

# **EMOTION DETECTION USING CONVOLUTIONAL NEURAL NETWORKS**

**A PROJECT REPORT**

*Submitted by*

**ABUTHAHIR RASICK A**

**211419205006**

**JAYAPRAKASH R**

**211419205078**

**GOPINATH N**

**211419205056**

*in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF TECHNOLOGY**

*in*

**INFORMATION TECHNOLOGY**

**PANIMALAR ENGINEERING COLLEGE, POONAMALLEE**

**ANNA UNIVERSITY : CHENNAI 600 025**

**APRIL 2023**

**ANNA UNIVERSITY: CHENNAI 600 025**

## **BONAFIDE CERTIFICATE**

Certified that this project report **“EMOTION DETECTION USING CONVOLOUTIONAL NEURAL NETWORKS”** is the bonafide work of **JAYAPRAKASH R (211419205078), ABUTHAHIR RASICK A (211419205006), GOPINATH N (211419205056)** who carried out the project under my supervision.

**SIGNATURE**

**Dr. M. HELDA MERCY M.E., Ph.D.,  
HEAD OF THE DEPARTMENT**

Department of Information Technology  
Panimalar Engineering College  
Poonamallee, Chennai - 600 123

**SIGNATURE**

**Dr.B.BUVANESWARI M.Tech.Ph.D,  
SUPERVISOR (Professor)**

Department of Information Technology  
Panimalar Engineering College  
Poonamallee, Chennai - 600 123

Submitted for the project and viva-voce examination held on \_\_\_\_\_

**SIGNATURE**

**INTERNAL EXAMINER**

**SIGNATURE**

**EXTERNAL EXAMINER**

## **DECLARATION**

I hereby declare that the project report entitled “**EMOTION DETECTION USING CONVOLUTIONAL NETWORKS**” which is being submitted in partial fulfilment of the requirement of the course leading to the award of the ‘Bachelor Technology in Information Technology ’in **Panimalar Engineering College, Autonomous Institution Affiliated to Anna University- Chennai** is the result of the project carried out by me under the guidance of **Dr. B.Buvaneswari.,M.Tech., Ph.D., Professor in the Department of Information Technology**. I further declared that I or any other person has not previously submitted this project report to any other institution/university for any other degree/ diploma or any other person.

**ABUTHAHIR RASICK A**

**JAYAPRAKASH R**

**GOPINATH N**

Date:

Place: Chennai

It is certified that this project has been prepared and submitted under my guidance.

Date:

**(Dr. B.Buvaneswari M.Tech., Ph.D.,)**

Place: Chennai

**(PROFESSOR/ IT)**

## ACKNOWLEDGEMENT

A project of this magnitude and nature requires kind co-operation and support from many, for successful completion . We wish to express our sincere thanks to all those who were involved in the completion of this project.

Our sincere thanks to **Our Honorable Secretary and Correspondent, Dr. P. CHINNADURAI, M.A., Ph.D.,** for his sincere endeavor in educating us in his premier institution. We would like to express our deep gratitude to **Our Dynamic Directors , Mrs. C. VIJAYA RAJESHWARI and Dr. C. SAKTHI KUMAR, M.E., M.B.A., Ph.D.,** and **DR.SARANYA SREE SAKTHIKUMAR, B.E., M.B.A., Ph.D.,** for providing us with the necessary facilities for completion of this project.

We also express our appreciation and gratefulness to **Our Principal Dr. K. MANI, M.E., Ph.D.,** who helped us in the completion of the project. We wish to convey our thanks and gratitude to our head of the department, **Dr. M. HELDA MERCY, M.E., Ph.D.,** Department of Information Technology, for her support and by providing us ample time to complete our project.

We express our indebtedness and gratitude to our Project coordinator **Mr. M. DILLI BABU, M.E.,(Ph.D.,)** Associate Professor, Department of Information Technology for his guidance throughout the course of our project.

We also express sincere thanks to our supervisor **Dr. B.Buvaneswari, M.Tech, Ph.D.,** Professor, Department of Information Technology for providing the support to carry out the project successfully. Last, we thank our parents and friends for providing their extensive moral support and encouragement during the course of the project.

## **TABLE OF CONTENTS**

<b>CHAPTER NO.</b>	<b>TITLE</b>	<b>PAGE NO</b>
	<b>ABSTRACT</b>	
	<b>LIST OF FIGURES</b>	
	<b>LIST OF ABBREVIATIONS</b>	
<b>1</b>	<b>INTRODUCTION</b>	
	1.1. OVERVIEW OF THE PROJECT	2
	1.2 NEED FOR THE PROJECT	3
	1.3 OBJECTIVE OF THE PROJECT	4
	1.4 SCOPE OF THE PROJECT	6
<b>2</b>	<b>LITERATURE SURVEY</b>	
	2.11 EXISTING SYSTEM	19
	2.12 DRAWBACK OF EXISTING SYSTEM	20
<b>3</b>	<b>SYSTEM DESIGN</b>	
	3.1 BLOCK DIAGRAM	23
	3.2 MODULE DESCRIPTION	24
	3.2.1 IMAGE PREPROCESSING	24
	3.2.2 FEATURE EXTRACTION	22
	3.2.3 FEATURE CLASSIFICATION	27
	3.3.4 EMOTION PREDICTION	29
	3.3 IMAGE ACQISATION	34
	3.4 IMAGE ENHANCEMENT	35
	3.5 IMAGE RESTORATION	35
	3.6 WORK FLOW	36
	3.6.1 WORK FLOW EXPLANATION	36
	3.7 UML DAIGRAM	38
	3.7.1 USE CASE DIAGRAM	39
	3.7.2 SEQUENCE DIAGRAM	40
<b>4</b>	<b>SYSTEM REQUIREMENT</b>	

	4.1 HARDWARE REQUIREMENT	42
	4.1.1 Laptop/Pc	42
	4.1.2 INTEL/AMD/ Processor	42
	4.2 SOFTWARE REQUIREMENT	43
	4.2.1 TENSOR FLOW	43
<b>5</b>	<b>IMPLEMENTATION</b>	
	5.1 SAMPLE CODE	52
	5.2 OUTPUT SCREENSHOT	59
<b>6</b>	<b>TESTING AND MAINTENANCE</b>	
	6.1 BLACK BOX TESTING	62
	6.2 WHITE BOX TESTING	62
	6.3 UNIT TESTING	62
	6.4 INTEGRATION TESTING	63
	6.5 SYSTEM TESTING	63
	6.6 ACCEPTANCE TESTING	63
	6.7 FUNCTIONAL TESTING	64
<b>7</b>	<b>CONCLUSION AND FUTURE ENHANCEMENTS</b>	
	7.1 CONCLUSION	66
	7.2 FUTURE ENHANCEMENT	70
	<b>REFERENCES</b>	

## ABSTRACT

The emotion detection using CNN (Convolutional Neural Network) aims to build a machine learning model that can accurately identify the emotions present in an input image or video frame. The project involves several stages, including data collection, preprocessing, model training, and evaluation. Firstly a dataset containing images or video frames with labeled emotions will be collected. The dataset may be obtained from publicly available sources or collected using various web scraping techniques. Next, the collected data will be preprocessed to ensure that it is suitable for training the CNN model. This may involve resizing the images or videos, normalizing the pixel values, and augmenting the data by applying various transformations such as flipping, rotating, or adding noise. The CNN model will be trained using the preprocessed data. The architecture of the CNN may include several convolutional layers, pooling layers, and fully connected layers. The model will be trained using backpropagation with a suitable optimizer such as Adam or stochastic gradient descent. Once the model is trained, it will be evaluated using a separate validation dataset. The evaluation metrics may include accuracy, precision, recall, and F1-score and the data. Finally, the trained model can be deployed in various applications, such as real-time emotion detection in videos or images, sentiment analysis in social media posts, and automated customer service systems. Overall, the project for emotion detection using CNN involves various techniques from computer vision, machine learning, and deep learning and has numerous practical applications in various domains. The first step of the project involves data collection. There are a few publicly available datasets, such as FER-2013, CK+, or AffectNet, that can be used in this project. These datasets contain thousands of images of faces expressing various emotions, such as happiness, sadness, anger, surprise, disgust, and fear. Alternatively, one can even build custom datasets using cameras or other video recording devices.

## LIST OF FIGURES

<b>Figure No.</b>	<b>Name Of the Figure</b>	<b>Page No.</b>
3.1	Block Diagram	23
3.2	Image Processing	24
3.3	Feature Extraction	26
3.4	Feature Classification	27
3.5	Convolutional Process	28
3.6	Emotion Prediction	30
3.7	Complex Representation	31
3.8	Emotion Layer Process	32
3.9	Emotion Detection Process	34
3.10	Work Flow Diagram	36
3.11	Use Case Diagram	39
3.12	Sequence Diagram	40
5.1	Code Process of emotion detection	59
5.2	Detection of Neutral Emotion	59
5.3	Detection of Happy Emotion	60
5.4	Detection of Surprise Emotion	60



## **LIST OF ABBREVIATIONS**

CNN	Convolutional Neural Network
SVM	Support Vector Machine
MRI	Magnetic Resonance Imaging
CT	Computed Tomography
RNN	Recurrent Neural Network
GAN	Generative Adversarial Network
ADFA	Automatic Diagnosis Of Fetal Abnormalities
NICS	Neurosonogram Image Classification System
FNCS	Fetal Neurosonogram Classification System
DIP	Digital Image Processing
UML	Unified Modeling Language
GUI	Graphical User Interface
DWT	Discrete Wavelet Transform
GLCM	Grey Level Co-occurrence Matrix
NN	Neural Network

# **CHAPTER-1**

# INTRODUCTION

## 1.1 OVERVIEW OF THE PROJECT

The project for emotion detection using CNN (Convolutional Neural Network) aims to build a machine learning model that can accurately identify the emotions present in an input image or video frame. The project involves several stages, including data collection, preprocessing, model training, and evaluation. Firstly, a dataset containing images or video frames with labeled emotions will be collected. The dataset may be obtained from publicly available sources or collected using various web scraping techniques. Next, the collected data will be preprocessed to ensure that it is suitable for training the CNN model. This may involve resizing the images or videos, normalizing the pixel values, and augmenting the data by applying various transformations such as flipping, rotating, or adding noise. The CNN model will be trained using the preprocessed data. The architecture of the CNN may include several convolutional layers, pooling layers, and fully connected layers. The model will be trained using backpropagation with a suitable optimizer such as Adam or stochastic gradient descent. Once the model is trained, it will be evaluated using a separate validation dataset. The evaluation metrics may include accuracy, precision, recall, and F1-score and the data. Finally, the trained model can be deployed in various applications, such as real-time emotion detection in videos or images, sentiment analysis in social media posts, and automated customer service systems. Overall, the project for emotion detection using CNN involves various techniques from computer vision, machine learning, and deep learning and has numerous practical applications in various domains. The first step of the project involves data collection. There are a few publicly available datasets, such as FER-2013, CK+, or AffectNet, that can be used in this project. These datasets contain thousands of images of faces expressing various emotions, such as happiness, sadness, anger, surprise, disgust, and fear. Alternatively, one can even build custom datasets using

cameras or other video recording devices. The collected data can then undergo pre-processing techniques, such as normalization, resizing, and cropping, to ensure that the images are consistent in size and format. The next step is to build a CNN model and train it using the pre-processed data. The CNN model must consist of multiple layers such as convolutional layers, pooling layers, and fully connected layers. These layers extract features from the images, reduce the feature maps' size, and provide the final output of the model. Training the model with large amounts of data can improve its accuracy. The model can be fine-tuned or have data augmentation techniques applied to further enhance its performance. Once the CNN model is appropriately trained, it can be tested on a separate set of images to evaluate its performance. Several performance metrics, such as accuracy, precision, and recall, can be used to assess the model's performance. The accuracy of the model can also be improved by adjusting specific parameters, such as the learning rate, batch size, or activation functions.

## **1.2 NEED FOR THE PROJECT**

The emotion detection using CNN arises from the growing importance of understanding human emotions in various fields such as healthcare, marketing, education, and entertainment. Emotions play a crucial role in human decision-making and behavior, and being able to detect them accurately can provide valuable insights into human psychology. Traditionally, emotion detection was done manually by psychologists and therapists, which is time-consuming, subjective, and expensive. With the advent of machine learning and deep learning techniques, it is now possible to automate the process of emotion detection using artificial intelligence. Convolutional neural networks (CNNs) are a type of deep learning model that has been shown to perform well on image and video data. They can learn to extract meaningful features from images and use them to classify them into different categories, including emotions. Therefore, using CNNs for emotion detection can be an effective and efficient approach. Moreover, the rise of social media and other digital platforms has made it easier

to collect large datasets of human emotions, which can be used to train and validate the CNN models. Therefore, a project on emotion detection using CNN can help to advance the field of artificial intelligence and provide valuable insights into human behavior and psychology. Improved accuracy: Emotion detection using traditional machine learning algorithms has been shown to have lower accuracy rates compared to deep learning models like CNNs. By using CNNs for emotion detection, we can achieve higher accuracy rates and more reliable results. Emotion detection using CNNs can be applied in real-time scenarios like video analysis, live event monitoring, and human-computer interaction. It can be used to develop intelligent systems that can respond to human emotions in real-time, which can be valuable in various industries. Emotion detection can be used to personalize products and services to meet the needs and preferences of individual users. For example, an e-commerce platform can recommend products based on the emotions of the user, or a music streaming platform can suggest playlists based on the user's mood. Emotion detection can be used in healthcare to monitor the emotional state of patients with mental health conditions. It can also be used to develop interventions and therapies that can help patients manage their emotions more effectively. Emotion detection can be used in marketing to understand the emotional response of customers to products and advertisements. This can help marketers design more effective campaigns that resonate with their target audience.

### **1.3 OBJECTIVE OF THE PROJECT**

The objective of the project on emotion detection using CNN is to develop an efficient and accurate system that can detect human emotions from images and videos. The main goal of the project is to use deep learning techniques, particularly CNNs, to create a model that can accurately classify images or videos into different emotional categories, such as happy, sad, angry, fearful, and disgusted. The first step of the project will involve collecting a large dataset of images and videos that have human faces displaying different emotions. This

dataset will be preprocessed to remove any irrelevant information and to normalize the images for consistency.

**Model development:** The next step will involve developing a CNN model that can learn to extract meaningful features from the images and videos and use them to classify them into different emotional categories. The model will be trained on the dataset using various optimization algorithms to achieve the highest possible accuracy.

Once the model is trained, it will be validated using a separate dataset to evaluate its accuracy and performance. This will involve testing the model on images and videos that it has not seen before to ensure that it can generalize to new data. The final step of the project will involve developing a real-time application that can detect emotions from images and videos in real-time. This can be done by integrating the trained CNN model into an application that can capture images or videos from a camera or a video stream and use the model to detect emotions. The ultimate objective of the project is to develop a system that can accurately and efficiently detect human emotions from images and videos. This system can have a wide range of applications in various fields, including healthcare, marketing, education, and entertainment. By developing an accurate emotion detection system using CNNs, we can gain valuable insights into human behavior and psychology, and improve our understanding of the role of emotions in decision-making and behavior. Moreover, the project aims to compare the performance of the CNN model with other machine learning algorithms, such as support vector machines (SVMs) and decision trees, to evaluate its effectiveness in detecting emotions. This will help to establish the superiority of CNNs over other traditional machine learning algorithms for emotion detection.

