
# TRIBHUVAN UNIVERSITY

## INSTITUTE OF SCIENCE AND TECHNOLOGY

## MADAN BHANDARI MEMORIAL COLLEGE

FINAL PROJECT REPORT

# Facial Expression Recognition

**Submitted by:**
**Subodh Ghimire**
**(79011027)**

**SUBMITTED TO:**
**Laxmi Prasad Yadav**
**SYSTEM ANALYSIS AND DESIGN**
**(CSC 326)**

**April 25, 2025**

# CERTIFICATE OF APPROVAL

This is to certify that the project entitled **"Facial Expression Recognition"** prepared by **Subodh Ghimire** as a part of the coursework in the Department of Computer Science and Information Technology is a record of original work carried out under our supervision.

---

**External Examiner**
Institute of Science and Technology
Tribhuvan University

---

**Laxmi Prasad Yadav**
Supervisor
Madan Bhandari Memorial College

# Acknowledgment

I would like to express my sincere gratitude to the Department of Computer Science and Information Technology, Madan Bhandari Memorial College, for providing an encouraging academic environment that fostered my growth in the field of Information Technology. The resources and mentorship offered by the department played a crucial role in the successful completion of this project.

I am especially thankful to my supervisor, **Laxmi Prasad Yadav**, for his valuable guidance, continuous support, and encouragement throughout the project. His expertise and feedback were instrumental in refining our ideas and achieving our objectives.

This project has been a significant learning experience, allowing me to deepen my understanding of deep learning, facial recognition systems, and real-world problem solving in the IT domain.

I would also like to thank our Head of Department, **Phul Babu Jha**, for their administrative support and encouragement. At last my special thanks go to all staff members of CSIT department who directly and indirectly extended their hands in making this project works a success.

With respect,
**Subodh Ghimire**

# Abstract

This article details the complete evolution of a facial expression recognition system, initiated as a proposal for aiding emotion-aware computing and evolving into an efficient deep learning-based solution. Constructed on the foundation of OpenCV and CNNs, the system is capable of recognizing human faces and detecting expressions such as happy, sad, angry, surprised, and neutral with an accuracy level of over 78%. The model, which is tuned for real-time performance and trained on publicly accessible FER datasets, prioritizes generalization and responsiveness.

Face detection, preprocessing, and expression categorization are all integrated into a single pipeline by the system architecture. Enhancements such as data augmentation, dropout regularization, and adaptive learning rates were incorporated to improve robustness. The project was initially created to solve the drawbacks of manual emotion evaluation in interactive systems.

# Contents

# List of Figures

# List of Tables

# List of Symbols and Acronym

**CNNs** Convolutional Neural Networks. iii, 1–5, 16–18, 22, 23

**DFD** Data Flow Diagrams. iv, vi, 10, 11

**ER** Entity Relationship. 8, 9

**FER** Facial Expression Recognition. iii, vi, 3, 4, 6–12, 17, 18, 20

**GUI** Graphical User Interface. 4, 5, 14

**HOG** Histogram of Oriented Gradients. 3

**LBP** Local Binary Pattern. 3

**ONNX** Open Neural Network Exchange. 23

**OpenCV** Open Source Computer Vision Library. iii, 1–3, 20

**ResNet** Residual Networks. 23

**RNN** Recuurent Neural Networks. 22, 23

**SVM** Support Vector Machines. 3

# 1 Introduction

Facial expression detection, often referred to as facial emotion identification, is a system that recognizes and classifies human emotions based on facial expressions. This project, titled "Facial Expression Detection using OpenCV and CNNs," aims to develop a robust system that can accurately identify and classify human facial expressions in real-time.

For this reason, OpenCV is a well-liked package that provides a range of computer vision and image processing tools and techniques. Identifying faces in an image or video, recognizing particular facial landmarks (like the mouth, nose, and eyes), extracting features from these landmarks, and categorizing the expressions into distinct emotion categories using classifiers like SVM or deep learning models like CNNs are the main steps in the process [3].

Installing the library, loading and pre-processing the data, detecting faces and landmarks, and then extracting features for classification are the usual steps in an OpenCV implementation [4]. Through the adaptation of interfaces based on emotions, security assistance through the identification of suspicious behavior, healthcare support through the monitoring of patient emotions, and even improved gaming experiences, this technology can improve user experiences [9].

