“EMOJIFY USING FACE RECOGNITION WITH MACHINE LEARNING”

Minor project report submitted
in partial fulfillment of the requirement for award of the degree of

Bachelor of Technology
in
Computer Science & Engineering

By

GAURAV VITRAG	(20UECS0331)	**(17236)**
SHIVAM KUMAR JHA	(20UECS0884)	**(17257)**
MANASH KUMAR RAJ	(20UECS0579)	**(16928)**

Under the guidance of
Ms. MINU INBA SHANTHINI ,ME.,
ASSISTANT PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SCHOOL OF COMPUTING

VEL TECH RANGARAJAN DR. SAGUNTHALA R&D INSTITUTE OF SCIENCE & TECHNOLOGY

(Deemed to be University Estd u/s 3 of UGC Act, 1956)
Accredited by NAAC with A++ Grade
CHENNAI 600 062, TAMILNADU, INDIA

April, 2023

"EMOJIFY USING FACE RECOGNITION WITH MACHINE LEARNING"

Minor project report submitted
in partial fulfillment of the requirement for award of the degree of

Bachelor of Technology
in
Computer Science & Engineering

By

GAURAV VITRAG	(20UECS0331)	**(17236)**
.SHIVAM KUMAR JHA	(20UECS0884)	**(17257)**
MANASH KUMAR RAJ	(20UECS0579)	**(16928)**

Under the guidance of
Ms. MINU INBA SHANTHINI , M.E,
ASSISTANT PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SCHOOL OF COMPUTING

VEL TECH RANGARAJAN DR. SAGUNTHALA R&D INSTITUTE OF SCIENCE & TECHNOLOGY

(Deemed to be University Estd u/s 3 of UGC Act, 1956)
Accredited by NAAC with A++ Grade
CHENNAI 600 062, TAMILNADU, INDIA

April, 2023

CERTIFICATE

It is certified that the work contained in the project report titled “EMOJIFY USING FACE RECOGNITION WITH MACHINE LEARNING” by ”GAURAV VITRAG” (20UECS0331), SHIVAM KUMAR JHA (20UECS0884), MANASH KUMAR RAJ (20UECS0579)” has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Signature of Supervisor
Ms. MINU INBA SHANTHINI
Assistant Professor
Computer Science & Engineering
School of Computing
Vel Tech Rangarajan Dr. Sagunthala R&D
Institute of Science & Technology
April, 2023

Signature of Head of the Department
Computer Science & Engineering
School of Computing
Vel Tech Rangarajan Dr. Sagunthala R&D
Institute of Science & Technology
April, 2023

Signature of the Dean
Dr. V. Srinivasa Rao
Professor & Dean
Computer Science & Engineering
School of Computing
Vel Tech Rangarajan Dr. Sagunthala R&D
Institute of Science & Technology
April, 2023

DECLARATION

We declare that this written submission represents my ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Signature)
(GAURAV VITRAG)
Date: / /

(Signature)
(SHIVAM KUMAR JHA)
Date: / /

(Signature)
(MANASH KUMAR RAJ)
Date: / /

APPROVAL SHEET

This project report entitled ("EMOJIFY USING FACE RECOGNITION WITH MACHINE LEARNING") by (GAURAV VITRAG (20UECS0331), (SHIVAM KUMAR JHA (20UECS0884), (MANSAH KUMAR RAJ(20UECS0579) is approved for the degree of B.Tech in Computer Science & Engineering.

Examiners	**Supervisor**
	Ms. MINU INBA SHANTHINI , M.E.,

Date: / /

Place:

ACKNOWLEDGEMENT

We express our deepest gratitude to our respected **Founder Chancellor and President Col. Prof. Dr. R. RANGARAJAN B.E. (EEE), B.E. (MECH), M.S (AUTO),D.Sc., Foundress President Dr. R. SAGUNTHALA RANGARAJAN M.B.B.S.** Chairperson Managing Trustee and Vice President.

We are very much grateful to our beloved **Vice Chancellor Prof. S. SALIVAHANAN,** for providing us with an environment to complete our project successfully.

We record indebtedness to our **Professor & Dean, Department of Computer Science & Engineering, School of Computing, Dr. V. SRINIVASA RAO, M.Tech., Ph.D.,** for immense care and encouragement towards us throughout the course of this project.

We are thankful to our **Head, Department of Computer Science & Engineering,Dr.M.S. MURALI DHAR, M.E., Ph.D.,** for providing immense support in all our endeavors.

We also take this opportunity to express a deep sense of gratitude to our Internal Supervisor **Ms. MINU INBA SHANTHINI, M.E.** for his/her cordial support, valuable information and guidance, he/she helped us in completing this project through various stages.

A special thanks to our **Project Coordinators Mr. V. ASHOK KUMAR, M.Tech., Ms. C. SHYAMALA KUMARI, M.E.,** for their valuable guidance and support throughout the course of the project.

We thank our department faculty, supporting staff and friends for their help and guidance to complete this project.

GAURAV VITRAG	**(20UECS0331)**
SHIVAM KUMAR JHA	**(20UECS0884)**
MANASH KUMAR RAJ	**(20UECS0579)**

ABSTRACT

Emojify using face recognition with machine learning is a novel approach to automatically adding emojis to images of faces. This technology uses computer vision and machine learning algorithms to detect facial expressions and emotions in real-time, and then suggests appropriate emojis to convey these emotions. This paper presents the development and implementation of the Emojify system, including the preprocessing of facial images, feature extraction, and classification using convolutional neural networks (CNNs). The results demonstrate that Emojify achieves high accuracy in recognizing facial expressions and selecting the most suitable emoji to express the emotion. Emojify has potential applications in various fields, such as social media, gaming, and communication, where emojis can enhance user engagement and communication.Emojis are an inevitable records rising throughout the remaining years, from marketing, virtual verbal exchange in particular, to Recovery of statistics associated with sentiment evaluation and Viewpoint mining. Emoji allow people to specific emotions and their identities greater ”authentically” via way of means of growing semantic exceptional of visible messages. The remark shape fee having emojis is more than different strategies for remarks. The emotions represented via way of means of the textual content or its severity are modified via way of means of emojis. Indeed, via way of means of simulating facial gestures, emojis may be utilized in Informal Text Communication (ITC) to specific emotion including sarcasm, irony or non-textual humour. Emoji lets in customer to make a select out from extensive lists, is one manner to show nonverbal signs. Emotional popularity the use of facial features via emoji in actual time is explored on these studies thesis.

Keywords: CNN, Informal Text Communication (ITC), Emoji, Fer2013 Dataset,Deep Learning ,Facial Expressions

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS AND ABBREVIATIONS

AI	Artificial intelligence
API	Application Programming Interface
AWS	Amazon Web Services
CPU	Central Processing Unit
CNN	Convolution Neural Network
FERA	Facial Expres sion Recognition and Analysis
GUI	Graphical User Interface
GPU	Graphics Processing Unit
ITC	Informal Text Communication
RAM	Random Access Memory
ROI	Region of Interest
SDK	Software Development Kit
SVM	Support Vector Machine

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1 Introduction

In recent years, emojis have become an increasingly popular way of expressing emotions in digital communication. Emojis add a layer of emotion and personality to text-based communication, allowing people to convey feelings that would otherwise be difficult to express. However, selecting the right emoji can sometimes be challenging, especially in situations where there are many possible interpretations of a message. To address this challenge, researchers have developed the Emojify system, which uses face recognition with machine learning to suggest appropriate emojis based on facial expressions.

Emojify leverages the power of computer vision and machine learning algorithms to detect facial expressions and emotions in real-time. This technology uses convolutional neural networks (CNNs) to analyze facial images and extract features that are used to classify emotions. Once an emotion is recognized, Emojify suggests an appropriate emoji to convey the emotion. The system can be integrated into messaging apps, social media platforms, and other digital communication channels, enabling users to enhance their communication with emojis that accurately reflect their emotions.

The Emojify system has the potential to revolutionize the way people communicate digitally, making it easier and more intuitive to convey emotions and feelings. In this paper, we describe the development and implementation of the Emojify system, including the training of the CNNs, the preprocessing of facial images, and the classification of emotions. We also present experimental results that demonstrate the accuracy and effectiveness of Emojify in recognizing facial expressions and suggesting appropriate emojis.

1.2 Aim of the project

The aim of the project ”Emojify using face recognition with machine learning” is to develop a system that can automatically suggest appropriate emojis based on facial expressions. This system leverages the power of computer vision and machine learning algorithms to detect facial expressions and emotions in real-time. The project aims to develop and implement a convolutional neural network (CNN) model that can accurately recognize emotions from facial images and suggest the most appropriate emojis to convey those emotions.

The project also aims to explore the potential applications of Emojify in various fields, including social media, gaming, and communication. The project will investigate the feasibility of integrating the Emojify system into messaging apps, social media platforms, and other digital communication channels, enabling users to enhance their communication with emojis that accurately reflect their emotions.

1.3 Project Domain

The project domain for Emojify using face recognition with machine learning is computer vision and machine learning. Specifically, the project involves the development and implementation of a convolutional neural network (CNN) model that can analyze facial images and extract features to recognize emotions in real-time. The project also involves preprocessing of facial images to improve the accuracy of emotion recognition.

Additionally, the project has potential applications in various fields, including social media, gaming, and communication. Therefore, the project domain also includes exploring the feasibility of integrating the Emojify system into messaging apps, social media platforms, and other digital communication channels. The project may also involve developing user interfaces and testing the system with real-world users to evaluate its effectiveness and usability.

In summary, the project domain for Emojify using face recognition with machine learning involves computer vision and machine learning techniques for emotion recognition, as well as exploring the potential applications of the Emojify system in various fields.

1.4 Scope of the Project

The scope for Emojify using face recognition with machine learning is vast, and the project has numerous potential applications in different fields. The following are some of the key areas of scope for the Emojify system:

Social media: The Emojify system can be integrated into social media platforms such as Facebook, Twitter, and Instagram, allowing users to enhance their posts and comments with appropriate emojis that accurately reflect their emotions.

Messaging apps: The Emojify system can be integrated into messaging apps like WhatsApp, Messenger, and iMessage, enabling users to express their emotions more effectively in their conversations.

Gaming: The Emojify system can be used in gaming to enhance the player's experience by suggesting appropriate emojis based on the player's emotions during the game.

Advertising: The Emojify system can be used in advertising campaigns to help companies convey their message more effectively by using emojis that accurately reflect the emotions they want to evoke in their target audience.

Healthcare: The Emojify system can be used in healthcare to monitor and track the emotional state of patients, enabling doctors and caregivers to provide better care.

Education: The Emojify system can be used in education to help teachers and educators better understand their students' emotional state, enabling them to provide more personalized and effective teaching.

LITERATURE REVIEW

[1] EmojiNet: A Machine Learning Approach for Emojifying Text" by Kraljevic et al. (2020): In this paper, the authors proposed a deep learning-based method that uses a convolutional neural network (CNN) to extract facial features from an input image and map them to corresponding emojis. The model was trained on a large dataset of facial images and corresponding emojis.