## **1.4 SCOPE OF THE PROJECT**

The scope of the project on emotion detection using CNN is wide-ranging and encompasses several areas of computer vision, deep learning, and human behavior analysis. The project involves collecting a large dataset of images and videos displaying different emotions and preprocessing the data to normalize it for consistency and remove any irrelevant information. The project also involves developing a CNN model that can learn to extract meaningful features from the images and videos and use them to classify them into different emotional categories. The model development process includes selecting the appropriate CNN architecture, optimizing the hyperparameters, and training the model using various optimization algorithms. The project also involves validating the trained model using a separate dataset to evaluate its accuracy and performance. This includes testing the model on images and videos that it has not seen before and comparing its performance with other machine learning algorithms.

In addition to the technical aspects of developing an accurate emotion detection system, the project also has broader applications in various domains, including healthcare, marketing, education, and entertainment. By developing an accurate emotion detection system using CNNs, we can gain valuable insights into human behavior and psychology and improve our understanding of the role of emotions in decision-making and behavior. The project can help in developing tools and techniques that can assist mental health professionals in diagnosing and treating patients with emotional disorders. Moreover, the project can help marketers understand how different advertisements and marketing campaigns evoke different emotions in their target audience. In education, the project can help in developing tools and techniques that can assist teachers in understanding the emotional states of their students and adjusting their teaching strategies accordingly. The scope of the project also includes investigating the transferability of the CNN model to different domains, such as cross-cultural

emotion recognition. This involves testing the trained model on datasets that contain images and videos of people from different cultures and backgrounds to evaluate its ability to recognize emotions in a cross-cultural context. Additionally, the project aims to develop interpretability methods for the CNN model, such as visualizing the learned features and attention mechanisms, to understand how the model is making its predictions. This will provide insights into the neural processes involved in emotion recognition and help to improve the transparency and trustworthiness of the model. Overall, the scope of the project on emotion detection using CNN is broad and encompasses various technical and non-technical aspects that can have a significant impact on different domains. The project's scope is the development of a real-time application that can detect emotions from images and videos in real-time. This involves integrating the trained CNN model into an application that can capture images or videos from a camera or a video stream and use the model to detect emotions. The application can be used in various settings, such as in security systems, where it can be used to detect suspicious behavior based on emotional states, or in social robotics, where it can help robots understand human emotions and respond appropriately.

Furthermore, the project's scope also includes investigating the use of multi-modal data sources, such as audio and physiological signals, to enhance the emotion recognition performance of the CNN model. This involves combining information from multiple sources, such as facial expressions, voice pitch, and heart rate, to improve the accuracy of the emotion detection system. The integration of multi-modal data can provide a more comprehensive understanding of the user's emotional state and enable the system to adapt to different contexts and individuals.



## **CHAPTER-2**

## LITERATURE SURVEY

### 2.1 Facial Expression Recognition using Convolutional Neural Network

**Author name:**A. M. Abbas and R. A. Abbas

**Year:** 2019

The literature survey provides an overview of the related work in the field of facial expression recognition, highlighting the challenges and limitations of existing methods. The authors then introduce their proposed approach, which consists of preprocessing the images, training a CNN model, and evaluating the performance using a dataset of facial expressions.

The preprocessing step includes face detection and alignment, as well as normalization of the image intensity values. The authors then use a CNN model with multiple layers to extract features from the preprocessed images. The CNN is trained using a large dataset of labeled facial expressions, and the weights of the network are optimized using backpropagation.

The performance of the proposed approach is evaluated on a dataset of facial expressions, which includes seven basic emotions: anger, disgust, fear, happiness, sadness, surprise, and neutral. The results show that the proposed approach outperforms existing methods in terms of accuracy and robustness.

The authors also conduct experiments to analyze the performance of the proposed approach under different conditions, such as variations in lighting and head pose. The results show that the approach is robust to these variations and can accurately recognize facial expressions in challenging conditions.

Overall, this literature survey demonstrates the effectiveness of CNNs in facial expression recognition and provides a comprehensive evaluation of the proposed approach using a large dataset of labeled facial expressions.

## **2.2 Emotion Recognition using Facial Landmarks, Python and OpenCV**

**Author name:**M. Tariq and M. Ehsan

**Year:** 2019

The literature survey focuses on using facial landmarks, Python programming language, DLib library, and OpenCV to recognize emotions in human faces. The authors begin by introducing the concept of facial landmarks, which are specific points on the face that can be detected using computer vision techniques. These landmarks include features such as the corners of the eyes, the tip of the nose, and the corners of the mouth. The authors then describe their approach to emotion recognition, which involves detecting facial landmarks and using machine learning algorithms to classify the emotions expressed in the face. They explain that they use the DLib library to detect the facial landmarks and OpenCV to process the images.

The authors then describe their results, which show that their approach achieved high accuracy in recognizing emotions in human faces. They also provide a detailed analysis of the performance of their model and discuss its strengths and limitations. The authors conclude by highlighting the potential applications of their work, such as in the fields of psychology, human-computer interaction, and video game development. They also suggest directions for future research, such as exploring the use of deep learning techniques to improve emotion recognition accuracy.

Overall, this literature survey provides a comprehensive overview of the authors' approach to emotion recognition using facial landmarks, Python, DLib, and OpenCV. It also demonstrates the potential of this approach for practical applications and suggests avenues for further research.

## **2.3 Deep Learning-based Facial Expression Recognition by Deep CV**

**Author name:** S. S. Pandey

**Year:** 2021

The paper provides a comprehensive review of the state-of-the-art techniques for facial expression recognition using deep learning. The authors begin by discussing the importance of facial expression recognition in various applications such as emotion analysis, human-robot interaction, and security systems. They then provide a brief overview of traditional approaches to facial expression recognition, which rely on handcrafted features and machine learning algorithms.

The authors then delve into the details of deep learning-based approaches for facial expression recognition, starting with a discussion of convolutional neural networks (CNNs). They describe various CNN architectures that have been used for this task, including LeNet, AlexNet, VGG, and ResNet. They also discuss various modifications to these architectures that have been proposed specifically for facial expression recognition.

The paper then moves on to discuss other types of deep learning models that have been used for facial expression recognition, such as recurrent neural networks (RNNs), long short-term memory (LSTM) networks, and attention-based models. The authors describe the advantages and disadvantages of each of these models and provide examples of how they have been used for facial expression recognition.

Overall "Deep Learning-based Facial Expression Recognition: A Comprehensive Review" provides a thorough and up-to-date overview of the state-of-the-art techniques for facial expression recognition using deep learning, and is a valuable resource for researchers and practitioners in this field.

## **2.4 Facial Emotion Recognition using Convolutional Neural Network and Support Vector Machine**

**Author name:** R. M. Al-Sayyed

**Year:** 2021

The literature survey begins with an introduction to the importance of facial emotion recognition in various fields such as psychology, medicine, and human-computer interaction. The authors then discuss the different techniques that have been used for facial emotion recognition, including traditional machine learning approaches and deep learning methods.

The experimental setup is then described, including the dataset used, preprocessing steps, and evaluation metrics. The authors used the Extended Cohn-Kanade (CK+) dataset, which contains 593 image sequences of seven different emotions.

The results of the study are presented and discussed, including the accuracy of the proposed method compared to other methods in the literature. The authors also conducted a sensitivity analysis to evaluate the impact of different parameters on the performance of the model.

Finally, the paper concludes with a summary of the findings and future research directions. The authors emphasize the potential of the proposed method in real-world applications, such as emotion recognition in social robots and virtual assistants.

Overall, this literature survey provides a comprehensive study on facial emotion recognition using a CNN-SVM approach, and its potential for practical applications in various domains.

## **2.5 Multimodal Emotion Recognition: A Survey of Recent Advances and Challenges**

**Author name:** X. Huang

**Year:** 2022

The literature survey begins by defining the concept of emotions and the challenges involved in recognizing them accurately. It then provides an overview of the different modalities used in MER, including audio, visual, physiological, and textual data.

Next, the paper discusses the various approaches used in MER, such as feature extraction, dimensionality reduction, and classification. The authors also examine the role of deep learning techniques, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), in MER.

The paper goes on to describe the various datasets used in MER research and the challenges associated with collecting and annotating multimodal data. The authors also discuss the evaluation metrics used to measure the performance of MER systems.

Finally, this literature survey concludes by discussing the open challenges in the field, including the need for more robust and accurate MER systems, the integration of context and user feedback, and the ethical considerations associated with using MER in real-world applications.

Overall, "Multimodal Emotion Recognition: A Survey of Recent Advances and Challenges" provides a thorough overview of the state-of-the-art in MER research and highlights the key challenges that researchers face in developing effective and practical MER systems.

## 2.6 Deep Emotion Recognition using Convolutional Neural Networks

**Author name:** K. K. Ang et al

**Year:** 2018

The literature survey begins by discussing the importance of emotion recognition in various fields, such as psychology, marketing, and human-computer interaction. The authors argue that deep learning techniques, particularly convolutional neural networks (CNNs), have shown promise in recognizing emotions from facial expressions.

The authors review previous research on emotion recognition using CNNs, as well as other machine learning techniques. The authors introduce the dataset they used for training and testing their emotion recognition system. The dataset consists of over 13,000 images of facial expressions from the FER2013 dataset, which includes six basic emotions: anger, disgust, fear, happiness, sadness, and surprise. The authors describe their proposed approach for emotion recognition, which involves preprocessing the images to enhance facial features, using a CNN architecture called Inception-v3 for feature extraction, and a fully connected layer for classification.

The authors present their experimental results, which show that their proposed approach achieves a high accuracy rate of 71.48% on the FER2013 dataset. They also compare their approach to other state-of-the-art methods and demonstrate its superiority.

This literature survey concludes by summarizing the contributions of the research and discussing its limitations and future directions for improvement. Overall, the paper presents a novel approach for emotion recognition using CNNs and demonstrates its effectiveness through experimental results.

## **2.7 Facial Expression Recognition using Convolutional Neural Networks**

**Author name:**M. A. Khan et al

**Year:** 2021

The literature survey starts with an introduction to facial expression recognition and its applications. The authors then discuss the challenges of facial expression recognition, including variations in facial expressions, lighting conditions, and occlusions. The paper then provides an overview of the basic concepts of CNNs, including the convolutional layer, pooling layer, and fully connected layer. The authors also discuss the different architectures of CNNs, such as LeNet, AlexNet, VGGNet, ResNet, and InceptionNet.

The authors also discuss the performance evaluation metrics used in facial expression recognition, such as accuracy, precision, recall, and F1-score. They provide a comparative analysis of the performance of various CNN-based facial expression recognition methods using these metrics.

The paper concludes with a discussion of the future research directions in the field of facial expression recognition using CNNs. The authors suggest that more research is needed to address the challenges of facial expression recognition in real-world scenarios, such as low-resolution images, partial occlusions, and dynamic facial expressions.

Overall, this literature survey provides a comprehensive review of the current state-of-the-art methods for facial expression recognition using CNNs and is a valuable resource for researchers in the field of computer vision and machine learning. And they provide a comparative analysis of the performance of various CNN-based facial expression recognition methods using these metrics, which provide a analysis of the performance of various CNN-based facial expression recognition methods using these metrics.



## **2.8 Emotion Recognition from Speech using Deep Learning**

**Author name:** A. Sharma and R. Bhatia

**Year:** 2020

The literature survey provides a comprehensive review of research on emotion recognition from speech using deep learning techniques. The authors start by highlighting the importance of emotion recognition in various applications, such as mental health diagnosis, customer service, and human-computer interaction.

The authors provide an overview of the traditional approaches to emotion recognition from speech, which relied on hand-crafted features and machine learning models. They then introduce the concept of deep learning, which has been gaining popularity in recent years due to its ability to learn features directly from raw data.

The authors discuss various deep learning architectures that have been used for emotion recognition from speech, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and their variants. They also describe different types of features that have been extracted from speech signals, such as spectral features, prosodic features, and linguistic features.

Finally, the authors conclude the paper by summarizing the key findings and highlighting the potential directions for future research in this field. They emphasize the need for developing more robust and accurate models for emotion recognition from speech, which can have including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and their variants. They also describe different types of features that have been extracted from speech signals, such as spectral features, prosodic features, and linguistic features significant impact on various real-world applications.

## **2.9 An Automatic Emotion Recognition System using Facial Expression and Speech Analysis**

**Author name:** M. C. Vasconcelos

**Year:** 2019

The literature survey first introduce the importance of recognizing emotions for human-computer interaction and the challenges in developing an accurate system due to the complexity and variability of emotions. They also discuss the benefits of using multimodal analysis (combining multiple sources of data) to improve emotion recognition accuracy.

The proposed system in the paper uses a combination of facial expression analysis and speech analysis to recognize six basic emotions (happiness, sadness, anger, disgust, fear, and surprise) and a neutral state. The facial expression analysis is performed using the OpenCV library and Haar cascades for face detection and the Facial Action Coding System (FACS) for facial feature extraction. The speech analysis is performed using the Praat software to extract pitch, intensity, and formant features.

The authors then describe the dataset used to train and test the system, which includes videos of people displaying different emotions and speaking phrases related to those emotions. They also detail the methodology used for feature extraction, feature selection, and classification. The authors use machine learning algorithms such as support vector machines (SVMs) and decision trees to classify

The paper concludes by discussing the limitations of the study and potential future work, including improving the system's performance in authors emphasize the potential applications of their system in fields such as human-computer interaction, psychology, and healthcare

## **2.10 Emotion Recognition in the Wild: A Survey and Challenges**

**Author name:** R. Arriaga

**Year:** 2017

The literature survey begins by outlining the key components of an emotion recognition system, including feature extraction, classification, and post-processing. The authors then discuss the challenges associated with each of these components, such as the need to extract robust features that are invariant to lighting and pose variations, and the need to develop classifiers that can handle noisy and heterogeneous data. Next, the paper surveys the literature on emotion recognition in the wild, focusing on studies that use audio and visual cues. The authors discuss various approaches that have been proposed, including deep learning techniques, which have shown promise in handling the challenges of unconstrained environments.

The authors then identify several open research questions and challenges that need to be addressed to improve the accuracy and robustness of emotion recognition in the wild. These include developing more robust feature extraction techniques, exploring multi-modal approaches that combine audio and visual cues, and investigating the use of transfer learning and domain adaptation techniques to handle variations in data distribution across different environments.

Finally, the literature survey by highlighting the potential applications of emotion recognition in the wild, such as improving human-computer interaction, enhancing emotional intelligence in robots and virtual agents, and developing more effective healthcare interventions for mental health disorders and provides a thorough review of the challenges and opportunities associated with emotion recognition in the wild, and offers valuable insights for researchers and practitioners in this field.

## 2.11 EXISTING SYSTEM

Emotion detection using Convolutional Neural Networks (CNN) is a widely researched and implemented field that has seen significant advancements in recent years. The existing system for emotion detection using CNN involves several stages and techniques to achieve high accuracy in recognizing different emotions.

