INDEX

EMOTION ANALYSER

Emotion analyser is an employing advanced machine learning and natural language processing techniques, serves as a pivotal tool in deciphering the intricate nuances of human sentiment across diverse mediums like audio, and video. The primary goal of an emotion analyser `is to automatically identify and classify emotions expressed in input data, providing valuable insights into the emotional context of communication, and enabling applications to respond appropriately to user emotions.

Abstract:

Multi-modal emotion recognition is an evolving field aiming to decipher human emotions across different modalities like facial expressions, body language, and speech. This research focuses on developing robust models capable of accurately discerning nuanced emotional states in diverse contexts, including social media, video conferencing, and surveillance systems. Utilizing deep learning techniques such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), this study investigates the fusion of multi-modal features to enhance emotion recognition performance. It explores feature fusion methodologies like concatenation and early fusion to integrate information from different modalities effectively. Additionally, the research explores training strategies like cross-validation and hyperparameter tuning to optimize model performance and ensure generalizability. Evaluation metrics such as accuracy, precision, recall, and F1-score are employed to assess the models' efficacy in recognizing emotions across various datasets, contributing to the advancement of empathetic and context-aware artificial intelligence systems.

Introduction:

It will automatically be identifying and categorizing emotions, these systems provide invaluable insights into the emotional context of interactions, facilitating personalized responses and enhancing user experiences. By analysing customer feedback to monitoring social media sentiment and even supporting mental health monitoring, emotion analysers find applications across a spectrum of industries, contributing to more empathetic and responsive digital environments. The machine learning model will process the audio-visual processing techniques, emotion analysers extract salient features from input data and classify emotions into predefined categories such as happiness, sadness, anger, or surprise. From traditional statistical methods like logistic regression to cutting-edge deep learning architectures such as convolutional neural networks (CNNs) these models evolve to handle the nuances of human emotions with increasing accuracy and granularity.

BACKGROUND:

Emotion analysis is a affective computing, it has become a crucial instrument for analysing human sentiment in digital communication. Initially tailored for textual data, its scope now spans across auditory and visual mediums, by advancements in machine learning, It particularly through sophisticated deep learning models. This progression has empowered more precise and nuanced emotion detection, fostering applications across diverse sectors including customer service, market research, and mental health monitoring. As the demand for personalized user experiences grows, emotion analysis stands as a cornerstone in crafting emotionally intelligent systems, reshaping the landscape of human-computer interaction in digital realms.

Problem statement:

- Detecting emotions accurately across different mediums like text, audio, and video remains tricky due to nuances and interpretability challenges in existing models. Deep learning methods, although powerful, are complex and may not be suitable for sensitive areas like mental health.

OBJECTIVE:

- To create a robust emotion analysis system capable of accurately detecting and understanding emotions in text, audio, and video data. We aim to overcome existing challenges, including interpretability and scalability issues, by employing innovative methodologies.
- To explore practical applications in areas such as customer feedback analysis, social media monitoring, and mental health assessment, with the aim of improving user experiences and decision-making processes across various domains.

SOFTWARE REQUIREMENTS:

Python: The primary programming language for implementing machine learning algorithms and building the application.

Deep Learning Frameworks: Libraries such as TensorFlow, Keras, or PyTorch for building and training deep learning models for image, audio, and text processing.

OpenCV: For image and video processing tasks, including face detection and feature extraction.

Librosa: A Python library for audio and music analysis, useful for extracting audio features like MFCCs.

Natural Language Processing (NLP) Libraries: Such as NLTK or SpaCy for text processing tasks, if text data is included in the multi-modal inputs.

Flask or Django: Web frameworks for building the application's backend to handle data processing, model inference, and serving APIs.

HTML/CSS/JavaScript: For building the frontend interface if you plan to create a web application.

Splunk (Optional): For log monitoring and analysis if you choose to integrate logging with Splunk.

Jupyter Notebook: For exploratory data analysis, prototyping machine learning models, and documenting the development process.

Git: Version control system for tracking changes in the codebase and collaborating with team members.

IDE (Integrated Development Environment): Such as PyCharm, VS Code, or JupyterLab for writing and debugging code efficiently.

HARDWARE REQUIREMENTS:

CPU: A multi-core CPU (e.g., Intel Core i5 or higher, AMD Ryzen series) is sufficient for basic development tasks, data preprocessing, and running inference on small-scale models.

GPU: For training deep learning models efficiently, especially large-scale models or models with complex architectures, a dedicated GPU is recommended. NVIDIA GPUs, such as GeForce GTX or RTX series for consumer-grade systems, or NVIDIA Tesla for enterprise-grade systems, are commonly used for deep learning tasks. GPUs with CUDA support are preferred for compatibility with popular deep learning frameworks like TensorFlow and PyTorch.

