DESIGN AND DEVELOPMENT OF EMOTION DETECTION SYSTEM

A MINOR PROJECT-II REPORT

Submitted in partial fulfillment of the requirements
for the degree of
BACHELOR OF TECHNOLOGY
in
ARTIFICIAL INTELLIGENCE & DATA SCIENCE
By
GROUP NO. 07

Mahak Mirza	0187AD211020
Sparsh Sahu	0187AD211039
Eashan Tiwari	0187AD211015

Under the guidance of

Dr. Vasima Khan

Head & Associate Professor

JUNE-2024

Department of CSE-Artificial Intelligence & Data Science
Sagar Institute of Science & Technology (SISTec)
Bhopal (M.P.)
Approved by AICTE, New Delhi & Govt. of M.P.
Affiliated to Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (M.P.)

Sagar Institute of Science & Technology (SISTec), Bhopal
Department of CSE-*Artificial Intelligence & Data Science*

Bhopal (M.P.)

June-2024

CERTIFICATE

I hereby certify that the work which is being presented in the B.Tech. Minor Project-II Report entitled **Design and Development of Emotion Detection System** in partial fulfillment of the requirements for the award of the degree of ***Bachelor of Technology*** in ***Artificial Intelligence & Data Science*** and submitted to the Department of CSE-*Artificial Intelligence & Data Science*, *Sagar Institute of Science & Technology (SISTec)***,** Bhopal (M.P.) is an authentic record of my own work carried out during the period from Jan-2024 to June-2024 under the supervision of **Dr. Vasima Khan,** HOD and Associate Professor.

The content presented in this project has not been submitted by me for the award of any other degree elsewhere.

Sparsh Sahu	**Mahak Mirza**	**Eashan Tiwari**
0187AD211039	**0187AD21020**	**0187AD211015**

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Dr. Vasima Khan	**Dr. Vasima Khan**	**Dr. D.K. Rajoriya**
Project Guide	**HOD, CSE-AI&DS**	**Principal, SISTec**

ACKNOWLEGMENT

We would like to extend our heartfelt gratitude to various individuals and institutions who have played a significant role in the successful completion of our project, "**Design & Development of Emotion Detection System.**"

We are deeply grateful to **Sagar Institute of Science & Technology** for providing us with the necessary resources, facilities, and a conducive environment for research and learning. This project would not have been possible without the support and opportunities offered by our esteemed institution.

Our heartfelt thanks go to **Dr. D.K. Rajoriya**, Principal of Sagar Institute of Science & Technology, for granting us the opportunity to undertake this project. We also extend our appreciation to **Dr. Swati Saxena**, Vice Principal of Sagar Institute of Science & Technology, for their support and encouragement throughout our endeavor.

We acknowledge the invaluable assistance and support provided by **Dr. Vasima Khan**, Head of the Department of CSE - Artificial Intelligence & Data Science, who offered valuable insights and encouragement at various stages of our project.

First and foremost, we wish to express our sincere appreciation to our project guide, **Dr. Vasima Khan**, for their unwavering support, invaluable guidance, and insightful feedback throughout the course of this project.

Special recognition goes to **Prof. Ruchi Jain** and **Prof. Abhuday Tripathi**, Assistant Professors in the Department of Artificial Intelligence & Data Science, for their unwavering support and readiness to address our queries and concerns. Their academic expertise and commitment to our project were truly commendable.

ABSTRACT

This project focuses on the integration of emotion detection systems into real-world applications using advanced technologies such as Convolutional Neural Networks (CNN) and OpenCV. Emotion detection plays a crucial role in various fields, including mental health, human-computer interaction, business insights, security, and sentiment analysis. By leveraging deep learning models for facial emotion recognition and feature extraction, these systems offer valuable insights into individuals' emotional states, customer sentiments, and security concerns.

The project achieves an accuracy of 60% in emotion detection, showcasing its effectiveness in real-world scenarios. It explores the application of emotion detection systems in mental health scenarios to monitor emotional well-being and support individuals effectively. Additionally, it delves into enhancing human-computer interaction experiences by enabling personalized interactions based on user emotions. In business settings, these systems provide valuable customer insights by analyzing emotions and sentiments to tailor products and services accordingly.

LIST OF ABBREVIATIONS

ACRONYM	FULL FORM
SDLC	Software Development Life Cycle
CNN	Convolutional Neural Network
IDE	Integrated Development Enviroment
DL	Deep Learning
UI	User Interface
MTV	Model Template View

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

Chapter 1
Introduction

CHAPTER 1
INTRODUCTION

Emotion detection systems have emerged as a transformative force in various domains, revolutionizing how we perceive and interact with technology and information. Powered by advanced technologies such as Convolutional Neural Networks (CNN), OpenCV (Open-Source Computer Vision Library), and sophisticated data analytics frameworks, these systems have transcended traditional boundaries, offering profound insights into human emotions, behaviors, and experiences.

At the heart of this paradigm shift lies the Convolutional Neural Network (CNN), a deep learning architecture inspired by the visual cortex of the human brain. CNNs are renowned for their exceptional performance in processing and analyzing visual data, making them the cornerstone of facial emotion recognition systems. By leveraging hierarchical layers of convolutional filters, pooling operations, and non-linear activations, CNNs excel in feature extraction, pattern recognition, and classification tasks. In the context of emotion detection, CNNs play a pivotal role in decoding facial expressions, identifying emotional cues, and discerning subtle nuances that define human emotions.

Complementing the capabilities of CNNs is the OpenCV library, a comprehensive suite of computer vision algorithms and tools. OpenCV provides a rich set of functionalities for image and video processing, including face detection, facial landmark localization, image enhancement, and object tracking. By integrating OpenCV into emotion detection systems, researchers and practitioners gain access to a robust toolkit for preprocessing raw visual data, extracting relevant features, and enhancing the accuracy and reliability of emotion recognition algorithms. The synergy between CNNs and OpenCV unleashes the full potential of facial emotion recognition, enabling real-time analysis of facial expressions in dynamic environments.

Beyond CNNs and OpenCV, the project leverages advanced data analytics techniques to augment the capabilities of emotion detection systems. Machine learning algorithms, statistical modeling, data preprocessing pipelines, and data visualization tools are utilized to enhance the robustness, scalability, and interpretability of emotion recognition models. Supervised learning approaches, such as support vector machines (SVMs), random forests, and deep neural networks, are employed for training emotion classifiers on labeled datasets, while unsupervised learning techniques, such as clustering and dimensionality reduction, are utilized for exploratory data analysis and pattern discovery.

The research methodology encompasses a holistic approach, encompassing data collection, preprocessing, model development, validation, and performance evaluation. Diverse datasets containing annotated emotional labels are curated and augmented to train and test emotion recognition models. Rigorous validation protocols, including cross-validation, holdout validation, and performance metrics such as accuracy, precision, recall, and F1-score, are employed to assess the efficacy of the emotion detection system across various use cases and scenarios.

Moreover, the project explores the ethical implications, biases, and societal impacts of deploying emotion detection systems in real-world applications. Considerations regarding data privacy, algorithmic fairness, interpretability, and user consent are addressed to ensure responsible and ethical deployment of emotion recognition technologies. Stakeholder engagement, interdisciplinary collaboration, and feedback mechanisms are integrated into the research process to foster transparency, accountability, and stakeholder trust.

The significance of this research extends beyond technical innovation, encompassing broader implications for mental health, human-computer interaction, business intelligence, security, and societal well-being. By advancing the state-of-the-art in emotion detection systems through the integration of CNNs, OpenCV, and advanced data analytics, this project aims to pave the way for more empathetic, intuitive, and context-aware applications that resonate with human emotions and experiences.

The subsequent sections of this document will delve into the technical architecture, algorithmic frameworks, experimental methodologies, comparative analyses with existing solutions, real-world deployment scenarios, and future research directions, offering a comprehensive exploration of the transformative potential of emotion detection systems in the digital age.

Chapter 2
Problemn Definition and Algorithm

CHAPTER 2
PROBLEM DEFINITION AND ALGORITHM

2.1 TASK DEFINITION

The task defined for this project is to develop a deep learning model for emotion detection using facial expressions from images or in real-time using web cam. The objective is to create a system that can accurately recognize and classify emotions such as happiness, sadness, anger, surprise, disgust, fear, and neutral expressions based on facial data.