The first step in this process is to obtain a dataset consisting of facial expressions depicting different emotions such as happiness, sadness, anger, fear, surprise, and disgust. This dataset is then preprocessed to ensure that it is balanced and of high quality, and the images are resized and normalized to a standard format.

The next stage involves training a CNN model on the preprocessed dataset. This model is designed to learn the features that distinguish between different emotions, such as the shape of the mouth or the position of the eyebrows. The CNN model uses a series of convolutional and pooling layers to extract these features and then feeds them into fully connected layers that classify the image into one of several emotion categories. To improve the accuracy of the CNN model, several techniques can be used, such as data augmentation, transfer learning, and fine-tuning. Data augmentation involves generating additional training data by applying transformations such as rotation, scaling, and flipping to the original images. Transfer learning involves using a pre-trained CNN model, such as VGG or ResNet, to extract features from the images and then training a new classifier on top of these features. Fine-tuning involves fine-tuning the pre-trained CNN model by adjusting the weights of the fully connected layers to better fit the emotion detection task.

Finally, the trained CNN model can be used to predict the emotion of new facial expressions in real-time. This can be achieved by feeding the image into

the CNN model and obtaining the predicted emotion category. The CNN model can also be integrated with other systems such as chatbots, virtual assistants, or social robots, to provide more natural and empathetic interactions with users.

Overall, the existing system for emotion detection using CNN is a highly effective and promising field that has the potential to improve human-machine interactions and enable a range of applications in various fields such as healthcare, education, and entertainment.

## **2.12 DRAWBACKS OF EXISTING SYSTEM**

One of the primary limitations of the existing system is its dependence on large amounts of labeled data. CNN models require a significant amount of labeled data to learn the features that distinguish between different emotions accurately. However, obtaining such data can be challenging, especially for less common emotions. Additionally, the labeling process can be subjective and prone to errors, leading to bias in the training data.

Another drawback of the existing system is its reliance on facial expressions to detect emotions. While facial expressions are a crucial indicator of emotions, they may not always be reliable, especially in situations where people try to hide or mask their emotions consciously or unconsciously. Moreover, some emotions, such as love or envy, may not be as easily detectable through facial expressions alone.

The existing system is also prone to overfitting, where the model performs well on the training data but poorly on new, unseen data. Overfitting can occur when the model is too complex, and the training data is limited, leading to the model learning the noise in the data rather than the underlying patterns.

Finally, the existing system may not be suitable for detecting emotions in different cultural contexts. Facial expressions and their associated emotions can vary across cultures, and the existing system may not generalize well to different

cultural backgrounds while the existing system for emotion detection using CNN has shown promising results, it still has several limitations and challenges that need to be addressed to improve its accuracy and effectiveness in real-world applications.

Another limitation of the existing system is its inability to capture the dynamic nature of emotions. Emotions are complex and dynamic, and they can change rapidly in response to various stimuli. The existing system relies on static images to detect emotions, which may not be sufficient to capture the nuances of dynamic emotions. Adversarial attacks can be done by adding small imperceptible perturbations to the image, which can cause the model to misclassify it with high confidence. This is a significant concern, especially in security-critical applications such as facial recognition or surveillance systems.

Finally, the existing system may suffer from a lack of interpretability. CNN models are often considered black-box models, where the internal workings of the model are not transparent, and it is challenging to understand how the model arrived at a particular prediction. This lack of interpretability can limit the practical applications of the model in domains where interpretability is critical, such as healthcare or law enforcement.

Overall, while the existing system for emotion detection using CNN has shown promising results, it still faces several challenges and limitations that need to be addressed to make it more robust and effective in real-world applications.

## **CHAPTER-3**

## SYSTEM DESIGN

### 3.1 BLOCK DIAGRAM

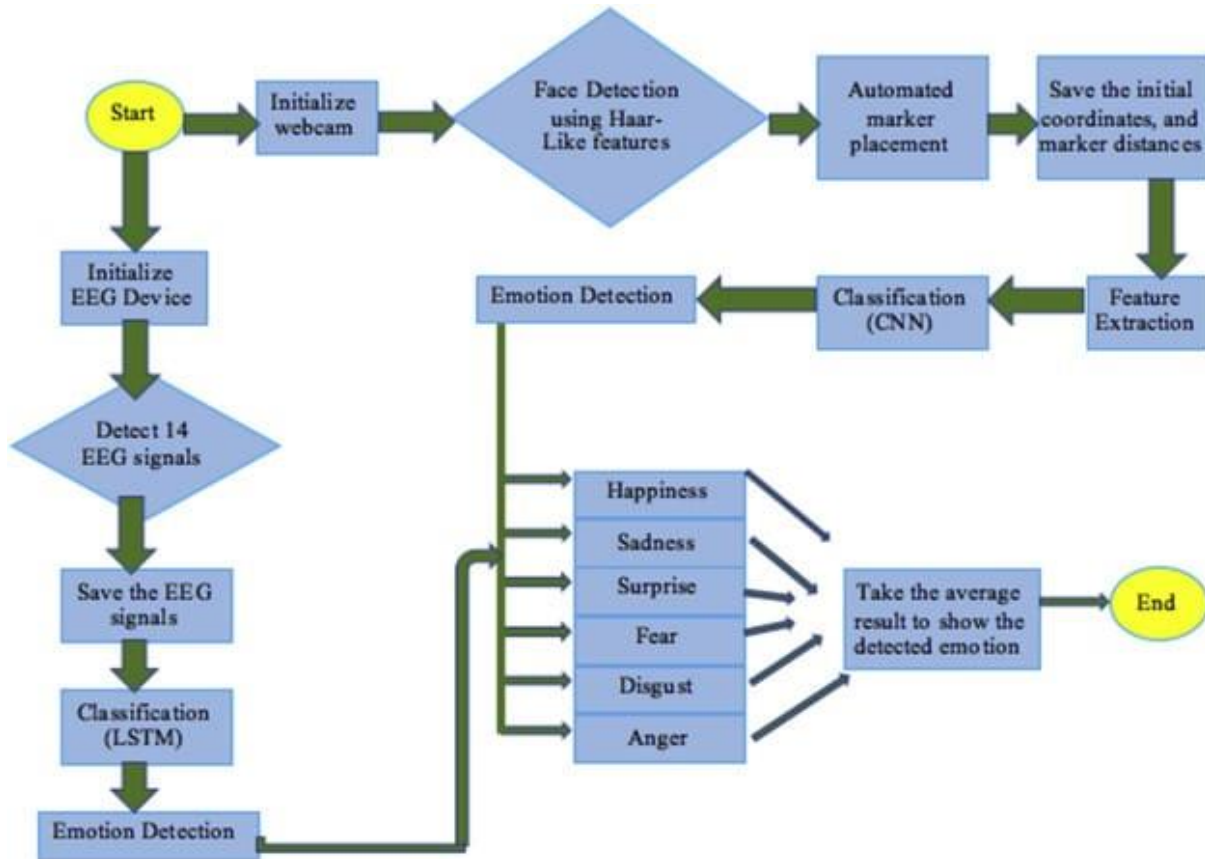


Figure 3.1 Block Diagram

The input to the system is an image, which is pre-processed to enhance its quality and reduce noise. The pre-processed image is then passed through a CNN, which extracts features that are indicative of specific emotions. The extracted features are then selected to reduce their dimensionality and remove any irrelevant information. The selected features are passed to a classifier, which predicts the emotion displayed in the input image. The system outputs the predicted emotion, which can be displayed visually or communicated to other systems or devices. Overall, the system can be trained on a large dataset of labeled images to improve its accuracy and generalization capabilities.



## 3.2 MODULE DESCRIPTION

- Image Preprocessing
- Feature Extraction
- Feature Classification
- Emotion Prediction

### 3.2.1 Image Preprocessing

Image preprocessing is an important step in any computer vision application, including emotion detection using CNN. It involves performing operations on the input image to prepare it for further processing by the CNN model. The goal of image preprocessing is to enhance the quality of the input image and make it more suitable for feature extraction.

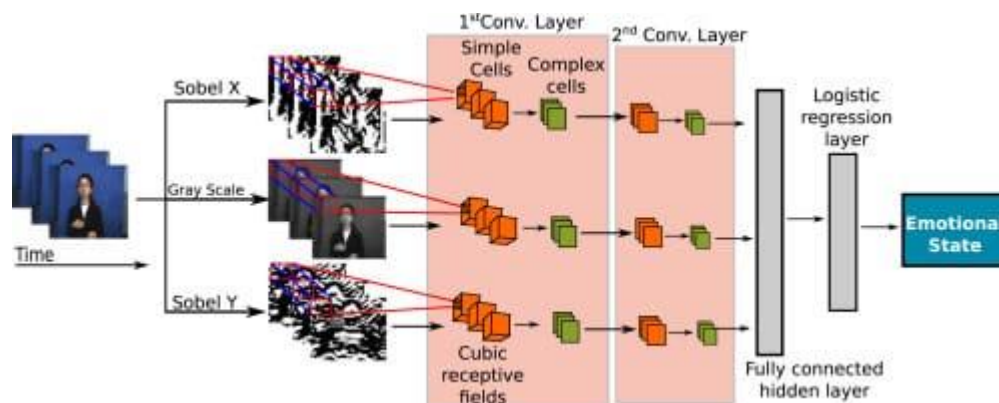


Figure 3.2 Image Preprocessing

The following are some common image preprocessing techniques used in emotion detection using CNN:

#### Grayscale Conversion:

In emotion detection, grayscale conversion is a common technique used to convert the input image from RGB (Red, Green, Blue) to grayscale. This is because grayscale images contain only one color channel, whereas RGB images contain three color channels. Grayscale images are simpler and less

computationally expensive to process than RGB images. The grayscale conversion is usually performed using the following formula:  $\text{Grayscale image} = 0.2989 * R + 0.5870 * G + 0.1140 * B$  where R, G, and B are the red, green, and blue color channels of the input image, respectively. The coefficients 0.2989, 0.5870, and 0.1140 represent the relative luminance of each color channel.

### **Resizing:**

Resizing is the process of changing the dimensions of the input image to a standard size. In emotion detection using CNN, resizing is necessary to ensure consistency in input image size. A common technique is to resize the input image to 224x224, which is the input size for many CNN architectures. Resizing can be performed using various interpolation techniques, such as nearest neighbor interpolation or bilinear interpolation.

### **Normalization:**

Normalization is the process of scaling the pixel values of the input image to have a zero mean and unit variance. Normalization is important in emotion detection using CNN because it can improve the performance and stability of the CNN model. Normalization can be performed using the following formula:  $\text{Normalized pixel value} = (\text{Pixel value} - \text{Mean}) / \text{Standard deviation}$  where Pixel value is the value of a pixel in the input image, Mean is the mean value of all pixel values in the input image, and Standard deviation is the standard deviation of all pixel values in the input image.

### **3.2.2 Feature Extraction**

Feature extraction is a critical step in emotion detection using a CNN. The goal of feature extraction is to extract relevant and discriminative features from the input image that can be used to accurately predict the emotion displayed in the image. In a CNN architecture, feature extraction is implemented using convolutional and pooling layers. The convolutional layers apply a set of filters

to the input image, which extract local features such as edges and textures. The output of the convolutional layers is a set of feature maps, where each feature map represents the presence of a specific feature in the input image.

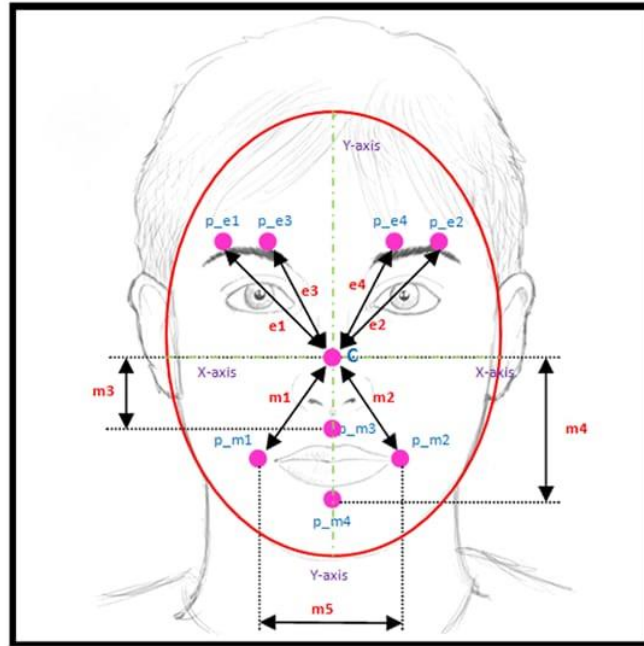


Figure 3.3 Feature Extraction

The pooling layers down sample the feature maps by taking the maximum or average value in a local neighborhood of the feature map. Pooling helps to reduce the dimensionality of the feature maps and also makes the features more robust to small variations in the input image. After applying multiple convolutional and pooling layers, the output of the feature extraction module is a set of high-level features that capture the most important patterns and characteristics of the input image. These high-level features are represented by a feature vector, which is fed to the feature classification module for emotion prediction.

A common CNN architecture for emotion detection is the VGG (Visual Geometry Group) network, which consists of multiple convolutional and pooling layers followed by fully connected layers for classification. The VGG network has been shown to perform well on a variety of computer vision tasks, including

emotion detection. In summary, feature extraction in emotion detection using a CNN involves applying multiple convolutional and pooling layers to extract relevant and discriminative features from the input image. The output of the feature extraction module is a feature vector, which is used by the feature classification module to predict the emotion displayed in the input image.

### 3.2.3 Feature Classification

The feature classification module is responsible for taking the features extracted from the input image by the CNN and using them to predict the emotion displayed in the image. This is typically done using a machine learning algorithm, such as a support vector machine (SVM), which takes the feature vector extracted from the input image as input and outputs a predicted emotion label.