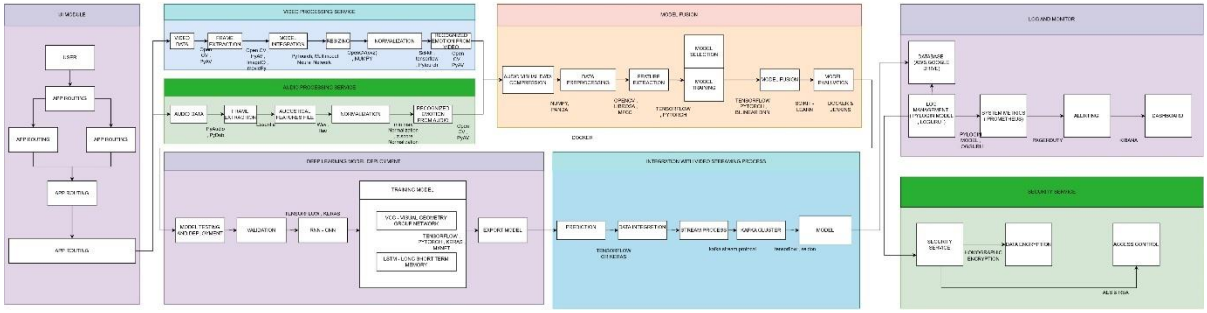
RAM: Adequate system memory (RAM) is essential for handling large datasets and model parameters during training. At least 16 GB of RAM is recommended for smooth operation, although more RAM may be required for larger models or batch sizes.

Storage: Sufficient storage space is necessary for storing datasets, trained models, and intermediate results. A solid-state drive (SSD) is preferred for faster data access and training times compared to traditional hard disk drives (HDD).

Internet Connection: A stable and high-speed internet connection is necessary for downloading datasets, model weights, and software packages, as well as accessing online resources and documentation.

Optional: Cloud Services: Cloud computing platforms like Splunk offer scalable compute resources, including GPU instances, for training deep learning models. Using cloud services can be cost-effective and convenient, especially for large-scale projects or when dedicated hardware is not available locally

ARCITECTURE:



Architecture Explanation (SERVICES):

Multimodal Fusion Service:

This service combines video and audio features for enhanced emotion recognition. Security measures include data encryption for sensitive information and access control to prevent unauthorized access to the fusion service. Integration with authentication services ensures secure access to the fusion service from other components.

Video Processing Service:

Handles video input and preprocessing tasks. Security measures involve validating input data to prevent injection attacks and ensuring secure communication channels for data transfer. Integration with authentication and authorization services ensures that only authorized users or services can access the video processing service.

Audio Analysis Service:

Analyses audio data to extract emotion-related features. Security measures include implementing secure APIs for data exchange and encrypting sensitive data at rest and in transit. Integration with identity management systems ensures proper authentication and access control for accessing audio analysis services.

Deep Learning Model Deployment:

Deploys deep learning models for emotion recognition. Security measures involve securing model endpoints with authentication and authorization mechanisms. Integration with security monitoring tools ensures continuous monitoring of model deployment for any security vulnerabilities or breaches.

Integration with Video Streaming Platform:

Integrates the emotion recognition system with video streaming platforms. Security measures include implementing secure communication protocols (e.g., HTTPS) and encryption of data in transit. Integration with identity and access management systems ensures secure authentication and authorization for accessing video streams.

Logging and Monitoring Service:

Ensures effective logging and monitoring of system performance and behaviour. Security measures involve protecting log data integrity and confidentiality through encryption and access control. Integration with

security information and event management (SIEM) systems enables real-time monitoring and detection of security incidents.

Security and Privacy Service:

This service ensures the overall security and privacy of the emotion recognition system. Implements measures such as data encryption, access control, and anonymization techniques to protect sensitive user data. Conducts regular security audits and vulnerability assessments to identify and mitigate potential security risks. Integrates with security frameworks and compliance standards to ensure adherence to regulatory requirements and industry best practices.

Functionality:

The multi-modal emotion recognition project for video content involves the development of a sophisticated system capable of accurately detecting and interpreting emotions conveyed through facial expressions, body language, and audio cues. This system integrates deep learning models, including convolutional neural networks (CNNs) for analysing video frames and for processing audio streams, with fusion techniques to combine information from multiple modalities. By training these models on labelled video datasets and evaluating their performance using standard metrics, the project aims to provide real-time emotion recognition capabilities for streaming video content. Integration with video streaming platforms enhances user engagement and facilitates valuable insights for content creators and advertisers. Additionally, the system offers customization options and visualization tools to tailor the emotion recognition process to

specific use cases or domains, ensuring its applicability across diverse scenarios.

TESTING:

Unit Testing: Write unit tests to verify the functionality of individual components, such as the detect_faces function and the route handlers. Use frameworks like unittest or pytest to automate the testing process.

Integration Testing: Perform integration testing to ensure that different components of your application work together correctly. This can involve testing the interaction between Flask, OpenCV, the emotion recognition model, and any external services such as Splunk.

End-to-End Testing: Conduct end-to-end testing to validate the entire application workflow, from capturing video frames to processing them for emotion recognition and streaming the results. This can involve manually testing the application in various scenarios or using automated testing frameworks like Selenium.

Performance Testing: Evaluate the performance of your application by measuring factors such as response time, throughput, and resource usage. Test the application under different load conditions to identify any performance bottlenecks or scalability issues.

User Acceptance Testing (UAT): Involve end-users or stakeholders in testing the application to ensure that it meets their requirements and expectations. Gather feedback from users to identify areas for improvement or additional features.

Security Testing: Perform security testing to identify and address potential vulnerabilities in your application, such as input validation, authentication, and data privacy. Ensure that sensitive data, such as video streams and emotion recognition results, are handled securely.

Error Handling and Logging Testing: Test error handling mechanisms to ensure that the application gracefully handles unexpected errors and logs relevant information for debugging purposes. Verify that logs are generated correctly and contain useful information for troubleshooting issues.

Usability Testing: Evaluate the usability of your application by observing how users interact with it and collecting feedback on the user interface, navigation, and overall user experience.