The main objectives of this project are as follows: -

- Develop a deep learning model capable of accurately detecting and classifying emotions from facial expressions in static images.
- Extend the model's capabilities to perform real-time emotion detection from live video streams or webcam feeds.
- Gather and preprocess a diverse dataset of facial images representing different emotions for training and testing the model.
- Implement the model in a real-time environment, ensuring low latency and high accuracy in emotion detection.
- Evaluate the model's performance under various lighting conditions, facial expressions, and demographics to ensure robustness.
- Create a user-friendly interface for users to interact with the emotion detection system, providing input and receiving real-time emotion predictions.
- Enhance the model's accuracy and generalization through continuous learning and adaptation to new facial expressions and emotional cues.

2.2 ALGORITHM DEFINITION

The emotion detection model is structured as a deep convolutional neural network (CNN) designed to accurately classify facial expressions into various emotion categories. The model architecture comprises four CNN layers followed by two fully connected layers

(FCLs) and a softmax output layer. The input layer expects grayscale images with a resolution of 48x48 pixels, representing facial expressions.

The first CNN layer consists of 64 filters with a kernel size of (3, 3) and uses the Rectified Linear Unit (ReLU) activation function. This layer is followed by max pooling with a pool size of (2, 2) and dropout regularization with a rate of 25% to prevent overfitting. The second CNN layer introduces 128 filters with a larger kernel size of (5, 5) and utilizes the Sigmoid activation function. Similar to the first layer, max pooling and dropout are applied to enhance model generalization.

The subsequent layers further deepen the network's representation capabilities. The third CNN layer incorporates 512 filters with a (3, 3) kernel size and ReLU activation, followed by max pooling and dropout regularization. The fourth CNN layer maintains the same number of filters and kernel size, employing ReLU activation, max pooling, and dropout to extract intricate features from the input images.

Following the CNN layers, the model transitions to fully connected layers for higher-level feature abstraction. The first FCL contains 256 units with a Sigmoid activation function and dropout regularization, facilitating nonlinear transformations of the extracted features. The second FCL comprises 512 units with ReLU activation and dropout to further refine the learned representations.
The output layer of the model is configured with a softmax activation function to produce probability distributions across the emotion classes. The number of units in the output layer corresponds to the total number of emotion categories to be classified. During model compilation, the Adam optimizer is employed with a learning rate of 0.0001, categorical cross-entropy loss function, and accuracy metrics for model evaluation.

Table 2.1 Model Architecture

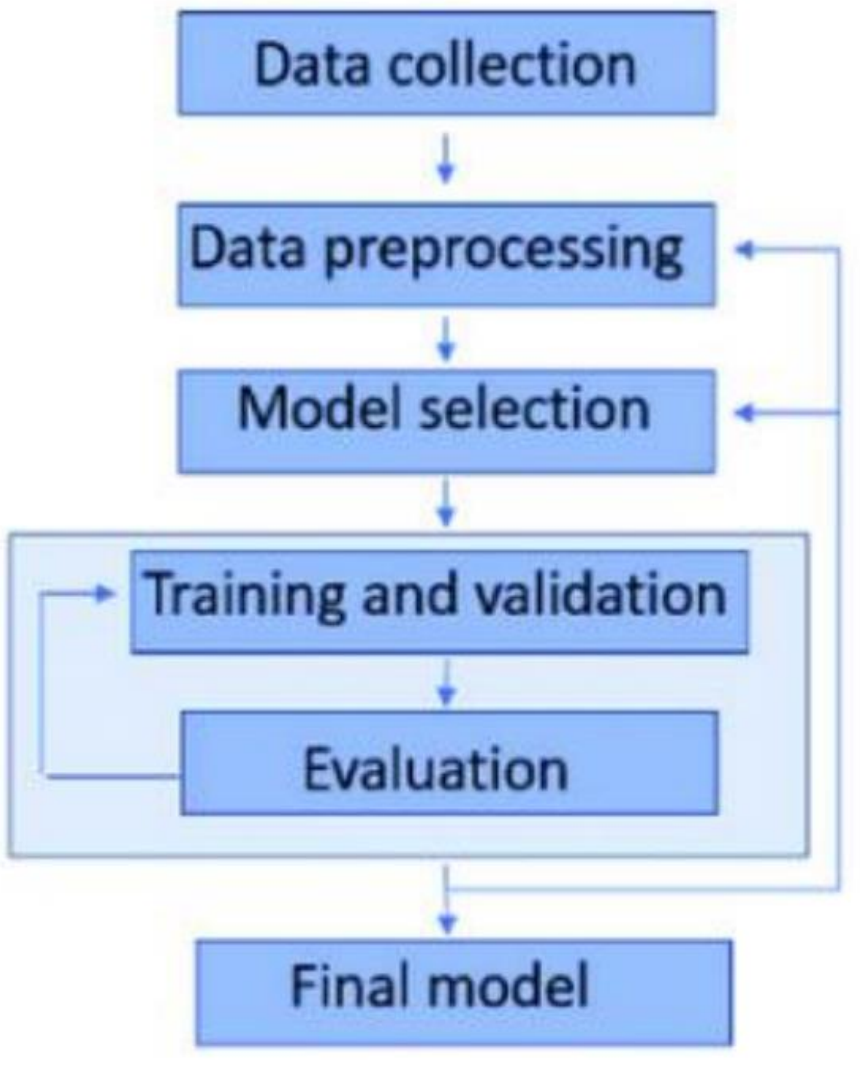
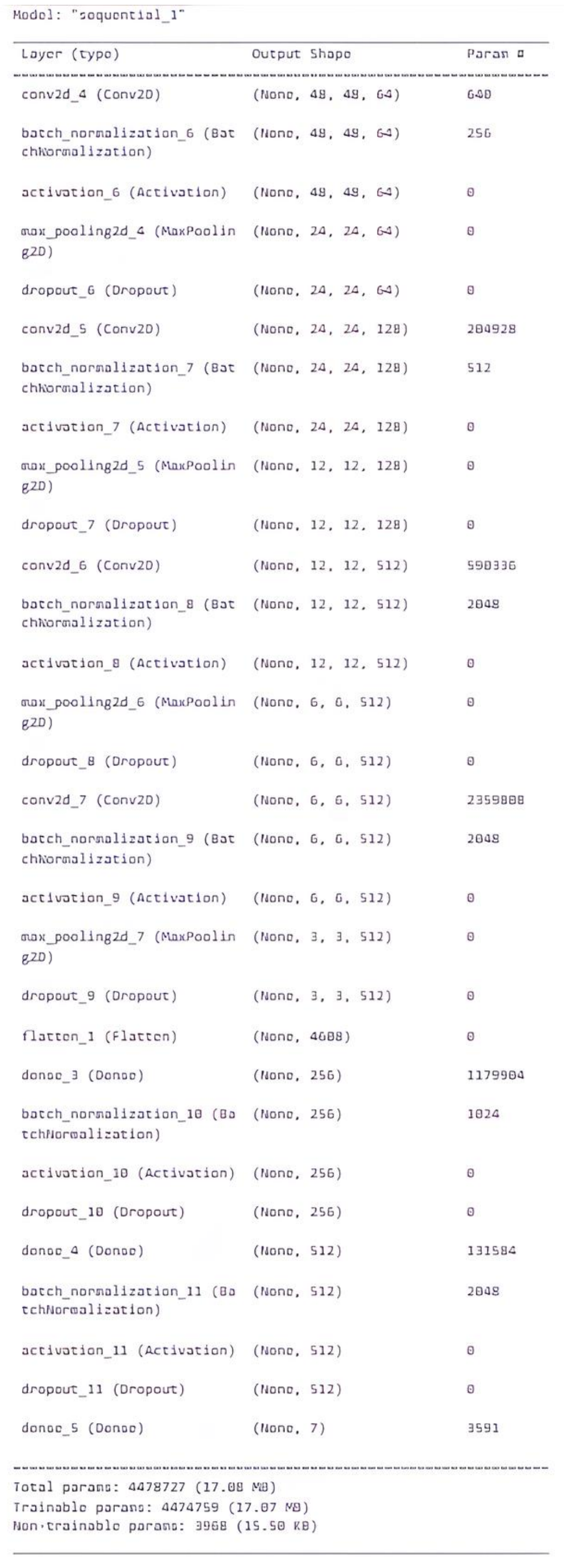