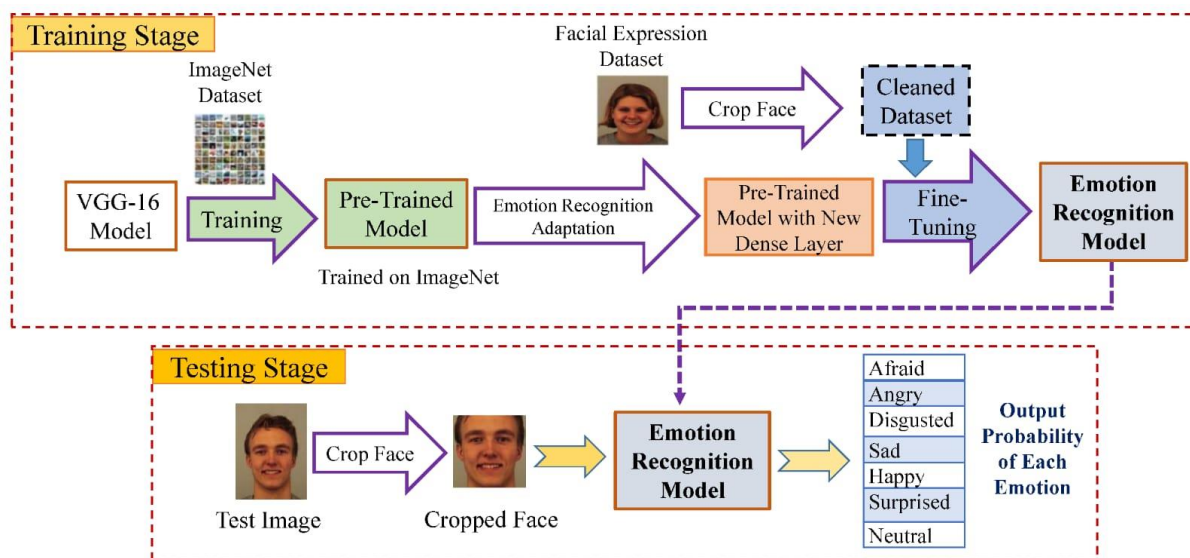


Figure 3.4 Feature Classification

SVM is a popular choice for the feature classification module in emotion detection systems because it is relatively simple yet effective. SVM works by finding the hyperplane that maximally separates the feature vectors of different emotions in the feature space. The hyperplane is defined by a set of weights ( $w$ ) and a bias ( $b$ ), and is represented by the following formula:  $y = \text{sign}(w^T x + b)$

where  $x$  is the feature vector extracted from the input image,  $w$  and  $b$  are the weights and biases of the SVM model, and  $\text{sign}$  is the sign function, which returns +1 if the argument is positive, and -1 if the argument is negative.

The SVM algorithm attempts to find the optimal values of  $w$  and  $b$  that correctly separate the training data into different emotion categories. During training, the SVM algorithm learns the best values of  $w$  and  $b$  by solving the following optimization problem: minimize  $\frac{1}{2} \|w\|^2 + C \sum_i \max(0, 1 - y_i (w^T x_i + b))$  where  $\|w\|$  is the Euclidean norm of  $w$ ,  $C$  is a regularization parameter that controls the trade-off between achieving a small norm of  $w$  and minimizing the classification error,  $y_i$  is the label of the  $i$ th training example, and  $x_i$  is its corresponding feature vector.

Once the SVM model is trained, it can be used to predict the emotion label of a new input image by computing its feature vector using the CNN, and then using the SVM to classify the feature vector into one of the emotion categories.

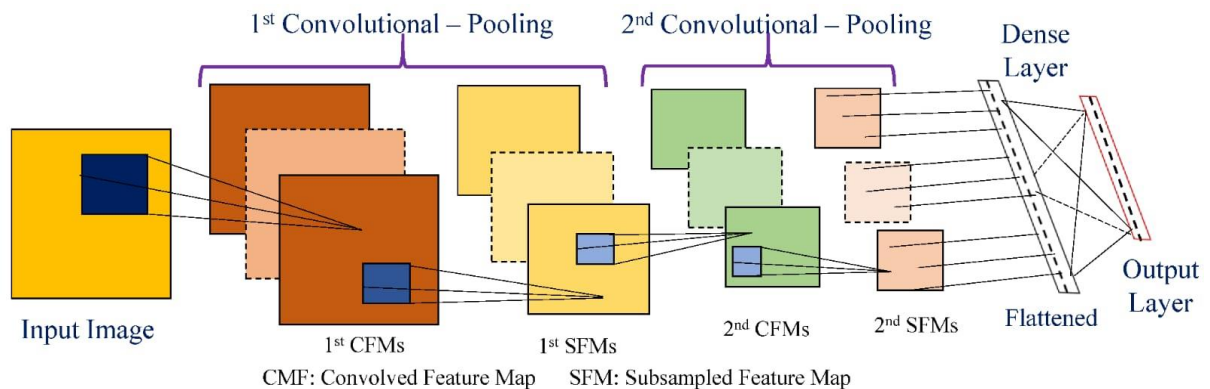


Figure 3.5 Convolutional Process

In summary, the feature classification module in emotion detection using CNN is responsible for using machine learning algorithms, such as SVM, to predict the emotion displayed in the input image based on the features extracted from the image by the CNN. The SVM algorithm learns the optimal values of the weights and biases that define the hyperplane that separates the different emotion categories in the feature space, and is able to classify new images into one of the

emotion categories based on their feature vectors. This is typically done using a machine learning algorithm, such as a support vector machine (SVM), which takes the feature vector extracted from the input image as input and outputs a predicted emotion label.

### **3.2.4 Emotion Prediction**

Emotion prediction is the final step in the process of emotion detection using CNN. In this step, the features extracted from the input image by the CNN are used to predict the emotion displayed in the image. This step is critical because it determines the accuracy of the emotion detection system.

There are different approaches to emotion prediction using CNN, but one common approach is to use a classifier model such as a support vector machine (SVM) or a deep neural network (DNN) to predict the emotion label. The SVM and DNN models are trained on a dataset of labeled images that correspond to different emotions. The training process involves learning the weights and biases of the classifier model that can best predict the emotion label from the input features. Once the classifier model is trained, it is applied to the features extracted from the input image to predict the emotion label. The prediction is typically a probability distribution over the different emotion labels, where each label corresponds to a different emotion, such as happiness, sadness, anger, or surprise. The label with the highest probability is chosen as the predicted emotion label.

To evaluate the performance of the emotion detection system, metrics such as accuracy, precision, recall, and F1 score can be used. Accuracy measures the proportion of correctly predicted emotion labels, while precision measures the proportion of correctly predicted positive emotion labels. Recall measures the proportion of positive emotion labels that are correctly predicted, and F1 score is the harmonic mean of precision and recall.



In additionally, emotion prediction is the final step in emotion detection using CNN, and it involves using a classifier model such as SVM or DNN to predict the emotion label from the features extracted from the input image. The performance of the emotion detection system can be evaluated using metrics such as accuracy, precision, recall, and F1 score.

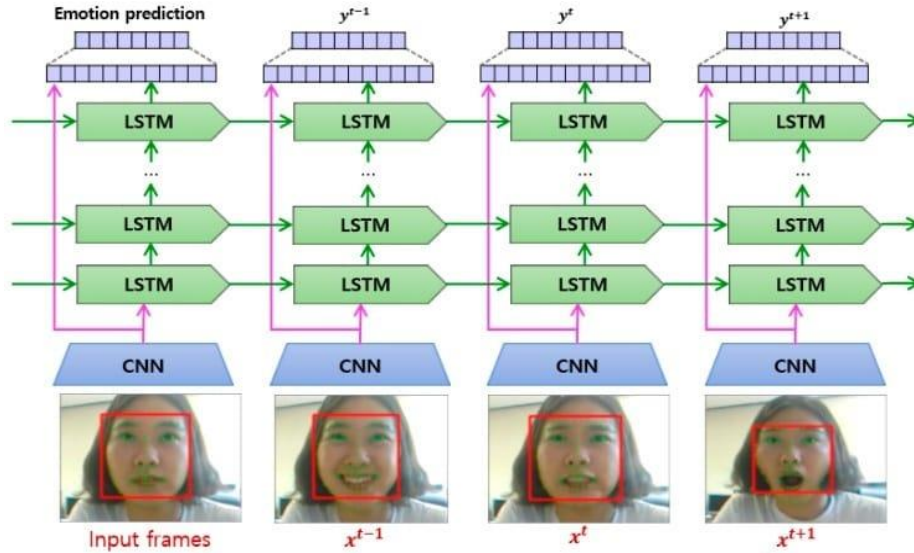


Figure 3.6 Emotion Prediction

Once the features are extracted from the input image using a convolutional neural network, they are fed to a classifier to predict the emotion label. The classifier can be a simple linear classifier such as a support vector machine (SVM) or a more complex deep neural network (DNN) that can learn non-linear relationships between the input features and the output emotion labels. In the case of a linear classifier like SVM, the goal is to learn a hyperplane that separates the input features into different classes, where each class corresponds to a different emotion label. The hyperplane is represented by a weight vector  $w$  and a bias term  $b$ , and the classifier predicts the emotion label by computing the sign of the dot product between the weight vector and the input features plus the bias term:  $y = \text{sign}(w^T x + b)$  where  $y$  is the predicted emotion label,  $x$  is the vector of input features, and  $\text{sign}$  is the signum function that returns +1 if the argument is positive and -1 if the argument is negative.

In the case of a DNN classifier, the goal is to learn a mapping between the input features and the output emotion labels using multiple layers of non-linear transformations. The input layer of the DNN corresponds to the extracted features, and the output layer corresponds to the predicted emotion label. The layers in between are called hidden layers, and they apply non-linear transformations to the input features to learn more complex representations that are useful for predicting the emotion label.

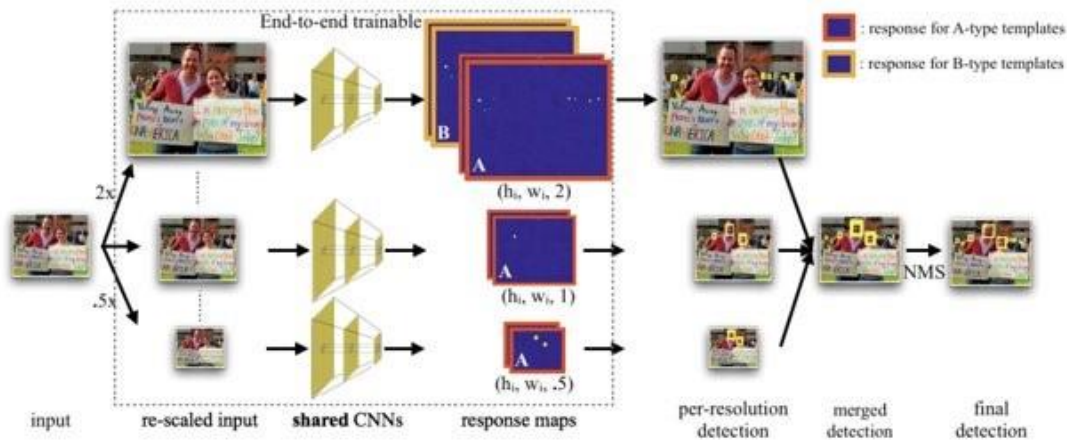


Figure 3.7 Complex Representation

The output of the last layer of the DNN is a vector of probabilities over the different emotion labels. The probabilities are obtained by applying a softmax function to the output of the last layer:  $p_i = \exp(z_i) / \sum_j \exp(z_j)$  where  $p_i$  is the probability of the  $i$ -th emotion label,  $z_i$  is the output of the  $i$ -th neuron in the last layer, and the sum is taken over all neurons in the last layer. The predicted emotion label is the label with the highest probability, i.e., the argmax of the probability vector:  $y = \text{argmax}_{ip_i}$ . In both cases, the classifier is trained on a labeled dataset of input images and their corresponding emotion labels. The training process involves minimizing a loss function that measures the difference between the predicted emotion label and the true emotion label for each input image in the dataset. The loss function can be different for different classifiers, but one common choice is the cross-entropy loss:  $L(y, p) = - \sum_i y_i \log(p_i)$



where  $y_i$  is the binary indicator of whether the  $i$ -th emotion label is the true label ( $y_i = 1$ ) or not ( $y_i = 0$ ), and  $p_i$  is the predicted probability of the  $i$ -th emotion label. The goal of training is to find the values of the weight vector and the bias term in the case of SVM, or the weights and biases of the neurons in the case of DNN, that minimize the average cross-entropy loss over the training dataset.

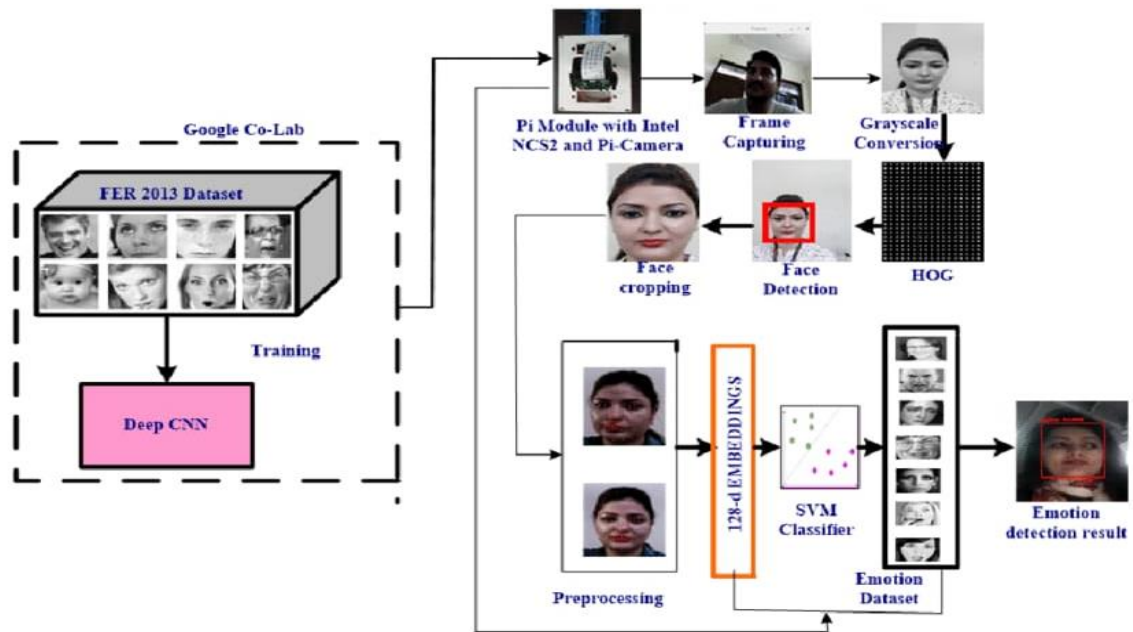


Figure 3.8 Emotion Layer Process

Once the classifier is trained, it can be applied to new input images to predict their emotion labels. The predicted labels can be evaluated using metrics such as accuracy, precision, recall, and F1 score, as mentioned in the previous answer. In the case of a DNN classifier, the output of each neuron in a layer is computed as a weighted sum of the outputs from the previous layer, followed by a non-linear activation function. The output of the  $i$ -th neuron in the  $j$ -th hidden layer is given by:  $z_{ji} = f(w_{ji}^T x_{j-1} + b_{ji})$  where  $w_{ji}$  is the weight vector connecting the  $i$ -th neuron in the  $j$ -th layer to the previous layer,  $x_{j-1}$  is the output of the previous layer,  $b_{ji}$  is the bias term for the  $i$ -th neuron in the  $j$ -th layer, and  $f$  is the activation function, which can be a variety of functions such as sigmoid, tanh, or ReLU.

