FACE EMOTION RECOGNITION FOR VISUALLY IMPAIRED PEOPLE USING TRANSFER LEARNING

Mini Project Report

Submitted in partial fulfillment of the requirements for

the award of degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

of

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

by

ANANDHU T G *(MAC21CS009)*

AREENA AJI *(MAC21CS017)*

JITHIN K A *(MAC21CS027)*

Department of Computer Science & Engineering

Mar Athanasius College of Engineering

Kothamangalam

JULY 2024

FACE EMOTION RECOGNITION FOR VISUALLY IMPAIRED PEOPLE USING TRANSFER LEARNING

Mini Project Report

Submitted in partial fulfillment of the requirements for

the award of degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

of

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

by

ANANDHU T G *(MAC21CS009)*

AREENA AJI *(MAC21CS017)*

JITHIN K A *(MAC21CS027)*

Department of Computer Science & Engineering

Mar Athanasius College of Engineering

Kothamangalam

JULY 2024

DECLARATION

We, undersigned hereby declare that the project report - Face Emotion Recognition for Visually Impaired People using Transfer Learning submitted for partial fulfillment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by us under supervision of Project coordinator. This submission represents the ideas in our own words and where ideas or words of others have been included, we have adequately and accurately cited and referenced the original sources. We also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in our submission. We understand that any violation of the above will be a cause for disciplinary action by the institute or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Date: 27.04.2024 Anandhu T G

Kothamangalam Areena Aji

Jithin K A

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MAR ATHANASIUS COLLEGE OF ENGINEERING

KOTHAMANGALAM

CERTIFICATE

This is to certify that the report entitled **FACE EMOTION RECOGNITION FOR VISUALLY IMPAIRED PEOPLE USING TRANSFER LEARNING** *submitted by* **Anandhu T G (MAC21CS009) , Areena Aji (MAC21CS017) and Jithin K A (MAC21CS027)** *towards partial fulfillment of the requirement for the award of Degree of Bachelor of Technology in Computer Science and Engineering from APJ Abdul Kalam Technological University for JULY 2024 is a bonafide record of the project carried out by them under our supervision and guidance.*

Prof. Sukanyathara J
Project Guide

Prof. Rotney Roy Meckamalil
Project Coordinator

Internal Examiner(s)

External Examiner(s)

Date:

Dept. Seal

ACKNOWLEDGEMENT

First and foremost, we sincerely thank the 'God Almighty' for his grace for the successful and timely completion of the project.

We express our sincere gratitude and thanks to Dr. Bos Mathew Jos, Principal and Prof. Joby George, Head of the Department for providing the necessary facilities and their encouragement and support.

We owe special thanks to the project guide Prof. Sukanyathara J. and project co-ordinator Prof. Rotney Roy Meckamalil for their corrections, suggestions and sincere efforts to coordinate the project under a tight schedule.

We express our sincere thanks to staff members in the Department of Computer Science and Engineering who have taken sincere efforts in guiding and correcting us in conducting this project.

Finally, we would like to acknowledge the heartfelt efforts, comments, criticisms, co-operation and tremendous support given to us by our dear friends during the preparation of the project and also during the presentation without which this work would have been all the more difficult to accomplish.

ABSTRACT

Title: Face Emotion Recognition for Visually Impaired People using Transfer Learning

Visually Impaired People often face limitations in social interactions, specifically at discerning emotional cues. The proposed framework tackles this issue head-on by devising a Facial Emotion Recognition system, by employing an advanced Transfer Learning approach within Convolutional Neural Networks (CNNs). By leveraging the rich datasets from FER 2013 and CK+, our model transcends the limitations of traditional emotion recognition methods. Transfer learning allows the model to benefit from pre-trained knowledge on vast datasets, making it more efficient and effective in capturing complex facial features associated with different emotions. This approach offers better accuracy and generalization capabilities than other conventional methods. The intricacies of facial expressions are comprehensively captured during training, enabling the system to not only identify individuals but also interpret nuanced changes in their emotional states throughout conversations. The innovative integration of an audio output system ensures a seamless and accessible experience for visually impaired users, facilitating a deeper understanding of social dynamics.Emphasizing transfer learning, enables creation of an efficient and robust framework that not only revolutionizes emotional understanding for visually impaired individuals but also sets a new standard in the field, showcasing the superior performance achievable through advanced machine learning techniques. It represents a crucial step towards bridging the social gap for the visually impaired, promoting inclusivity, independence, and safety in their daily lives.

Table of Contents

List of Figures

List of Abbreviations

FER	Face Emotion Recognition
CNN	Convolutional Neural Network
GPU	Graphics Processing Unit
TPU	Tensor Processing Unit
API	Application Programming Interface
TFX	TensorFlow Extended
PyPI	Python Package Index
OpenCV	Open Source Computer Vision Library
ReLU	Rectified Linear Unit
SDLC	Software Development Life Cycle
VIP	Visually Impaired People

Chapter 1

INTRODUTION

Project Outline

Project Outline: Real-time Facial Emotion Recognition System

I. Introduction

This project focuses on developing a real-time facial emotion recognition system [2]. The ability to accurately detect and classify facial expressions in real-time has numerous applications, including assistive technologies for visually impaired individuals and human-computer interaction systems. The objective is to create a system that utilizes computer vision techniques and machine learning algorithms to recognize and classify emotions from live webcam feeds. The system will be capable of processing facial images in real-time and providing audio output to convey the detected emotions to users.

II. Literature Review

A comprehensive review of existing techniques and approaches for facial emotion recognition will be conducted. This will include an overview of computer vision algorithms for facial feature extraction and classification, as well as machine learning models commonly used for emotion recognition tasks. The review will also encompass studies on real-time emotion detection systems and their applications in various domains.

III. Methodology

The project's methodology will involve several stages, starting with data collection and preprocessing. Annotated datasets containing facial images with labeled emotions will be used to train and fine-tune the emotion recognition model. Computer vision techniques, such as Haar cascade classifiers for face

detection, will be utilized to extract facial features. Machine learning models, including convolutional neural networks (CNNs) and deep learning architectures, will be employed for emotion classification. The system will be implemented using Python and TensorFlow for model development and deployment.

IV. System Architecture

The system architecture will be outlined, highlighting the components involved and their interactions. The front end will include modules for capturing live webcam feeds and displaying real-time emotion predictions. The back end will consist of the emotion recognition model and audio output components. The data flow and processing steps within the system will be described to provide a comprehensive understanding of the system's operation.

V. Implementation Details

This section will delve into the implementation specifics of the project. It will cover the data collection process, annotation procedures, and preprocessing techniques applied to the collected data. The configuration and training of the emotion recognition model will be described, along with the integration of real-time webcam feeds for emotion prediction. The audio output functionality for conveying emotions to users will also be explained.

VI. Experimental Evaluation

The project's performance will be evaluated using appropriate metrics for facial emotion recognition, including accuracy, precision, and recall. Real-time testing will be conducted to assess the system's performance under various conditions and environments. The results will be analyzed, and the findings will be discussed, including strengths and limitations of the system in terms of accuracy, efficiency, and real-world applicability.

VII. Conclusion and Future Work

The project's conclusions will summarize the achievements and contributions made in real-time facial emotion recognition. The potential applications and impact of the developed system will be discussed, along with suggestions for future enhancements and research directions. The project aims to contribute to the advancement of assistive technologies and human-computer interaction systems by providing a reliable and efficient facial emotion recognition solution.

VIII. References

A comprehensive list of references used throughout the project will be provided.