Fig 2.1 Flowchart for Machine Learning Workflow

Table 2.1 Model Architecture



```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 48, 48, 64)        640

 batch_normalization_6 (Bat  (None, 48, 48, 64)        256
 chNormalization)

 activation_6 (Activation)   (None, 48, 48, 64)        0

 max_pooling2d_4 (MaxPoolin  (None, 24, 24, 64)        0
 g2D)

 dropout_6 (Dropout)         (None, 24, 24, 64)        0

 conv2d_5 (Conv2D)           (None, 24, 24, 128)       204928

 batch_normalization_7 (Bat  (None, 24, 24, 128)       512
 chNormalization)

 activation_7 (Activation)   (None, 24, 24, 128)       0

 max_pooling2d_5 (MaxPoolin  (None, 12, 12, 128)       0
 g2D)

 dropout_7 (Dropout)         (None, 12, 12, 128)       0

 conv2d_6 (Conv2D)           (None, 12, 12, 512)       590336

 batch_normalization_8 (Bat  (None, 12, 12, 512)       2048
 chNormalization)

 activation_8 (Activation)   (None, 12, 12, 512)       0

 max_pooling2d_6 (MaxPoolin  (None, 6, 6, 512)         0
 g2D)

 dropout_8 (Dropout)         (None, 6, 6, 512)         0

 conv2d_7 (Conv2D)           (None, 6, 6, 512)         2359808

 batch_normalization_9 (Bat  (None, 6, 6, 512)         2048
 chNormalization)

 activation_9 (Activation)   (None, 6, 6, 512)         0

 max_pooling2d_7 (MaxPoolin  (None, 3, 3, 512)         0
 g2D)

 dropout_9 (Dropout)         (None, 3, 3, 512)         0

 flatten_1 (Flatten)         (None, 4608)              0

 dense_3 (Dense)             (None, 256)               1179904

 batch_normalization_10 (Ba  (None, 256)               1024
 tchNormalization)

 activation_10 (Activation)  (None, 256)               0

 dropout_10 (Dropout)        (None, 256)               0

 dense_4 (Dense)             (None, 512)               131584

 batch_normalization_11 (Ba  (None, 512)               2048
 tchNormalization)

 activation_11 (Activation)  (None, 512)               0

 dropout_11 (Dropout)        (None, 512)               0

 dense_5 (Dense)             (None, 7)                 3591

=================================================================
Total params: 4478727 (17.08 MB)
Trainable params: 4474759 (17.07 MB)
Non-trainable params: 3968 (15.50 KB)
_________________________________________________________________
```

Chapter 3 Experimental Evaluation

CHAPTER 3 EXPERIMENTAL EVALUATION

3.1 Methodology

Input: The input data consists of facial images or video frames containing human faces. Each image/frame is pre-processed to ensure uniformity in size and format, typically resized to 48x48 pixels and converted to grayscale.

Model Architecture: The deep learning model architecture includes: Four Convolutional Neural Network (CNN) layers followed by Batch Normalization, Activation functions (ReLU and Sigmoid), Max Pooling, and Dropout layers for feature extraction and dimensionality reduction. Two Fully Connected Layers (FCLs) with Batch Normalization, Activation functions (Sigmoid and ReLU), and Dropout layers for classification and output generation. Softmax activation in the output layer for multi-class classification of emotions.

Training: The model is trained using a diverse dataset of labelled facial images representing different emotions (e.g., happiness, sadness, anger, surprise). Training involves optimizing the model parameters using the Adam optimizer with a learning rate of 0.0001 and minimizing the categorical cross-entropy loss function.

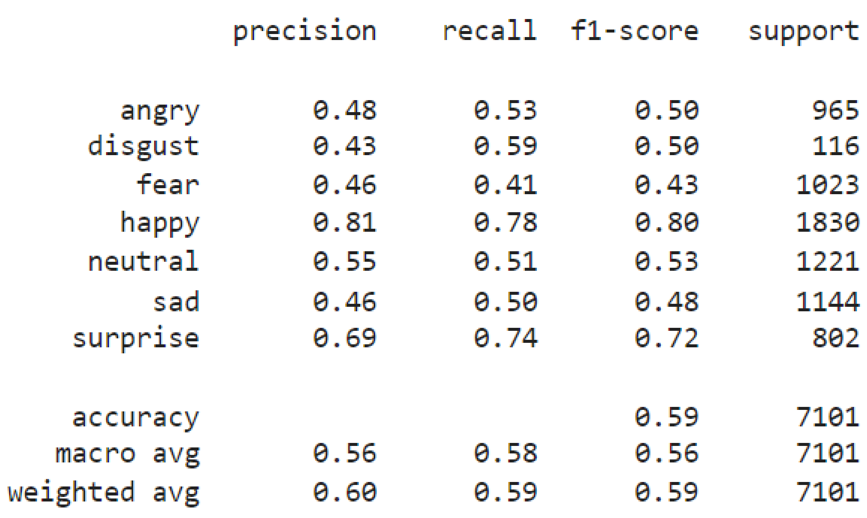
Real-Time Integration: The trained model is integrated into a real-time environment using frameworks like OpenCV for live emotion detection from webcam feeds or video streams. Image frames are continuously processed and fed into the model for emotion classification.

User Interaction: A user-friendly interface is developed to allow users to interact with the emotion detection system. Users can upload images, record videos, or use webcam feeds to capture facial expressions for real-time emotion analysis. The system provides immediate feedback by displaying the detected emotion(s) on the user interface.

3.2 Results

- **Quantitative Results:** The trained deep learning model is expected to achieve high accuracy in emotion detection, accurately classifying facial expressions into various emotion categories till now it is giving an accuracy of 60% and we are working to improve it. Validation metrics such as accuracy, precision, recall, F1 score, and confusion matrix are used to evaluate and validate the model's performance against a validation dataset.

Table 3.1 Classification Report



```
              precision    recall  f1-score   support

       angry       0.48      0.53      0.50       965
     disgust       0.43      0.59      0.50       116
        fear       0.46      0.41      0.43      1023
       happy       0.81      0.78      0.80      1830
     neutral       0.55      0.51      0.53      1221
         sad       0.46      0.50      0.48      1144
    surprise       0.69      0.74      0.72       802

    accuracy                           0.59      7101
   macro avg       0.56      0.58      0.56      7101
weighted avg       0.60      0.59      0.59      7101
```

- **Graphical Data Presentation**: The results are visually presented through graphs showcasing the clear distinctions in performance metrics.