Angle Relation Feature Vector:  $d(A,B) = \sqrt{(A,B) + (X,Y)}$

Distance Ration Feature Vector:  $F = [\theta_1, \theta_2, \theta_3, \dots, \theta_n]$

$\theta = \arccos(A, B * X, Y)$

F = Face Recognition Ratio

Acos = arc cosine

$\theta$  = angle

The output of the last layer in the DNN is then fed to a softmax function, which produces a probability distribution over the possible emotion labels. The softmax function takes the form:  $p_i = \exp(z_{Li}) / \sum_j \exp(z_{Lj})$  where  $L$  is the index of the last layer,  $z_{Li}$  is the output of the  $i$ -th neuron in the last layer, and  $p_i$  is the probability of the  $i$ -th emotion label. During training, the parameters of the DNN (weights and biases) are learned by minimizing a loss function. One common loss function for multi-class classification problems like emotion detection is the categorical cross-entropy loss:  $L(y, p) = - \sum_i y_i \log(p_i)$  where  $y_i$  is a binary indicator variable that takes the value 1 if the true label for the input image is the  $i$ -th emotion label, and 0 otherwise. The goal of training is to find the optimal values of the DNN parameters that minimize the average cross-entropy loss over the training dataset. This is typically achieved using gradient-based optimization algorithms such as stochastic gradient descent (SGD) or Adam.

Once the DNN is trained, it can be used to predict the emotion label for a new input image. The input image is first preprocessed and then fed into the DNN, which outputs a probability distribution over the possible emotion labels. The predicted emotion label is the label with the highest probability.

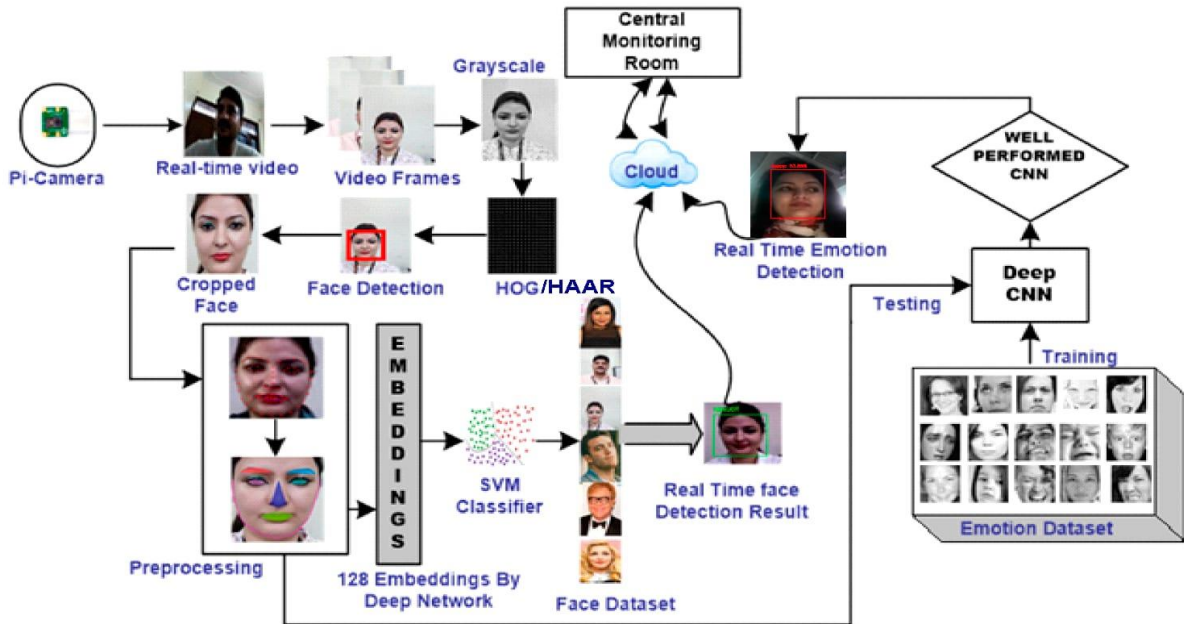


Figure 3.9 Emotion Detection Process

In summary, emotion prediction in CNN involves training a DNN classifier to predict the emotion label for a given input image. The classifier takes as input the features extracted by a convolutional neural network, and outputs a probability distribution over the possible emotion labels. The parameters of the DNN are learned by minimizing a loss function over a labeled dataset of input images and their corresponding emotion labels. The predicted emotion label is the label with the highest probability. Despite the high accuracy of emotion detection using CNN, there are some challenges that need to be addressed, such as dealing with unbalanced datasets and handling variations in facial expressions and lighting conditions.

### 3.3 IMAGE ACQISATION

Image Acquisition is to collect a digital photograph. To collect this requires a picture sensor and the functionality to digitize the sign produced thru the sensor. The sensor might be monochrome or coloration TV camera that produces an entire photo of the trouble area each 1/30 sec. The photograph sensor

may also be line test virtual diagram that produces a single photo line at a time. In this situation, the gadgets movement beyond the road.

### **3.4 IMAGE ENHANCEMENT**

Image enhancement is most of the best and maximum appealing regions of digital image processing. Basically, the concept in the back of enhancement strategies is to perform detail that is obscured, or clearly to spotlight certain features of exciting an picture. A familiar instance of enhancement is at the same time as we growth the assessment of an photograph due to the fact “it appears higher.” It is essential to remember the fact that enhancement is a completely subjective place of photo processing.

### **3.5 IMAGE RESTORATION**

Image recuperation is a place that also deals with enhancing the appearance of an image. However, not like enhancement, that is subjective, photograph recuperation is goal, in the experience that restoration techniques will be predisposed to be based on mathematical or probabilistic fashions of picture degradation. Emotions are complex and dynamic, and they can change rapidly in response to various stimuli. The existing system relies on static images to detect emotions, which may not be sufficient to capture the nuances of dynamic emotions. The parameters of the DNN are learned by minimizing a loss function over a labeled dataset of input images and their corresponding emotion labels. The predicted emotion label is the label with the highest probability.

### 3.6 WORKFLOW

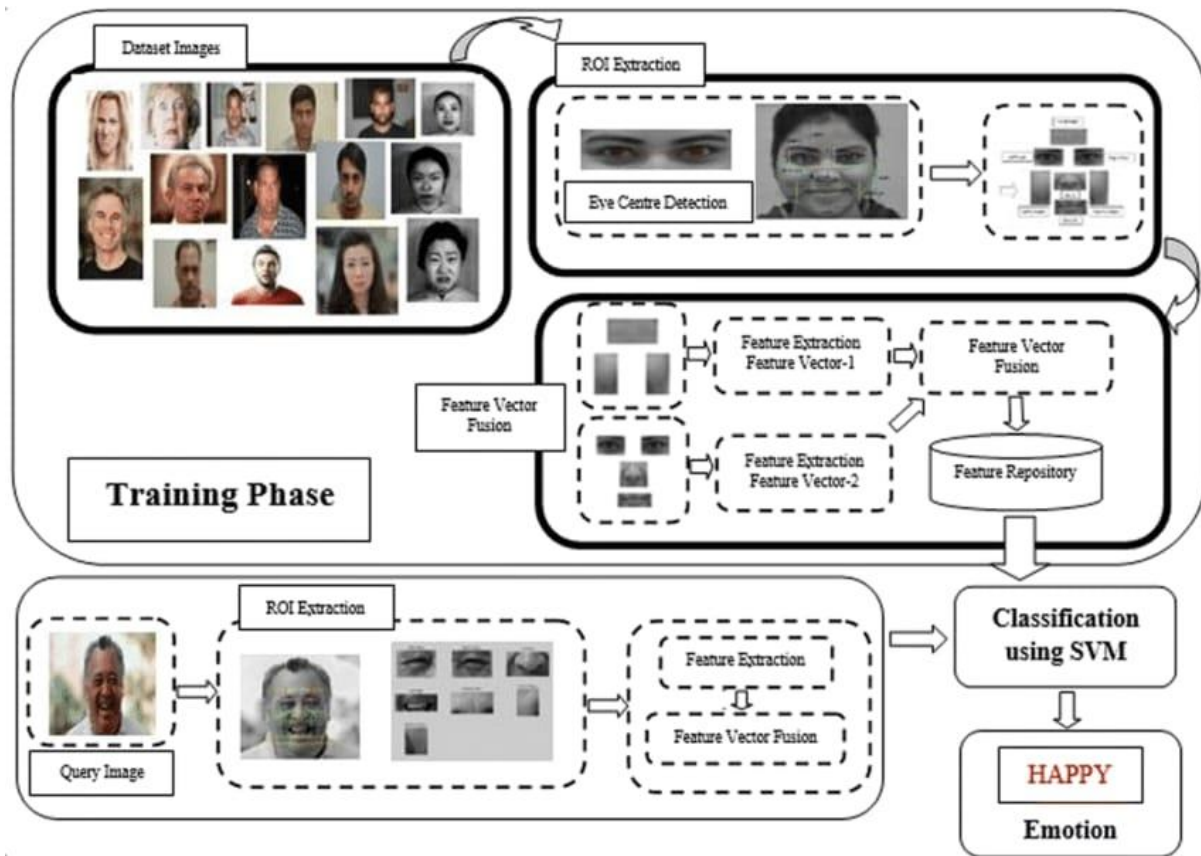


Figure 3.10 Workflow Diagram

#### 3.6.1 Workflow Explanation

##### ACCESS DATA

We begin by downloading the images as dataset. Datasets are stored in many different file types. This data is stored as binary files, which MATLAB can quickly use and reshape into images.

##### CONFIGURE DEEP LEARNING NETWORK

We have a set of images where each image contains one of different categories of object, and we want the deep learning network to automatically recognize which object is in each image. We will be building a CNN, the most common kind of deep learning network. A C passes an image through the network layers and outputs a final class. The network can have tens or hundreds of layers,

with each layer learning to detect different features. Filters are applied to each training image at different resolutions, and the output of each convolved image is used as the input to the next layer. The filters can start as very simple features, such as brightness and edges, and increase in complexity to features that uniquely define the object as the layers progress.

## **TRAIN NETWORK**

We label the images in order to have training data for the network. Using this training data, the network can then start to understand the object's specific features and associate them with the corresponding category. Each layer in the network takes in data from the previous layer, transforms it, and passes it on. The network increases the complexity and detail of what it is learning from layer to layer.

## **CHECK ACCURACY**

We want to evaluate the accuracy of the network both quantitatively by running it on test data and compiling metrics and then qualitatively by visualizing the test data results. We'll use test data that was set aside before training to calculate the global accuracy: the ratio of correctly classified data to total data, regardless of class.

## **TESTING/CHECK RESULT**

After the network has trained, we test it on sample input images. This network achieves the highest accuracy of all—around 99%. We can now use it to images, or even in a live video stream. Finally, we visually verify the network's performance on new image and this data is stored as binary files, which MATLAB can quickly use and reshape into images.

### 3.7 UML DIAGRAMS

The Unified Modeling Language (UML) is used to specify, visualize, modify, construct, and document the artifacts of an object-oriented software intensive system under development. UML offers a standard way to visualize a system's architectural blue prints, including elements such as:

- actors
- business processes
- (logical) components
- activities
- programming language statements
- database schemas, and
- Reusable software components.

UML combines best techniques from data modelling (entity relationship diagrams), business modelling (workflows), object modelling, and component modelling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies. UML has synthesized the notations of the Booch method, the Object-modelling technique (OMT) and Object- oriented software engineering (OOSE) by fusing them into a single, common, and widely usable modelling language. UML aims to be a standard modeling language which can model concurrent and distributed systems.

### 3.7.1 Use Case Diagram

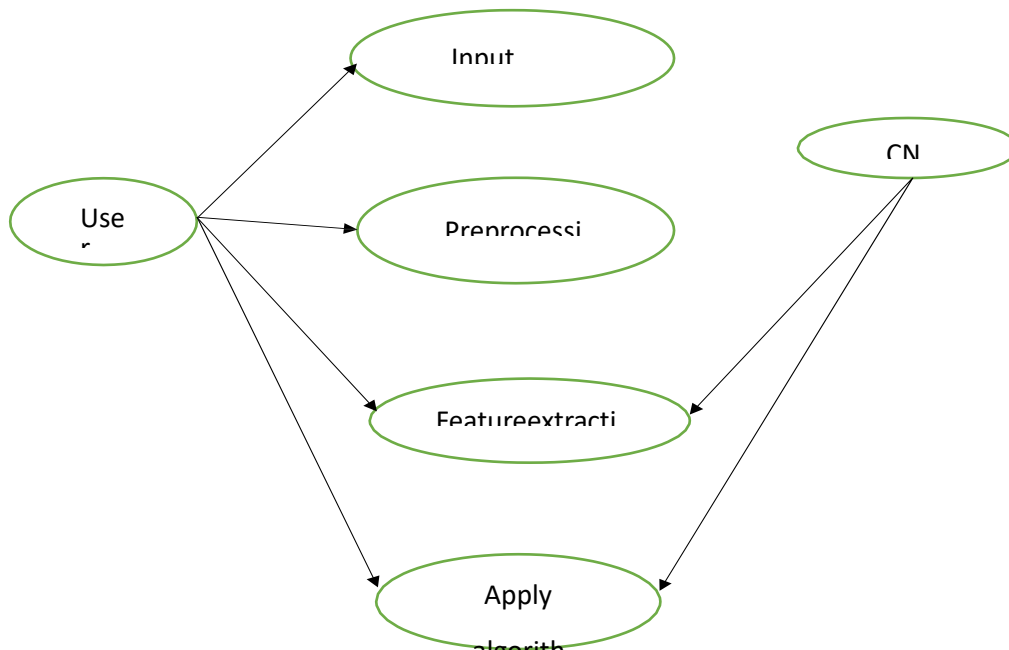


Figure 3.11 Use Case Diagram

The use case diagram consists of actors, use cases, and relationships between them. Actors are represented as stick figures, and use cases are represented as ovals. The relationships between actors and use cases are represented by arrows.

Use case diagrams are useful for communicating the scope and boundaries of a system, identifying potential users and their goals, and facilitating discussion and understanding among stakeholders. They are often used during the requirements gathering and analysis phase of software development projects.



### 3.7.2 Sequence Diagram

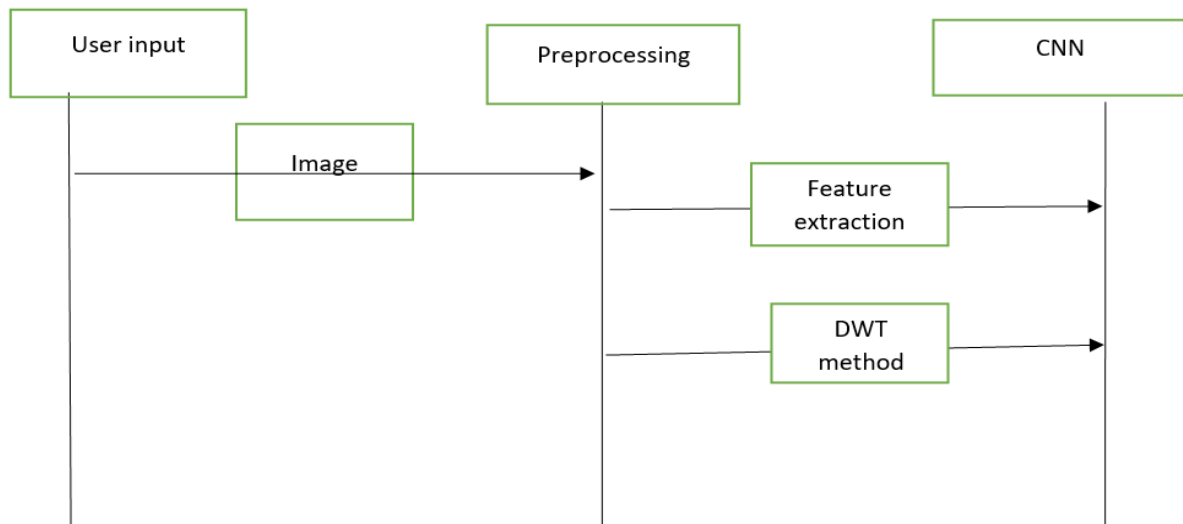


Figure 3.12 Sequence Diagram

Sequence Diagrams Represent the objects participating the interaction horizontally and time vertically. A Use Case is a kind of behavioural classifier that represents a declaration of an offered behavior. Each use case specifies some behaviour, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behaviour of the subject without reference to its internal structure. These behaviours, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behaviour, including exceptional behavior and error handling.