Note: This condensed version of the project outline has been formatted to fit a single page, providing an overview of the project's key aspects while maintaining its main structure.

Chapter 2

BACKGROUND

2.1 Python

Python is a versatile and widely-used programming language known for its simplicity, readability, and extensive ecosystem of libraries and frameworks. It is an interpreted, high-level language that emphasizes code readability and developer productivity, making it popular among beginners and experienced programmers alike.

One of Python's key strengths is its rich ecosystem of libraries and frameworks for various domains, including data science, machine learning, web development, and automation. The Python Package Index (PyPI) hosts over 300,000 packages, providing developers with a vast array of tools and resources to streamline development workflows.

In the context of machine learning and artificial intelligence, Python is the de facto language of choice for researchers, developers, and data scientists. Libraries such as TensorFlow, PyTorch, and scikit-learn offer powerful tools for building and training machine learning models, while frameworks like Django and Flask simplify the development of web-based applications.

Python's simplicity and readability contribute to its widespread adoption across industries and domains. Its clean and expressive syntax allows developers to write concise and maintainable code, reducing development time and effort. Additionally, Python's strong community support and active development community ensure ongoing innovation and improvement.

Overall, Python's versatility, ease of use, and extensive ecosystem make it an ideal choice for a wide range of applications, from simple scripting tasks to complex

machine learning projects. Its popularity continues to grow, cementing its position as one of the most widely-used programming languages in the world.

2.1.1 TensorFlow

TensorFlow [9] is a leading open-source machine learning framework developed and maintained by Google. It provides a comprehensive ecosystem of tools, libraries, and resources for building and training deep learning models [1]. TensorFlow's versatility makes it suitable for a wide range of applications, including computer vision, natural language processing, and reinforcement learning.

At its core, TensorFlow operates using a dataflow graph paradigm, where computational tasks are represented as nodes in a graph. These nodes are interconnected by edges that represent the flow of data between operations. This allows for efficient execution of computations across multiple devices, including CPUs, GPUs, and TPUs (Tensor Processing Units).

One of TensorFlow's key features is its flexibility and scalability. It supports both high-level APIs, such as Keras, for rapid prototyping and low-level APIs for fine-grained control over model architecture and training. TensorFlow's extensive documentation and community support make it accessible to both beginners and experienced practitioners.

In addition to its core functionality, TensorFlow provides a suite of specialized libraries and tools. TensorFlow Hub offers a repository of pre-trained models and modules that can be easily integrated into custom workflows. TensorFlow Extended (TFX) provides end-to-end machine learning pipelines for production deployment and monitoring.

Overall, TensorFlow's robustness, performance, and extensive feature set make it a go-to choice for researchers, developers, and businesses alike for building state-of-the-art machine learning solutions.

2.1.2 OpenCV

OpenCV (Open Source Computer Vision Library) [8] is an open-source computer vision and machine learning software library designed to provide a common infrastructure for computer vision applications. Originally developed by Intel, OpenCV has since become a community-driven project supported by a large and active user base.

OpenCV offers a wide range of functionalities for image and video analysis, including image processing, object detection and tracking, feature extraction, and stereo vision. It provides a collection of over 2500 optimized algorithms that can be used to perform various computer vision tasks efficiently.

One of the key strengths of OpenCV is its cross-platform compatibility, with support for multiple programming languages, including C++, Python, Java, and MATLAB/Octave. This makes it accessible to developers working across different platforms and environments.

OpenCV's modular design allows developers to leverage specific modules based on their project requirements, ensuring lightweight deployments and efficient resource utilization. Its extensive documentation and tutorials make it easy for developers to get started with building computer vision applications.

Overall, OpenCV's versatility, performance, and ease of use make it a popular choice for both academic research and industrial applications in fields such as robotics, augmented reality, surveillance, and medical imaging.

2.1.3 Pyttsx3

Pyttsx3 is a Python library for text-to-speech (TTS) conversion, providing a simple and intuitive interface for synthesizing speech from text. It is built on top of

the cross-platform text-to-speech engine, SAPI5, which is widely supported on Windows operating systems.

Pyttsx3 offers several features for customizing speech synthesis, including voice selection, rate adjustment, and volume control. It supports multiple languages and accents, allowing for the generation of speech in different languages and regional dialects.

One of the key advantages of Pyttsx3 is its ease of use and seamless integration with Python applications. Developers can quickly incorporate speech synthesis capabilities into their projects without the need for complex setup or configuration.

Pyttsx3's compatibility with various Python environments, including Jupyter notebooks, IDEs, and command-line interfaces, makes it suitable for a wide range of use cases. It can be used to add voice-based interfaces to applications, create audio notifications, or generate spoken content for accessibility purposes.

Overall, Pyttsx3 provides a convenient and efficient solution for adding text-to-speech functionality to Python applications, enabling developers to create more engaging and accessible user experiences.

2.2 MobileNetV2

MobileNetV2 is a convolutional neural network (CNN) architecture [3] specifically designed for resource-constrained environments, such as mobile and embedded devices, where computational resources are limited. Introduced by Google in 2018, MobileNetV2 builds upon the success of its predecessor, MobileNet, by incorporating novel design strategies to improve both accuracy and efficiency.

The architecture of MobileNetV2 revolves around two key concepts: depthwise separable convolutions and linear bottlenecks. Depthwise separable convolutions

decompose the standard convolutional operation into two separate steps: depthwise convolution and pointwise convolution. This factorization significantly reduces the computational cost and number of parameters, making the network more lightweight while preserving its ability to capture complex features.

Linear bottlenecks are introduced to further enhance the efficiency of the network. These bottlenecks consist of a sequence of convolutional layers followed by a bottleneck layer with linear activation. The bottleneck layer reduces the number of channels, allowing the network to learn compact representations efficiently. This design enables MobileNetV2 to achieve a good balance between model size, computational cost, and accuracy.

Here is a high-level description of the MobileNetV2 model

1. Depthwise Separable Convolution:

   Depthwise separable convolution is a key component of MobileNetV2's architecture. It decomposes the standard convolution operation into two separate steps: depthwise convolution and pointwise convolution. Depthwise convolution applies a single convolutional filter per input channel, while pointwise convolution combines the output of depthwise convolution across all channels using 1x1 convolutions. This factorization reduces the number of computations and parameters, making the network more efficient.

2. Linear Bottleneck Residual Block:

   The linear bottleneck residual block consists of a sequence of convolutional layers followed by a bottleneck layer with linear activation. This design allows the network to capture rich feature representations while maintaining computational efficiency. The bottleneck layer reduces the number of channels, compressing the feature maps and enabling the network to learn compact representations efficiently.

3. Expansion and Contraction:

   MobileNetV2 employs a design strategy based on the concept of expansion and contraction. In the expansion phase, the number of channels is increased to capture richer feature representations. In the contraction phase, the number of channels is reduced to compress the feature maps and maintain computational efficiency. This design strategy enables MobileNetV2 to achieve high performance while minimizing computational overhead.

To fine-tune the MobileNetV2 model for facial emotion recognition, we adopt a transfer learning approach, capitalizing on its pre-trained weights from the ImageNet dataset. Initially, we load the MobileNetV2 architecture, excluding its classification head, while retaining its input and penultimate output layers. This preserves the learned features extracted from ImageNet's diverse images. Subsequently, we augment the model with custom layers to adapt it to our emotion recognition task.