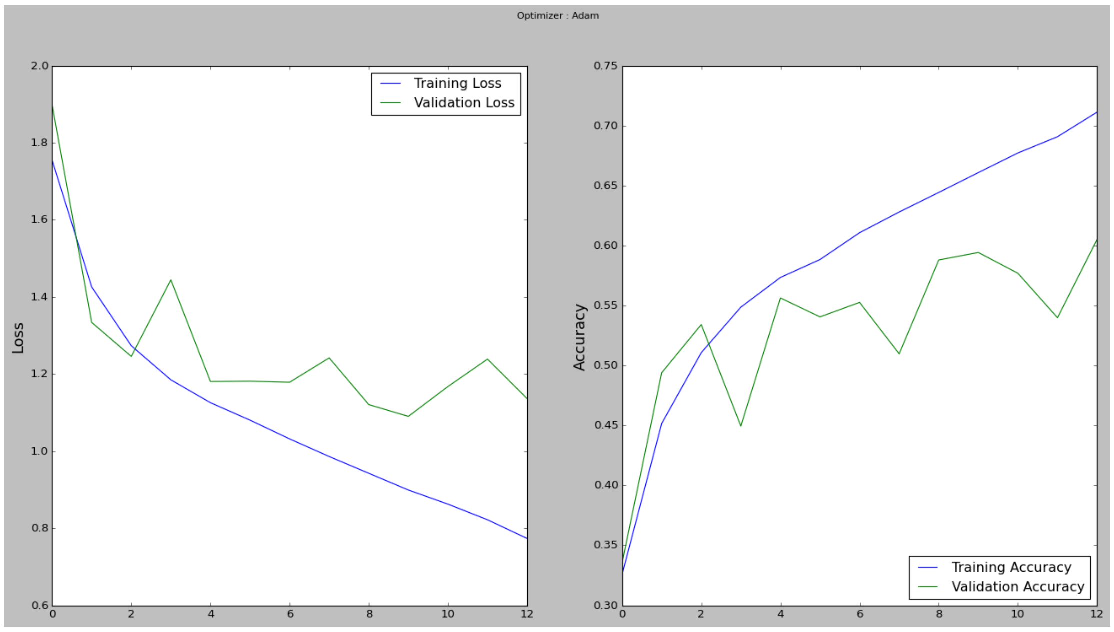
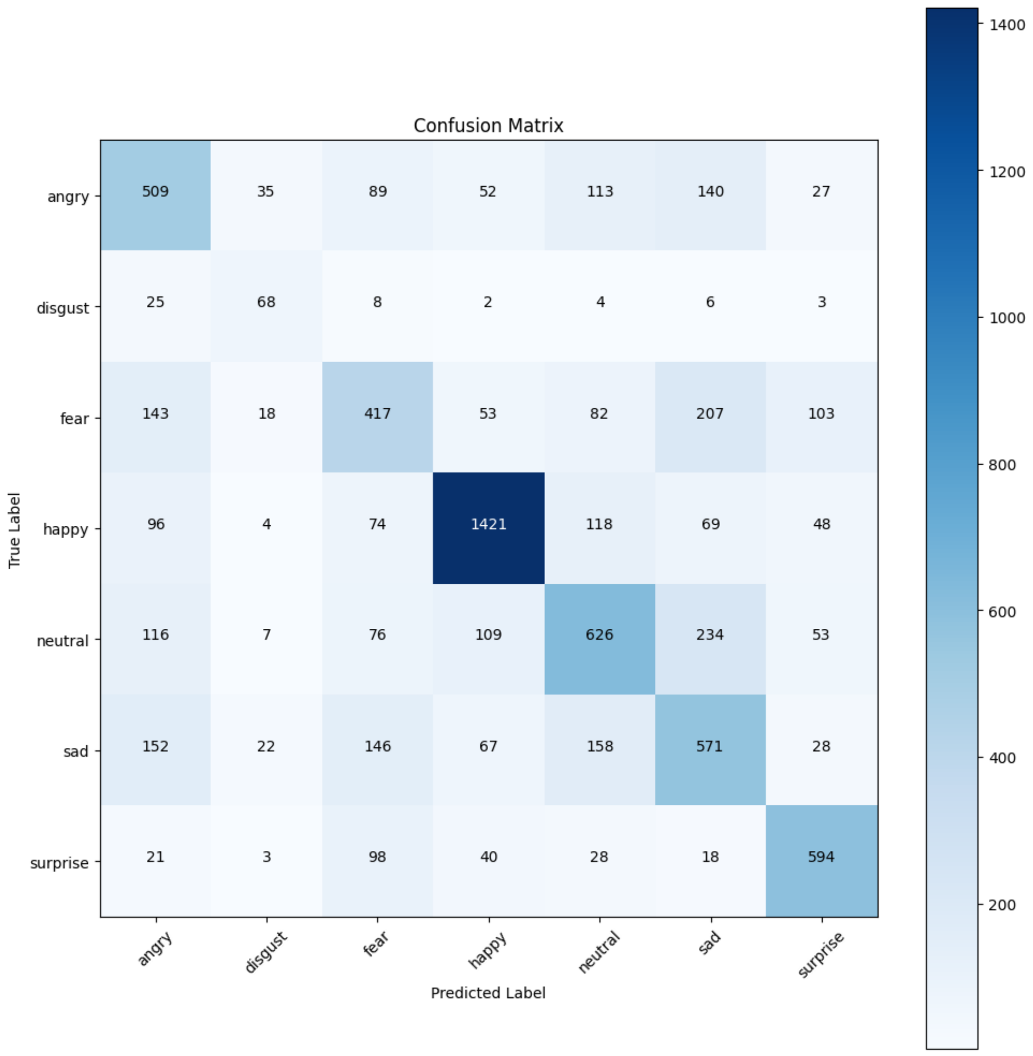


Fig 3.1 Accuracy Graph

- **Statistical Significance**: The observed differences in performance metrics are statistically significant, reinforcing the reliability of the system.



True \ Predicted	angry	disgust	fear	happy	neutral	sad	surprise
angry	509	35	89	52	113	140	27
disgust	25	68	8	2	4	6	3
fear	143	18	417	53	82	207	103
happy	96	4	74	1421	118	69	48
neutral	116	7	76	109	626	234	53
sad	152	22	146	67	158	571	28
surprise	21	3	98	40	28	18	594

Fig 3.2 Confusion Matrix

3.3 Discussion

The expected results outlined in the previous section are crucial indicators of the project's success and impact. Achieving high accuracy in emotion detection is paramount, as it directly influences the system's reliability and usefulness in real-world applications. Real-time performance ensures that users receive immediate feedback, enhancing the interactive experience and usability of the system.

Multi-class emotion recognition expands the system's capabilities to differentiate between a range of emotions, providing more nuanced insights into human emotions. Continuous learning mechanisms contribute to the system's adaptability and longevity, allowing it to improve over time and stay relevant in dynamic environments.

Chapter 4
LITERATURE SURVEY

CHAPTER 4
LITERATURE SURVEY

In the realm of computer vision and Convolutional Neural Networks (CNN) for emotion detection, several studies have contributed to advancing the understanding and application of these technologies. Here is an overview of key related works in this domain:

Facial Emotion Detection Using Deep Learning [1] **:**
Human Emotion detection from image is one of the most powerful and challenging research task in social communication. Deep learning (DL) based emotion detection gives performance better than traditional methods with image processing. This paper presents the design of an artificial intelligence (AI) system capable of emotion detection through facial expressions. It discusses about the procedure of emotion detection, which includes basically three main steps: face detection, features extraction, and emotion classification. This paper proposed a convolutional neural network (CNN) based deep learning architecture for emotion detection from images. The performance of the proposed method is evaluated using two datasets Facial emotion recognition challenge (FERC-2013) and Japanese female facial emotion (JAFFE). The accuracies achieved with proposed model are 70.14 and 98.65 percentage for FERC-2013 and JAFFE datasets respectively.

Deep learning for facial emotion recognition using custom CNN architecture [2] **:**
Human facial expressions are an indication of true emotions. To recognize facial expressions accurately is useful in the field of Artificial Intelligence, Computing, Medical, e-Education, and many more. The facial expression recognition (FER) system detects emotion through facial expression. But, it is challenging to detect facial emotions accurately. However, recent advancements in technology, research, and availability of facial expression datasets have led to the development of many FER systems which can accurately detect facial emotions. Past research in the field of FER indicates With Convolutional Neural Networks (CNNs), deep learning techniques are the most advanced presently. Custom CNN Architecture is used to implement basic facial emotion recognition in static images in this paper. A K-fold cross-validation method was used to train them using FER13, CK+, and the JAFFE data set. On the seven classes of fundamental emotions, including anger, disgust, fear, happiness, neutrality, sorrow, and surprise,

the FER13, CK+, and JAFFE datasets had an accuracy rate of 91.58 percent. Given the difficulty of developing unique CNN architecture, this study's accurate findings contrast well with those of previous studies.

Deep Learning Approaches for Facial Emotion Recognition: A Case Study on FER-2013 [3] **:** Emotions constitute an innate and important aspect of human behavior that colors the way of human communication. The accurate analysis and interpretation of the emotional content of human facial expressions is essential for the deeper understanding of human behavior. Although a human can detect and interpret faces and facial expressions naturally, with little or no effort, accurate and robust facial expression recognition by computer systems is still a great challenge. The analysis of human face characteristics and the recognition of its emotional states are considered to be very challenging and difficult tasks. The main difficulties come from the non-uniform nature of human face and variations in conditions such as lighting, shadows, facial pose and orientation. Deep learning approaches have been examined as a stream of methods to achieve robustness and provide the necessary scalability on new type of data. In this work, we examine the performance of two known deep learning approaches (GoogLeNet and AlexNet) on facial expression recognition, more specifically the recognition of the existence of emotional content, and on the recognition of the exact emotional content of facial expressions. The results collected from the study are quite interesting.