TRAINING :

Data Collection: Gather a diverse dataset containing samples of facial expressions, body language, speech, and other modalities expressing different emotions. Ensure the dataset is balanced across different emotions and includes a variety of individuals, backgrounds, and contexts.

Data Preprocessing: Preprocess the data to ensure consistency and compatibility across modalities. This may involve tasks

such as face detection and alignment, audio feature extraction (e.g., MFCCs), and normalization.

Feature Extraction: Extract relevant features from each modality to represent the emotional content. For example, facial features may include landmarks, expressions, and head movements, while audio features may include pitch, intensity, and spectral characteristics.

Model Architecture Design: Design a deep learning architecture capable of integrating features from multiple modalities and predicting emotional states. This may involve architectures such as multi-input CNNs, recurrent neural networks (RNNs), or attention mechanisms.

Model Training: Train the multi-modal emotion recognition model using the preprocessed data. Utilize appropriate loss functions, optimizers, and regularization techniques to optimize the model parameters. Consider techniques like transfer learning or multi-task learning to leverage pre-trained models or jointly learn related tasks.

Model Evaluation: Evaluate the trained model using validation data to assess its performance in recognizing emotions across different modalities. Use evaluation metrics such as accuracy, precision, recall, F1-score, and confusion matrices to quantify the model's performance.

Hyperparameter Tuning: Fine-tune the model's hyperparameters (e.g., learning rate, batch size, architecture depth) to improve its performance and generalization ability.

Model Deployment: Deploy the trained model in a production environment, considering factors such as scalability, latency, and resource constraints. Use frameworks like TensorFlow Serving or ONNX Runtime for efficient model deployment.

Monitoring and Maintenance: Continuously monitor the deployed model's performance and retrain it periodically using new data to adapt to evolving patterns and maintain accuracy.

EVALUTION:

Accuracy: Measure the overall accuracy of the model in correctly predicting emotions across all modalities.

Precision and Recall: Calculate precision (the proportion of true positive predictions among all positive predictions) and recall (the proportion of true positive predictions among all actual positives) for each emotion class. This helps assess the model's ability to make correct predictions and avoid false positives or false negatives.

F1-Score: Compute the F1-score, which is the harmonic mean of precision and recall, to balance both metrics and provide a single measure of model performance.

Confusion Matrix: Generate a confusion matrix to visualize the model's predictions compared to the ground truth labels for each emotion class. This helps identify specific areas where the model may be struggling to accurately classify emotions.

Cross-Validation: Perform k-fold cross-validation to evaluate the model's performance on multiple subsets of the data. This helps assess the model's generalization ability and detect overfitting or underfitting.

ROC Curve and AUC: If applicable (e.g., for binary classification tasks), plot the Receiver Operating Characteristic (ROC) curve and calculate the Area Under the Curve (AUC) to assess the model's performance across different decision thresholds.

Error Analysis: Conduct error analysis to understand common patterns or types of errors made by the model. This can help identify areas for improvement and guide further model refinement.

User Studies: In some cases, conduct user studies or surveys to gather feedback from human annotators or end-users on the

model's performance and subjective impressions of emotion recognition accuracy.

Benchmarking: Compare the performance of your model against existing state-of-the-art approaches or baseline models to assess its relative effectiveness and identify areas where further improvements can be made.

CONCLUSION:

Training a multi-modal fusion model for emotion recognition involves integrating data from diverse modalities like images, audio, and text. Modality-specific models capture unique emotional cues, while fusion architectures combine these for enhanced prediction accuracy. Joint training optimizes parameters across modalities, ensuring seamless integration of information. Careful selection of loss functions and metrics ensures model robustness and accuracy. Fine-tuning further refines parameters to enhance performance. In conclusion, training multi-modal fusion models holds promise for advancing emotion recognition technology.