We add two Dense layers, the first with 128 neurons followed by ReLU activation, serving as a feature extractor. The subsequent layer, comprising 64 neurons with ReLU activation, further refines the extracted features. Finally, a Dense layer with 7 neurons and softmax activation predicts the probability distribution over the seven emotion classes. This sequential architecture allows us to leverage the robust feature extraction capabilities of MobileNetV2 while tailoring the model to our specific task. By fine-tuning the pre-trained model in this manner, we expedite training and improve performance on our facial emotion recognition task, achieving more accurate and efficient predictions.

Chapter 3

REQUIREMENT SPECIFICATION

3.1 Methodology

The methodology for designing the facial emotion recognition system [2] involves several crucial steps and components. Below is an outline of the typical methodology used in the development of such a system:

1. **Data Collection:**

   - Gather facial image datasets containing labeled facial expressions from diverse sources, ensuring representation across various demographics and contexts.

   - Utilize publicly available datasets such as FER2013 or CK+ and consider augmenting the data to increase diversity and robustness.

   - Employ web scraping techniques or utilize APIs from platforms like Kaggle or Open Images to gather additional data if necessary.

2. **Data Preprocessing**

   - Perform preprocessing on the collected facial images to enhance data quality and consistency.

   - Resize images to a standard size to ensure uniformity in input dimensions for the neural network.

   - Normalize pixel values to a common scale to facilitate model convergence during training.

3. **Feature Extraction**

   - For face detection, utilize Haar Cascade Classifier (e.g., haarcascade frontalface) for face detection to localize and extract facial regions of interest.

   - Extract meaningful features from preprocessed facial images to capture

relevant information for emotion recognition.

- Explore methods such as facial landmarks detection or facial action unit analysis to extract additional facial attributes or expressions.

4. **Machine Learning Models**

   - Choose appropriate machine learning models for facial emotion recognition based on the complexity of the task and available resources.

   - Consider leveraging pre-trained models or transfer learning to expedite model training and improve performance, especially with limited data.

5. **Training and Evaluation**

   - Split the dataset into training, validation, and testing sets to train and evaluate the performance of the developed models.

   - Utilize metrics such as accuracy, precision, recall, and F1-score to assess model performance and identify areas for improvement.

6. **Model Integration and Deployment**

   - Integrate the trained facial emotion recognition model into a unified system capable of real-time inference. - Implement a user interface optimized for the visually impaired, leveraging audio signals as the primary mode of interaction to provide emotion predictions.

7. **Continuous Improvement**

   - Monitor the performance of the deployed system and collect user feedback for ongoing improvements. - Regularly update the models with new data to adapt to evolving facial expressions and improve accuracy over time. - Stay abreast of the latest research and advancements in facial emotion recognition to incorporate novel techniques and enhance system capabilities.

The overall architecture and design may vary based on specific project requirements, available resources, and the complexity of the analysis. It is important to iterate

and refine the design based on experimentation, evaluation, and feedback to build an effective and reliable facial emotion recognition system

3.1.1 Justification for this methodology

This model can be used when the requirements of the complete system are clearly defined and understood, like the case of this project where:

- Major requirements were evidently defined. However, some details evolved with time.
- There was a need to complete the project within a short time schedule.
- A new technology is being used or the resources with needed skill set are not available

The incremental model is much better equipped to handle change. Each incremental functionality is verified by the developer and hence the relative risk in managing large and complex projects is substantially reduced. In a nutshell, incremental SDLC provide plethora of advantages inducing:

- Generates working software quickly and early during the software life cycle
- This model is more flexible and less costly to change scope and requirements
- It is easier to test and debug during a smaller iteration
- Iteration process of the product being developed will be easy

3.2 System Analysis

Analysis is an important part of any project; if analysis is not done properly then the whole project moves in the wrong direction. By conducting an analysis the developer can get a brief idea of the challenges that they will be facing during the development of the project. It also provides a schedule for proper project work.

3.2.1 Feasibility Study

Feasibility study of the system is a very important stage during system design. Feasibility study is a test of a system proposal according to its workability impact on the organization, ability to meet user needs, and effective use of resources. Feasibility study decides whether the system is properly developed or not. There are five types of feasibility as mentioned below:

1. Technical Feasibility
2. Time Schedule feasibility
3. Operational feasibility
4. Implementation feasibility
5. Economic Feasibility

3.2.1.1 Technical Feasibility

Technical feasibility corresponds to determination of whether it is technically feasible to develop the software. Here those tools are considered, which will be required for developing the project. The tools, which are available, and tools, which will be required, are taken into account. Considering all above points and aspects it is observed that the cost incurred in developing this project from a technical perspective would not be too high. Thus, it is feasible for companies to develop this system.

3.2.1.2 Time Feasibility

Time feasibility corresponds to whether sufficient time is available to complete the project. Time feasibility is a measure of how reasonable the project timetable is. Given our technical expertise, are the project deadlines reasonable? Some projects are initiated with specific deadlines. It is necessary to determine whether the deadlines are mandatory or desirable.

Parameters considered are:

• Schedule of the project

• Time by which the project has to be completed

• Reporting period

• Requirement gathering

• Human resources available

Considering all the above factors it was decided that the allotted time that is 2 months was sufficient to complete the project.

3.2.1.3 Operational Feasibility

Operational feasibility corresponds to whether users are aware of the interface environment and sufficient resources are available or not. Parameters considered are people with a basic knowledge of computers would be able to use the system very effectively and easily, as the system would have an intuitive GUI. All the relevant necessary resources for implementing and operating this system are already present in office. Bearing in mind the above factor, it was observed that the cost would be incurred in developing this project from an operational standpoint would be low. Thus, it would be operational feasible for the company.

3.2.1.4 Implementation feasibility

Implementation feasibility is about basic infrastructure required to develop the system. Considering all the points below, it is feasible to develop a system.

Factors considered:

• All the minimum infrastructure facility required like PC, books, technical manuals are provided

• Proper guidance is provided

• All necessary data and files are provided

3.2.1.5 Economic Feasibility

Economic feasibility involves assessing the overall cost of the system. The proposed project utilizes Python for backend development, incorporating TensorFlow and OpenCV for facial emotion recognition. For the frontend user interface, technologies such as pyttsx3 for speech synthesis and Flask for web-based interaction are employed

3.3 Requirements Analysis and Specification

A complete understanding of software requirements is essential to the success of a web-development effort. No matter how well designed or well coded, a poorly analysed and specific program will disappoint users and bring grief to the developers. The requirement analysis task is the process of discovery, refinement, modification and specification. The software scope, initially established by the system engineer and refined during project planning, is refined in detail. Models of the required data, information and control flow, and operational behaviour are created. Alternative solutions are analysed and various project elements. This application should be developed with an aim to detect facial emotions accurately and easily.

3.3.1 Functional Requirements

The following is the desired functionality of the new system:

• Users can input real-time facial expressions captured through a live camera feed.

• Users should be able to predict the emotions conveyed in the input facial expressions.

• Users receive output predictions in audio format, facilitating accessibility for visually impaired individuals.

3.3.2 Non Functional requirements

It specifies the quality attribute of a project. They judge the application based on responsiveness, usability, security, portability and other non-functional standards that are critical to the success of the project.

• Availability: The system should remain operational in any day and place

• Accuracy: There is a need to optimize the system to ensure more accurate results

3.4 Software Requirements

- Operating System: Android, MacOS, or Windows for compatibility with various user devices.
- Development Environment: Python programming language and Anaconda distribution for managing dependencies and libraries used in the project.