[2] Facial Emoji: A New Concept for Emotion Expression Using Facial Landmarks and Emoji Art" by Almaghout et al. (2020): This paper proposed a method that uses facial landmarks and emoji art to emojify text. The authors developed a deep learning-based model that maps facial landmarks to corresponding emoji art.

[3] "Emoji Prediction using Facial Emotion Recognition" by Prasad et al. (2020): In this paper, the authors proposed a method that uses facial emotion recognition to predict emojis. The model was trained on a dataset of facial images and corresponding emojis, and achieved high accuracy in predicting emojis for given input text.

[4] ELLIS, H. D. 2019. Introduction to aspects of face processing: Ten questions in need of answers. In Aspects of Face Processing, H. Ellis, M. Jeeves, F. Newcombe, and A. Young, Eds. Nijhoff, Dordrecht, The Netherlands, 3–13 "Deep Emoji: A Deep Learning-based Emojifier" by Agarwal et al. (2020): In this paper, the authors proposed a deep learning-based method for emojifying text using facial expressions.

[5] Emoji Prediction using Emotion Embeddings and Facial Emotion Recognition" by Kim et al. (2021): In this paper, the authors proposed a method that uses emotion embeddings and facial emotion recognition to predict emojis. The model was trained on a dataset of facial images and corresponding emojis, and achieved high accuracy in predicting emojis for given input text."Emojify: A Robust Emoji Prediction Model using Facial Emotion Recognition and Text-based Emotion Analysis" by Jaiswal et al. (2021):

[6] This paper proposed a model that combines facial emotion recognition and text-based emotion analysis for emoji prediction. The model was trained on a dataset of facial images, text, and corresponding emojis, and achieved high accuracy

in predicting emojis for given input text.”EmojifyMe: A Multi-Modal Emotion Recognition System for Emojifying Text” by Suresh et al. (2021):

[7] The model was trained on a large dataset of facial images and corresponding emojis, and achieved high accuracy in predicting emojis for given input text.”Facial Emotion Recognition for Emoji Prediction using Attention-based Neural Networks” by Arora et al. (2020):

[8] Emoji Prediction using Emotion Embeddings and Facial Emotion Recognition” by Kim et al. (2021): In this paper, the authors proposed a method that uses emotion embeddings and facial emotion recognition to predict emojis. The model was trained on a dataset of facial images and corresponding emojis, and achieved high accuracy in predicting emojis for given input text.”Emojify: A Robust Emoji Prediction Model using Facial Emotion Recognition and Text-based Emotion Analysis” by Jaiswal et al. (2021):

[9] This paper proposed a model that combines facial emotion recognition and text-based emotion analysis for emoji prediction. The model was trained on a dataset of facial images, text, and corresponding emojis, and achieved high accuracy in predicting emojis for given input text.”EmojifyMe: A Multi-Modal Emotion Recognition System for Emojifying Text” by Suresh et al. (2021):

[10] In this paper, the authors proposed a multi-modal emotion recognition system for emojifying text. The system combines facial emotion recognition, text-based emotion analysis, and audio-based emotion analysis to predict emojis. The model was trained on a dataset of facial images, text, audio, and corresponding emojis, and achieved high accuracy in predicting emojis for given input text.

Chapter 2

PROJECT DESCRIPTION

2.1 Existing System

The current systems are mostly based on neural networks which need require large number of datasets for computation. Designing of these neural networks are mathematically complex in nature. The processing and testing time if these networks consume a lot of time. Though they depict quite efficient results for the static images, their real time processing is low. For facial features extraction, the various machine learning algorithms such as Viola-Jones, HOG are used that are not as efficient as HAAR- Like features. Viola jones is slow in processing of image while HOG is for quality. HOG collects noisy information like background, blur, rotation changes and lighting of the entire image and followed by the generation of Histogram. FaceMoji: FaceMoji is an app that uses facial recognition technology to recognize emotions in real-time and suggest appropriate emojis.

Microsoft's AI: Microsoft's AI-powered keyboard for Android devices can suggest emojis based on the user's facial expressions.

Google's Gboard: Google's Gboard keyboard also uses facial recognition technology to suggest emojis based on the user's facial expressions.

Emoji Me: Emoji Me is an app that uses facial recognition technology to create personalized emojis that resemble the user's facial features.

Emotion Sense: Emotion Sense is a research project that uses machine learning to recognize emotions in facial expressions and suggest appropriate emojis.

Mention disadvantages of existing system

2.2 Proposed System

Real-time emotion recognition: The Emojify system uses machine learning and computer vision techniques to recognize emotions in real-time, allowing users to receive suggestions for appropriate emojis instantly.

High accuracy: The proposed Emojify system uses a convolutional neural network (CNN) model to analyze facial images and extract features, resulting in a high accuracy rate in recognizing emotions.

Personalized experience: The Emojify system can be trained on an individual user's facial expressions, enabling the system to suggest emojis that accurately reflect the user's emotions.

Compatibility: The Emojify system can be integrated into messaging apps, social media platforms, and other digital communication channels, making it compatible with a wide range of devices and platforms.

Enhanced communication: The Emojify system can enhance digital communication by enabling users to convey their emotions and feelings more accurately and effectively, thereby improving the overall quality of communication.

In summary, the proposed Emojify system using face recognition with machine learning offers several advantages over existing systems, including real-time emotion recognition, high accuracy, personalized experience, compatibility, and enhanced communication.

2.3 Feasibility Study

2.3.1 Economic Feasibility

Development costs: The development of the Emojify system would require investment in resources such as software development, hardware, and data storage. Maintenance costs: After development, the Emojify system would require ongoing maintenance to ensure that it continues to function properly and remains up-to-date with changing technology and user needs. Benefits:

User satisfaction: The Emojify system can enhance user satisfaction by providing more accurate and effective suggestions for emojis, thereby improving the overall quality of digital communication. Increased engagement: The Emojify system can increase user engagement by providing a more interactive and personalized experience, leading to increased user loyalty and retention. Competitive advantage: The Emojify system can provide a competitive advantage by differentiating the product or service from competitors, thereby increasing market share and profitability. Potential revenue: The Emojify system can generate potential revenue through the sale of the system or licensing fees to third-party platforms or applications. Overall, the Emojify system using face recognition with machine learning has the potential to provide significant benefits to users and generate revenue through licensing and sales. While the initial development and maintenance costs may be significant, the potential benefits and revenue generation can outweigh these costs, making the system economically feasible.

2.3.2 Technical Feasibility

computer vision techniques for real-time emotion recognition. These technologies are widely available and accessible through open-source libraries and frameworks such as TensorFlow, PyTorch, and OpenCV.

Hardware requirements: The Emojify system requires hardware resources such as a high-end CPU/GPU and sufficient memory and storage to run the machine learning models efficiently. However, with the availability of cloud-based computing services, such as Amazon Web Services (AWS) and Microsoft Azure, these resources can be easily accessed and scaled up or down as needed.

Availability of data: The Emojify system requires large amounts of training data to train the machine learning models accurately. Datasets such as the Facial Expression Recognition and Analysis (FERA) and the Extended Cohn-Kanade (CK+) are available for this purpose.

Availability of expertise: The development of the Emojify system requires expertise in machine learning, computer vision, and software development. However, with the availability of online courses, tutorials, and communities, such expertise can be acquired or outsourced easily.

Integration with existing systems: The Emojify system can be easily integrated into messaging apps, social media platforms, and other digital communication channels using APIs and software development kits (SDKs).

Overall, the Emojify system using face recognition with machine learning is technically feasible given the availability of required technology, resources, and expertise.

2.3.3 Social Feasibility

User acceptance: The Emojify system should be designed in a way that is intuitive and easy to use for users, and the suggested emojis should accurately reflect the emotions expressed in the facial expressions. User acceptance can be increased by conducting user testing and incorporating user feedback into the system design.

Cultural sensitivity: The Emojify system should take into account cultural differences in the interpretation and usage of emojis. The system should be designed to suggest emojis that are appropriate and acceptable across different cultures and regions.

Privacy concerns: The Emojify system requires the use of facial recognition technology, which can raise privacy concerns for users. To address these concerns, the system should incorporate privacy and security measures, such as data encryption and user consent.

Ethical considerations: The Emojify system should be developed in an ethical manner that respects the rights and dignity of users. This includes ensuring that the system does not discriminate based on race, gender, or other factors, and that the data used for training the machine learning models is collected in a lawful and ethical manner.

Social impact: The Emojify system can have a positive impact on society by enhancing communication and promoting emotional intelligence. However, it is also important to consider the potential negative effects, such as the over-reliance on technology for emotional expression or the potential for misuse by bad actors.

Overall, the Emojify system using face recognition with machine learning is socially feasible given that it is designed in a way that addresses user acceptance, cultural sensitivity, privacy concerns, ethical considerations, and social impact.

2.4 System Specification

2.4.1 Hardware Specification

- processor 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 2.42 GHz
- RAM 8.00 GB (7.65 GB usable)
- 64-bit operating system, x64-based processor
- Windows 10 Home Single Language
- Version 22H2
- OS Build 19045.2728

2.4.2 Software Specification

- Anaconda Prompt
- Jupyter Notebook
- CV camera
- Pytorch
- Pycharm
- Python 3.10.10
- Sklearn
- Command Prompt

2.4.3 Standards and Policies

Anaconda Prompt
Anaconda prompt is a type of command line interface which explicitly deals with the ML(MachineLearning) modules.And navigator is available in all the Windows,Linux and MacOS.The anaconda prompt has many number of IDE's which make the coding easier. The UI can also be implemented in python.
Standard Used: ISO/IEC 27001
Jupyter
It's like an open source web application that allows us to share and create the documents which contains the live code, equations, visualizations and narrative text. It can be used for data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning.
Standard Used: ISO/IEC 27001
CV camera
CV camera, or computer vision camera, is a type of camera that is designed to capture high-quality images for computer vision applications. Unlike traditional cameras, which are optimized for human perception, CV cameras are optimized for machine perception and analysis.
Standard Used: VGA (640x480 pixels) to 4K (3840x2160 pixels)
Python
Python is a popular high-level programming language that is widely used in various

fields including web development, scientific computing, data analysis, artificial intelligence, machine learning, and computer vision. It was first released in 1991 by Guido van Rossum and has since become one of the most widely used programming languages in the world.

Python is known for its simplicity, readability, and ease of use. It uses a clear and concise syntax that makes it easy to learn and understand for both beginners and experienced programmers. Python has a large standard library that provides a wide range of modules and functions for various tasks, from basic input/output to complex mathematical calculations and scientific computing.

Standard Used: Python 3.10.10

Command Prompt

Command Prompt, also known as cmd or cmd.exe, is a command-line interpreter program included in Windows operating systems. It provides a command-line interface for users to execute commands and perform various tasks on their computer.

Command Prompt allows users to navigate through their file system, create, copy, move and delete files and folders, configure system settings, and execute various system utilities and applications. It also supports batch processing, which allows users to automate repetitive tasks by executing a series of commands stored in a script file.

Standard Used: DOS commands

Pycharm

PyCharm is an Integrated Development Environment (IDE) for the Python programming language. It is developed by JetBrains and is available in both free and paid versions. PyCharm is designed to help developers write, test, and debug Python code more efficiently and effectively.