SOURCE CODE:

```
from keras.preprocessing.image import img_to_array
from flask import Flask, render_template, Response,request,session,flash,url_for,redirect,jsonify
import cv2
import numpy as np
from flask_pymongo import PyMongo
from passlib.hash import pbkdf2_sha256
import os
from flask_login import LoginManager, UserMixin, login_user, login_required, logout_user, current_user
from keras.models import load_model
from mtcnn.mtcnn import MTCNN
import librosa
import wave
import pyaudio
import logging
from logging.handlers import RotatingFileHandler


app = Flask(_name_)
```

```
app.config['SECRET_KEY'] = '123456'
app.config['MONGO_URI'] = 'mongodb://localhost:27017/your_database_name'

# Configure logging
handler = RotatingFileHandler('app.log', maxBytes=10000, backupCount=1)
handler.setLevel(logging.INFO)
formatter = logging.Formatter('%(asctime)s - %(levelname)s - %(message)s')
handler.setFormatter(formatter)
app.logger.addHandler(handler)



emotion_classifier = load_model("data_mini_XCEPTION.106-0.65.hdf5", compile=False)
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
Emotions = ["angry", "disgust", "scared", "happy", "sad", "surprised", "neutral"]

model = load_model('mymodel.h5')
```

```
wav_filepath=r"C:\Users\ELDERGOD MAN 24\PycharmProjects\pythonProject8\EMAN\recorded_audio.wav"
output_filename = 'recorded_audio.wav'


detector = MTCNN()


genderProto = "gender_deploy.prototxt" #structure of the model
genderModel = "gender_net.caffemodel" #gender prediction model


MODEL_MEAN_VALUES = (78.4263377603, 87.7689143744, 114.895847746)
genderList = ['Male', 'Female']


face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
genderNet = cv2.dnn.readNet(genderModel, genderProto)


def detect_faces(frame):
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))
    for (x, y, w, h) in faces:
```

```
            roi = gray[y:y + h, x:x + w]
            roi = cv2.resize(roi, (48, 48))
            roi = roi.astype("float") / 255.0
            roi = img_to_array(roi)
            roi = np.expand_dims(roi, axis=0)
            preds = emotion_classifier.predict(roi)[0]
            label = Emotions[preds.argmax()]
            cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)
            cv2.putText(frame, f"{label}", (x, y - 5),
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 0))
            app.logger.info(f"Detected emotion: {label}")
    return frame


def generate_frames():
    camera = cv2.VideoCapture(0)
    while True:
        ret, frame = camera.read()
        if not ret:
            break
        else:
            frame_with_faces = detect_faces(frame)
            ret, buffer = cv2.imencode('.jpg', frame_with_faces)
            frame = buffer.tobytes()
```

```
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame +
b'\r\n')


def detect_gen(frame):
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    faces = face_cascade.detectMultiScale(image=gray,
scaleFactor=1.3, minNeighbors=5)
    for (x, y, w, h) in faces:
        roi = gray[y:y + h, x:x + w]
        blob = cv2.dnn.blobFromImage(roi, 1.0, (227, 227),
MODEL_MEAN_VALUES, swapRB=False)
        genderNet.setInput(blob)
        genderPreds = genderNet.forward()
        gender = genderList[genderPreds[0].argmax()]
        cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)
        cv2.putText(frame, f'{gender}', (x, y - 5),
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255))
    return frame


def gender_frames():
    camera = cv2.VideoCapture(0)
    while True:
        ret, frame = camera.read()
```

```
        if not ret:
            break
        else:
            frame_with_faces = detect_gen(frame)
            ret, buffer = cv2.imencode('.jpg', frame_with_faces)
            frame = buffer.tobytes()
            yield (b'--frame\r\n'
                   b'Content-Type: image/jpeg\r\n\r\n' + frame +
b'\r\n')


def extract_mfcc(wav_file_name):
    # This function extracts mfcc features and obtain the mean
of each dimension
    # Input : path_to_wav_file
    # Output: mfcc_features'''
    y, sr = librosa.load(wav_file_name)
    mfccs = np.mean(librosa.feature.mfcc(y=y, sr=sr,
n_mfcc=40).T, axis=0)


    return mfccs


def predict(model, wav_filepath):
    emotions = {1: 'neutral', 2: 'calm', 3: 'happy', 4: 'angry', 5:
'sad', 6: 'fearful', 7: 'disgust', 8: 'surprised'}
```

```
    test_point = extract_mfcc(wav_filepath)
    test_point = np.reshape(test_point, newshape=(1, 40, 1))
    predictions = model.predict(test_point)
    text=emotions[np.argmax(predictions[0]) + 1]

    return text

def record_audio(output_filename, duration=5, sample_rate=44100, chunk_size=1024, audio_format=pyaudio.paInt16, channels=1):
    # Initialize PyAudio
    audio = pyaudio.PyAudio()

    # Open microphone stream
    stream = audio.open(format=audio_format, channels=channels, rate=sample_rate, input=True, frames_per_buffer=chunk_size)

    print("Recording...")

    frames = []

    # Record audio for the specified duration
    for i in range(int(sample_rate / chunk_size * duration)):
```

```
        data = stream.read(chunk_size)
        frames.append(data)

    print("Finished recording.")

    # Stop and close the stream
    stream.stop_stream()
    stream.close()
    audio.terminate()

    # Save the recorded audio to a .wav file
    with wave.open(output_filename, 'wb') as wf:
        wf.setnchannels(channels)
        wf.setsampwidth(audio.get_sample_size(audio_format))
        wf.setframerate(sample_rate)
        wf.writeframes(b".join(frames))

    print(f"Audio saved as {output_filename}")
    #predict(model, wav_filepath)



mongo = PyMongo(app)
login_manager = LoginManager(app)
```

```
login_manager.login_view = 'signin'

# Define the User class for Flask-Login
class User(UserMixin):
    def _init_(self, user_id):
        self.id = user_id

# Callback to reload the user object
@login_manager.user_loader
def load_user(user_id):
    return User(user_id)



@app.route('/')
def index():
    return render_template('index.html')

@app.route('/signin', methods=['GET', 'POST'])
def signin():
    if request.method == 'POST':
        users = mongo.db.users
        existing_user = users.find_one({'username':
request.form['username']})
```

```
        if existing_user and pbkdf2_sha256.verify(request.form['password'], existing_user['password']):
            user_obj = User(existing_user['_id'])
            login_user(user_obj)
            flash('Login successful!', 'success')
            return redirect(url_for('dash'))
        else:
            flash('Invalid username or password. Please try again.', 'danger')

    return render_template('signin.html')



@app.route('/signup', methods=['GET', 'POST'])
def signup():
    if request.method == 'POST':
        users = mongo.db.users
        existing_user = users.find_one({'username': request.form['username']})

        if existing_user is None:
```

```
        hashed_password = pbkdf2_sha256.hash(request.form['password'])
        user_id = users.insert_one({'username': request.form['username'], 'password': hashed_password,'email':request.form['email']}).inserted_id
        user_obj = User(user_id)
        login_user(user_obj)
        flash('Account created successfully!', 'success')
        return redirect(url_for('signin'))
    else:
        flash('Username already exists. Please choose a different one.', 'danger')

  return render_template("signup.html")



@app.route('/dash')
@login_required
def dash():
  return render_template('dash.html', username=current_user.id)


@app.route('/eman')
def eman():
```

```
    return render_template('eman.html')

@app.route('/about')
def about():
    return render_template('about.html')

@app.route('/feedback', methods=['GET', 'POST'])
def feedback():
    if request.method == 'POST':
        name = request.form.get('name')
        email = request.form.get('email')
        subject = request.form.get('subject')
        message = request.form.get('message')

        data = {
            'name': name,
            'email': email,
            'subject': subject,
            'message': message
        }
        try:
            mongo.db.feedback_report.insert_one(data)
            flash("Feedback Submitted Successfully", 'success')
```

```
        except Exception as e:
            flash("Error submitting form", 'danger')

        return redirect(url_for('dash'))

    # If it's a GET request, you might want to render a template or return a response.
    return render_template('feedback.html')

@app.route('/contact')
def contact():
    return render_template('contact.html')

@app.route('/video_feed')
def video_feed():
    return Response(generate_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')

@app.route('/video_feed2')
def video_feed2():
    return  Response(gender_frames(),  mimetype='multipart/x-mixed-replace; boundary=frame')

@app.route('/get_text', methods=['POST'])
```

```
def get_text():
    # Here you can generate or fetch the text dynamically
    record_audio(output_filename)
    text=predict(model, output_filename)
    print(text)
    return jsonify({'text': text})

@app.route('/logout')
@login_required
def logout():
    logout_user()
    flash('You have been logged out.', 'success')
    return redirect(url_for('index'))

if _name_ == '_main_':
    app.run(debug=True)
```