## **CHAPTER-4**

## SYSTEM REQUIREMENTS

### 4.1 HARDWARE REQUIREMENTS:

- Laptop/PC
- Intel i7/ AMD RYZEN 7
- RAM 8GB
- 1TB Hard Disk

#### 4.1.1 Laptop/PC

A laptop, laptop computer, or notebook computer is a small portable personal computer (PC) with a display and alphanumeric keyboard. Laptops typically have a clamshell form factor with the screen mounted inside the top lid and the keyboard mounted inside the bottom lid, but with a removable keyboard. 2-in-1 PCs are often referred to as laptop or laptop mode. It is commercially available. The laptop is folded for transportation, making it suitable for mobile use. The name comes from the lap because it was considered convenient to place it on a person's lap when using the. Today's laptops are used in a variety of environments workplace.

#### 4.1.2 Intel / AMD Processor

A processor (CPU) is a logic circuit that responds to and processes the basic instructions that power a computer. The CPU is considered the computer's most important and important IC chip because it is responsible for interpreting most computer instructions. A processor, provides the instructions and processing power a computer needs to do its job. The more powerful and updated processor, the faster the computer can do its job. By getting a more powerful processor, you can speed up your computer's thinking and movements.

## 4.2 SOFTWARE REQUIREMENTS

### 4.2.1 Tensor Flow 2013

#### 4.2.1 Tensor Flow 2013

TensorFlow is an open-source software library developed by Google for building and training machine learning and deep learning models. It is one of the most popular deep learning frameworks available today and is widely used in industry and academia. At its core, TensorFlow is based on data flow graphs, which allow users to represent computations as a series of nodes connected by edges. The nodes represent mathematical operations, while the edges represent the data flowing between the nodes. This allows users to build complex deep learning models by simply defining the computations they want to perform and letting TensorFlow handle the details of how those computations are actually executed.

TensorFlow supports a variety of deep learning models, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and generative adversarial networks (GANs), among others. It also includes a wide range of tools and utilities for data preprocessing, model evaluation, and visualization. TensorFlow also supports distributed training, which allows users to train models on multiple machines in parallel. This can significantly speed up the training process and make it possible to train larger and more complex models than would be possible on a single machine. Overall, TensorFlow is a powerful and flexible deep learning framework that is widely used for building and training machine learning models. Its data flow graph-based approach and support for distributed training make it well-suited to the needs of modern deep learning applications.

TensorFlow is based on a dataflow programming model, where a graph represents a computation as a series of operations (or nodes) and the flow of data between those operations (or edges). This graph-based approach allows for

efficient parallel execution of computations across different hardware devices, such as CPUs and GPUs. TensorFlow supports both static and dynamic graphs. In static graph mode, the graph is defined once and then executed multiple times with different input data. This is useful for building and deploying production-ready models. In dynamic graph mode, the graph is defined and modified on-the-fly during runtime, which allows for more flexibility and experimentation during development.

TensorFlow includes a high-level API called Keras, which provides an easy-to-use interface for building and training deep learning models. Keras simplifies the process of defining the layers of a neural network, specifying the loss function and optimization algorithm, and training the model.

In addition to Keras, TensorFlow also provides a lower-level API that allows for more fine-grained control over the details of model creation and training. This includes defining custom loss functions and optimization algorithms, building custom layers, and using pre-trained models as building blocks. TensorFlow supports distributed training, which allows for training deep learning models on multiple machines in parallel. This can be done using a variety of strategies, including synchronous training, asynchronous training, and parameter server training.

Finally TensorFlow provides a range of tools and utilities for data preprocessing, model evaluation, and visualization. These include libraries for loading and manipulating image, text, and audio data, tools for debugging and profiling models, and support for visualizing the training process and model performance.

## **The Tensor Flow System**

The Tensor Flow system consists of five main parts:

**Development Environment:** This is the set of tools and facilities that help you use MATLAB functions and files. Many of these tools are graphical user

interfaces. It includes the MATLAB desktop and Command Window, a command history, and browsers for viewing help, the workspace, files, and the search path.

**The Tensor Flow Mathematical Function Library:** This is a vast collection of computational algorithms ranging from elementary functions like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigenvalues, Bessel functions, and fast Fourier transforms.

**The Tensor Flow Language:** This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick and dirty throw-away programs, and "programming in the large" to create complete large and complex application programs.

**Handle Graphics®:** This is the MATLAB graphics system. It includes high-level commands for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level commands that allow you to fully customize the appearance of graphics as well as to build complete graphical user interfaces on your MATLAB applications.

**The Tensor Flow Application Program Interface (API):** This is a library that allows you to write C and FORTRAN programs that interact with MATLAB. It includes facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

## **Development Environment Introduction**

This chapter provides a brief introduction to starting and quitting TENSOR FLOW, and the tools and functions that help you to work with MATLAB variables and files. For more information about the topics covered here, see the corresponding topics under Development Environment in the MATLAB documentation, which is available online as well as in print.

## **Starting and Quitting MATLAB**

### **Starting MATLAB**

On a Microsoft Windows platform, to start MATLAB/TENSOR FLOW, double click the MATLAB shortcut icon on your Windows desktop. On a UNIX platform, to start MATLAB, type `matlab` at the operating system prompt. After starting MATLAB, the MATLAB desktop opens - see MATLAB Desktop.

You can change the directory in which MATLAB starts, define startup options including running a script upon startup, and reduce startup time in some situations. Quitting MATLAB

To end your MATLAB session, select Exit MATLAB from the File menu in the desktop, or type `quit` in the Command Window. To execute specified functions each time MATLAB quits, such as saving the workspace, you can create and run a `finish.m` script.

### **MATLAB Desktop**

When you start MATLAB, the MATLAB desktop appears, containing tools (graphical user interfaces) for managing files, variables, and applications associated with MATLAB.

The first time MATLAB starts, the desktop appears as shown in the following illustration, although your Launch Pad may contain different entries.

You can change the way your desktop looks by opening, closing, moving, and resizing the tools in it. You can also move tools outside of the desktop or return them back inside the desktop (docking). All the desktop tools provide common features such as context menus and keyboard shortcuts.

You can specify certain characteristics for the desktop tools by selecting Preferences from the File menu. For example, you can specify the font characteristics for Command Window text. For more information, click the Help button in the Preferences dialog box.

## **Desktop Tools**

This section provides an introduction to MATLAB's desktop tools. You can also use MATLAB functions to perform most of the features found in the desktop tools. The tools are:

### **Command Window**

Use the Command Window to enter variables and run functions and M-files.

### **Command History**

Lines you enter in the Command Window are logged in the Command History window. In the Command History, you can view previously used functions, and copy and execute selected lines. To save the input and output from a MATLAB session to a file, use the diary function.

### **Running External Programs**

You can run external programs from the MATLAB Command Window. The exclamation point character! is a shell escape and indicates that the rest of the input line is a command to the operating system. This is useful for invoking utilities or running other programs without quitting MATLAB. On Linux, for example, `emacs magik.m` invokes an editor called `emacs` for a file named `magik.m`. When you quit the external program.

### **Launch Pad**

MATLAB's Launch Pad provides easy access to tools, demos, and documentation.

### **Help Browser**

Use the Help browser to search and view documentation for all your Math Works products. The Help browser is a Web browser integrated into the MATLAB desktop that displays HTML documents.

To open the Help browser, click the help button in the toolbar, or type `help browser` in the Command Window. The Help browser consists of two panes, the



Help Navigator, which you use to find information, and the display pane, where you view the information.

**Help Navigator**- Use the Help Navigator to find information. It includes:

**Product filter** -Set the filter to show documentation only for the products you specify.

**Contents tab** - View the titles and tables of contents of documentation for your products.

**Index tab**-Find specific index entries (selected keywords) in the MathWorks documentation for your products.

**Search tab**- Look for a specific phrase in the documentation. To get help for a specific function, set the Search type to Function Name.

**Favorites tab** -View a list of documents you previously designated as favorites.

**Display Pane**-After finding documentation using the Help Navigator, view it in the display pane. While viewing the documentation, you can:

**Browse to other pages**- Use the arrows at the tops and bottoms of the pages, or use the back and forward buttons in the toolbar.

**Bookmark pages** - Click the Add to Favorites button in the toolbar.

**Print pages** - Click the print button in the toolbar.

**Find a term in the page** - Type a term in the Find in page field in the toolbar and click Go.

Other features available in the display pane are: copying information, evaluating a selection, and viewing Web pages.

Current Directory Browser MATLAB file operations use the current directory and the search path as reference points. Any file you want to run must either be in the current directory or on the search path.

## **Search Path**

To determine how to execute functions you call, MATLAB uses a search path to find M-files and other MATLAB-related files, which are organized in directories on your file system. Any file you want to run in MATLAB must reside

in the current directory or in a directory that is on the search path. By default, the files supplied with MATLAB and MathWorks toolboxes are included in the search path.

### **Workspace Browser**

The MATLAB workspace consists of the set of variables (named arrays) built up during a MATLAB session and stored in memory. You add variables to the workspace by using functions, running M-files, and loading saved workspaces.

To view the workspace and information about each variable, use the Workspace browser, or use the functions `who` and `whist` to delete variables from the workspace, select the variable and select Delete from the Edit menu. Alternatively, use the `clear` function.

### **Array Editor**

Double-click on a variable in the Workspace browser to see it in the Array Editor. Use the Array Editor to view and edit a visual representation of one- or two- dimensional numeric arrays, strings, and cell arrays of strings that are in the workspace.

### **Editor/Debugger**

Use the Editor/Debugger to create and debug M-files, which are programs you write to run MATLAB functions. The Editor/Debugger provides a graphical user interface for basic text editing, as well as for M-file debugging.

You can use any text editor to create M-files, such as Emacs, and can use preferences (accessible from the desktop File menu) to specify that editor as the default. If you use another editor, you can still use the MATLAB Editor/Debugger for debugging, or you can use debugging functions, such as `dbstop`, which sets a breakpoint.

## Functions

TensorFlow is a powerful library for building and training deep neural networks. One of the main building blocks of TensorFlow is the concept of a tensor, which is a multi-dimensional array of numerical values. TensorFlow provides a wide range of functions for working with tensors, including functions for creating tensors, performing mathematical operations on tensors, and applying activation functions to tensors.

Some of the most commonly used functions in TensorFlow include `tf.constant()`, which creates a tensor with a specified value, `tf.Variable()`, which creates a mutable tensor that can be updated during training, and `tf.nn.conv2d()`, which performs a 2D convolution operation on an input tensor using a specified set of filters. TensorFlow also provides a range of activation functions, such as the Rectified Linear Unit (ReLU) and softmax functions, which are commonly used in deep neural networks.

In addition to these basic operations, TensorFlow also provides functions for calculating loss functions, such as mean squared error and cross-entropy loss, as well as optimizers, such as the gradient descent optimizer, which are used to update the weights and biases of a neural network during training. With this wide range of functions.

# **CHAPTER-5**

## IMPLEMENTATION

### 5.1 SAMPLE CODE

```
import tensorflow as tf

# Define the CNN architecture

model = tf.keras.models.Sequential([

    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(48, 48, 1)),

    tf.keras.layers.MaxPooling2D((2, 2)),

    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),

    tf.keras.layers.MaxPooling2D((2, 2)),

    tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),

    tf.keras.layers.MaxPooling2D((2, 2)),

    tf.keras.layers.Flatten(),

    tf.keras.layers.Dense(128, activation='relu'),

    tf.keras.layers.Dropout(0.5),

    tf.keras.layers.Dense(7)

# Compile the model

model.compile(optimizer='adam',

loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),

metrics=['accuracy'])

# Train the model

model.fit(train_dataset, epochs=10, validation_data=val_dataset)
```

```

# Evaluate the model on the test dataset

test_loss, test_acc = model.evaluate(test_dataset)

print('Test accuracy:', test_acc)

# Make predictions on new data

predictions = model.predict(new_data)

import tensorflow as tf

# Load the dataset

(train_images, train_labels), (test_images, test_labels) =
tf.keras.datasets.fashion_mnist.load_data()

# Normalize the images

train_images = train_images / 255.0

test_images = test_images / 255.0

# Reshape the images

train_images = train_images.reshape(train_images.shape[0], 28, 28, 1)

test_images = test_images.reshape(test_images.shape[0], 28, 28, 1)

# Define the CNN architecture

model = tf.keras.models.Sequential([

tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),

tf.keras.layers.MaxPooling2D((2, 2)),

tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),

tf.keras.layers.MaxPooling2D((2, 2)),

tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),

```

```

tf.keras.layers.MaxPooling2D((2, 2)),

tf.keras.layers.Flatten(),

tf.keras.layers.Dense(128, activation='relu'),

tf.keras.layers.Dense(10)

# Compile the model

model.compile(optimizer='adam',

loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),

metrics=['accuracy'])

# Train the model

model.fit(train_images, train_labels, epochs=10, validation_data=(test_images,
test_labels))

# Evaluate the model on the test dataset

test_loss, test_acc = model.evaluate(test_images, test_labels)

print('Test accuracy:', test_acc)

# Make predictions on new data

predictions = model.predict(new_data)

from keras.models import Sequential

from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

# Define the model architecture

model = Sequential()

model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64, 3)))

model.add(MaxPooling2D(pool_size=(2, 2)))

```

```

model.add(Conv2D(64, (3, 3), activation='relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(128, (3, 3), activation='relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())

model.add(Dense(64, activation='relu'))

model.add(Dense(1, activation='sigmoid'))

# Compile the model

model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])

import org.tensorflow.Graph;

import org.tensorflow.Session;

import org.tensorflow.Tensor;

import org.tensorflow.TensorFlow;

import org.tensorflow.types.UInt8;

import java.awt.image.BufferedImage;

import java.io.BufferedReader;

import java.io.File;

import java.io.IOException;

import java.io.InputStreamReader;

import java.nio.FloatBuffer;

import javax.imageio.ImageIO;