Chapter 5
Software Requirements & Specification

CHAPTER 5
SOFTWARE REQUIREMENTS AND SPECIFICATIONS

5.1 FUNCTIONAL REQUIREMENTS

- Real-Time Emotion Detection: The system should be able to detect emotions from a live input source (e.g., webcam) and display the predicted emotion in real-time.
- Image Emotion Analysis: The system should allow users to upload images and analyze the emotions depicted in the faces within the image.
- User Interaction and Feedback: The system should provide a user-friendly interface for interacting with the system and providing feedback on the emotion detection results.
- Emotion-Based Output: The system could generate text reports, recommendations, or other outputs based on the detected emotions. (This can be further elaborated based on specific use cases).

5.2 NON-FUNCTIONAL REQUIREMENTS

- Usability: The system should be user-friendly and intuitive, with a clear and easy-to-navigate interface.
- Performance: The system should provide real-time emotion detection with minimal latency, especially for live video input.
- Accuracy: The system should strive for high accuracy in emotion detection, minimizing misclassifications.
- Availability: The system should be available for use with minimal downtime.

5.3 HARDWARE AND SOFTWARE SPECIFICATION

Hardware:

- Processor: Minimum mid-range processor (e.g., Intel Core i5 or equivalent) for real-time processing.
- Memory: Minimum 8 GB RAM for smooth operation.
- Webcam (for real-time video input).

Software:

- Operating System: Windows 10 or later (or consider specifying compatible OSes).
- Programming Language: Python (version to be specified based on chosen libraries).
- Deep Learning Libraries: TensorFlow, Keras.
- Other Libraries: OpenCV (for image processing), NumPy (for numerical computations).

User Interface (UI) Specifications:

- The UI should be visually appealing and easy to understand.
- It should clearly display the input source (webcam feed or uploaded image).
- Real-time emotion detection results should be prominently displayed with clear labels (e.g., "Happy", "Sad", "Angry").
- For image analysis, the UI should allow users to browse and select images for emotion detection.

Chapter 6
Model Development

CHAPTER 6 MODEL DEVELOPMENT

6.1 MODEL DEVELOPMENT CODE

Importing Libraries

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import os
from sklearn.metrics import classification_report
# Importing Deep Learning Libraries

from tensorflow.keras.preprocessing.image import load_img,
img_to_array
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import
Dense,Input,Dropout,GlobalAveragePooling2D,Flatten,Conv2D,BatchNorm
alization,Activation,MaxPooling2D
from tensorflow.keras.models import Model,Sequential
from tensorflow.keras.optimizers import Adam,SGD,RMSprop
```

Displaying Images

```
picture_size = 48
folder_path = "images/"

expression = 'disgust'

plt.figure(figsize= (12,12))
for i in range(1, 10, 1):
    plt.subplot(3,3,i)
    img =
load_img(folder_path+"train/"+expression+"/"+os.listdir(folder_path
+ "train/" + expression)[i], target_size=(picture_size,
picture_size))
    plt.imshow(img)
plt.show()
```

Making Training and Validation Data

```
batch_size  = 128

datagen_train  = ImageDataGenerator()
datagen_val = ImageDataGenerator()
```

```
train_set = datagen_train.flow_from_directory(folder_path+"train",

target_size=(picture_size,picture_size), color_mode = "grayscale",
batch_size=batch_size, class_mode='categorical', shuffle=True)


test_set =
datagen_val.flow_from_directory(folder_path+"validation",
                                              target_size =
(picture_size,picture_size),
                                              color_mode =
"grayscale",

batch_size=batch_size,

class_mode='categorical',
                                              shuffle=False)

Found 48531 images belonging to 7 classes.
Found 7101 images belonging to 7 classes.
```

Model Building

```
from keras.optimizers import Adam,SGD,RMSprop


no_of_classes = 7

model = Sequential()

#1st CNN layer
model.add(Conv2D(64,(3,3),padding = 'same',input_shape =
(48,48,1)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Dropout(0.25))

#2nd CNN layer
model.add(Conv2D(128,(5,5),padding = 'same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Dropout (0.25))

#3rd CNN layer
model.add(Conv2D(512,(3,3),padding = 'same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Dropout (0.25))
```

```
#4th CNN layer
model.add(Conv2D(512,(3,3), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())

#Fully connected 1st layer
model.add(Dense(256))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))


# Fully connected layer 2nd layer
model.add(Dense(512))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))

model.add(Dense(no_of_classes, activation='softmax'))



opt = Adam(learning_rate = 0.0001)
model.compile(optimizer=opt,loss='categorical_crossentropy',
metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 48, 48, 64)        640

 batch_normalization_6 (Bat  (None, 48, 48, 64)        256
 chNormalization)

 activation_6 (Activation)   (None, 48, 48, 64)        0

 max_pooling2d_4 (MaxPoolin  (None, 24, 24, 64)        0
 g2D)

 dropout_6 (Dropout)         (None, 24, 24, 64)        0

 conv2d_5 (Conv2D)           (None, 24, 24, 128)       204928

 batch_normalization_7 (Bat  (None, 24, 24, 128)       512
 chNormalization)
```

```
activation_7 (Activation)   (None, 24, 24, 128)       0

max_pooling2d_5 (MaxPoolin  (None, 12, 12, 128)       0
g2D)

dropout_7 (Dropout)         (None, 12, 12, 128)       0

conv2d_6 (Conv2D)           (None, 12, 12, 512)       590336

batch_normalization_8 (Bat  (None, 12, 12, 512)       2048
chNormalization)

activation_8 (Activation)   (None, 12, 12, 512)       0

max_pooling2d_6 (MaxPoolin  (None, 6, 6, 512)         0
g2D)

dropout_8 (Dropout)         (None, 6, 6, 512)         0

conv2d_7 (Conv2D)           (None, 6, 6, 512)         2359808

batch_normalization_9 (Bat  (None, 6, 6, 512)         2048
chNormalization)

activation_9 (Activation)   (None, 6, 6, 512)         0

max_pooling2d_7 (MaxPoolin  (None, 3, 3, 512)         0
g2D)

dropout_9 (Dropout)         (None, 3, 3, 512)         0

flatten_1 (Flatten)         (None, 4608)              0

dense_3 (Dense)             (None, 256)               1179904

batch_normalization_10 (Ba  (None, 256)               1024
tchNormalization)

activation_10 (Activation)  (None, 256)               0

dropout_10 (Dropout)        (None, 256)               0

dense_4 (Dense)             (None, 512)               131584

batch_normalization_11 (Ba  (None, 512)               2048
tchNormalization)

activation_11 (Activation)  (None, 512)               0

dropout_11 (Dropout)        (None, 512)               0

dense_5 (Dense)             (None, 7)                 3591
```

```
=================================================================
Total params: 4478727 (17.08 MB)
Trainable params: 4474759 (17.07 MB)
Non-trainable params: 3968 (15.50 KB)
_________________________________________________________________
```

Fitting the Model with Training and Validation Data

```
from keras.optimizers import RMSprop,SGD,Adam
from keras.callbacks import ModelCheckpoint, EarlyStopping,
ReduceLROnPlateau

checkpoint = ModelCheckpoint("./model.keras", monitor='val_acc',
verbose=1, mode='max')

early_stopping = EarlyStopping(monitor='val_loss',
                          min_delta=0,
                          patience=3,
                          verbose=1,
                          restore_best_weights=True
                          )

reduce_learningrate = ReduceLROnPlateau(monitor='val_loss',
                              factor=0.2,
                              patience=3,
                              verbose=1,
                              min_delta=0.0001)

callbacks_list = [early_stopping,checkpoint,reduce_learningrate]

epochs = 48

model.compile(loss='categorical_crossentropy',
              optimizer = Adam(learning_rate=0.001),
              metrics=['accuracy'])

# history = model.fit(train_set,
#
steps_per_epoch=train_set.n//train_set.batch_size,
#                       epochs=epochs,
#                       validation_data = test_set,
#                       validation_steps =
test_set.n//test_set.batch_size,
#                       callbacks=callbacks_list
#                       )
```

Plotting Accuracy & Loss

```
plt.style.use("classic")
plt.figure(figsize=(20,10))
plt.subplot(1, 2, 1)
plt.suptitle('Optimizer : Adam', fontsize=10)
plt.ylabel('Loss', fontsize=16)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')

plt.subplot(1, 2, 2)
plt.ylabel('Accuracy', fontsize=16)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation
Accuracy')
plt.legend(loc='lower right')
plt.show()



from tensorflow.keras.models import load_model

# Load the model from the file
model = load_model('model.h5')

model.summary()



results = model.predict(test_set)

