accuracy and effectiveness.
In summary, analyzing the results of a real-time face emotion recognition system involves evaluating its accuracy, performance in real-world scenarios, limitations, challenges, and potential areas for improvement. This analysis can guide you in refining your system and addressing any issues that arise during deployment.

CHAPTER 6: CONCLUSION & FUTURE SCOPE





The future scope of real-time face emotion recognition systems is promising and includes several exciting areas of development and innovation. Here are some potential future directions and advancements:

1. Improved Accuracy: Ongoing research in deep learning and computer vision will likely lead to more accurate emotion recognition models. This could involve better feature extraction techniques, more advanced neural network architectures, and enhanced training strategies.

2. Multi-Modal Approaches: Combining facial analysis with other modalities, such as voice analysis or physiological signals, can provide a more comprehensive understanding of emotions. Integrating these modalities could lead to more accurate and robust emotion recognition systems.

3. Temporal Context: Incorporating temporal information from video sequences can enhance emotion recognition by capturing how emotions evolve over time. Long Short-Term Memory (LSTM) networks and other sequential models can be employed to model temporal dependencies.

4. Transfer Learning: Applying transfer learning techniques, where models pre trained on large datasets are fine-tuned for emotion recognition, can expedite the development of accurate models, especially when labeled emotion datasets are limited.

5. Adaptation to Individual Differences: Future systems might adapt to individual differences in facial expressions, personal characteristics, and cultural variations. Customized models that take these differences into account could lead to more personalized emotion recognition.

6. Real-world Variability: Emotion recognition systems could be trained and tested on more diverse and challenging real-world datasets, accounting for different lighting conditions, angles, poses, and occlusions.

7. Edge Computing: Advancements in hardware and optimization techniques will likely make it feasible to deploy emotion recognition systems on edge devices like smartphones, wearable devices, and IoT devices, enabling more privacy-preserving and localized applications.

8. Ethical and Privacy Considerations: Researchers and developers will continue to address ethical concerns, privacy issues, and potential biases in data to ensure that emotion recognition technology is used responsibly and respects users' rights.

9. Affective Computing Applications: Emotion recognition can be integrated into a wide range of applications, including human-computer interaction, virtual reality, gaming, health monitoring, marketing, and education, creating new opportunities for innovation.

10. Real-world Context Integration: Emotion recognition systems might incorporate contextual information from the environment and user interactions to provide a deeper understanding of emotions in different situations.

11. Continuous Learning and Feedback Loop: Emotion recognition systems could continually learn and adapt from user feedback, improving their accuracy and relevance over time.

12. Emotion Generation: Some systems might progress beyond recognition to generating emotional responses, such as generating appropriate emotional responses in virtual assistants or chatbots.
As technology advances and research in artificial intelligence and computer vision continues, the scope for real-time face emotion recognition systems is likely to expand, leading to more accurate, versatile, and ethically sound applications in various domains.

In conclusion, the development and implementation of a face emotion recognition system have significant implications for human-computer interaction, psychological research, and various industries. This technology leverages the power of deep learning, computer vision, and real-time processing to automatically detect and classify emotions from facial expressions. The journey from data collection to model training and real-time deployment involves careful consideration of ethical concerns, privacy issues, and accuracy improvements.

Through this system, we can gain insights into human emotional responses, enabling improved user experiences, adaptive interfaces, and tailored content delivery. However, it's essential to acknowledge the challenges, including variations in facial expressions, cultural differences, and limitations in current technology. Researchers and developers must work collaboratively to address biases, ensure privacy, and mitigate unintended consequences of emotion recognition systems.

As technology continues to evolve, the potential applications of real-time face emotion recognition are vast. From aiding clinical diagnoses to enhancing virtual reality experiences, the impact spans industries such as healthcare, entertainment, marketing, education, and more. The future holds the promise of more accurate, adaptable, and context-aware emotion recognition systems that truly understand and respond to human emotions in ways that benefit society.

In this journey, it's vital to remain committed to responsible development, user consent, and transparent communication about how data is collected, processed, and utilized. By combining innovation with ethical considerations, we can harness the capabilities of face emotion recognition systems to create meaningful and positive interactions between humans and technology.

REFERENCES

What is Transfer Learning? Transfer Learning in Keras | Fine Tuning Vs Feature Extraction
https://www.youtube.com/watch?v=WWcgHjuKVqA&t=591s

OpenCV Course - Full Tutorial with Python
https://www.youtube.com/watch?v=oXlwWbU8l2o&t=1032s

TensorFlow v2.13.0
https://www.tensorflow.org/api_docs/python/tf