Research Papers:

1.A Review of Face Recognition Technology

Authors: LIXIANG LI, XIAOHUI MU1, SIYING LI, AND HAIPENG PENG.

2.A Survey of Deep Learning-Based Multimodal Emotion Recognition: Speech, Text, and Face

Authors:Hailun Lian, Cheng Lu, Sunan Li, Yan Zhao, Chuangao Tang and Yuan Zong .

3. Multimodal Emotion Recognition Model Based on a Deep Neural Network with Multiobjective Optimization

Authors:Mingyong Li , Xue Qiu , Shuang Peng, Lirong Tang, Qiqi Li, Wenhui Yang, and Yan Ma.

4.Exploiting EEG Signals and Audiovisual Feature Fusion for Video Emotion Recognition

Authors:XING, HUI ZHANG1, KEJUN ZHANG, LEKAI ZHANG, XINDA WU,XIAOYING SHI , SHANGHAI YU AND SANYUAN ZHANG1

5. Speech Emotion Recognition Using Deep Learning Techniques: A Review

Authors:AMIN KHALIL, EDWARD JONES 2, MOHAMMAD INAYATULLAH BABAR 1,TARIQULLAH JAN1, MOHAMMAD HASEEB ZAFAR 3, AND THAMER ALHUSSAIN4

6.End-to-End Speech Emotion Recognition With Gender Information

Author:TING-WEI SUN

7.Deep Learning Approaches for Bimodal Speech Emotion Recognition: Advancements, Challenges, and a Multi-Learning Model

Author:SAMUEL KAKUBA 1,2, ALWIN POULOSE 3, AND DONG SEOG HAN

8.DEEP MULTIMODAL LEARNING FOR AUDIO-VISUAL SPEECH RECOGNITION

Authors:Youssef Mroueh , Etienne Marcheret , Vaibhava Goel

9.Multimodal Learning Using 3D Audio-Visual Data for Audio-Visual Speech Recognition

Authors:Rongfeng Su, Lan Wang,Xunying Liu

10.Multimodal and Temporal Perception of Audio-visual Cues for Emotion Recognition

Authors:Esam Ghaleb, Mirela Popa, and Stylianos Asteriadis

11.Designing a Multimodal Corpus of Audio-Visual Speech using a High-Speed Camera

Authors:Alexey Karpov, Andrey Ronzhin, Irina Kipyatkova

12. DEEP LEARNING FOR ROBUST FEATURE GENERATION IN AUDIOVISUAL EMOTION RECOGNITION

Authors:Yelin Kim, Honglak Lee, and Emily Mower Provost

Websites and Online Resources:

OpenFace: The official website for OpenFace provides documentation, tutorials, and resources for understanding and using the toolkit for facial behavior analysis.

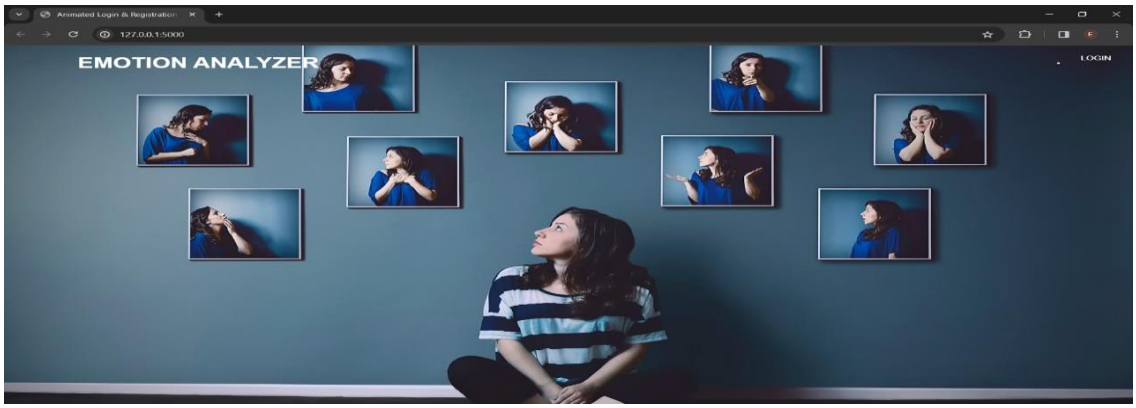
TensorFlow and PyTorch: The official websites for TensorFlow and PyTorch offer extensive documentation, tutorials, and examples for building deep learning models, which can be applied to multi-modal emotion recognition tasks.