3.5 Hardware Requirements

- Camera: A webcam or integrated camera capable of capturing real-time facial expressions.
- Speaker/Earphone: An audio output device such as speakers or earphones to receive audio predictions.
- Processor: An Intel Core i5 processor or above to handle real-time processing of facial emotion recognition algorithms.
- Display: A monitor with a minimum size of 13 inches for displaying system output and user interfaces.

3.6 Summary

In this chapter the details of the requirements are specified. All the feasibility factors are considered in this chapter. Technical, time, operational, economic, implementation feasibility discussions are also included. The functional and non-functional requirements of the project are also elaborated.

Chapter 4

DESIGN AND IMPLEMENTATION

In this chapter the discussion is about the architecture of the proposed system which have the facility to detect emotions and identify saved faces. Input to the system can be videos or images. Output from the system will be in two formats of sound and text.

Design Part: The system employs a modular architecture to facilitate real-time face emotion recognition for visually impaired individuals [5]. It integrates modules for video input processing, face detection, emotion classification using a MobileNetV2-based pre-trained model, and person identification. These modules interact seamlessly to provide accurate and timely feedback to users. The architecture ensures scalability and adaptability for future enhancements.

Implementation Part: The implementation utilizes Python for its flexibility and robustness. OpenCV is employed for video processing, enabling efficient frame extraction and face detection. Transfer learning techniques are applied to MobileNetV2 for emotion classification. Person identification is achieved through custom algorithms integrated with facial recognition libraries. The system's output is delivered through both audio feedback via a lightweight end device and text messages in the terminal, ensuring accessibility and usability for visually impaired users.

4.1 Work Flow Diagram

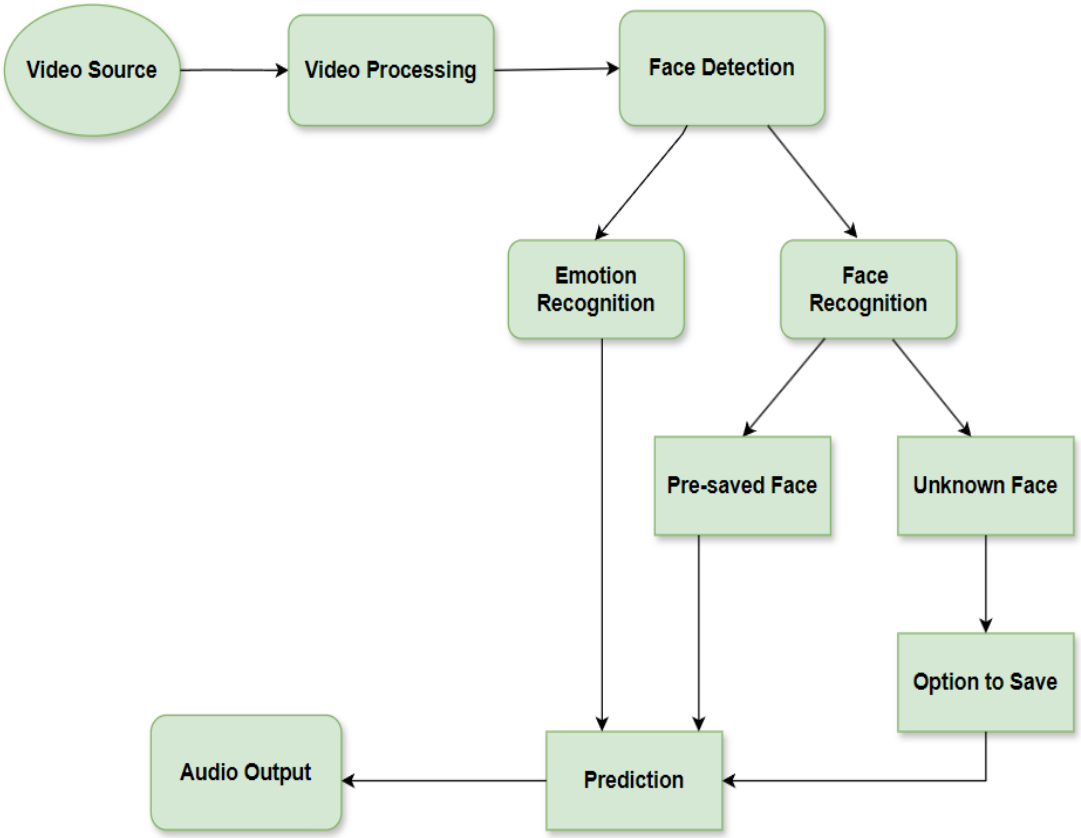

Figure 4.1: Video processing and recognition work flow diagram

The flow diagram outlines the process of real-time Face Emotion Recognition (FER) system for visually impaired users. It starts with receiving video input, proceeds to process the video for frame extraction and face detection. Detected faces undergo emotion recognition and person identification. The system then generates audio feedback for users through a lightweight device and displays text messages for monitoring and debugging purposes, ensuring comprehensive and accessible emotion analysis.

4.2 Application Diagram

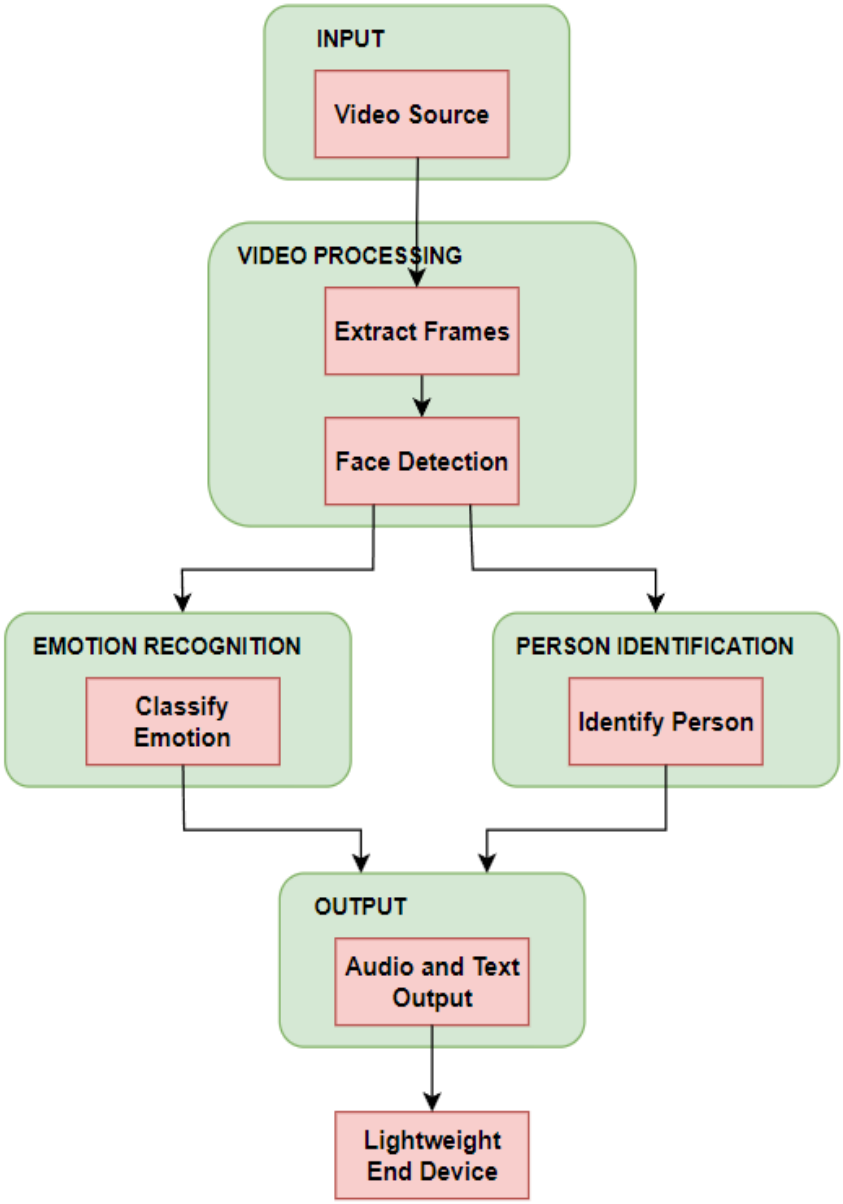

Figure 4.2: Application diagram of FER system

The application diagram visualizes a real-time Face Emotion Recognition (FER) system for visually impaired users. It encompasses modules for video input, processing, emotion recognition, and person identification. Outputs include audio feedback for users and text-based messages for monitoring. This diagram ensures efficient and accessible emotion analysis for diverse user needs.