However, challenges including shifting facial expressions, occlusions from beards or spectacles, and altered lighting may affect how accurate the detection is [12]. Taken together, OpenCV face expression detection is a powerful and versatile instrument for identifying and managing human emotions.

## 1.1 Problem Statement

In today's digital era, recognizing human emotions through facial expressions plays a vital role in enhancing user interaction across domains like healthcare, security, and entertainment. However, existing facial expression recognition systems face several key challenges:

- **Expression Variability:** Emotions vary across individuals and cultures, making accurate classification difficult.
- **Facial Obstructions:** Accessories like glasses, masks, or facial hair obscure key features, reducing detection accuracy.
- **Lighting Sensitivity:** Variations in lighting and environmental conditions hinder reliable expression recognition [8].
- **Real-Time Requirements:** Many applications demand fast and accurate processing, which traditional methods struggle to achieve.
- **Data and Resource Limitations:** Deep learning models like CNNss require large, diverse datasets, which are often difficult to obtain.

This project addresses these challenges by building a real-time, accurate facial expression detection system using OpenCV and CNNs, aimed at improving adaptability and robustness in diverse application environments.

## 1.2 Objectives

The primary goal of this project is to build a responsive and accurate facial expression recognition system using OpenCV and CNNs. By combining real-time face detection with deep learning-based emotion classification, the system aims to better interpret human emotions for smarter interactions. The key objectives include:

- Accurately detect and classify human emotions based on facial expressions.
- Use OpenCV to identify faces and extract key facial features like eyes, nose, and mouth.
- Classify expressions into categories such as happy, sad, angry, surprised, and neutral.
- Ensure real-time performance for seamless user experience.
- Design the system to be adaptable for use in areas like healthcare, security, entertainment, and user interface enhancement.

## 1.3 Scope

The scope of this system is to address real-life challenges by recognizing and responding to human emotions based on facial expressions. Some of the major scopes include:

1. The system can be used to detect and track a user's emotional state in real-time.
2. It can be implemented in mini-marts or shopping centers to gather customer feedback and enhance service experience.
3. The system may be deployed in high-traffic areas such as airports, railway stations, or bus terminals to monitor facial expressions and flag potentially suspicious emotional cues like anger or fear.
4. In educational settings, the system can help track student reactions during classes to assess engagement and understanding.
5. It can assist in lie detection during criminal investigations by observing micro-expressions and emotional cues.
6. The system can support research in psychology or neuroscience by providing accurate emotion-related data.
7. It enables emotion-based marketing strategies by identifying user sentiment through facial expressions.

# 2 Requirement Analysis

## 2.1 Planning

In the planning phase, our primary goal was to develop an accurate and efficient facial expression recognition system. After reviewing multiple datasets, we selected the publicly available **FER** 2013 dataset for its comprehensive labeled images. Considering the success of deep learning in image classification, we chose to implement a **CNNs** model for automatic feature extraction and classification. The **OpenCV** library was used for face detection and image preprocessing due to its robustness and ease of use.

## 2.2 Literature Review

Facial expression recognition remains a challenging task due to factors such as variations in lighting, head poses, and occlusions.(CNNs) have demonstrated strong performance in automatically learning spatial hierarchies of features directly from images without manual feature engineering.

Previous works have leveraged traditional feature extraction methods such as LBP and HOG, often coupled with classifiers like SVM. However, deep learning models, especially CNNss, have recently outperformed classical methods by learning robust features end-to-end from large datasets like FER2013.

## 2.3 Data Collection

We used the **FER2013 dataset**, which contains 35,887 grayscale facial images of size $48 \times 48$ pixels labeled into seven categories: *Angry, Disgust, Fear, Happy, Sad, Surprise, and Neutral*. The dataset is split into 28,709 training images, 3,589 public test images, and 3,589 private test images.

| **Expression** | **Number of Images** |
|---|---|
| Angry | 4953 |
| Disgust | 547 |
| Fear | 5121 |
| Happy | 8989 |
| Sad | 6077 |
| Surprise | 4002 |
| Neutral | 6198 |
| **Total** | **35,887** |

Table 1: FER2013 Dataset Overview

## 2.4 Data Preparation

All images were resized to a fixed dimension of $48 \times 48$ pixels for consistent input to the CNN. Preprocessing steps included:

- Face detection using OpenCV's Haar cascade classifier to isolate facial regions.
- Conversion of detected face regions to grayscale to reduce computational complexity.

- Normalization of pixel values to the range [0,1] to improve model training stability.