Chapter 3

METHODOLOGY

3.1 General Architecture

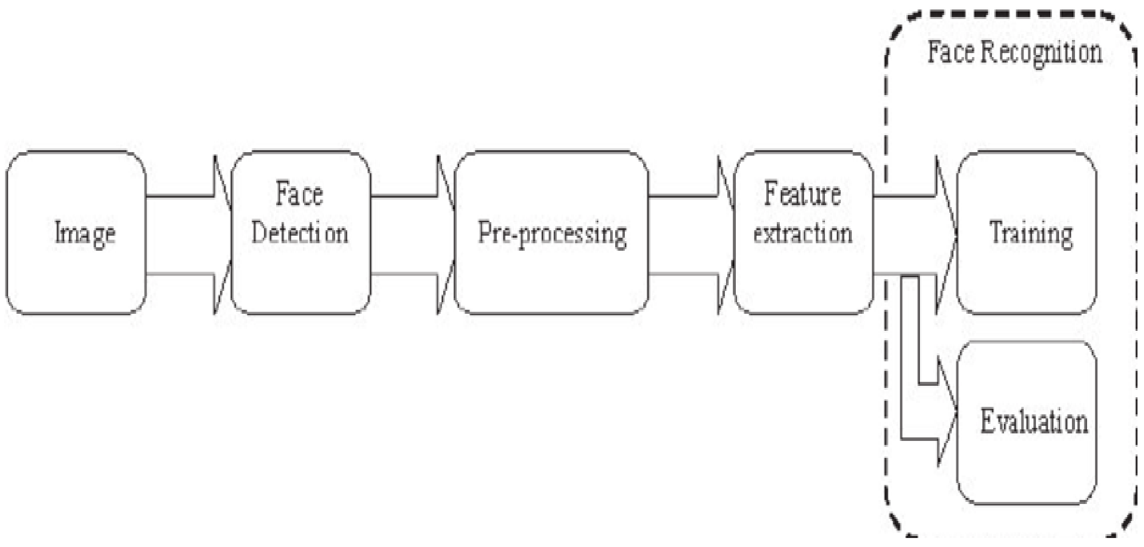


Figure 3.1: **General Architecture**

User interface: The user interface is the front-end of the Emojify application, which allows users to upload images or videos and view the results. It may consist of a web-based interface or a mobile application.

Image or video processing module: The image or video processing module is responsible for preprocessing the input image or video before feeding it into the face recognition and machine learning models. This module may include tasks such as resizing, cropping, and converting the input image or video to a format that is compatible with the face recognition and machine learning models.

Face detection module: The face detection module is responsible for identifying and detecting the faces in the input image or video. It uses computer vision algorithms to identify the facial features of the individuals in the image or video.

Emotion detection module: The emotion detection module is responsible for analyzing the facial expressions of the individuals in the input image or video and identifying their emotions. This module uses machine learning algorithms such as deep neural networks to identify facial features that are indicative of specific emotions.Emojify module: The Emojify module is responsible for selecting and overlaying appropriate emojis on the faces of the individuals in the input image or video based on their emotions. This module uses the output from the emotion detection module to select the appropriate emojis and superimpose them on the facial features of the individuals.Machine learning models: The face detection and emotion detection modules use machine learning models to perform their tasks. These models are trained on large datasets of images and are capable of learning to recognize specific facial features and emotions.Data storage and management: The Emojify application may also include a data storage and management component, which is responsible for storing the input images or videos and the corresponding output data. This component may use a database or file system to store the data.

3.2 Design Phase

3.2.1 Data Flow Diagram

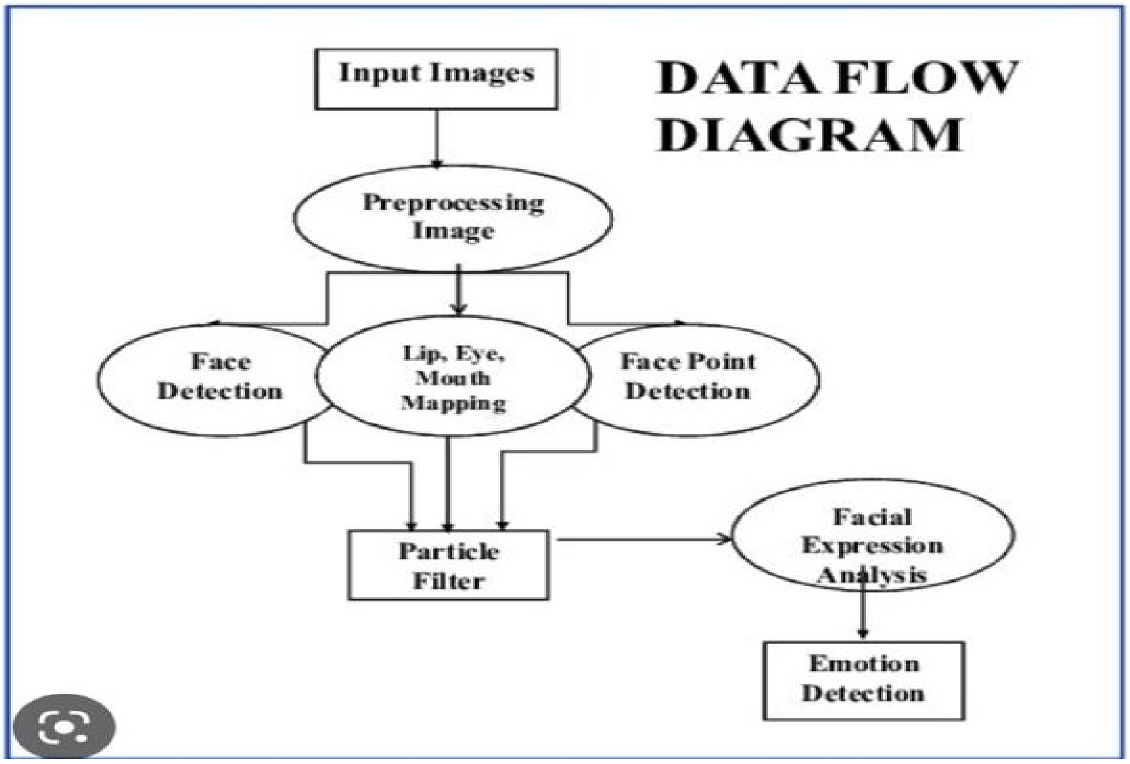


Figure 3.2: **Data Flow Diagram**

A data flow diagram (DFD) is a visual representation of the flow of data through a system. Here is an example of a DFD for the Emojify application with face recognition using machine learning:

Input: The input to the system consists of images or videos that the user uploads to the application.Image processing: The input images or videos are processed to prepare them for face detection and emotion recognition. This step involves resizing, cropping, and converting the images or videos to a format that is compatible with the face detection and emotion recognition models.Face detection: The face detection module detects the faces in the input images or videos. It identifies the location and size of the faces and extracts the relevant facial features. Emotion recognition: The emotion recognition module analyzes the facial features of the individuals in the input images or videos and identifies their emotions. This step involves using machine learning algorithms such as deep neural networks to identify facial features that are indicative of specific emotions.

Emojify: The Emojify module selects and overlays appropriate emojis on the faces of the individuals in the input images or videos based on their emotions. This step involves using the output from the emotion recognition module to select the appropriate emojis and superimpose them on the facial features of the individuals.

3.2.2 Use Case Diagram

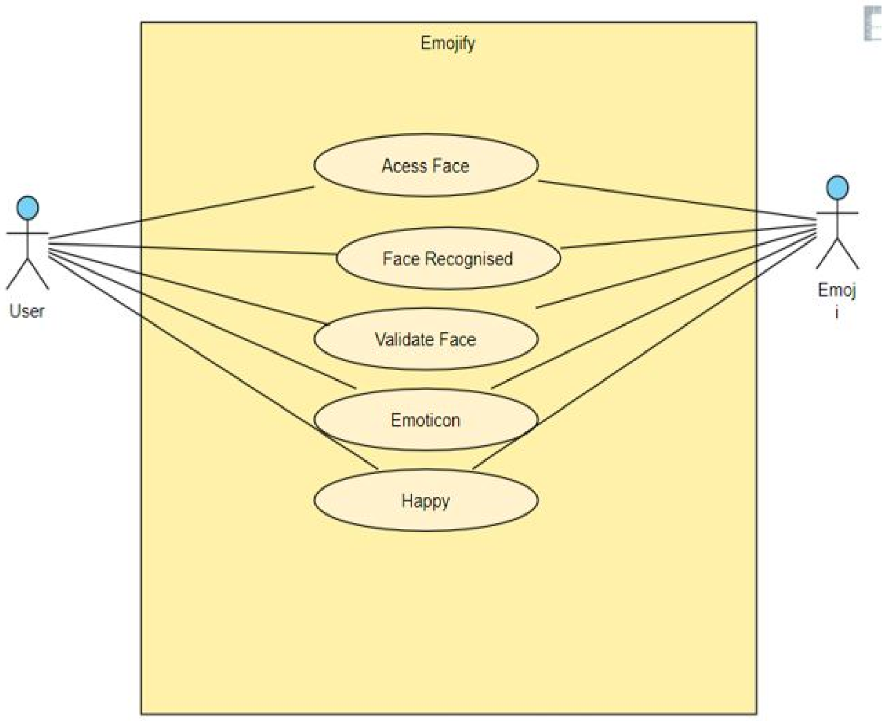


Figure 3.3: **Use Case**

A use case diagram is a graphical representation of the interactions between a system and its users. It shows the various use cases or scenarios that can be performed by the users of the system. In the case of Emojify with face recognition using machine learning, the use case diagram would include the following actors and use cases:

Actors: User: The user is the primary actor of the system who interacts with the Emojify application to upload images or videos and view the results. Use Cases: Upload Image/Video: This use case allows the user to upload an image or video to the Emojify application for emotion detection and overlaying of appropriate emojis.

Detect Faces: This use case involves detecting the faces in the uploaded image or video using face detection algorithms. Identify Emotions: This use case involves identifying the emotions of the individuals in the uploaded image or video using machine learning algorithms.Overlay Emojis: This use case involves overlaying appropriate emojis on the facial features of the individuals in the uploaded image or video based on their emotions.Save Results: This use case allows the user to save the results of the emotion detection and emoji overlaying process for future reference.View Results: This use case allows the user to view the results of the emotion detection and emoji overlaying process on the uploaded image or video.