4.3 Use-Case Diagram

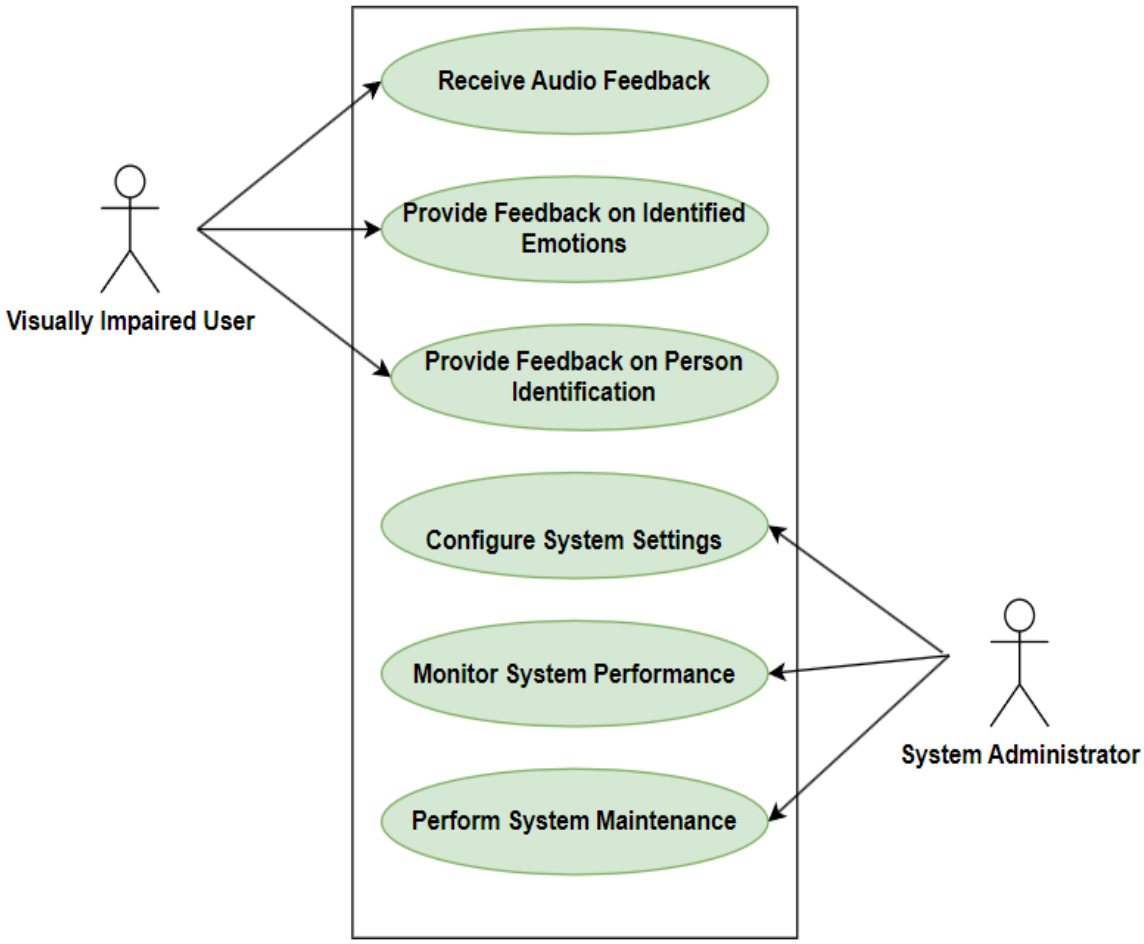

Figure 4.3: Use-Case diagram of FER system

The use case diagram illustrates the interactions between user and System Administrator in the FER system. VIPs engage with functionalities such as receiving real-time video input and providing feedback on emotions, while the System Administrator oversees system configuration, monitoring, and maintenance.

Chapter 5

RESULTS

5.1 Test 1 (Pre-saved face)

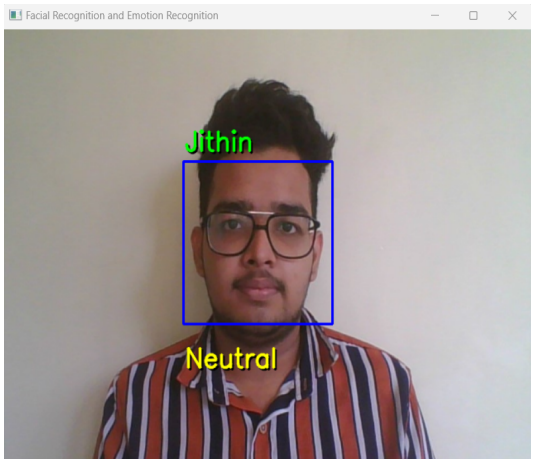

Figure 5.1: Known Face
(Emotion: Neutral)

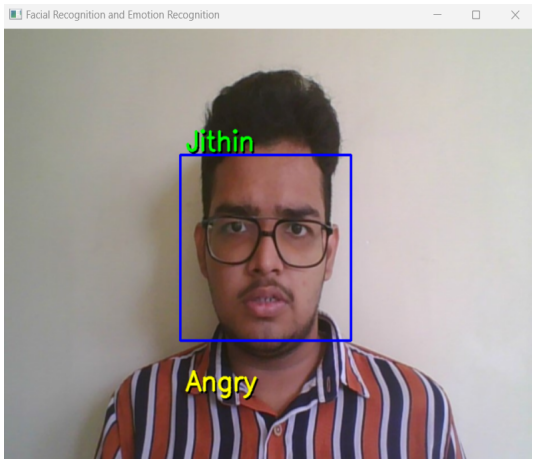

Figure 5.2: Known Face
(Emotion: Angry)

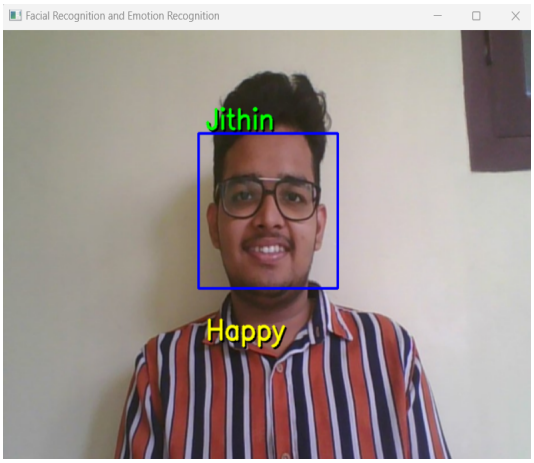

Figure 5.3: Known Face
(Emotion: Happy)