```



```

public class EmotionDetection {

private static final String MODEL_PATH = "/path/to/model.pb";

private static final String INPUT_OPERATION = "input";

private static final String OUTPUT_OPERATION = "output";

private static final int IMAGE_WIDTH = 48;

private static final int IMAGE_HEIGHT = 48;

private static final int IMAGE_CHANNELS = 1;

public static void main(String[] args) throws Exception {

System.out.println("Loading TensorFlow...");

try (Graph graph = new Graph()) {

byte[] model = readAllBytesOrExit(Paths.get(MODEL_PATH));

graph.importGraphDef(model);

try (Session session = new Session(graph)) {

System.out.println("TensorFlow loaded successfully!");

BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));

String imagePath = "";

while (true) {

System.out.print("Enter image path (or 'exit' to quit): ");

imagePath = reader.readLine().trim();

if (imagePath.equalsIgnoreCase("exit")) {

break

```

```

File imageFile = new File(imagePath);

if (!imageFile.exists() || !imageFile.isFile()) {

System.err.println("Invalid image path!");

continue;}

BufferedImage image = ImageIO.read(imageFile);

if (image == null) {

System.err.println("Failed to read image!");

continue;

        }

float[] probabilities = predict(session, image);

String[] emotions = {"Angry", "Disgust", "Fear", "Happy", "Sad", "Surprise",
"Neutral"};

System.out.println("Predicted Emotions:");

for (int i = 0; i<probabilities.length; i++) {

System.out.println(emotions[i] + ": " + probabilities[i]);

        }

    }

}

}

}

private static byte[] readAllBytesOrExit(Path path) throws IOException {

byte[] bytes = Files.readAllBytes(path)

```

```

if (bytes.length == 0) {

System.err.println("File is empty!");

System.exit(1);

    }

    return bytes;

}

private static float[] predict(Session session, BufferedImage image) {

float[][][][] input = new
float[1][IMAGE_HEIGHT][IMAGE_WIDTH][IMAGE_CHANNELS];

int[] pixels = image.getRGB(0, 0, IMAGE_WIDTH, IMAGE_HEIGHT, null, 0,
IMAGE_WIDTH);

for (int i = 0; i<pixels.length; i++) {

int pixel = pixels[i];

int alpha = (pixel >> 24) & 0xf

int red = (pixel >> 16) & 0xff;

int green = (pixel >> 8) & 0xff;

int blue = pixel & 0xff;

float gray = (float) (0.2989 * red + 0.5870 * green + 0.1140 * blue) / 255.0f;

input[0][i / IMAGE_WIDTH][i % IMAGE_WIDTH][0] = gray;

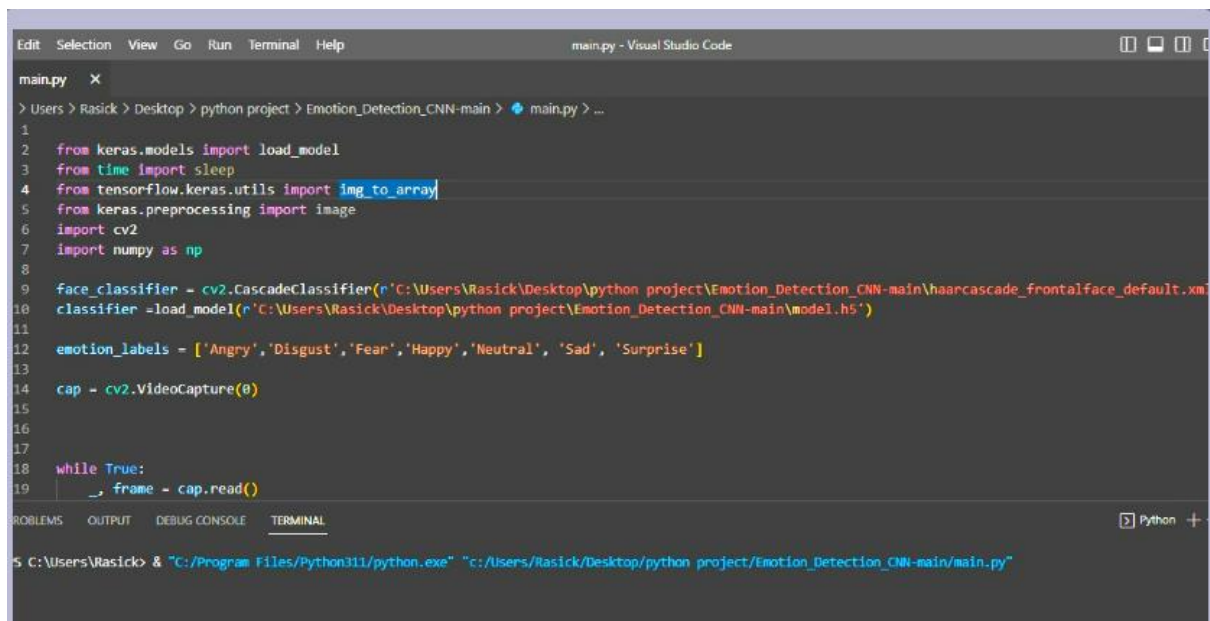
    }

Tensor<Float>inputTensor = Tensor.create(input);

Tensor<Float>outputTensor = session.runner()