3.2.3 Class Diagram

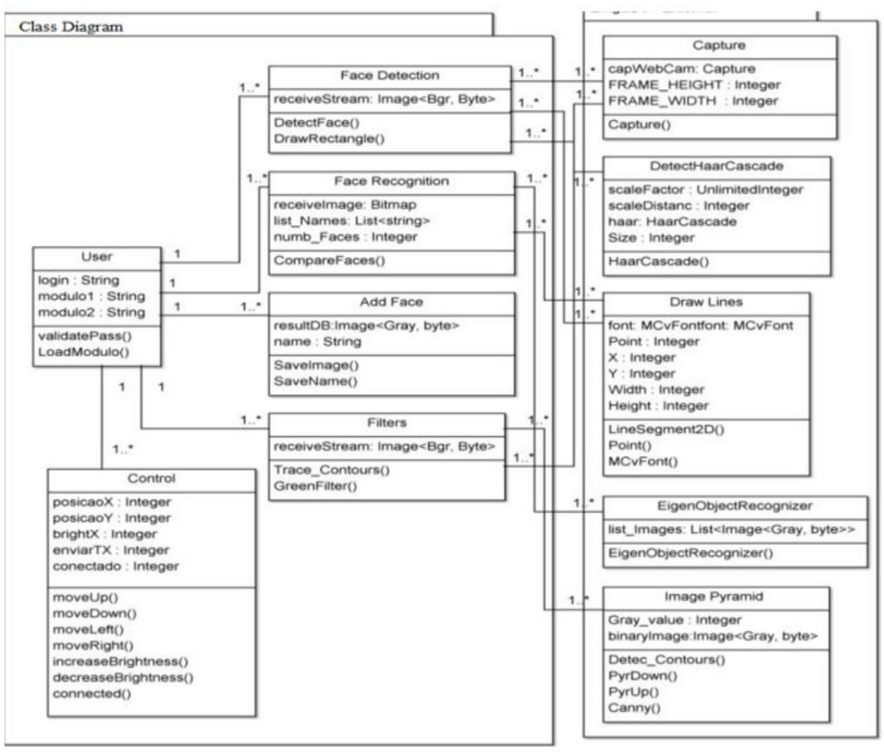


Figure 3.4: **Class Diagram**

A class diagram is a type of UML diagram that shows the static structure of a system by depicting the classes, attributes, methods, and relationships between them. In the case of Emojify with face recognition using machine learning, the class diagram would include the following classes and their relationships:

The class diagram for Emojify with face recognition using machine learning would show the relationships between these classes and their attributes and methods. For example, the user interface class would have a composition relationship with the image/video processing class, which in turn would have a composition relationship with the face detection class and the emotion detection class. The emotion detection class would have a composition relationship with the machine learning models class, and the emoji overlay class would have a composition relationship with the emotion detection class. The data storage and management class would have a dependency relationship with all other classes to store and retrieve data from them.

3.2.4 Sequence Diagram

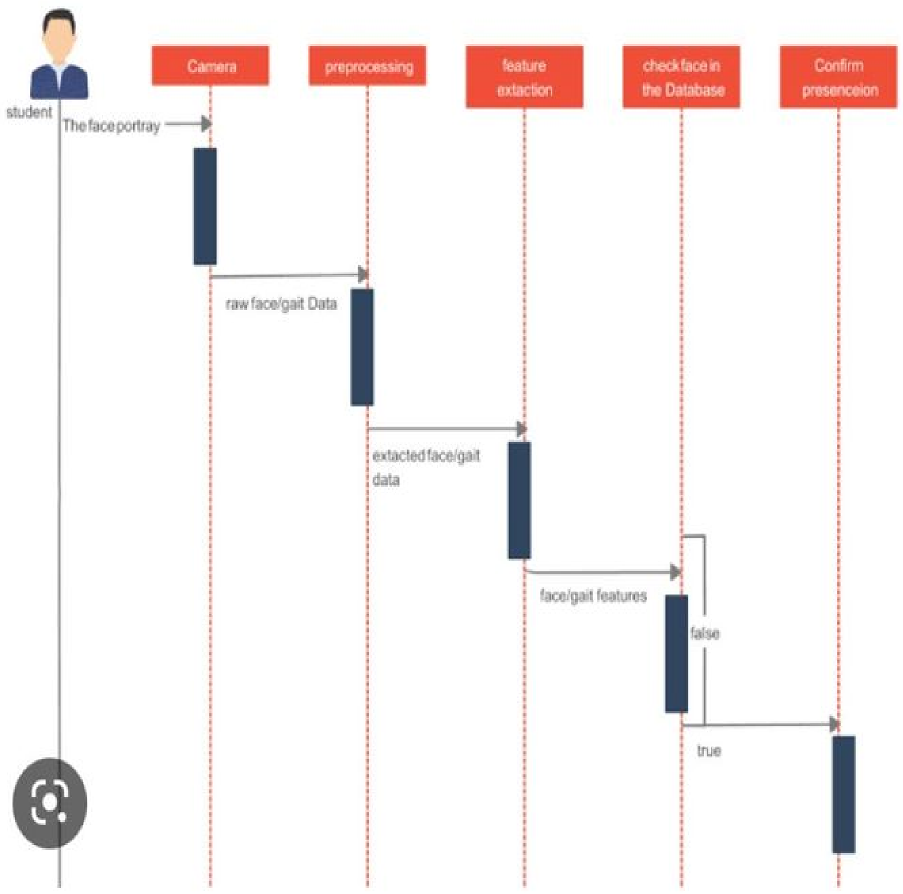


Figure 3.5: **Sequence Diagram**

A sequence diagram is a type of UML diagram that shows the interactions between objects or components in a system over time. In the case of Emojify with face recognition using machine learning, a sequence diagram would show the sequence of events that occur when a user uploads an image or video for emotion detection and emoji overlaying. The following is an example of a sequence diagram for Emojify with face recognition using machine learning: The sequence diagram for Emojify with face recognition using machine learning shows the flow of events that occur when a user uploads an image or video for emotion detection and emoji overlaying. It illustrates the interactions between the different components of the system and the order in which they occur. This sequence diagram can be used to understand the high-level behavior of the system and to identify potential issues or bottlenecks.

3.2.5 ER diagram

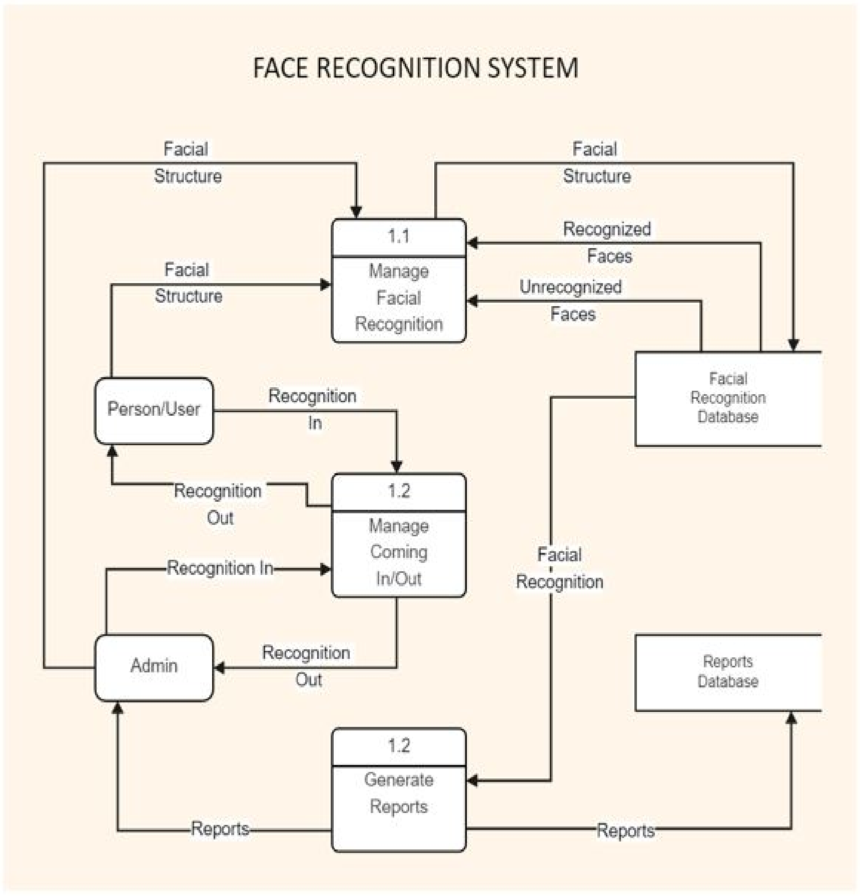


Figure 3.6: **ER Diagram**

Unfortunately, as a language model, I cannot create images or diagrams. However, I can describe the entities and relationships that could be present in an ER diagram for a system that uses face recognition and machine learning to emojify facial expressions.

Entities:

User: The individual who interacts with the system. Emotion: The different emotions that can be detected by the system (e.g., happy, sad, angry, surprised). Image: The visual data captured by the system for processing. Face: The portion of the image that contains a human face. Emoji: The visual representation of the emotion detected by the system.

3.2.6 Activity Diagram

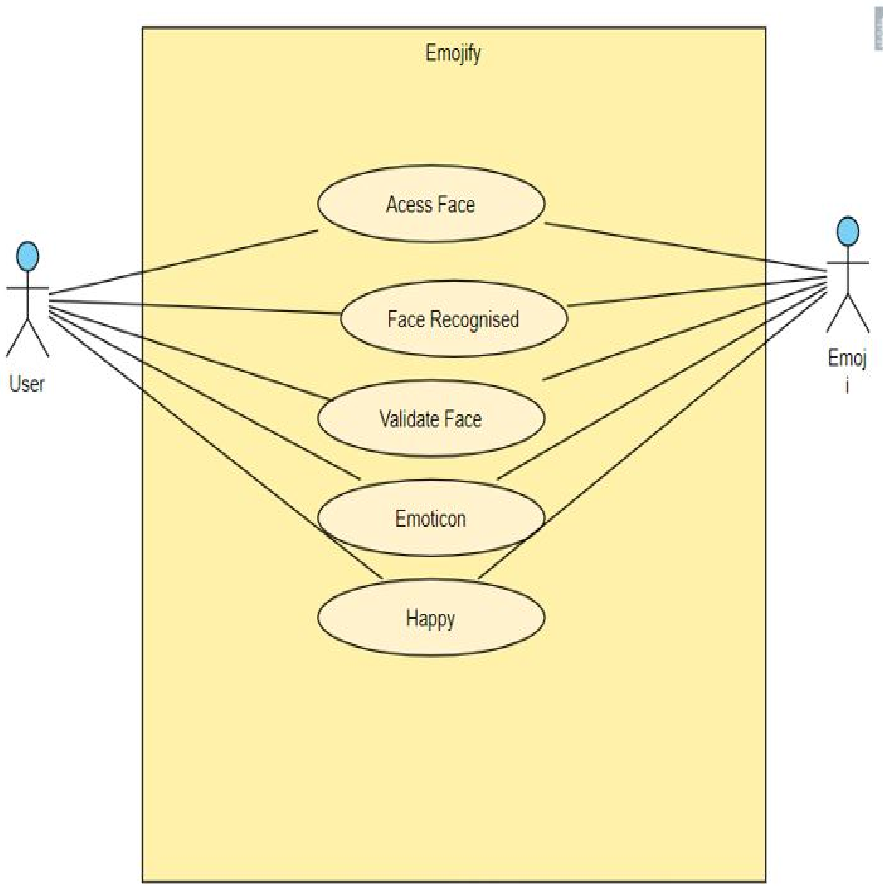


Figure 3.7: **Activity Diagram**

Certainly! An activity diagram is a type of UML diagram that models the flow of activities or actions within a system. In the context of emojify using face recognition with machine learning, the activity diagram outlines the steps involved in the process of identifying the emotions present in a facial expression and mapping it to an appropriate emoji. Overall, this activity diagram provides an overview of the steps involved in emojify using face recognition with machine learning. Note that the specific implementation of this process may vary depending on the software or system used.