5.2 Test 2 (Unknown face)

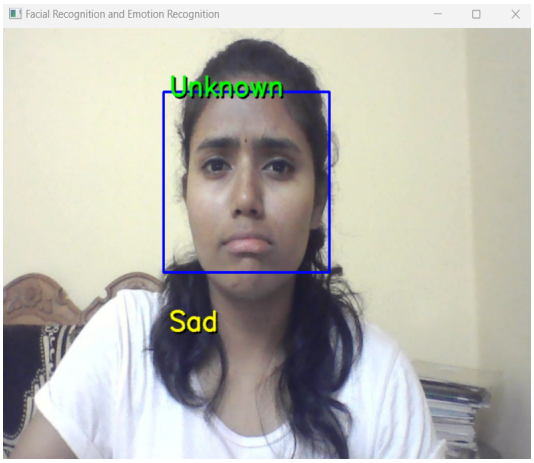

Figure 5.4: Unknown Face

(Emotion: Sad)

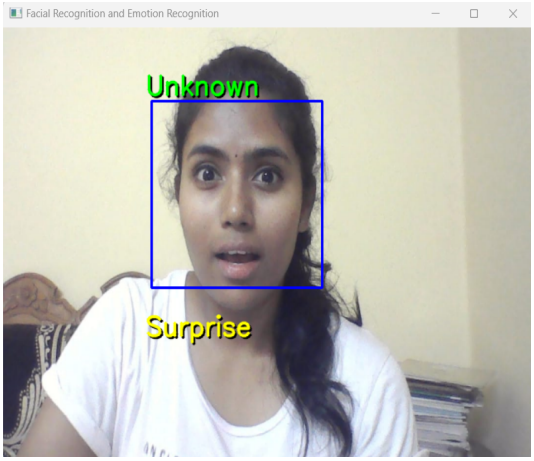

Figure 5.5: Unknown Face

(Emotion: Surprise)

5.3 Test 3 (Unknown with Save Face feature)

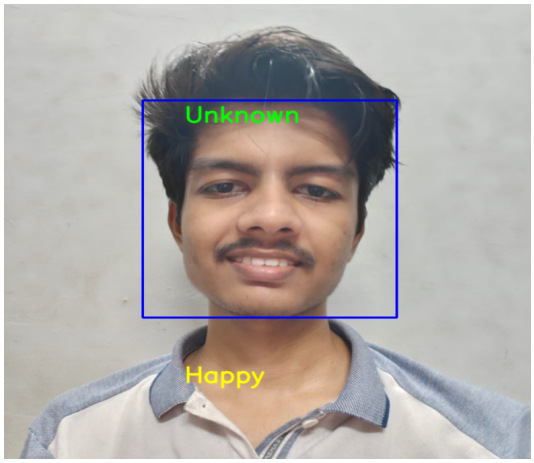

Figure 5.6: Unknown Face (Emotion- Happy)

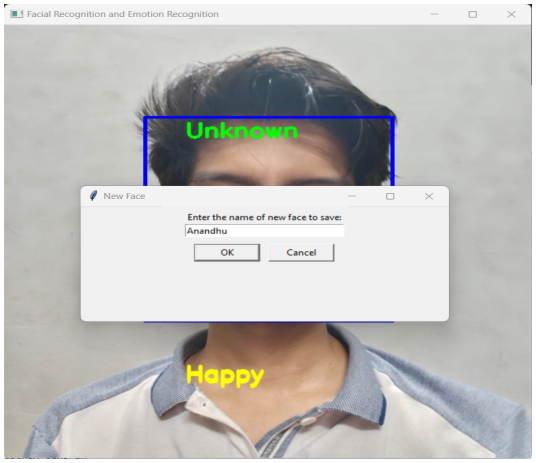

Figure 5.7: Saving a new face

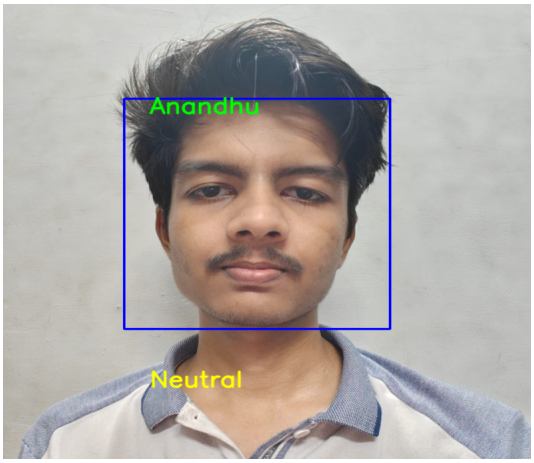

Figure 5.8: Known Face (Emotion- Neutral)

Chapter 6

FUTURE SCOPE

The future scope and developments of facial emotion recognition systems with audio output for visually impaired individuals using transfer learning hold immense potential for enhancing social interactions. There are several areas that can be explored to further improve the capabilities of these systems.

Integrating multimodal inputs such as touch or smell can provide a more holistic understanding of social cues beyond facial expressions, enhancing the user experience and emotional interpretation accuracy.

Real-time learning algorithms can be implemented to continually update emotional recognition models based on user feedback, leading to personalized emotional interpretations and improved system accuracy over time.

Transfer learning techniques can be employed to leverage pre-trained models and adapt them to handle a larger database of faces and emotions, improving system performance and generalization capabilities.

Customization features can be introduced to allow users to define their own preferences and thresholds for emotional cues, tailoring the system to individual needs and enhancing user satisfaction.

Ethical considerations such as data privacy and bias mitigation should be prioritized in the future development of these systems to ensure responsible and fair usage across diverse user groups.

Collaborative interfaces that enable seamless communication between visually impaired users and sighted individuals should also be explored, promoting inclusivity and fostering meaningful social interactions in various settings.

By focusing on these areas of development,facial emotion recognition with audio output for visually impaired individuals using transfer learning aims to contribute to the ongoing advancement of assistive technologies, empowering visually impaired individuals to navigate social interactions confidently and independently while promoting a more inclusive and empathetic society.

Chapter 7

CONCLUSION

In conclusion, the development of a facial emotion recognition system with audio output for visually impaired individuals using transfer learning represents a significant advancement in enhancing social interactions and emotional understanding for this user group. Through this project, we have successfully designed and implemented an architecture that combines image processing, deep learning models [1], and audio feedback systems to interpret and convey emotional cues in real-time.

The project has showcased the effectiveness of transfer learning techniques in leveraging pre-trained models for facial recognition, thereby reducing the computational burden and improving system performance. By integrating audio output capabilities, we have enhanced the user experience by providing auditory feedback corresponding to recognized emotions, fostering more natural and meaningful interactions.

Moreover, the project underscores the importance of continuous refinement and adaptation in assistive technologies. Future scope and developments in facial emotion recognition systems include multi-modal integration, real-time learning algorithms for personalized interpretations, privacy and ethics considerations, and collaborative interfaces for inclusive social interactions.

By embracing these advancements and building upon the foundational work of this project, we are poised to make significant strides in empowering visually impaired individuals to navigate social interactions with greater confidence and independence. Ultimately, our efforts contribute to creating a more inclusive and empathetic society by leveraging technology to bridge communication barriers and foster genuine human connections.