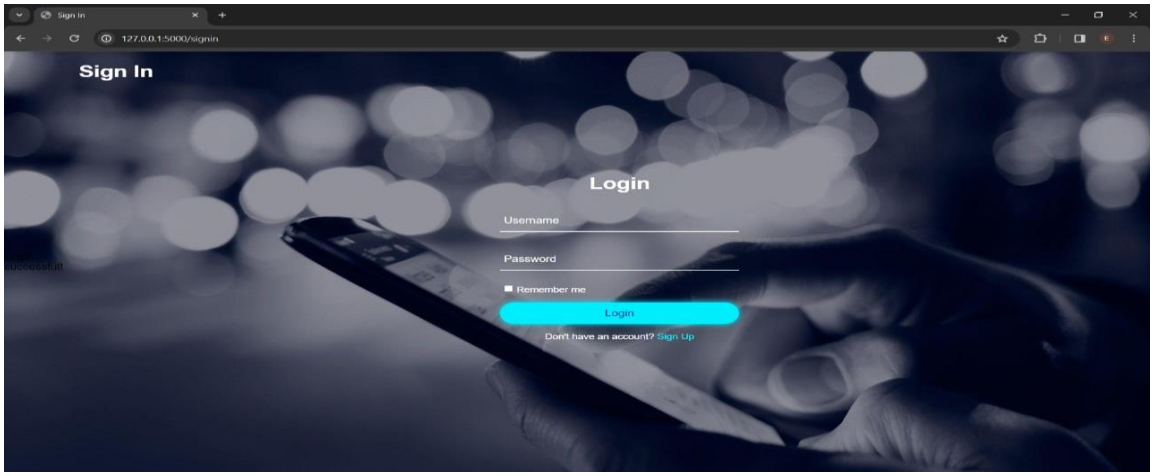
Librosa: The Librosa website provides documentation and examples for using the library to extract audio features, such as MFCCs, which are crucial for multi-modal emotion recognition.

Speech-to-Text Services: Websites for Google Cloud Speech-to-Text, Microsoft Azure Speech Services, and IBM Watson Speech to Text offer documentation and resources for integrating speech recognition capabilities into multi-modal emotion recognition systems.

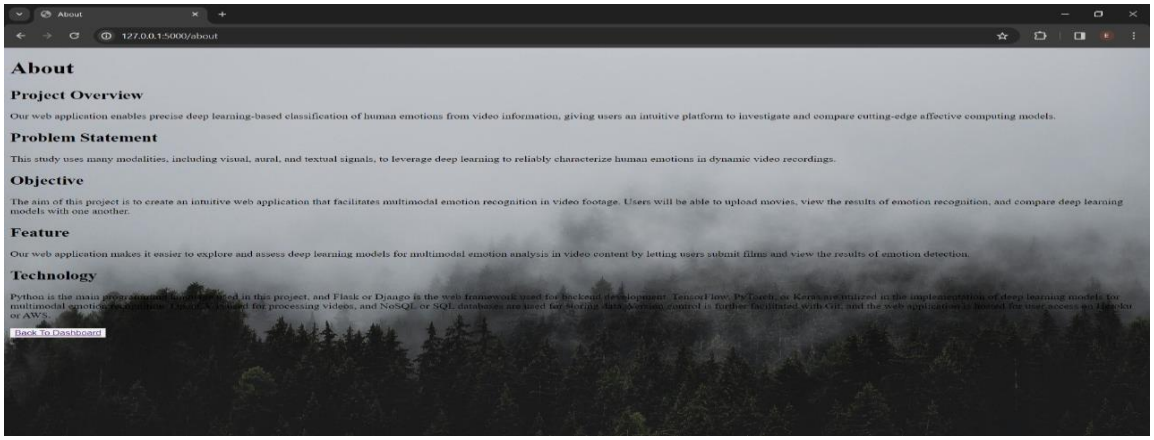
University Repositories: Explore digital repositories of universities, such as their library websites or institutional repositories, to find theses and dissertations related to emotion recognition, affective computing, and deep learning.