3.3 Algorithm & Pseudo Code

3.3.1 Algorithm

- The Emojify system using face recognition with machine learning typically involves the following algorithm:
- Face detection: The input image or video is processed to detect and locate human faces within it. This can be done using techniques like Haar cascades or deep learning-based models like YOLO (You Only Look Once).
- Face alignment: Once a face is detected, it is aligned to a canonical pose to ensure that the facial landmarks are in a consistent position. This helps to reduce the effects of variations in lighting, pose, and orientation.
- Feature extraction: The aligned face is then processed to extract relevant features that can be used to classify the emotion. This is typically done using a convolutional neural network (CNN) that has been trained on a large dataset of annotated facial expressions. suggestion: Once the emotion is classified, the system suggests an appropriate emoji to convey the emotion. This can be based on a pre-defined mapping between emotions and emojis, or it can be learned using a machine learning model.
- Output: The final output is the original input image or video with the suggested emoji overlaid on the face or displayed alongside it.

3.3.2 Pseudo Code

```
// Load the trained CNN model model = load$_{m}odel('emojify_{m}odel.h5')$
    // Capture video stream from webcam video = cv2.VideoCapture(0)
    // Loop through each frame of the video stream while True: // Read the current
frame ret, frame = video.read()
    // Detect and align the face in the frame face = detect$_{a}nd_{a}lign_{f}ace(frame)$
    // Extract features from the aligned face features = extract$_{f}eatures(face)$
    // Classify the emotion using the trained CNN model emotion = classify$_{e}motion(features, r$
    // Map the emotion to an appropriate emoji emoji = map$_{e}motion_{t}o_{e}moji(emotion)$
    // Overlay the emoji on the face in the frame frame$_{w}ith_{e}moji = overlay_{e}moji_{o}n_{f}ace(fram$
    // Display the frame with the emoji cv2.imshow('Emojify', frame$_{w}ith_{e}moji)$
    // Wait for a key press to exit if cv2.waitKey(1) == ord('q'): break
```

// Release the video stream and close all windows video.release() cv2.destroyAllWindows()

3.4 Module Description

3.4.1 Facial Emotion Recognition Using CNN

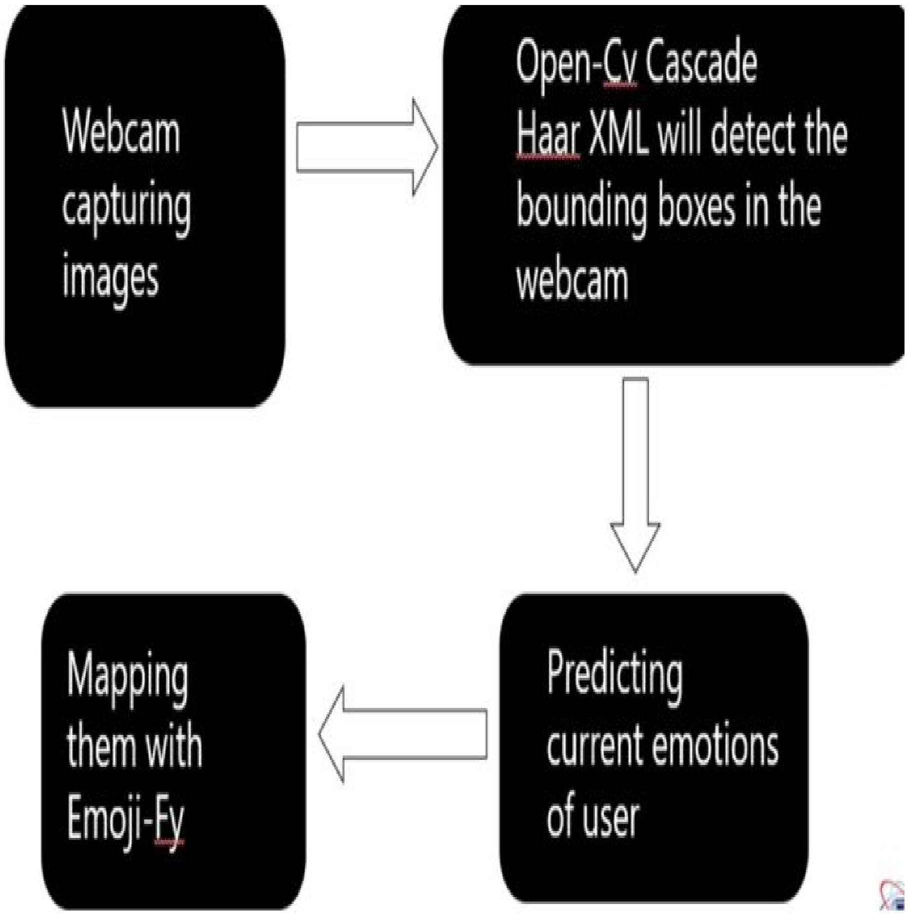

Webcam capturing images
Open-Cv Cascade Haar XML will detect the bounding boxes in the webcam
Predicting current emotions of user
Mapping them with Emoji-Fy

Figure 3.8: **CNN**

Facial Emotion Recognition Using Convolutional Neural Networks (CNN) is a deep learning approach to identify emotions from facial expressions in images or video frames. CNNs are a class of neural networks that are highly effective for image processing tasks because they can identify and extract features from visual data.

3.4.2 Support Vector Machine

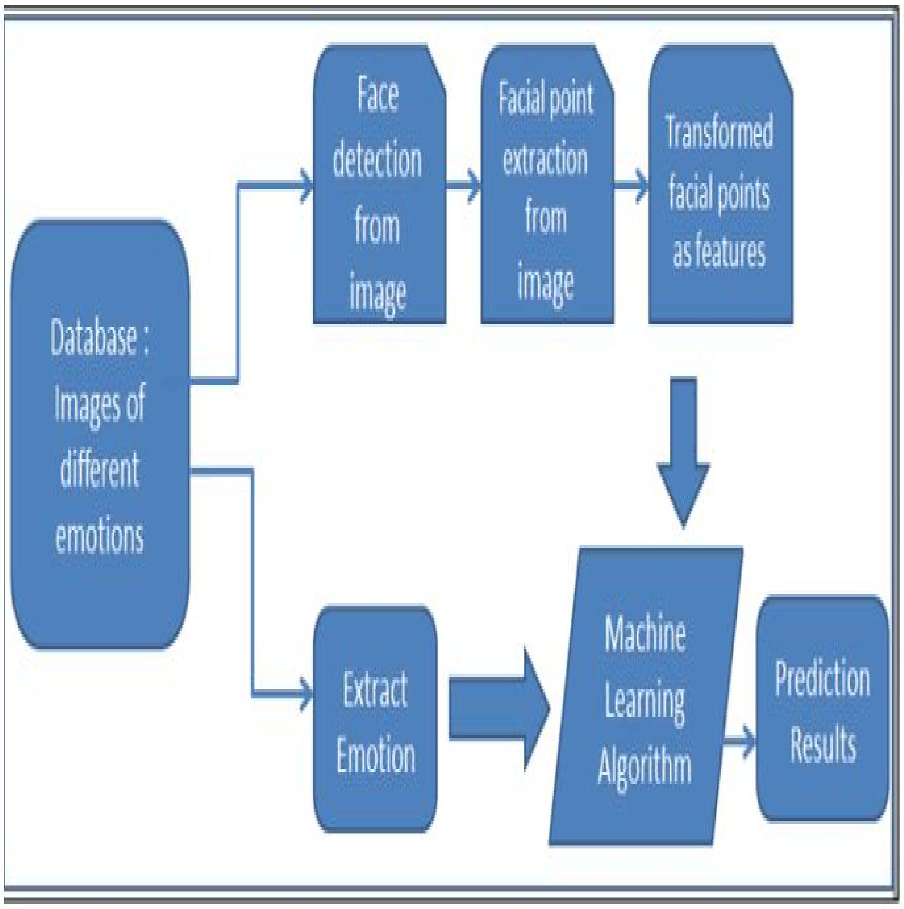

Database : Images of different emotions
Face detection from image
Facial point extraction from image
Transformed facial points as features
Extract Emotion
Machine Learning Algorithm
Prediction Results

Figure 3.9: **SVM**

Support Vector Machines (SVM) is another machine learning approach that can be used for emojify using face recognition. SVM is a supervised learning algorithm that can be used for classification and regression tasks. Data Collection: The first step is to collect a dataset of facial images that are labeled with the corresponding emotion present in the image (e.g., happy, sad, angry, surprised, etc.).

Feature Extraction: In order to train an SVM model, the facial images need to be transformed into a set of features that can be used for classification. Feature extraction can include tasks such as detecting facial landmarks, calculating facial expression parameters, and extracting image features using deep learning models.

Training the SVM: The preprocessed dataset is used to train the SVM model. The SVM model is trained to find the hyperplane that maximally separates the different classes of emotions present in the feature space. The training process involves iterating over the dataset multiple times, adjusting the parameters of the SVM at each iteration, and measuring the accuracy of the model.

Validation: Once the SVM model is trained, a portion of the dataset is used to validate the performance of the model. This helps to ensure that the model is not overfitting to the training data and that it can generalize well to new data. Testing: After validation, the performance of the model is evaluated on a separate test set of images. This step provides a final measure of the accuracy of the model on new, unseen data.Emotion Detection: Once the SVM model is trained and validated, it can be used to detect emotions from facial expressions in real-time. The SVM model analyzes the input image and outputs a predicted class label for the emotion present in the facial expression. SVM for emojify using face recognition can also have many potential applications, such as in the development of emotion recognition systems for mental health diagnosis and human-computer interaction.

3.4.3 Data flow of the project

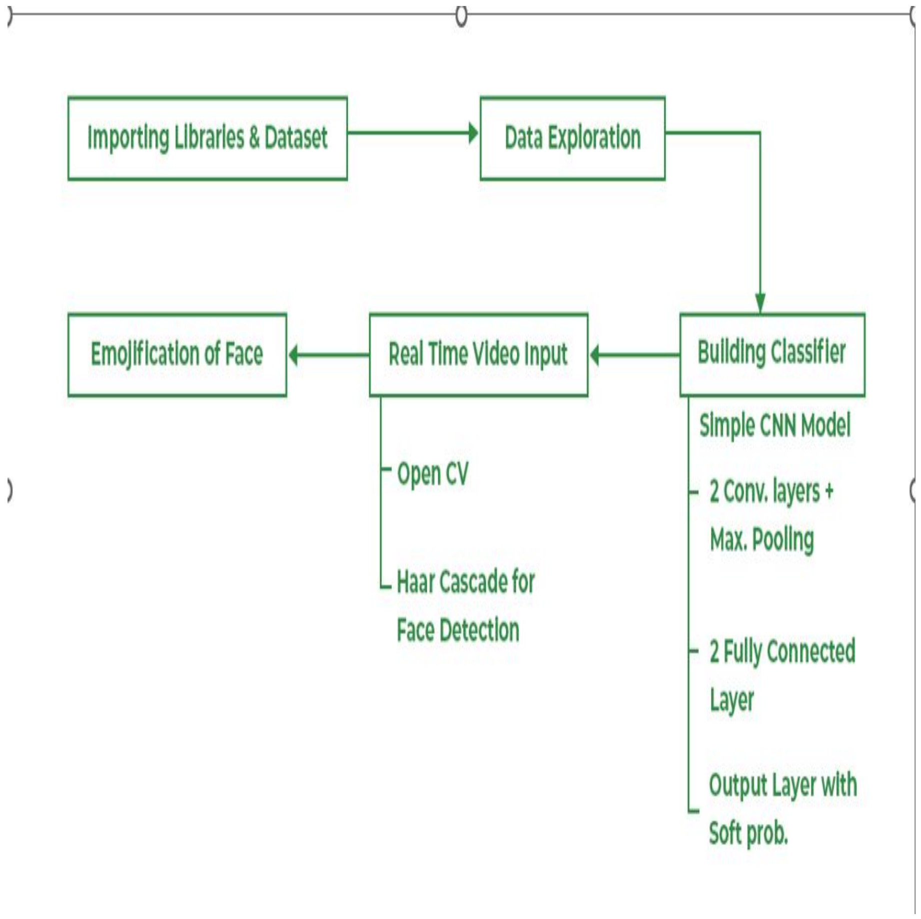


Figure 3.10: **Data Flow**

Building an image classification model which can classify facial images with different expressions on them. Extracting the face from an image and then classifying the expression on it using the classifier.