56/56 [==============================] - 14s 254ms/step

y_pred = np.argmax(results, axis=1)
true_labels = test_set.classes
g_dict = test_set.class_indices
classes = list(g_dict.keys())

print(classification_report(true_labels, y_pred,  target_names=
classes))

import matplotlib.pyplot as plt
import numpy as np

# Get test images and their corresponding labels
test_images = test_set[0][0]
test_labels = test_set[0][1]
predicted_labels = model.predict(test_images)
predicted_classes = np.argmax(predicted_labels, axis=1)

# Define the class labels
class_labels = ['angry', 'disgusted', 'fearful', 'happy',
'neutral', 'sad', 'surprised']

# Select 14 random images to visualize
num_images = 14
random_indices = np.random.choice(len(test_images),
size=num_images, replace=False)

# Plot the test images with their predicted classes
```

```
fig, axes = plt.subplots(2, 7, figsize=(15, 6))

for i, index in enumerate(random_indices):
    img = test_images[index]
    label = class_labels[np.argmax(test_labels[index])]
    predicted_label = class_labels[predicted_classes[index]]

    row = i // 7
    col = i % 7

    axes[row, col].imshow(img)
    axes[row, col].set_title(f'True: {label}\nPredicted:
{predicted_label}')
    axes[row, col].axis('off')

plt.tight_layout()
plt.show()

4/4 [==============================] - 0s 88ms/step
```

1. **Libraries:** The project utilizes the following libraries:
    - TensorFlow: Deep learning framework for building and training the CNN model.
    - Keras: High-level API built on top of TensorFlow for easier model development.
    - NumPy: Used for numerical computations and image manipulation.
    - Matplotlib: Used for plotting training and validation curves.
    - Scikit-learn: Provides functions for model evaluation metrics.

2. **Data Preparation:**
    - Data Loading: Functions are defined to load facial expression images from a designated folder.
    - Data Preprocessing: ImageDataGenerator from TensorFlow is used for resizing images to a consistent size and data augmentation techniques like random flipping and scaling to increase training data variety and prevent overfitting.
    - Data Splitting: The loaded data is split into training and validation sets using a specific ratio (e.g., 80% for training, 20% for validation).
    - Batching: Training and validation data are divided into batches for efficient model training.

3. **Model Building:** The CNN architecture is defined as a sequential model in Keras:
    - Convolutional Layers: Multiple convolutional layers are used with activation functions (e.g., ReLU) to extract features from the images.
    - Pooling Layers: Max pooling layers are added to reduce the dimensionality of the data and improve generalization.
    - Dropout Layers: Dropout layers are used to randomly drop units during training, further preventing overfitting.
    - Fully Connected Layers: Dense layers are added at the end to perform classification. The number of neurons in the final layer corresponds to the number of facial expressions to be recognized.

4. **Model Training and Evaluation:**
    - Optimizer and Loss Function: Adam optimizer is used for efficient training, and categorical cross-entropy loss function is used due to the multi-class classification nature of the problem.
    - Model Compilation: The model is compiled with the chosen optimizer, loss function, and metrics (e.g., accuracy).
    - Callbacks: Callbacks are defined to monitor the training process and improve model performance:
        - ModelCheckpoint: Saves the best performing model based on validation accuracy.
        - EarlyStopping: Stops training if validation accuracy doesn't improve for a certain number of epochs.
        - ReduceLROnPlateau: Reduces learning rate if validation loss stops improving.
    - Training: The model is trained on the prepared data batches using the compiled configuration.
    - Evaluation: The trained model is evaluated on the validation set to assess its performance using metrics like accuracy and loss.

5. **Model Prediction and Visualization:**
    - Loading Saved Model: The pre-trained model saved during training is loaded for further use.

- Prediction: The model is used to predict labels (facial expressions) for new, unseen test data.
- Evaluation Metrics: The classification_report function is used to evaluate the model's performance on the test data, providing precision, recall, F1-score, and support for each facial expression class.
- Visualization: Functions are defined to visualize test images along with their predicted labels.

Chapter 7 Model Deployement

CHAPTER 7
MODEL DEPLOYMENT

7.1 MODEL DEPLOYMENT CODE

```
import tkinter as tk
from tkinter import filedialog, messagebox
import cv2
import numpy as np
from PIL import Image, ImageTk
from tensorflow.keras.models import load_model

# Load your CNN model
model = load_model("model.h5")  # Replace "model.h5" with your model file path

# Function to get prediction from the CNN model for an image file
def get_prediction_from_file(file_path):
    frame = cv2.imread(file_path)
    if frame is not None:
        faces = faceCascade.detectMultiScale(
            cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY),
            scaleFactor=1.2,
            minNeighbors=6,
            minSize=(48, 48),
            flags=cv2.CASCADE_SCALE_IMAGE
        )
        for (x, y, w, h) in faces:
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
            cropped_face = frame[y:y + h, x:x + w]
            pred = get_prediction(cropped_face)
            cv2.putText(frame, str(pred), (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9,
(0, 255, 0), 2)
        return frame
    else:
        messagebox.showerror("Error", "Failed to load image.")
        return None

# Function to get prediction from the CNN model for webcam feed
def get_prediction_from_webcam():
    while True:
        ret, frame = video_capture.read()
        if ret:
            frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)  # Convert BGR to RGB
            faces = faceCascade.detectMultiScale(
                cv2.cvtColor(frame, cv2.COLOR_RGB2GRAY),
                scaleFactor=1.2,
                minNeighbors=6,
                minSize=(48, 48),
                flags=cv2.CASCADE_SCALE_IMAGE
            )
            for (x, y, w, h) in faces:
                cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
                cropped_face = frame[y:y + h, x:x + w]
                pred = get_prediction(cropped_face)
                cv2.putText(frame, str(pred), (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX,
0.9, (0, 255, 0), 2)
            display_prediction(frame)
            window.update()  # Update the Tkinter window to show the new frame
        else:
            messagebox.showerror("Error", "Failed to capture video from camera.")
            break

# Function to display prediction result in the Tkinter window
```

```
def display_prediction(frame):
    if frame is not None:
        # Convert frame to ImageTk format and display in the GUI
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        img = Image.fromarray(frame)
        img = ImageTk.PhotoImage(image=img)
        video_label.img = img
        video_label.config(image=img)
        video_label.image = img  # Keep reference to avoid garbage collection
    else:
        messagebox.showerror("Error", "Failed to display prediction.")

# Function to perform prediction based on user choice
def perform_prediction():
    if option.get() == "Image from Computer":
        # Release the video capture object if webcam feed is active
        if video_capture.isOpened():
            video_capture.release()
        file_path = filedialog.askopenfilename()
        if file_path:
            frame = get_prediction_from_file(file_path)
            display_prediction(frame)
    elif option.get() == "Webcam":
        frame = get_prediction_from_webcam()
        display_prediction(frame)

# Create the Tkinter window
window = tk.Tk()
window.title("Emotion Prediction")

# Set the minimum size of the window
window.minsize(400, 300)
```

```
# Function to handle window resize events
def on_resize(event):
    # Get the new size of the window
    new_width = event.width
    new_height = event.height

# Bind the resize event to the window
window.bind("<Configure>", on_resize)

# Create a label to prompt the user to select an option
option_label = tk.Label(window, text="Select an option:")
option_label.pack()

# Create a variable to store the selected option
option = tk.StringVar()
option.set(None)
# Create radio buttons for user selection
radio_button1 = tk.Radiobutton(window, text="Image from Computer", variable=option,
value="Image from Computer")
radio_button1.pack(anchor=tk.W)
radio_button2 = tk.Radiobutton(window, text="Webcam", variable=option,
value="Webcam")
radio_button2.pack(anchor=tk.W)

# Create a button to perform prediction based on user choice
predict_button = tk.Button(window, text="Predict", command=perform_prediction)
predict_button.pack()

# Create a label to display the prediction result
video_label = tk.Label(window)
video_label.pack()

# Load the Haar cascade classifier for face detection
```

```
faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades +
"haarcascade_frontalface_default.xml")