Chapter 8

APPENDIX

8.1 Dataset Description

The FER 2013 dataset [6], a cornerstone for emotion recognition research, offers a rich collection of data images for training and testing. These images, sourced from diverse environments, capture nuanced facial expressions across seven distinct emotional categories: Angry, Disgust, Fear, Happy, Sad, Surprise, and Neutral. We utilized approximately 15,000 photos from the FER 2013 dataset for training our model and 5000 for testing its performance. Each image underwent meticulous preprocessing, including normalization of sizes and augmentation techniques like rotation and scaling to enhance dataset variability.

Furthermore, the dataset's balanced distribution ensured equitable representation across emotional classes, mitigating biases and promoting robust model training. Each image is annotated with its corresponding emotional label, enabling supervised learning approaches for emotion classification. The careful curation and comprehensive coverage of the FER 2013 dataset make it an invaluable resource for training and evaluating emotion recognition models, fostering advancements in understanding and interpreting human emotions.

8.2 Training Process

The training process involved several iterative steps to develop a robust emotion recognition model. Initially, the FER 2013 dataset underwent preprocessing, including resizing images to a standardized format and applying augmentation techniques such as rotation and flipping to increase dataset diversity. A deep learning architecture, particularly a convolutional neural network (CNN) [4], was chosen for its effectiveness

in image classification tasks.

Hyperparameters such as batch size, learning rate, and optimizer were carefully tuned through experimentation and validation on a separate validation set. Additionally, techniques such as dropout regularization and batch normalization were employed to prevent overfitting and improve generalization performance.

Training proceeded in epochs, with the model learning from the training data and adjusting its parameters to minimize a predefined loss function. Regular monitoring of training and validation metrics, including accuracy and loss, guided model refinement and optimization.

Throughout the training process, model performance was evaluated using the validation set to ensure consistent improvement and avoid overfitting. Fine-tuning and adjustment of hyperparameters were iteratively performed to optimize model performance.

Ultimately, the trained model demonstrated the ability to accurately classify human emotions, providing a valuable tool for emotion recognition tasks in various applications.

8.3 Evaluation Metrics

Confusion Matrix

The confusion matrix [7] provides a detailed summary of the model's performance by displaying the counts of true positive, true negative, false positive, and false negative predictions for each of the seven emotional categories: Angry, Disgust, Fear, Happy, Sad, Surprise, and Neutral. Each row of the matrix corresponds to the true class labels, while each column represents the predicted class labels. This allows for a visual assessment of the model's ability to accurately classify emotions and identify any patterns of misclassification or confusion between different emotions.

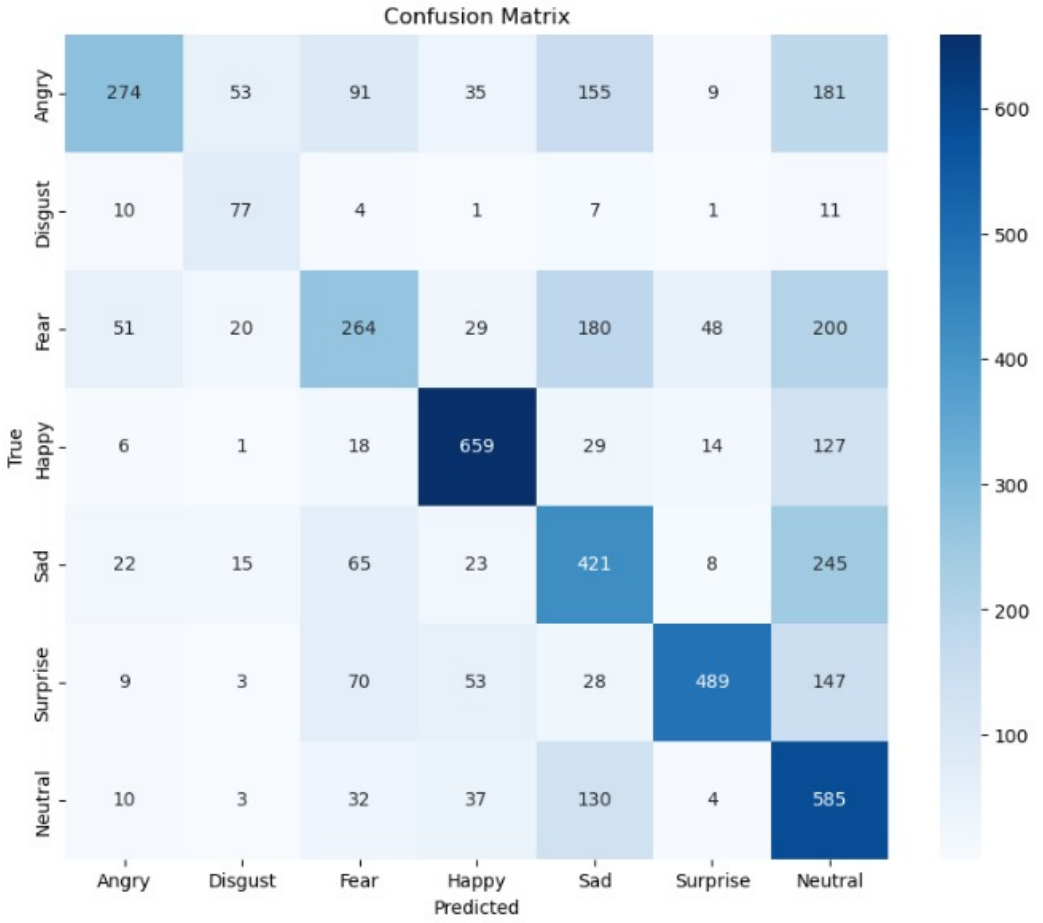

Figure 8.1: Confusion Matrix

8.4 FER Model Training Code

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
import os
import tensorflow as tf


Datadirectory = "Training" ##training dataset
Classes = ["0","1","2","3","4","5","6"]
##List of classes=> exact name of your folders
```

```
training_Data=[] ##data

def create_training_Data():
    for category in Classes:
        path = os.path.join(Datadirectory,category)
        class_num = Classes.index(category)  ## 0 1,  ## Label
        for img in os.listdir(path):
            try:
                img_array = cv2.imread(os.path.join(path,img))
                new_array = cv2.resize(img_array, (img_size,img_size))
                training_Data.append( [new_array,class_num] )
            except Exception as e:
                pass


create_training_Data()

import random
random.shuffle(training_Data)

x = [] ## data/feature
y = [] ## label
for features,label in training_Data:
    x.append(features)
    y.append(label)

x = np.array(x).reshape(-1, img_size, img_size, 3)
## converting it to 4 dimensions

# Normalizing the data
X = x/255.0
```

```
from tensorflow import keras
from tensorflow.keras import layers
import tensorflow as tf

model = tf.keras.applications.MobileNetV2() ## Pre-trained Model
base_input = model.layers[0].input
base_ouput = model.layers[-2].output

final_output = layers.Dense(128)(base_ouput)
## adding new layer after output of global pooling layer
final_ouput  = layers.Activation('relu')(final_output)
## activation function
final_output = layers.Dense(64)(final_ouput)
final_ouput  = layers.Activation('relu')(final_output)
final_output = layers.Dense(7,activation='softmax')(final_ouput)
## we need 7 classes ## Classification layers

from tensorflow import keras
from tensorflow.keras.models import Model

new_model = keras.Model(inputs = base_input, outputs = final_output)

new_model.compile(loss="sparse_categorical_crossentropy",
                        optimizer= "adam", metrics = ["accuracy"] )
new_model.fit(X, Y, epochs=20)
new_model.save('fer_model_01.h5')
```

8.5 Live Demo Code