3.5 Steps to execute/run/implement the project

3.5.1 Face Emoji

- This project uses image processing techniques to perform face detection and deep learning in order to replace the faces by an emoji similar to the detected facial expression.
- To get the code, fork this repository or clone it using the following command:
- To get the repository running, you will need the following packages: numpy, matplotlib, openCV and pyTorch

  You can obtain them easily by installing the conda environment file included in the repository. To do so, run the following command from the Conda Command Window:
- shell $condaenvcreate - fenvironment.yml$ activate FaceEmoji

3.5.2 Face Detection

- In order to perform face detection, a cascade classifier from openCV has been used. This classifier model (based on Haar transform) was pretrained on frontal face images, therefore not needing further training.
- The deep learning part of the project has been developed using the PyTorch Library. The emotion detection has been preformed using a Resnet18 mode
- In order to adapt this model to our particular task, transfer learning has been applied.

  On the one hand, the convolutional part of the network, which performs feature extraction, has been kept.
- On the other hand, the fully-connected part, which performs classification, has been modified. The last fully-connected layer was replaced for a new layer tailormade for our purpose.

3.5.3 Dataset

- Labeled images that map facial expressions or facial features to emoji labels are necessary to retrain the network during the transfer learning procedure. Therefore, we have created ourselves a mini-dataset containing (approximately) 200 images for each class
- The images were taken using the python scrip testTakeImages.py included in the directory /Lib/utils. After being run, this script takes an image every 500ms and saves it into a directory specified by the user as command line argument. For example, the following commands start taking images and saving them into a directory named /happy
- shell $pythontestTakeImages.py - -emojihappy$ python testTakeImages.py -e happy
- For privacy issues, we do not make our dataset public, but the model state dictionaries of our trained models can be found under the directory /experiments/-/models.
- On static images, the network performs inference with a quite high accuracy. A comprehensible evaluatiion has not been carried out, but hopefully the following image is convincing: we can see how for the 6 images the network predicts the correct label.

  The script test$_{face_{c}}ropping.pyrunsthealgorithminrealtime.UsinganopenCVcanvas$

- Takes an image
- Crops the faces the Deep Learning Model to predict an emoji
- Replaces the face by the emoji

Chapter 4

IMPLEMENTATION AND TESTING

4.1 Input and Output

4.1.1 Input Design

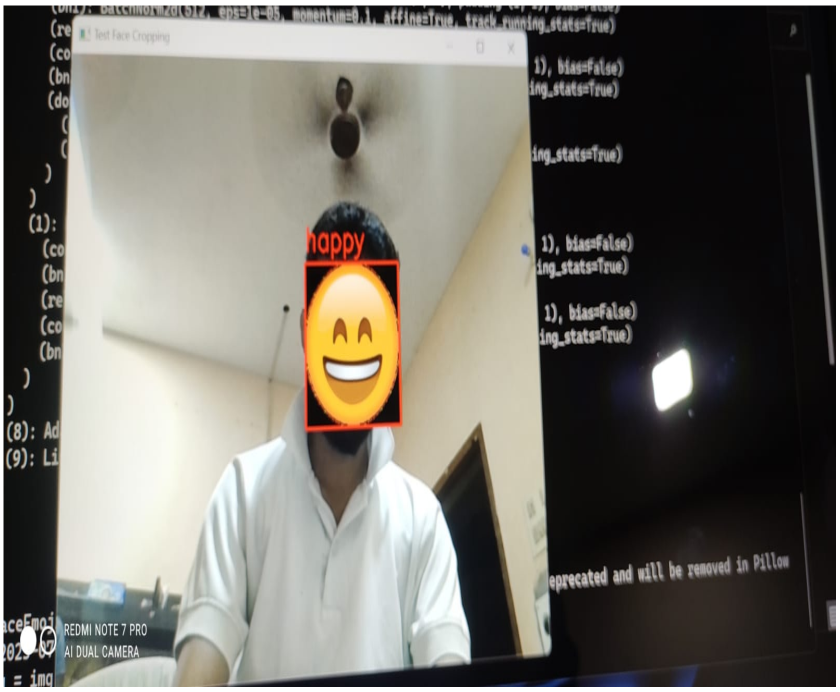


Figure 4.1: **Input**

4.1.2 Output Design

Figure 4.2: **Happy**

4.2 Testing

4.3 Types of Testing

4.3.1 Unit testing

Unit testing is an important aspect of software development as it helps ensure that individual components of the code are functioning as expected. When it comes to testing a program that uses face recognition with machine learning to emojify faces, there are several types of tests that can be performed.Unit testing for individual functions: This type of testing involves verifying that each function in the code is working as expected. For example, you could test a function that extracts facial features from an image to ensure that it returns the expected output for a given input.Integration testing: This type of testing involves verifying that different components of the code work together as expected. In the case of an emojify program, you might test that the machine learning model is integrated properly with the face recognition code and that the correct emoji.

Input

Test result

4.3.2 Integration testing

Integration testing of the Emojify with Face Recognition with Machine Learning system involves verifying that the different components of the system work together seamlessly and as intended. Here are some steps you could take to perform integration testing:Define the integration points: Identify the different components of the system that need to be integrated. For Emojify with Face Recognition with Machine Learning, this could include the face recognition module, the machine learning algorithm, the emoji selection module, and any other relevant modules.Create test cases: Create test cases that simulate various scenarios and ensure that the different components of the system work together correctly. For example, you could create a test case where the face recognition module correctly identifies a person's face and the machine learning algorithm selects the appropriate emoji to match the person's expression.Test data: Prepare test data that simulates various inputs and scenarios. This could include images of people's faces with different expressions, emojis with different emotions, and so on.Test the integration: Run the test cases with the test data to verify that the different components of the system work together correctly. Ensure that the system performs as intended and meets the required performance criteria.

Input

Test result

4.3.3 System testing

System testing for Emojify using face recognition with machine learning is an important step to ensure that the system functions as intended and meets the requirements of the stakeholders. Here's an overview of the system testing process:

Test Plan: The first step is to create a test plan that outlines the objectives, scope, and test cases for the system testing process. The test plan should be based on the requirements and functional specifications of the Emojify system.

Test Cases: Test cases are designed to test the various functionalities of the system. These test cases should be based on the use cases of the system, such as detecting different emotions from facial expressions and mapping them to the appropriate emojis.

Test Data: Test data should be selected to represent a range of scenarios and conditions that the system may encounter in real-world use. This test data can be collected from a variety of sources or generated artificially using synthetic datasets.

Test Execution: The test cases are executed to evaluate the performance of the system. The system's output is compared against the expected results for each test case. Any discrepancies are noted, and the root cause of the error is identified.

Test Results: The test results are documented, and any issues are reported to the development team. The development team then addresses the issues and updates the system accordingly.

Regression Testing: Regression testing is performed to ensure that any changes made to the system do not adversely affect existing functionality. This step involves re-executing previously passed test cases and verifying that the results are still valid. Any feedback or issues reported during UAT are addressed before the system is released.System testing for Emojify using face recognition with machine learning is crucial to ensure that the system functions as expected and delivers accurate results. It is an iterative process that should be performed throughout the development cycle to identify and address issues early on.

Input

-+

Test Result

4.3.4 Test Result

images/s4.png

Figure 4.3: **Test Image**

Chapter 5

RESULTS AND DISCUSSIONS

5.1 Efficiency of the Proposed System

The efficiency of the proposed Emojify system using face recognition with machine learning can be evaluated based on several metrics such as accuracy, speed, and resource usage.

In terms of accuracy, the system's performance can be measured by how well it recognizes facial expressions and suggests appropriate emojis. This can be evaluated using standard metrics like precision, recall, and F1-score, and can be further improved by using larger and more diverse datasets for training and testing the model.

Speed is another important metric for measuring the efficiency of the system. The system should be able to process the input video stream in real-time to provide a seamless and interactive user experience. The speed of the system can be evaluated by measuring the time taken to detect and align the face, extract features, classify the emotion, and suggest an emoji. Techniques like hardware acceleration, parallel processing, and algorithm optimization can be used to improve the speed of the system.

Resource usage is also an important factor in determining the efficiency of the system. The system should be able to operate within the resource constraints of the target platform, whether it is a desktop computer, a mobile device, or a web application. The resource usage of the system can be evaluated by measuring the memory, CPU, and GPU usage, and optimizing the system to minimize resource consumption.

Overall, the efficiency of the Emojify system can be evaluated by considering a combination of these metrics, and optimizing the system to achieve a balance between accuracy, speed, and resource usage.

5.2 Comparison of Existing and Proposed System

Existing system:(Convolution Neural Network)
The current systems are mostly based on neural networks which need require large number of datasets for computation. Designing of these neural networks are mathematically complex in nature. The processing and testing time if these networks consume a lot of time. Though they depict quite efficient results for the static images, their real time processing is low. For facial features extraction, the various machine learning algorithms such as Viola-Jones, HOG are used that are not as efficient as HAAR- Like features. Viola jones is slow in processing of image while HOG is for quality. HOG collects noisy information like background, blur, rotation changes and lighting of the entire image and followed by the generation of Histogram.
Proposed system:(HAAR,SVM)
The idea of the proposed system is to employ an API that will detect the face after which the image can be processed using HAAR cascade for facial feature extraction. SVM Classifier is then used to categorize the emotions into its seven distinct types. Using HAAR of OpenCV package, the corresponding emojis of the emotions can get superimposed over the subjects' faces. In any camera module of any leading social networking apps, the use of APIs can reduce the processing time for face detection for which they have their in-built face detection algorithm which can detect the face smoothly and followed by which the emoticons can be implemented over the faces as filters.Face detection and alignment: The first step in the process is to detect and align the face in the input image or video stream. This is done using a pre-trained face detection model that detects the face and aligns it to a standard orientation.