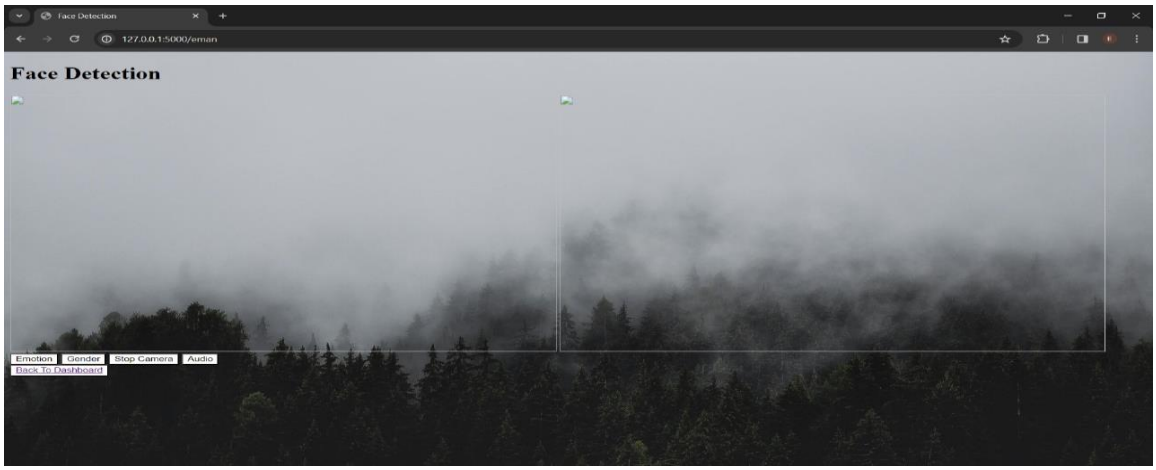
GitHub hosts numerous repositories related to multi-modal emotion recognition, including implementations of deep learning models, datasets, and pre-trained models. You can explore these repositories to find code, examples, and resources for your project.