# Initialize the video capture object
video_capture = cv2.VideoCapture(0)

# Run the Tkinter event loop
window.mainloop()

# Release the video capture object
video_capture.release()
```

7.2 MODEL DEPLOYMENT EXPLANATION

Libraries:

- tkinter: Used for creating the graphical user interface (GUI).
- filedialog: Provides functions for opening file selection dialogs.
- messagebox: Used for displaying pop-up messages.
- cv2: OpenCV library for image processing and video capture.
- numpy: Used for numerical computations.
- PIL: Python Imaging Library for image manipulation.
- tensorflow.keras.models: Used to load the pre-trained CNN model.

Model Loading:

The code loads the pre-trained model saved during training using load_model. The path to the model file ("model.h5") needs to be replaced with the actual location.

Prediction Functions:

get_prediction_from_file(file_path):

Takes an image file path as input. Reads the image using OpenCV. Detects faces using a Haar cascade classifier. For each detected face: Crops the face region from the image.

Calls the get_prediction function (not shown in the provided code) to predict the facial expression using the loaded model. Draws a rectangle around the detected face and displays the predicted emotion on the image. Returns the processed image with predictions.

get_prediction_from_webcam():

Captures video frames from the webcam using OpenCV. Converts each frame from BGR (OpenCV color format) to RGB (used by PIL). Uses the same face detection and prediction logic as get_prediction_from_file for each frame. Continuously displays the processed video frames with predictions in the GUI window.

GUI Implementation with Tkinter:

The code creates a Tkinter window with the title "Emotion Prediction." It defines functions for handling window resizing events and displaying prediction results. Labels and radio buttons are created to allow users to choose between processing an image or using the webcam. A "Predict" button triggers the perform_prediction function based on the user's selection. A label is used to display the processed image or video frame with predicted emotions.

Face Detection:

The code utilizes a pre-trained Haar cascade classifier (cv2.CascadeClassifier) to detect faces in images or video frames.

Video Capture:

OpenCV's VideoCapture object is used to initialize and capture video frames from the webcam.

Main Loop:

The window.mainloop() function starts the Tkinter event loop, waiting for user interactions like button clicks. After the program exits, the video capture object is released using video_capture.release().

Chapter 8
Model Screenshot

CHAPTER 8
MODEL SCREENSHOT

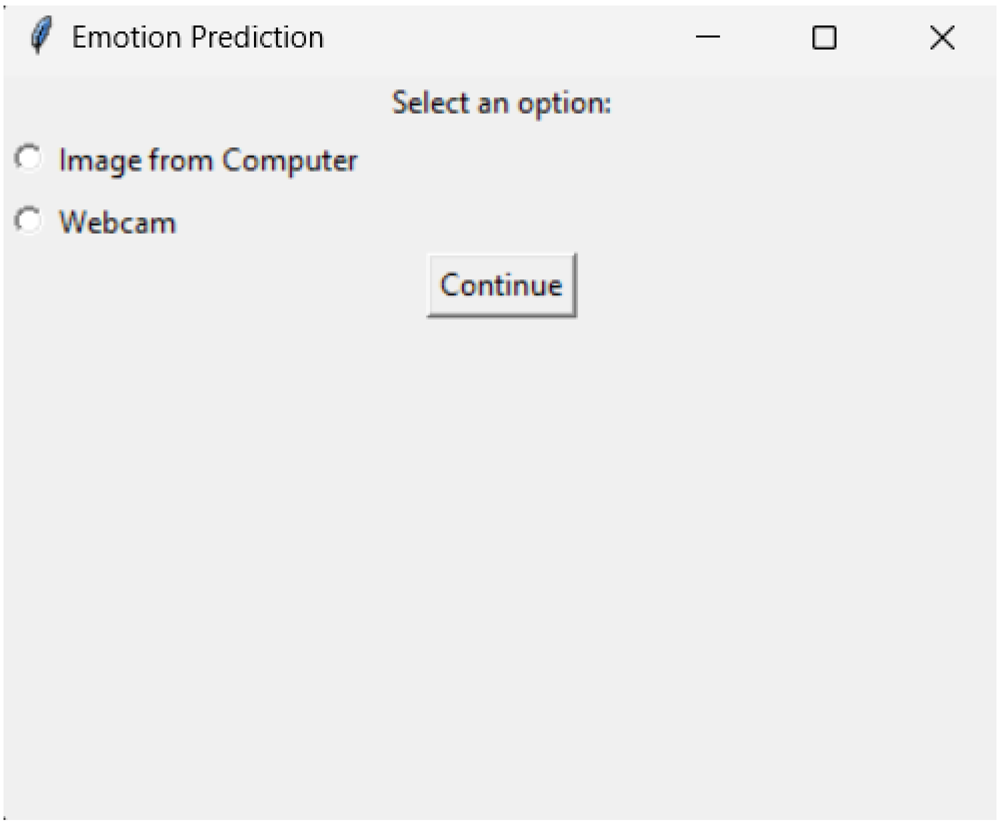


Fig 8.1 GUI Interface

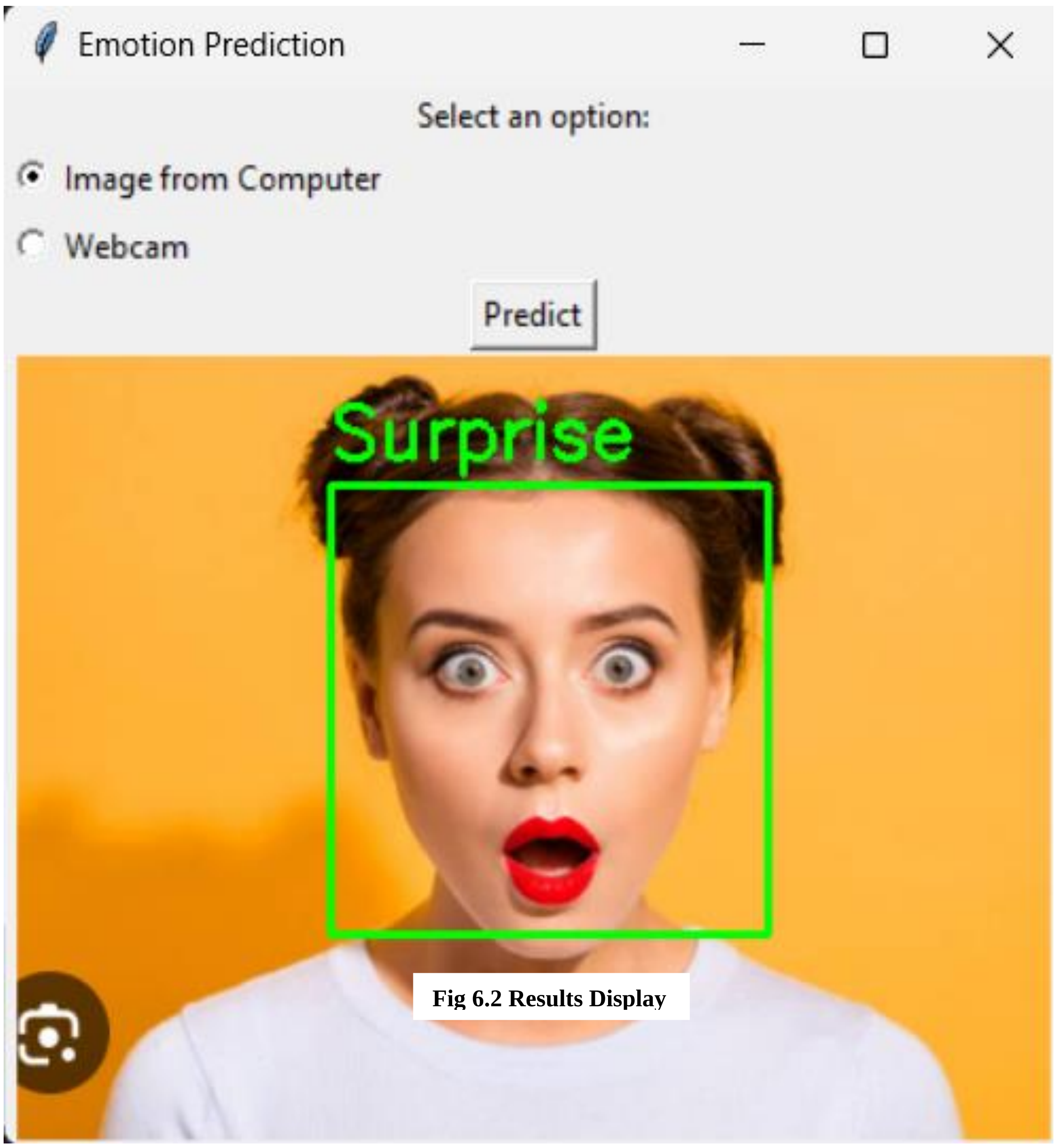


Fig 8.1 Result Display

REFERENCES

JOURNALS / RESEARCH PAPERS

1. A. Jaiswal, A. Krishnama Raju and S. Deb, "Facial Emotion Detection Using Deep Learning," *2020 International Conference for Emerging Technology (INCET)*, Belgaum, India, 2020, pp. 1-5, doi: 10.1109/INCET49848.2020.9154121.
2. Mr. Rohan Appasaheb Borgalli and Dr. Sunil Surve 2022 *J. Phys.: Conf. Ser.* **2236** 012004**DOI** 10.1088/1742-6596/2236/1/012004.
3. Giannopoulos, P., Perikos, I., Hatzilygeroudis, I. (2018). Deep Learning Approaches for Facial Emotion Recognition: A Case Study on FER-2013. In: Hatzilygeroudis, I., Palade, V. (eds) Advances in Hybridization of Intelligent Methods. Smart Innovation, Systems and Technologies, vol 85. Springer, Cham. https://doi.org/10.1007/978-3-319-66790-4_1.

WEBSITES (with exact URL up to page)

4. https://towardsdatascience.com/face-detection-models-which-to-use-and-why-d263e82c302c
5. https://github.com/opencv/opencv/tree/master/data/haarcascades
9. https://www.kaggle.com/datasets/msambare/fer2013

PROJECT SUMMARY

About Project

Title of the project	Design and Development of Emotion Detection System
Semester	VI
Members	Sparsh Sahu Mahak Mirza Eashan Tiwari
Team Leader	Mahak Mirza
Describe role of every member in the project	Sparsh Sahu created the GUI and processed the data, while Mahak Mirza & Eashan Tiwari collaborated on designing, and training the model.
What is the motivation for selecting this project?	The motivation for selecting the emotion detection project includes its relevance in various domains, opportunities for innovation and skill development, interest in human behavior and technology, social impact, and alignment with academic and career goals.
Project Type **(Desktop Application, Web Application, Mobile App, Web)**	Desktop Application

Tools &Technologies

Programming language used	Python
Compiler used **(with version)**	Python 3.11.5
IDE used **(with version)**	VS Code v18.18.2
Front End Technologies **(With version, wherever Applicable)**	Tkinter Version- 8.6
Back End Technologies **(With version, wherever applicable)**	***Not Applicable***

Database used	***Not Applicable***

Software Design& Coding

Is prototype of the software developed?	No
SDLC model followed **(Waterfall, Agile, Spiral etc.)**	Agile Model
Why above SDLC model is followed?	The Agile model is likely chosen for flexibility, continuous feedback, iterative development, risk mitigation, user involvement, and complexity management in the snake recognition project.
Justify that the SDLC model mentioned above is followed in the project.	The Agile SDLC model is justified for the Emotion Detection System due to its adaptability to changing requirements, continuous feedback, iterative development, and user-centric approach, essential for complex technology implementation.
Software Design approach followed **(Functional or Object-Oriented)**	Object Oriented
Name the diagrams developed **(According to the Design approach followed)**	Dataflow diagram