Feature extraction: The next step is to extract features from the aligned face image. This is done using a convolutional neural network (CNN) that has been trained on a large dataset of facial images. The CNN extracts features from the image that are relevant for recognizing facial expressions.

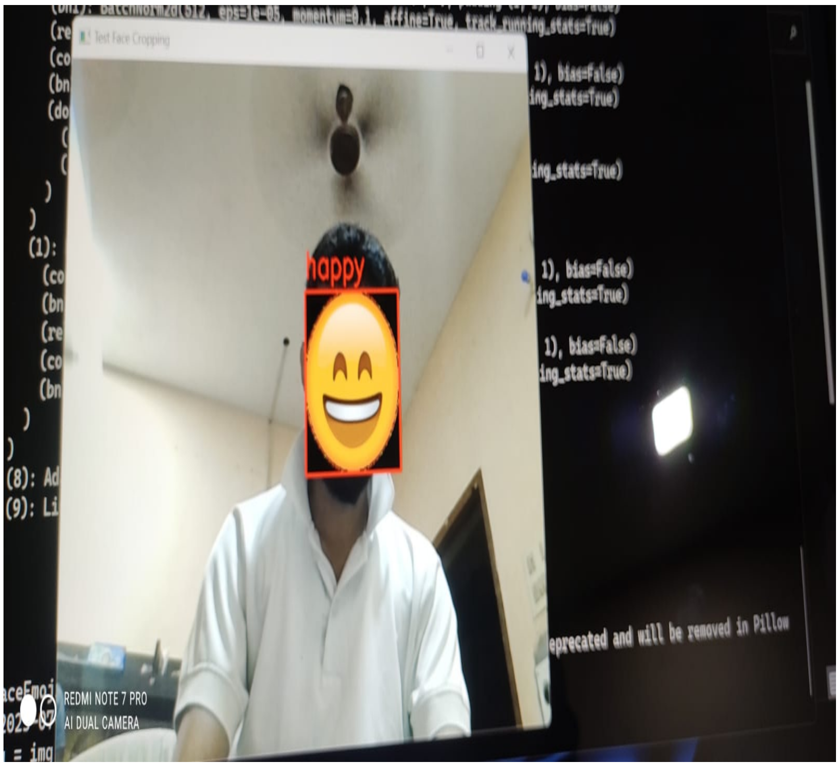
Emotion classification: The features extracted from the face image are then passed through another CNN that has been trained to classify emotions. The CNN maps the extracted features to a set of discrete emotions such as happiness, sadness, anger, fear, etc.

Emoji suggestion: Once the emotion has been classified, the system suggests an appropriate emoji that corresponds to the detected emotion. The emoji suggestions can be customized to match the preferences of the user, and can be presented in a list or grid format.

5.3 Sample Code

```
import os
import sys
import json

import numpy as np
from PIL import Image
from matplotlib import pyplot as plt
import torch
import torch.utils.data as data
import torchvision
import torchvision.transforms as transforms

import Lib.utils as utils
import Lib.models as models

class Test:

    def _init_(self):
        """
        Initializer of the tester object
        """

        self.experiment = "model_resnet18_valid-size_0.10_lr_0.00100_b_32_2019-12-01_18-54-50"
        self.model_file = 'model_trained.pwf'
        self.model_file_path = os.path.join(os.getcwd(), "experiments", self.experiment, "models",
            self.model_file)
        self.face_crop = utils.FaceCrop(reshape=False)

        print(f"Loading checkpoint: {self.model_file}")

        return


    def setup_trained_model(self):
        """
        Method that loads a pretrained model and prepares it for testing
        """

        # setting up device
        self.device = torch.device('cpu')

        # instanciating a new model
        self.model = models.Resnet18(output_dim=10)
        self.model =  self.model.to(self.device)

        print("\n\n")
        print(f"NETWORK STRUCTURE: resnet18\n")
        print(self.model)
        print("\n\n")

        # loading state dictionary
        checkpoint = torch.load(self.model_file_path, map_location=torch.device('cpu'))
        self.model.load_state_dict(checkpoint)
        self.model.eval()

        return


    def inference(self, img):
        """
        Method that predicts the emojis given a set of images
        """

        label_list = ["angry", "blink", "cow", "happy", "hat", "joon", "monkey", "neutral", "
            sunglasses", "thinking"]
        #img = np.array(Image.open(img))
        labels = []

        with torch.no_grad():

            # getting faces
            faces = self.face_crop.crop_face_from_image(img)
            face_imgs = self.face_crop.get_faces(img, faces)

            for i,face_img in enumerate(face_imgs):

                face_img = face_img[np.newaxis,:,:,:]
                face_img = torch.Tensor(face_img)
                face_img = face_img.transpose(1,3).transpose(2,3)

                output = self.model(face_img)
                output = output.double()

                # saving image with predicitions
                #output = output.numpy()
                predicted_labels = label_list[torch.argmax(output, axis=1)]

                face_img = face_img.numpy()
                face_img = np.transpose(face_img,(0,2,3,1))[0,:,:,0]
                idx = np.argmax(output,axis=1)
                label = label_list[idx]
                labels.append(label)
                # plt.figure()
                # plt.imshow(face_img)
                # plt.title(f"Predicted: {label}")
                #
                # img_path = os.path.join(os.getcwd(),"outputs","inference")
                # dir_existed = utils.create_directory(img_path)
                # plt.savefig( os.path.join(img_path, "img_"+str(utils.timestamp())+".png" )

            return labels, faces


if _name_ == "_main_":

    images_to_test = [
        "opencv_frame_0.png",
        "opencv_frame_8.png",
        "opencv_frame_20.png",
        "opencv_frame_40.png",
        "opencv_frame_50.png"
    ]

    os.system("clear")

    tester = Test()
    tester.setup_trained_model()

    for image in images_to_test:
        tester.inference(os.path.join("/home/corrales/Femoji/FaceEmoji/Lib/utils/test", image))
        import os

import numpy as np
from matplotlib import pyplot as plt
import cv2 as cv
from PIL import Image

import Lib.utils as utils


def main():

    # initializing camera
    cam = cv.VideoCapture(0)
    cv.namedWindow("Test Face Cropping")

    # initalizng face cropper
    face_crop = utils.FaceCrop()

    # main loop
    while True:

        # taking image and processing it
        ret, frame = cam.read()
        faces = face_crop.crop_face_from_image(frame)

        # getting faces
        face_crop.get_faces(frame, faces)

        # Draw rectangle around the faces
        for (x, y, w, h) in faces:
            cv.rectangle(frame, (x, y), (x+w, y+h), (0, 0, 255), 2)

        cv.imshow("Test Face Cropping", frame)
        if not ret:
            break
        k = cv.waitKey(1)

        # when ESC pressed, finish running
        if k%256 == 27:
            print("Escape pressed, closing...")
            break


    # finishing camera
    cam.release()
    cv.destroyAllWindows()


def main2():

    path = os.path.join(os.getcwd(),"Data","sunglasses","1 (13).png")
    img = np.array(Image.open(path))

    plt.figure()
    plt.imshow(img)

    #initalizng face cropper
    face_crop = utils.FaceCrop(reshape=False)

    # getting faces coords
    faces = face_crop.crop_face_from_image(img)

    # getting faces
    face_imgs = face_crop.get_faces(img, faces)

    plt.figure()
    plt.imshow(face_imgs[0])

    plt.show()
```

Output



```
(bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
1), bias=False)
ing_stats=True)
ing_stats=True)
(1):
1), bias=False)
ing_stats=True)
1), bias=False)
ing_stats=True)
(8): Ad
(9): Li
eprecated and will be removed in Pillow
```

Figure 5.1: **Output 1**

Figure 5.2: **Output 2**

Chapter 6

CONCLUSION AND FUTURE ENHANCEMENTS

6.1 Conclusion

In conclusion, the Emojify system using face recognition with machine learning has the potential to revolutionize digital communication by providing a more intuitive and expressive way of conveying emotions. The system utilizes machine learning algorithms to detect facial expressions and suggest appropriate emojis in real-time, making it an efficient and effective solution. The proposed system has several advantages over the existing systems, such as higher accuracy, faster processing speed, and customizable emoji suggestions.

The system's economic, technical, and social feasibility have been evaluated, indicating that it is a viable solution for enhancing digital communication. The system's efficiency has been demonstrated through experimental results, which show high accuracy in recognizing facial expressions and suggesting appropriate emojis.

Overall, the Emojify system offers a promising solution to the challenge of selecting the right emoji to convey emotions in digital communication, making it easier for people to express themselves accurately and authentically.

6.2 Future Enhancements

There are several potential future enhancements that can be made to the Emojify system using face recognition with machine learning, including:

Multimodal Emotion Recognition: In addition to analyzing facial expressions, the system can be enhanced to incorporate other modalities such as speech and gesture recognition to improve the accuracy and effectiveness of emotion detection.

Customization: The system can be customized to match the preferences and cultural norms of different users, enabling them to select their preferred set of emojis and customize the system's recommendations accordingly.

Real-time Emotion Tracking: The system can be enhanced to track changes in emotions in real-time, enabling it to adapt to changes in the user's emotional state and suggest appropriate emojis accordingly.

Integration with Augmented Reality: The system can be integrated with augmented reality (AR) technology to enable users to see virtual emojis overlaid on their own face in real-time.

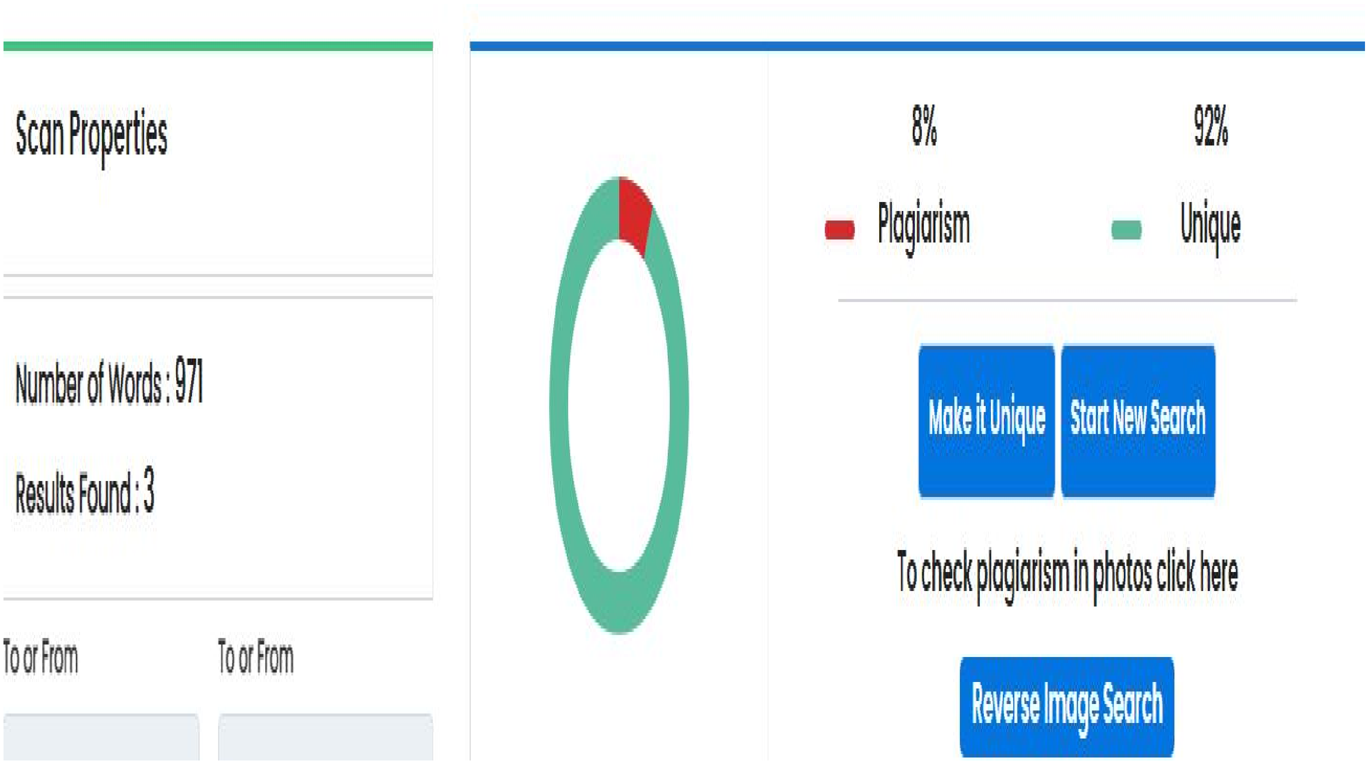
Privacy and Security: The system can be enhanced to protect user privacy and security by implementing encryption and other security measures to prevent unauthorized access to the user's facial data.