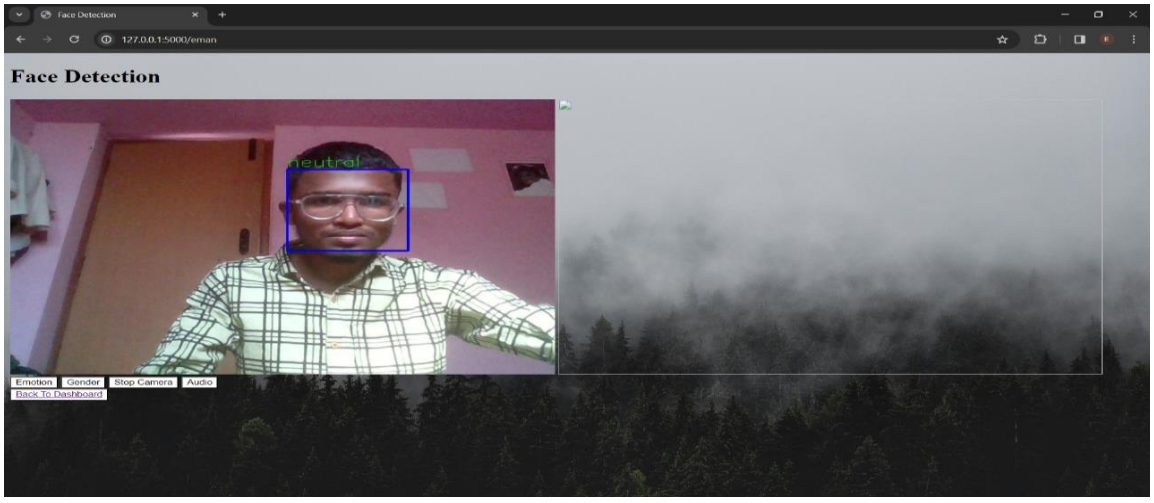

EMOTION ANALYZER
LOGIN

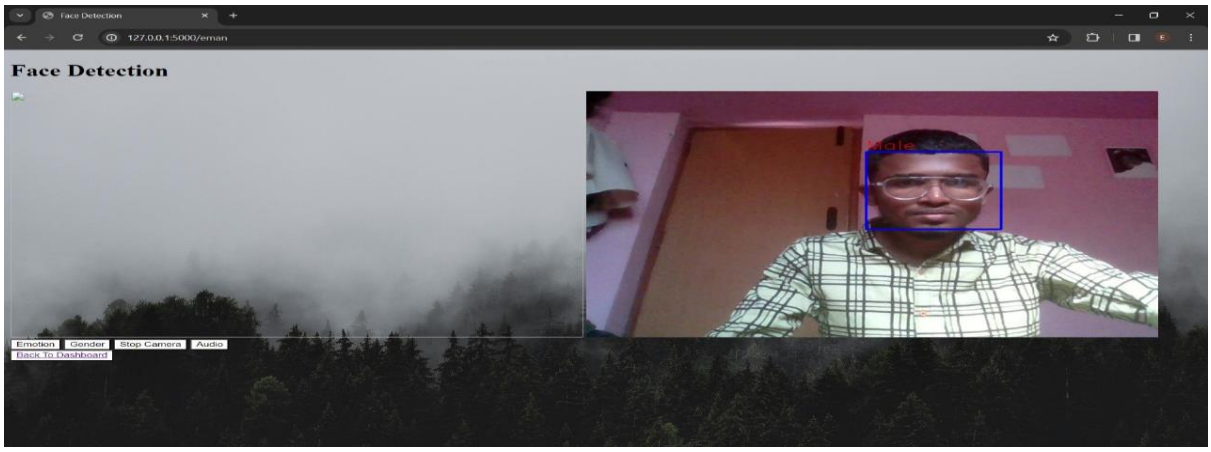

Sign In
Login
Username
Password
Remember me
Login
Don't have an account? Sign Up

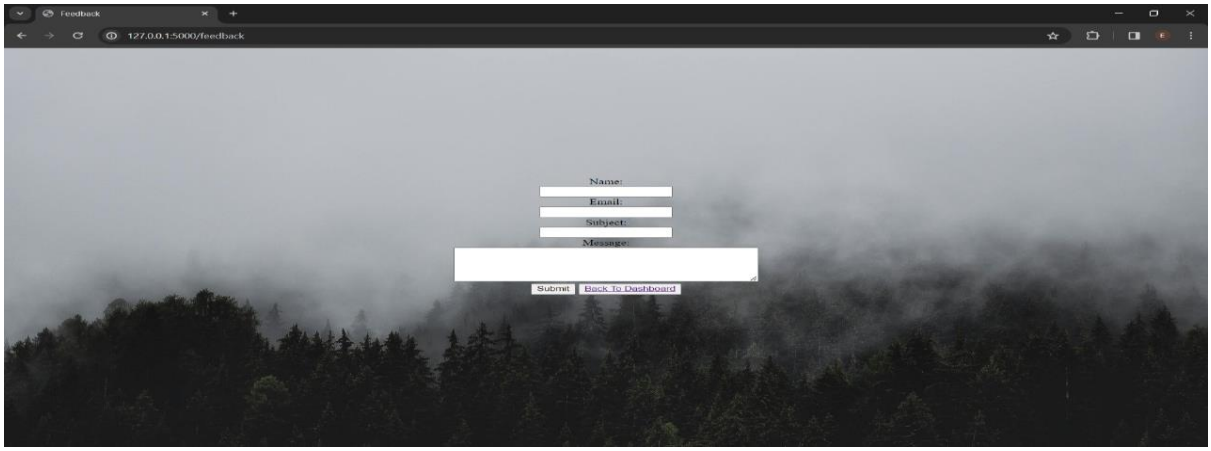

Dashboard
Home
About
Contact
Feedback
Eman
LOGOUT
In our organization, we are proud to present the Multimodal Emotion Recognition for Video Content project, a pioneering endeavor in the field of deep learning. This groundbreaking challenge aims to accurately analyze and classify human emotions expressed in video content through the integration of visual, auditory, and textual cues. Leveraging cutting-edge deep learning techniques, our project strives to push the boundaries of emotion recognition, paving the way for enhanced human-computer interaction and affective computing applications. Join us on this journey as we delve into the intricate realm of emotions and embark on a transformative exploration of multimedia content analysis.

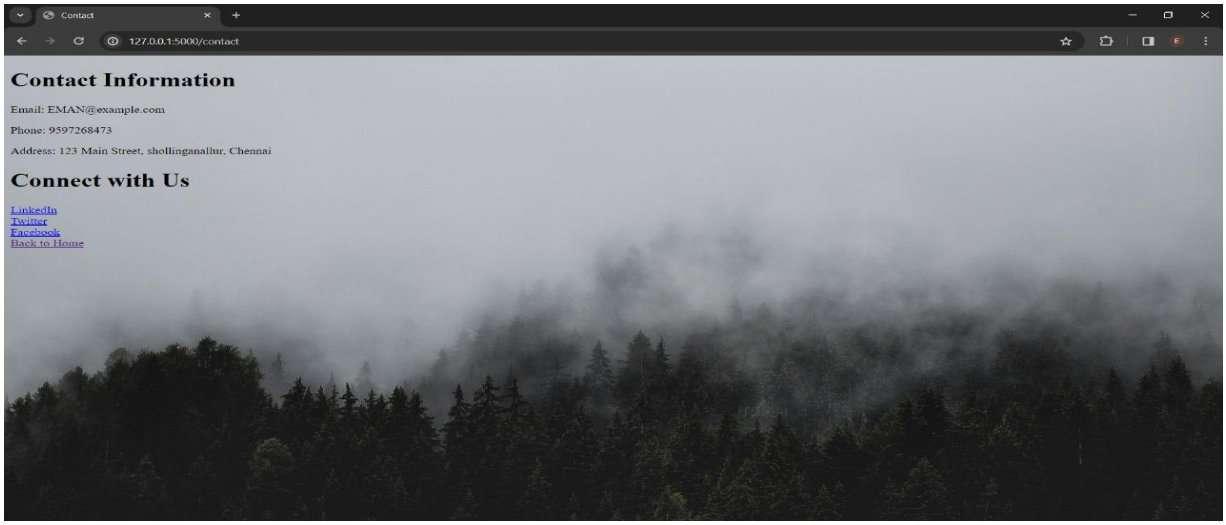

About
127.0.0.1:5000/about
About
Project Overview
Our web application enables precise deep learning-based classification of human emotions from video information, giving users an intuitive platform to investigate and compare cutting-edge affective computing models.
Problem Statement
This study uses many modalities, including visual, aural, and textual signals, to leverage deep learning to reliably characterize human emotions in dynamic video recordings.
Objective
The aim of this project is to create an intuitive web application that facilitates multimodal emotion recognition in video footage. Users will be able to upload movies, view the results of emotion recognition, and compare deep learning models with one another.
Feature
Our web application makes it easier to explore and assess deep learning models for multimodal emotion analysis in video content by letting users submit films and view the results of emotion detection.
Technology
Back To Dashboard


Face Detection
127.0.0.1:5000/eman
Face Detection
Emotion
Gender
Stop Camera
Audio
Back To Dashboard


Face Detection
127.0.0.1:5000/eman
Face Detection
neutral
Emotion
Gender
Stop Camera
Audio
Back To Dashboard


Face Detection
127.0.0.1:5000/eman
Face Detection
Male
Emotion
Gender
Stop Camera
Audio
Back To Dashboard


Feedback
127.0.0.1:5000/feedback
Name:
Email:
Subject:
Message:
Submit
Back To Dashboard


Contact
127.0.0.1:5000/contact
Contact Information
Email: EMAN@example.com
Phone: 9597268473
Address: 123 Main Street, shollinganallur, Chennai
Connect with Us
LinkedIn
Twitter
Facebook
Back to Home