```
import cv2
import numpy as np
import tensorflow as tf
```

```
import time
import pyttsx3
import os
import glob
import face_recognition
import tkinter as tk
from tkinter import simpledialog

class SimpleFacerec:
    def __init__(self):
        self.known_face_encodings = []
        self.known_face_names = []
        self.frame_resizing = 0.25

    def load_encoding_images(self, images_path):
        images_path = glob.glob(os.path.join(images_path, "*.jpg"))

        for img_path in images_path:
            img = cv2.imread(img_path)
            rgb_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

            basename = os.path.basename(img_path)
            (filename, ext) = os.path.splitext(basename)
            face_encodings = face_recognition.face_encodings(rgb_img)

            if len(face_encodings) == 0:
                print(f"No faces found in {img_path}. Skipping...")
                continue

            img_encoding = face_encodings[0]

            self.known_face_encodings.append(img_encoding)
```

```
            self.known_face_names.append(filename)


    def detect_known_faces(self, frame):
        small_frame = cv2.resize(frame, (0, 0),
            fx=self.frame_resizing, fy=self.frame_resizing)
        rgb_small_frame = cv2.cvtColor(small_frame,
                        cv2.COLOR_BGR2RGB)
        face_locations = face_recognition.face_locations(
                        rgb_small_frame)
        face_encodings = face_recognition.face_encodings(
                         rgb_small_frame, face_locations)

        face_names = []
        for face_encoding in face_encodings:
            matches = face_recognition.compare_faces(
            self.known_face_encodings, face_encoding)
            name = "Unknown"

            face_distances = face_recognition.face_distance(
                        self.known_face_encodings, face_encoding)
            best_match_index = np.argmin(face_distances)
            if matches[best_match_index]:
                name = self.known_face_names[best_match_index]
            face_names.append(name)

        face_locations = np.array(face_locations)
        face_locations = face_locations / self.frame_resizing
        return face_locations.astype(int), face_names

def detect_emotion(face_roi):
    final_image = cv2.resize(face_roi, (224, 224))
```

```
    final_image = np.expand_dims(final_image, axis=0)
    final_image = final_image / 255.0
    predictions = new_model.predict(final_image)
    emotion_label = emotion_labels[np.argmax(predictions)]
    return emotion_label

def speak_emotion_and_name(name, emotion):
    if name == "Unknown":
        engine.say(f"Unknown person is {emotion}")
    else:
        engine.say(f"{name} is {emotion}")
    engine.runAndWait()



def capture_screenshot(frame, save_path, filename):
    cv2.imwrite(os.path.join(save_path, filename), frame)
    print("Screenshot saved as:", filename)

def get_custom_name():
    root = tk.Tk()
    root.withdraw()  # Hide the main window

    # Create a simple dialog box to get user input
    custom_name = simpledialog.askstring("Custom Name", "Enter
                            a custom name for the new face:")

    # Check if user clicked cancel or input is empty
    if custom_name is None or custom_name.strip() == "":
        print("No custom name provided.")
        return None

    return custom_name
```

```
path = "haarcascade_frontalface_default.xml"
font_scale = 1
font = cv2.FONT_HERSHEY_SIMPLEX
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy',
                  'Sad', 'Surprise', 'Neutral']

cap = cv2.VideoCapture(0)
if not cap.isOpened():
    cap = cv2.VideoCapture(1)
if not cap.isOpened():
    raise IOError("Cannot open webcam")

new_model = tf.keras.models.load_model('fer_model4.h5')
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
               'haarcascade_frontalface_default.xml')
engine = pyttsx3.init()

IMAGES_FOLDER = "images"
sfr = SimpleFacerec()
sfr.load_encoding_images(IMAGES_FOLDER)

new_face_detected = True
emotion_last_spoken = time.time()

while True:
    ret, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.1, 4)

    if len(faces) == 0:
        if not new_face_detected:
```

```
            cv2.putText(frame, "No face detected", (20, 50), font,
                    font_scale, (255, 255, 255), 2, cv2.LINE_AA)
        new_face_detected = False
        cv2.imshow("Facial Recognition and Emotion Recognition",
                    frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
        continue

    for (x, y, w, h) in faces:
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = frame[y:y+h, x:x+w]
        cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)

        if new_face_detected:
            engine.say("New face detected")
            engine.runAndWait()
            new_face_detected = False

        emotion = detect_emotion(roi_color)
        face_locations, face_names = sfr.detect_known_faces(frame)


        for face_loc, name in zip(face_locations, face_names):
            y1, x2, y2, x1 = face_loc[0], face_loc[1], face_loc[2],
                face_loc[3]
            cv2.putText(frame, name, (x1, y1 - 30), font,
                    font_scale, (0, 255, 0), 2, cv2.LINE_AA)
            cv2.putText(frame, emotion, (x1, y1 + h + 30), font,
                    font_scale, (0, 255, 255), 2, cv2.LINE_AA)

            if time.time() - emotion_last_spoken > 5:
```

```
                emotion_last_spoken = time.time()
                speak_emotion_and_name(name, emotion)

    cv2.imshow("Facial Recognition and Emotion Recognition", frame)

    key = cv2.waitKey(1)
    if key == 27:  # Press Esc to exit
        break
    elif key == ord('n'):  # Press 'n' to capture and save screenshot
        custom_name = get_custom_name()
        filename = custom_name + ".jpg"
        capture_screenshot(frame, IMAGES_FOLDER, filename)
        sfr.load_encoding_images(IMAGES_FOLDER)
        # Reload encodings to include the new face

cap.release()
cv2.destroyAllWindows()
```

REFERENCES

[1] S. K. Lalitha, J. Aishwarya, N. Shivakumar, T. Srilekha and G. C. R. Kartheek, "A deep learning model for face expression detection", Proc. Int. Conf. Recent Trends Electron. Inf. Commun. Technol. (RTEICT), pp. 647-650, Aug. 2022.

[2] D. Shehada, A. Turky, W. Khan, B. Khan and A. Hussain, "A Lightweight Facial Emotion Recognition System Using Partial Transfer Learning for Visually Impaired People," in IEEE Access, vol. 11, pp. 36961-36969, 2023

[3] U. Kulkarni et al., "Facial Key points Detection using MobileNetV2 Architecture," 2023 IEEE 8th International Conference for Convergence in Technology (I2CT), Lonavla, India, 2023

[4] N. Zhou, R. Liang and W. Shi, "A lightweight convolutional neural network for real-time facial expression detection", IEEE Access, vol. 9, pp. 5573-5584, 2023.

[5] A. Ashok and J. John, "Facial expression recognition system for visually impaired", Proc. Int. Conf. Intell. Data Commun. Technol. Internet of Things, pp. 244-250, 2022.

[6] Ö. Ezerceli and M. T. Eskil, "Convolutional Neural Network (CNN) Algorithm Based Facial Emotion Recognition (FER) System for FER-2013 Dataset,"in IEEE Access,vol. 3, no. 1, pp. 39-53, 2023

[7] L. -E. Pommé, R. Bourqui, R. Giot and D. Auber, "Relative Confusion Matrix: Efficient Comparison of Decision Models," in IEEE Access, vol. 11, pp. 36961-36969, 2023

[8] L. Bai, T. Zhao and X. Xiu, "Exploration of computer vision and image processing technology based on OpenCV," 2022 International Seminar on Computer Science and Engineering Technology (SCSET), Indianapolis, IN, USA, 2022

[9] B. Zheng, T. Masuda and T. Shibata, "An Indoor Positioning with a Neural Network Model of TensorFlow for Machine Learning," 2022 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), Hualien City, Taiwan, 2022