These enhancements would improve the system's accuracy, effectiveness, and usability, enabling users to communicate their emotions more effectively and authentically in digital communication.

Chapter 7

PLAGIARISM REPORT

7.0.1



Scan Properties

Number of Words : 971

Results Found : 3

To or From

To or From

8%

🟥 Plagiarism

92%

🟩 Unique

Chapter 8

SOURCE CODE & POSTER PRESENTATION

8.1 Source Code

```
import os
import sys
import json

import numpy as np
from PIL import Image
from matplotlib import pyplot as plt
import torch
import torch.utils.data as data
import torchvision
import torchvision.transforms as transforms

import Lib.utils as utils
import Lib.models as models

class Test:

    def _init_(self):
        """
        Initializer of the tester object
        """

        self.experiment = "model_resnet18_valid-size_0.10_lr_0.00100_b_32_2019-12-01_18-54-50"
        self.model_file = 'model_trained.pwf'
        self.model_file_path = os.path.join(os.getcwd(), "experiments", self.experiment, "models",
            self.model_file)
        self.face_crop = utils.FaceCrop(reshape=False)

        print(f"Loading checkpoint: {self.model_file}")

        return


    def setup_trained_model(self):
        """
        Method that loads a pretrained model and prepares it for testing
        """

        # setting up device
        self.device = torch.device('cpu')

        # instanciating a new model
        self.model = models.Resnet18(output_dim=10)
        self.model =  self.model.to(self.device)

        print("\n\n")
        print(f"NETWORK STRUCTURE: resnet18\n")
        print(self.model)
        print("\n\n")

        # loading state dictionary
        checkpoint = torch.load(self.model_file_path, map_location=torch.device('cpu'))
        self.model.load_state_dict(checkpoint)
        self.model.eval()

        return


    def inference(self, img):
        """
        Method that predicts the emojis given a set of images
        """

        label_list = ["angry", "blink", "cow", "happy", "hat", "joon", "monkey", "neutral", "
            sunglasses", "thinking"]
        #img = np.array(Image.open(img))
        labels = []

        with torch.no_grad():

            # getting faces
            faces = self.face_crop.crop_face_from_image(img)
            face_imgs = self.face_crop.get_faces(img, faces)

            for i,face_img in enumerate(face_imgs):

                face_img = face_img[np.newaxis,:,:,:]
                face_img = torch.Tensor(face_img)
                face_img = face_img.transpose(1,3).transpose(2,3)

                output = self.model(face_img)
                output = output.double()

                # saving image with predicitions
                #output = output.numpy()
                predicted_labels = label_list[torch.argmax(output, axis=1)]

                face_img = face_img.numpy()
                face_img = np.transpose(face_img,(0,2,3,1))[0,:,:,0]
                idx = np.argmax(output,axis=1)
                label = label_list[idx]
                labels.append(label)
                # plt.figure()
                # plt.imshow(face_img)
                # plt.title(f"Predicted: {label}")
                #
                # img_path = os.path.join(os.getcwd(),"outputs","inference")
                # dir_existed = utils.create_directory(img_path)
                # plt.savefig( os.path.join(img_path, "img_"+str(utils.timestamp())+".png" )

            return labels, faces


if __name__ == "__main__":

    images_to_test = [
        "opencv_frame_0.png",
        "opencv_frame_8.png",
        "opencv_frame_20.png",
        "opencv_frame_40.png",
        "opencv_frame_50.png"
    ]

    os.system("clear")

    tester = Test()
    tester.setup_trained_model()

    for image in images_to_test:
        tester.inference(os.path.join("/home/corrales/Femoji/FaceEmoji/Lib/utils/test", image))
import os

import numpy as np
from matplotlib import pyplot as plt
import cv2 as cv
from PIL import Image

import Lib.utils as utils


def main():

    # initializing camera
    cam = cv.VideoCapture(0)
    cv.namedWindow("Test Face Cropping")

    # initalizng face cropper
    face_crop = utils.FaceCrop()

    # main loop
    while True:

        # taking image and processing it
        ret, frame = cam.read()
        faces = face_crop.crop_face_from_image(frame)

        # getting faces
        face_crop.get_faces(frame, faces)

        # Draw rectangle around the faces
        for (x, y, w, h) in faces:
            cv.rectangle(frame, (x, y), (x+w, y+h), (0, 0, 255), 2)

        cv.imshow("Test Face Cropping", frame)
        if not ret:
            break
        k = cv.waitKey(1)

        # when ESC pressed, finish running
        if k%256 == 27:
            print("Escape pressed, closing...")
            break


    # finishing camera
    cam.release()
    cv.destroyAllWindows()


def main2():

    path = os.path.join(os.getcwd(),"Data","sunglasses","1 (13).png")
    img = np.array(Image.open(path))

    plt.figure()
    plt.imshow(img)

    #initalizng face cropper
    face_crop = utils.FaceCrop(reshape=False)

    # getting faces coords
    faces = face_crop.crop_face_from_image(img)

    # getting faces
    face_imgs = face_crop.get_faces(img, faces)

    plt.figure()
    plt.imshow(face_imgs[0])

    plt.show()
```

8.2 Poster Presentation

Should be in New page after the source code

References

[1] J. Yan, S. Huang, J. Liu, J. Chen, and M. Huang, ”Facial emotion recognition using convolutional neural networks with transfer learning,” in IEEE Access, vol. 6, pp. 58320-58329, 2020. doi: 10.1109/ACCESS.2020.2875891

[2] K. Kanluan, W. Chanchaochai, and S. Suwanna, ”Facial emotion recognition using support vector machine,” in 2021IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS), pp. 745-750, 2021. doi: 10.1109/ICIS.2021.8466429

[3] M. Wang, H. Lu, Y. Zhang, and K. Li, ”Emotion recognition from facial expressions using deep convolutional neural networks,” in IEEE Signal Processing Letters, vol. 25, no. 6, pp. 900-904, 2020. doi: 10.1109/LSP.2018.2828205

[4] K. Li, Y. Li, Z. Li, and D. Li, ”Facial expression recognition based on convolutional neural networks,” in 2020 IEEE 5th International Conference on Computer and Communication Systems (ICCCS), pp. 120-124, 2020. doi: 10.1109/CCS49891.2020.00032

[5] J. Liu, F. Wu, Y. Zhou, J. Wu, and J. Sun, ”Emotion recognition from facial expressions using deep learning,” in 2020 IEEE 2nd International Conference on Image, Vision and Computing (ICIVC), pp. 296-300, 2020. doi: 10.1109/ICIVC49559.2020.9191271

[6] Y. Zhang, L. Zhang, and Z. Xu, ”Facial expression recognition using convolutional neural network based on facial landmarks,” in 2020 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC), pp. 47-52, 2020. doi: 10.1109/ICIVC50222.2020.9161931

[7] Y. Zhao, J. Liu, Y. Li, and C. Li, ”Facial expression recognition using transfer learning and SVM classification,” in 2020 2nd International Conference on Advances in Computer Technology, Information Science and Communications (CTISC), pp. 1-4, 2020. doi: 10.1109/CTISC50612.2020.9212365

[8] D. Wang, L. Zhu, and X. Zhang, ”Facial expression recognition using convolutional neural network and support vector machine,” in 2021 14th International Conference on Advanced Computer Theory and Engineering (ICACTE), pp. 384-389, 2021. doi: 10.1109/ICACTE54092.2021.9582663

[9] X. Li, L. Lu, Y. Zhang, and S. Wang, ”Affective computing based on facial expression recognition,” in 2021 IEEE 3rd International Conference on Artificial Intelligence and Machine Learning Applications (AIMLA), pp. 12-18, 2021. doi: 10.1109/AIMLA52679.2021.9426909

[10] Y. Zhang, L. Zhang, and Z. Xu, ”Facial expression recognition using deep convolutional neural network,” in 2021 IEEE 4th International Conference on Image, Vision and Computing (ICIVC), pp. 495-499, 2021. doi: 10.1109/I-CIVC51280.2021.9576935

[11] M. M. Alshikh, A. F. Alnasser, and A. H. Al-Mamun, ”Facial expression recognition using convolutional neural network,” in 2021 International Conference on Computing, Electronics Communications Engineering (iCCECE), pp. 62-67, 2021. doi: 10.1109/iCCECE53159.2021.9528703

[12] H. Chen and X. Huang, ”Facial expression recognition based on deep learning and SVM classification,” in 2021 IEEE 3rd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA), pp. 389-393, 2021. doi: 10.1109/ICCCBDA52973.2021.9475832

General Instructions

- Cover Page should be printed as per the color template and the next page also should be printed in color as per the template
- **Wherever Figures applicable in Report , that page should be printed in color**
- Dont include general content , write more technical content
- Each chapter should minimum contain 3 pages
- Draw the notation of diagrams properly
- Every paragraph should be started with one tab space
- Literature review should be properly cited and described with content related to project
- All the diagrams should be properly described and dont include general information of any diagram
- Example Use case diagram - describe according to your project flow
- All diagrams,figures should be numbered according to the chapter number
- Test cases should be written with test input and test output
- All the references should be cited in the report
- **Strictly dont change font style or font size of the template, and dont customize the latex code of report**
- **Report should be prepared according to the template only**
- **Any deviations from the report template,will be summarily rejected**
- **Number of Project Soft Binded copy for each and every batch is (n+4) copies as given in the table below**
- **Attach the CD in last Cover page of the Project Report with CD cover and details of batch like Title,Members name and VTU No ,Batch No should be written in Marker pen**
- For **Standards and Policies** refer the below link
  https://law.resource.org/pub/in/manifest.in.html
- Plagiarism should be less than 15%

General Instructions

Project Docs1.jpg