In case Object Oriented approach is followed, which of the OOPS principles are covered in design?	The Object-Oriented design approach in our project adheres to OOP principles: SRP, OCP, LSP, ISP, and DIP, ensuring modularity and flexibility.
No. of Tiers **(example 3-tier)**	2-Tiers (application tier & presentation tier)
Total no. of front end pages	1
Total no. of tables in database	***Not Applicable***
Database is in which Normal Form?	***Not Applicable***
Are the entries in database encrypted?	***Not Applicable***
Front end validations applied **(Yes / No)**	No

Session management done **(in case of web applications)**	***Not Applicable***
Is application browser compatible **(in case of web applications)**	***Not Applicable***
Exception handling done **(Yes / No)**	Yes
Commenting done in code **(Yes / No)**	Yes
Naming convention followed **(Yes / No)**	Yes
What difficulties faced during deployment of project?	Difficulties faced during the deployment of the emotion detection project include hardware compatibility, real-time performance optimization, managing model size and complexity, integration with existing systems, and comprehensive documentation and training requirements.
Total no. of Use-cases	10
Give titles of Use-cases	1. Real-Time Emotion Detection 2. Image Emotion Analysis 3. User Interaction and Feedback 4. Emotion-Based Content Recommendation 5. Emotion Monitoring and Reporting 6. Emotion-Based Marketing and Advertising 7. Emotion Recognition in Video Content 8. Emotion-Enhanced User Experience 9. Emotion-Based Healthcare Applications 10. Security and Access Control

Project Requirements

MVC architecture followed **(Yes / No)**	**No**
If yes, write the name of MVC architecture followed **(MVC-1, MVC-2)**	***Not Applicable***
Design Pattern used **(Yes / No)**	Yes
If yes, write the name of Design Pattern used	MTV (Model-Template View) GU
Interface type **(CLI / GUI)**	GUI
No. of Actors	1

Name of Actors	User
Total no. of Functional Requirements	5
List few important non-Functional Requirements	Usability Performance Accuracy Availability

Testing

Which testing is performed? **(Manual or Automation)**	Manual
Is Beta testing done for this project?	No

Write project narrative covering above mentioned points

The "Design and Development of Emotion Detection System" project, led by Team Leader Mahak Mirza and team members Sparsh Sahu and Eashan Tiwari, encompasses various key aspects. Sparsh Sahu contributed significantly by creating the GUI and handling data processing tasks, while Mahak Mirza and Eashan Tiwari collaborated on designing and training the model. The project's motivation stemmed from its relevance across diverse domains, offering opportunities for innovation, skill development, and aligning with academic and career goals. Utilizing Python programming language, specifically Python 3.11.5 in VS Code v18.18.2, and Tkinter Version- 8.6 for the GUI, the team adopted an Agile SDLC model for its flexibility, continuous feedback, and iterative development approach. The Object-Oriented design approach was followed, covering OOP principles like SRP, OCP, LSP, ISP, and DIP, ensuring modularity and flexibility. The system, being a desktop application, implemented the MTV (Model-Template View) GUI design pattern and focused on functional requirements such as usability, performance, accuracy, and availability. Despite challenges during deployment, including hardware compatibility and real-time performance optimization, the project successfully addressed ten use cases ranging from real-time emotion detection to security and access control. Manual testing was conducted, with no beta testing performed. Overall, the project narrative reflects a comprehensive journey in creating an advanced Emotion Detection System with a user-centric and robust design.

Sparsh Sahu
0187AD211039

Guide Signature
Dr. Vasima Khan
(Head of Department –
CSE-Artificial Intelligence & Data Science)

Mahak Mirza
0187AD211020

Eashan Tiwari
0187AD211015

APPENDIX-1 GLOSSARY OF TERMS

A

Accuracy	The proportion of correctly classified emotions in the test set.
Agile	An iterative software development methodology that emphasizes flexibility, continuous feedback, and user involvement.

C

CNN	Convolutional Neural Network: A type of deep learning architecture commonly used for image classification and feature extraction from visual data (like facial expressions).

D

Data Augmentation	(for images): Techniques like random cropping, flipping, and rotation to artificially increase the size and diversity of a dataset.
Data Preprocessing	Cleaning, formatting, and transforming raw data to prepare it for training a deep learning model.
DFD	Dataflow Diagram (DFD): A graphical representation of the flow of data through a system, showing the processes, data stores, and data flows involved.
DL	Deep Learning: A subfield of machine learning concerned with artificial neural networks with multiple hidden layers.

F

F1-Score	A harmonic mean of precision and recall, combining both metrics into a single measure.

L

Loss Function	A function that measures the difference between the model's predictions and the true labels during training.

M

MTV	Model-Template-View (MTV) GUI Pattern: A design pattern for web applications where the Model handles data, the Template defines the presentation, and the View interacts with the user.

O

OOP	Object-Oriented Programming: A programming paradigm that uses objects (data and methods) to structure code and promote modularity.
Optimizer	An algorithm that updates the weights of the deep learning model to minimize the loss function and improve its performance.

P

Precision	The proportion of true positives (correctly identified emotions) out of all positive predictions.

R

Recall	The proportion of true positives out of all actual occurrences of the emotion in the data.