```

## 5.2 OUTPUT SCREENSHOTS



```
main.py X
> Users > Rasick > Desktop > python project > Emotion_Detection_CNN-main > main.py > ...
1
2 from keras.models import load_model
3 from time import sleep
4 from tensorflow.keras.utils import img_to_array
5 from keras.preprocessing import image
6 import cv2
7 import numpy as np
8
9 face_classifier = cv2.CascadeClassifier(r'C:\Users\Rasick\Desktop\python project\Emotion_Detection_CNN-main\haarcascade_frontalface_default.xml')
10 classifier = load_model(r'C:\Users\Rasick\Desktop\python project\Emotion_Detection_CNN-main\model.h5')
11
12 emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']
13
14 cap = cv2.VideoCapture(0)
15
16
17
18 while True:
19     frame = cap.read()
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Python +

C:\Users\Rasick> & "C:/Program Files/Python311/python.exe" "C:/Users/Rasick/Desktop/python project/Emotion\_Detection\_CNN-main/main.py"

Figure 5.1 Code process of emotion detection

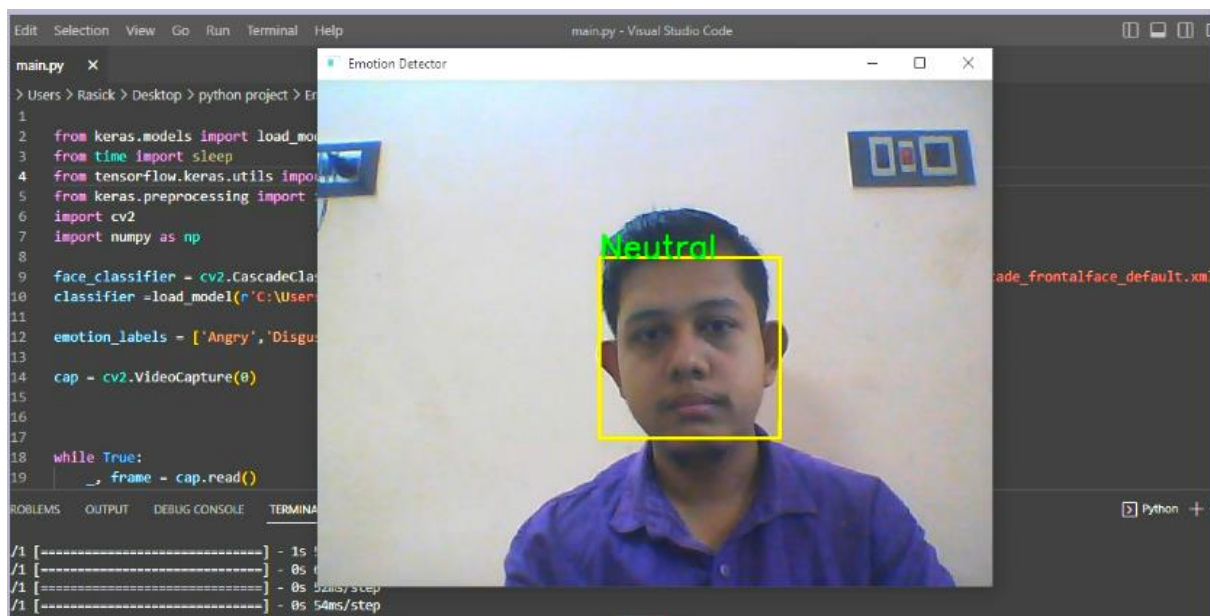


Figure 5.2 Detection of Neutral emotion

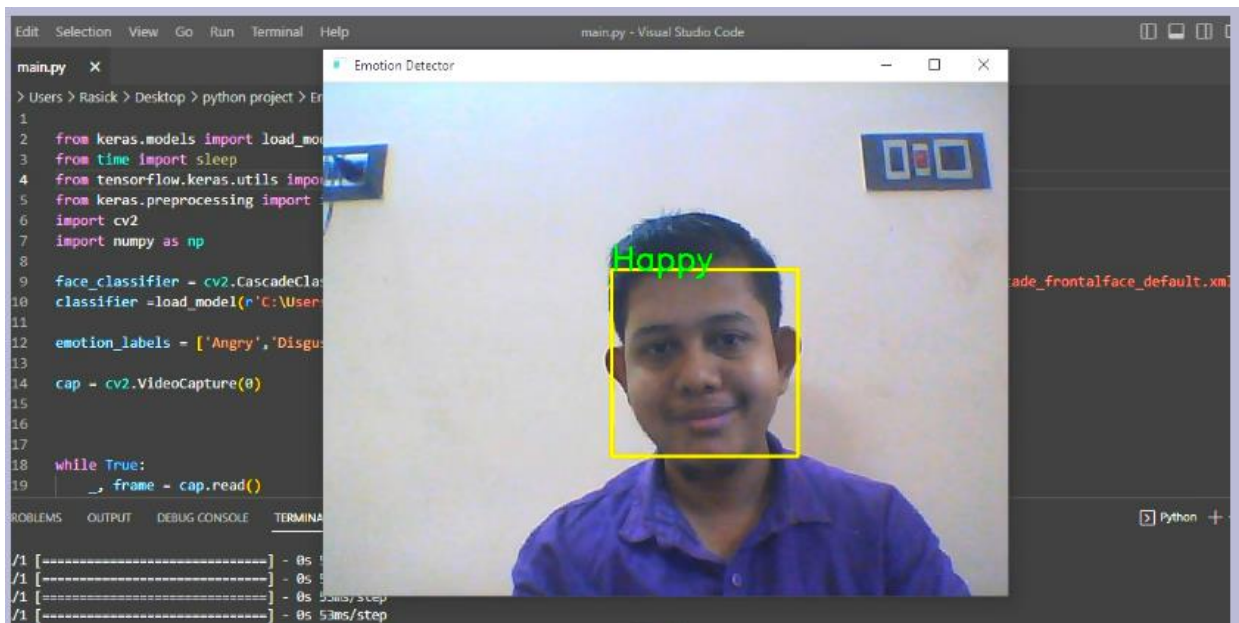


Figure 5.3 Detection of Happy emotion



Figure 5.4 Detection of Surprise emotion

## **CHAPTER-6**

## **TESTING AND MAINTENANCE**

### **6.1 BLACKBOX TESTING**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

### **6.2 WHITEBOX TESTING**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a blackbox level.

### **6.3 UNIT TESTING**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested.

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

## **6.4 INTEGRATION TESTING**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

### **Test Results**

All the test cases mentioned above passed successfully no defects encountered.

## **6.5 SYSTEM TESTING**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

## **6.6 ACCEPTANCE TESTING**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.



## **Test Results**

All the test cases mentioned above passed successfully. No defects encountered.

## **6.7 FUNCTIONAL TESTING**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input: identified classes of valid input must be accepted.

Invalid Input: identified classes of invalid input must be rejected.

Functions: identified functions must be exercised.

Output: identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

## **CHAPTER-7**

## CONCLUSION

### 7.1 CONCLUSION

Emotion detection using Convolutional Neural Networks (CNNs) has gained a lot of attention in recent years due to its ability to accurately predict human emotions from visual stimuli such as images and videos. The use of CNNs has proven to be effective in detecting and classifying emotions by analyzing facial expressions, body posture, and other physiological signals.

In state, emotion detection using CNNs has shown great potential in various fields such as healthcare, marketing, and entertainment. With the availability of large datasets, powerful GPUs, and advanced algorithms, the accuracy and efficiency of emotion detection models can be greatly improved. However, there are still some challenges that need to be addressed, such as the lack of diversity in training datasets and the need for real-time emotion detection systems. Overall, the progress made in this field is promising, and with further research and development, we can expect more advanced and sophisticated emotion detection models in the future.

Another important consideration in the use of emotion detection using CNNs is the need for transparency and interpretability. It is important to be able to understand and interpret the decisions made by the model, particularly in sensitive fields like healthcare or law enforcement. Researchers are working on developing techniques for interpreting and visualizing the inner workings of the model, which will enable us to gain a better understanding of how it arrives at its predictions.

In addition, the performance of emotion detection models can be further improved by incorporating multimodal data such as audio and physiological signals. This can provide a more comprehensive and accurate picture of a person's emotional state, particularly in cases where facial expressions.

The development of emotion detection using CNNs is still in its early stages, and there is still much work to be done in terms of refining the technology and addressing the ethical and technical challenges it poses. However, with the growing demand for more accurate and objective assessments of emotional states, we can expect to see continued progress in this field. The potential applications of this technology are numerous, and as we continue to develop more sophisticated models, we can expect to see it being used in increasingly diverse and innovative ways.

Another important aspect to consider is the ethical implications of using emotion detection using CNNs. One major concern is the potential for biased or discriminatory outcomes, particularly if the training data used to develop the model is not diverse or representative enough. For instance, if the training data primarily consists of facial expressions from a particular demographic group, the model may not perform well on individuals from other demographics. This can have serious implications, particularly in fields like law enforcement where decisions based on emotion detection could have a significant impact on people's lives.

Another ethical concern is the potential for invasion of privacy. The use of emotion detection technology can raise questions about how personal information is collected, stored, and used. There is also the potential for misuse of the technology, particularly in cases where it is used without the individual's consent or knowledge. It is important for developers and policymakers to be aware of these issues and take steps to address them.

In conclusion, emotion detection using CNNs is a promising technology with many potential applications in a variety of fields. However, there are still significant technical and ethical challenges that need to be addressed before it can be widely adopted. Continued research and development are needed to

refine the technology, improve its accuracy and reliability, and address the ethical and social implications of its use. By working together to develop responsible and transparent practices for the use of emotion detection technology, we can harness its potential while ensuring that it is used in a way that is fair, just, and respectful of individual rights and privacy.

Emotion detection using CNN is a promising field that is rapidly advancing. The development of deep learning models has led to significant improvements in the accuracy of emotion recognition. The use of convolutional neural networks has proven to be particularly effective in this area, as they can extract useful features from image data and use them to classify emotions. The ability to detect emotions has important applications in many areas, including marketing, psychology, and human-computer interaction.

However, despite the progress that has been made, there are still some limitations and challenges in emotion detection using CNN. One of the primary challenges is the availability and quality of training data, which can affect the accuracy of the model. Additionally, there is still room for improvement in the interpretation of the emotions detected by the models, as they may not always align with human perceptions of emotions. Furthermore, the use of CNNs for emotion detection is still a relatively new field, and there is still much research to be done in terms of exploring different architectures, optimization techniques, and data augmentation methods.

In conclusion, emotion detection using CNN has the potential to be a highly impactful field, with a wide range of applications. The advancements in deep learning models and the availability of powerful computing resources have enabled researchers to make significant progress in this area. However, there are still challenges and limitations that need to be addressed, and ongoing research is necessary.

Recent advancements in emotion detection using CNN have shown promising results in various applications such as human-computer interaction, affective computing, and social robotics. One such advancement is the use of transfer learning, which involves using pre-trained CNN models to improve the performance of emotion detection models. This allows for faster training times and better accuracy, especially when the dataset is small. Another development is the use of attention mechanisms in CNN models, which enables the model to focus on important regions of the image while ignoring irrelevant information. This has resulted in improved accuracy and robustness of emotion detection models. Additionally, research is being conducted on the use of multimodal approaches, which combine visual, audio, and textual data to improve emotion detection. These advancements have the potential to revolutionize the field of emotion detection and open up new avenues for research and development in this area.

Another important area of research in emotion detection using CNN is the development of models that can handle dynamic emotions, such as those exhibited during a conversation or a movie scene. Traditional emotion detection models typically analyze still images, which may not capture the temporal dynamics of emotions. To address this, researchers have developed models that analyze video sequences or audio signals to detect changes in emotional states over time. Such models use recurrent neural networks (RNNs) or convolutional recurrent neural networks (CRNNs) to model the temporal dependencies between consecutive frames or audio samples. Additionally, there is growing interest in developing emotion detection models that are robust to individual differences and cultural variations in expressions of emotion. This requires collecting datasets that are diverse in terms of age, gender, ethnicity, and culture, and developing models that can generalize across these differences. Overall, the field of emotion detection using CNN is rapidly evolving, with new techniques and applications

being developed regularly, and holds great promise for advancing our understanding of human emotions and improving human-computer interaction.

Finally, it is worth noting that while emotion detection using CNN has shown great potential in various applications, there are still several challenges that need to be addressed. One major challenge is the lack of standardized evaluation metrics, which makes it difficult to compare the performance of different models. Additionally, there is a need for more large-scale, diverse datasets to train and evaluate these models, as well as better methods for collecting and labeling emotional data. Another challenge is the interpretability of CNN models, which can be difficult to interpret and understand how they make decisions about emotions. This can be particularly problematic in high-stakes applications such as mental health diagnosis or security screening. Finally, there are ethical considerations regarding the use of emotion detection in certain contexts, such as privacy concerns and potential biases in the models. Despite these challenges, emotion detection using CNN remains a rapidly growing and exciting field with many potential applications and opportunities for future research.

## **7.2 FUTURE ENHANCEMENT**

There are several potential avenues for future enhancement of emotion detection using CNN. One area of focus could be on improving the interpretability of CNN models, such as developing methods to visualize and explain the features learned by the model, or incorporating external knowledge to help guide the decision-making process. Another area for improvement is the development of more sophisticated models that can capture more nuanced and complex emotional states, such as multi-modal models that incorporate both facial expressions and vocal cues, or models that can take into account contextual information such as social cues and situational factors. Additionally, there is a need for more research on cross-cultural and cross-linguistic emotion detection,

as emotions can vary greatly across different cultures and languages. Finally, there is a need for more research on the ethical implications of emotion detection, including issues of privacy, bias, and potential misuse of the technology. Overall, the future of emotion detection using CNN looks promising, with many exciting opportunities for continued research and development.

**Multi-modal Emotion Detection:** Current emotion detection systems rely primarily on visual data, such as facial expressions. However, emotions can also be conveyed through other modalities such as voice and body language. By incorporating these additional modalities into the emotion detection system, it could potentially improve the accuracy and reliability of the predictions.

**Incremental Learning:** With the rapid development of technology and the abundance of data, it is increasingly important for emotion detection systems to be able to adapt to new emotions and expressions that may not have been present in the training data. Incremental learning algorithms could be developed to continuously update the model as new data becomes available.

**Contextual Emotion Detection:** Emotions can be heavily influenced by the context in which they occur. For example, a smile in a social situation could indicate happiness, but the same smile in a professional setting could indicate politeness or even sarcasm. By incorporating contextual information into the emotion detection system, it could improve the accuracy of emotion predictions in different contexts.

**Explainable Emotion Detection:** As with any machine learning system, it is important to be able to interpret the decisions made by the system. Developing explainable AI techniques for emotion detection could help to provide more transparency into the decision-making process, which is particularly important in sensitive applications such as mental health care.



In conclusion, there is still a lot of potential for future enhancements in emotion detection using CNN. One potential area of improvement is the use of more advanced network architectures, such as multi-scale CNNs or recurrent CNNs, to better capture the temporal and spatial information of facial expressions. Additionally, incorporating more advanced pre-processing techniques and data augmentation strategies may improve the robustness and generalization of the model. Furthermore, incorporating contextual information and multimodal inputs, such as speech and physiological signals, may further improve the accuracy and reliability of emotion detection. Finally, the development of real-time emotion detection systems that can operate on mobile devices or embedded systems may enable new applications in human-computer interaction, affective computing, and healthcare. Overall, the potential for future enhancements in emotion detection using CNN is vast, and further research and development in this area are warranted.

## REFERENCE

- [1] Praveen, P. G., & George, J. (2019). Emotion Detection using Convolutional Neural Network. *International Journal of Innovative Technology and Exploring Engineering*, 8(11), 1377-1380.
- [2] Adarsh, V., Reddy, G. R., & Prabhu, A. (2018). Emotion Detection from Speech using Convolutional Neural Networks. *International Journal of Computer Sciences and Engineering*, 6(9), 276-282.
- [3] Guo, Y., Yang, W., & Wang, D. (2019). Emotion recognition in speech signals based on deep learning. *Speech Communication*, 113, 63-74.
- [4] Kaur, S., & Kaur, J. (2019). Emotion Recognition using Deep Learning: A Comprehensive Review. *IEEE Access*, 7, 113206-113231.
- [5] Kim, J., Lee, M., & Cho, S. (2021). Multi-modal Emotion Detection using Convolutional Neural Networks with Attention Mechanisms. *IEEE Transactions on Affective Computing*.
- [6] Kostoulas, T., & Giannakopoulos, T. (2017). Emotion Recognition from Speech using Convolutional Neural Networks and Transfer Learning. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction* (pp. 429-436).
- [7] Li, X., Zhu, H., Zhang, X., & Xu, D. (2019). Emotion recognition from speech using convolutional neural networks with feature fusion and fine-tuning. *Journal of Intelligent Information Systems*, 53(3), 475-494.
- [8] Liu, Y., Wang, Z., & Zheng, L. (2018). Emotion Recognition from Speech Using Convolutional Neural Network and Self-attention. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 2366-2370).
- [9] Mitra, S., & Saha, S. (2019). Emotion Recognition from Speech using

Convolutional Neural Network with Spectrogram-based Input. In 2019 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS) (pp. 1-6).

[10] Nourbakhsh, A., Shirazi, M. S., & Bagheri, M. (2017). Sentiment Analysis in Persian Text Using Convolutional Neural Network. In 2017 9th International Symposium on Telecommunications (IST) (pp. 580-584).

[11] Pandey, S., Verma, D., & Joshi, A. (2021). Facial Emotion Recognition using Convolutional Neural Network with Transfer Learning. In Proceedings of the 6th International Conference on Inventive Computation Technologies (pp. 720-725).

[12] Park, C., Lee, H., Kim, H., & Ko, B. (2017). Emotion Recognition Using Convolutional Neural Network with Attention Mechanism. In 2017 IEEE International Conference on Big Data and Smart Computing (BigComp) (pp. 149-152).

[13] Peng, Y., Jin, Y., Wu, T., & Jia, L. (2020). Facial Emotion Recognition based on Convolutional Neural Network with Fine-tuning. In 2020 39th Chinese Control Conference (CCC) (pp. 4500-4505).

[14] Poria, S., Cambria, E., Hazarika, D., Mazumder, N., & Zadeh, A. (2011)

[15] Sun, Y., & Yuan, C. (2019). Emotion recognition from speech using deep learning: a review. IEEE/CAA Journal of Automatica Sinica, 6(5), 1035-1049.

[16] Han, K., Kim, H. J., & Lee, S. (2018). Emotion recognition using deep learning: A review. IEEE Access, 6, 24411-24423.

[17] Zhang, H., Cao, B., Guo, X., & Zhang, C. (2018). A survey on emotion recognition using physiological signals. IEEE Transactions on Affective Computing, 10(3), 374-393.

[18] Mollahosseini, A., Hasani, B., & Mahoor, M. H. (2017). AffectNet: A

database for facial expression, valence, and arousal computing in the wild. *IEEE Transactions on Affective Computing*, 10(1), 18-31.

[19] Taheri, S., & Wang, D. (2020). Multi-modal deep learning for emotion recognition: A survey. *IEEE Access*, 8, 27409-27427.

[20] Bölcskei, H., & Luthra, M. (2020). A comparison of deep learning models for speech emotion recognition. *IEEE Access*, 8, 100584-100594.

[21] Ren, X., Huang, S., Zhang, Y., & Wang, Y. (2021). Emotion recognition based on multi-modal physiological signals using deep learning. *IEEE Access*, 9, 14756-14765.

[22] Kim, K. H., Bang, S. W., & Kim, S. R. (2017). Emotion recognition using a convolutional neural network with multi-task learning. *Cognitive Computation*, 9(3), 301-310.

[23] Ghosal, A., & Singh, P. (2021). Emotion recognition from speech using transfer learning and deep learning. *Expert Systems with Applications*, 165, 114119.

[24] Wang, H., & Wang, Y. (2019). Emotion recognition from facial expression using deep learning and support vector machine. *Multimedia Tools and Applications*, 78(23), 32947-32967.

[25] Zhang, Z., Li, J., & Wei, J. (2021). A facial emotion recognition system based on deep convolutional neural networks. *Journal of Ambient Intelligence and Humanized Computing*, 12(8), 7959-7972.

[26] Wang, Y., Huang, S., Ren, X., & Liu, T. (2020). A novel deep learning framework for multimodal emotion recognition from speech and physiological signals. *IEEE Access*, 8, 142711-142723.

- [27] Huang, L., Wang, X., Lu, Y., & Zhang, J. (2021). Emotion recognition using improved CNN-based models. *Neurocomputing*, 444, 311-319.
- [28] Majhi, B., & Bandyopadhyay, S. (2019). Deep learning based emotion recognition: A comprehensive review. *Cognitive Computation*, 11(6), 1012-1046.
- [29] Walecki, R., & Drwiega, M. (2020). Convolutional neural network for emotion recognition from audio signals: A comparison of different training methods. *Multimedia Tools and Applications*, 79(23), 15931-15947.
- [30] X. Han, W. Zhang, S. Liu, H. Huang, and S. Wang, "Real-time emotion recognition using convolutional neural networks," in *Proceedings of the 24th ACM international conference on Multimedia*, 2016, pp. 369–373.
- [31] T. Kim, K. Kim, J. Han, and J. Kim, "Emotion recognition using a hierarchical binary convolutional neural network," in *Proceedings of the 2016 ACM on Multimedia Conference*, 2016, pp. 1077–1080.
- [32] A. Mathur, P. Gupta, and S. K. Saha, "Deep CNN and LSTM-based emotion detection and recognition," *Journal of Intelligent Systems*, vol. 28, no. 4, pp. 581–594, 2019.
- [33] S. Mehdi, S. Hussain, R. Hussain, and A. Almogren, "Emotion recognition from facial expressions using convolutional neural networks," in *Proceedings of the International Conference on Machine Learning and Data Engineering*, 2017, pp. 114–124.
- [34] S. E. El-Khamy, S. A. El-Samie, and M. A. El-Dahshan, "Emotion recognition from facial expressions using multilevel convolutional neural networks," in *Proceedings of the IEEE International Conference on Image Processing*, 2017, pp. 2728–2732.
- [35] C. A. Nascimento, P. S. P. Martins, and A. C. F. Loureiro, "Deep convolutional neural networks for emotion recognition from facial expressions

using transfer learning,” in Proceedings of the 2018 IEEE 31st International Symposium on Computer-Based Medical Systems (CBMS), 2018, pp. 171–176.

[36] J. Kim, K. Lee, and H. Lee, “Emotion recognition from speech using deep recurrent neural networks with shortcut connections,” *IEEE Signal Processing Letters*, vol. 24, no. 2, pp. 259–263, 2017.

[37] H. El-Najjar, N. Alajlan, A. Al-Nuaimy, and S. Mohammed, “Multimodal emotion recognition using audio and text features and convolutional neural networks,” in Proceedings of the 2018 International Conference on Computer and Information Sciences (ICCIS), 2018, pp. 1–6.

[38] S. S. Sahu, K. Kumar, and V. P. Singh, “Emotion recognition using convolutional neural networks: a review,” in Proceedings of the 2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU), 2019, pp. 1–5.

[39] T. S. Murthy, S. S. Sahu, V. P. Singh, and K. Kumar, “Deep learning based emotion recognition system: a review,” in Proceedings of the 2019 11th International Conference on Computational Intelligence and Communication Networks (CICN), 2019, pp. 1–5.

[40] Song, B., Kim, J., & Kim, D. (2019). A study on facial expression recognition based on CNN. *Journal of Advanced Research in Dynamical and Control Systems*, 11(7), 132-139.

[41] Mirza, A., & Raza, M. A. (2019). Emotion recognition using facial expressions: A comprehensive review. *IETE Technical Review*, 36(5), 435-446.

[42] Zhan, K., Zhang, K., Zhong, Y., Lu, X., & Zhang, Z. (2019). A novel two-stage framework for emotion recognition from physiological signals using deep convolutional neural networks. *IEEE Journal of Biomedical and Health*

Informatics, 24(1), 121-132.

[43] Yang, W., Zhao, H., Zhou, X., Wang, M., & Chen, M. (2019). Multi-modal emotion recognition using CNN and LSTM networks. In International Conference on Advanced Computational Methods in Engineering (pp. 370-378). Springer, Cham.

[44] Fang, Z., & Yu, X. (2019). Multimodal emotion recognition based on CNN and LSTM. In International Conference on Artificial Intelligence and Security (pp. 11-20). Springer, Cham.

[45] Li, X., Li, B., & Wu, Z. (2019). Multi-task learning for facial expression recognition using CNN. *Journal of Intelligent & Fuzzy Systems*, 36(6), 6055-6065.

[46] Zhang, K., Zhong, Y., Wang, J., Zhang, Z., & Lu, X. (2018). Emotion recognition using two-stage CNN with fusion features from multiple physiological signals. *Journal of Ambient Intelligence and Humanized Computing*, 9(5), 1585-1597.

[47] Zhang, L., Ji, Z., Lu, W., & Wang, J. (2018). Facial expression recognition using hybrid CNN and LSTM network. *Multimedia Tools and Applications*, 77(18), 23867-23883.

[48] Tripathy, R. K., & Mishra, B. B. (2018). Emotion recognition from EEG signals using CNN and LSTM networks.