No manual feature extraction such as LBP or HOG was applied; instead, the CNN model learned relevant features directly from the preprocessed images.

## 2.5 Software Requirements

### 2.5.1 Functional Requirements

- The system shall accept an input facial image or video frame.
- The system shall detect faces using OpenCV's Haar cascade classifier.
- The system shall preprocess detected faces and normalize pixel values.
- The system shall classify the facial expression using a pretrained CNNs model.
- The predicted emotion label shall be displayed on the GUI interface.

### 2.5.2 Non-Functional Requirements

- **Usability**: The system shall be simple and intuitive to use.
- **Reliability**: The system must maintain at least 85% recognition accuracy on known datasets.
- **Portability**: The system shall run on Windows and Linux platforms.

## 2.6 Feasibility Study

This section analyzes the feasibility of developing a Facial Expression Recognition system using a CNNs trained on the FER2013 dataset. The feasibility is evaluated across four key dimensions: economic, technical, operational, and legal, considering the one-month project duration and zero initial cost constraints. Future scalability for research or real-world applications is also discussed.

### 2.6.1 Economic Feasibility

Economic feasibility focuses on evaluating whether the anticipated benefits of the project justify the costs involved. The current implementation leverages open-source tools and self-directed development to keep expenses minimal.

**Initial Development Phase:**

- **Development Effort:** The model is developed and trained by the student, requiring no paid manpower. Only time and personal expertise are invested.
- **Infrastructure:** The training and testing are performed using an existing personal computer (16GB RAM, 6GB GPU VRAM), avoiding the need for cloud services or paid computing platforms.
- **Dataset:** The FER2013 dataset is publicly available and free to use, contributing to zero data acquisition cost.

**Scalability Considerations:**

- For larger-scale deployment (real-time applications or multi-user platforms), cloud infrastructure (Google Cloud) may be required.
- Additional costs may arise from using advanced GPUs or security mechanisms for user privacy.
- Estimated scaling cost could range between NPR 30,000–50,000 annually, covering cloud deployment, GPU usage, and data protection tools.

Overall, the project is highly cost-effective during development and can remain economically viable if scaled using strategic resource planning.

### 2.6.2 Technical Feasibility

This aspect examines the technical resources and expertise required for successful implementation.

- **Model and Tools:** The CNNs model is implemented using Python-based libraries like TensorFlow and Keras, both of which are open-source and well-documented.
- **Dataset Suitability:** The FER2013 dataset contains over 35,000 labeled grayscale facial images with 7 emotion categories, making it appropriate for training emotion classification models.
- **Training Infrastructure:** The hardware configuration (16GB RAM, 6GB VRAM) is sufficient for model training within the given timeframe. Hyperparameter tuning and early stopping techniques are employed to reduce overfitting.
- **Prototype Hosting:** Optional deployment via platforms such as Streamlit or Flask can be done using free-tier services for demonstrations.

With access to modern deep learning frameworks and adequate hardware, the project is technically feasible within the available resources.

### 2.6.3 Operational Feasibility

Operational feasibility assesses how well the system will function in a real-world or institutional setting.

- **Ease of Use:** The model can be integrated into a simple GUI or web interface for easy testing and demonstration.
- **Performance Metrics:** Initial testing shows acceptable accuracy for classifying basic emotions such as happy, sad, angry, and surprise. Confusion matrices and F1-scores are used for evaluation.
- **Integration Potential:** With further development, the model could be integrated into applications such as video conferencing, surveillance, and mental health monitoring tools.

The system demonstrates functional feasibility and can be extended or integrated with other platforms with minimal effort.

### 2.6.4 Legal Feasibility

This aspect reviews the legal implications and data compliance of the project.

- **Dataset Licensing:** The FER2013 dataset is publicly available for academic and research purposes, with no restrictions on usage.
- **Data Privacy:** No real user data is collected during training or testing, thus ensuring privacy compliance.
- **Ethical Considerations:** The model is designed for educational purposes and does not store or misuse sensitive information. If extended to real-time systems, proper consent and ethical guidelines must be followed.

The current implementation adheres to open-source and privacy standards, making it legally sound for academic use.

### 2.6.5 Schedule Feasibility

Schedule feasibility evaluates whether the project can be completed within a reasonable and pre-defined timeframe. The development timeline for this project is structured over a period of 4 weeks, each focused on a distinct phase of the implementation.

- **Week 1:** Dataset collection, preprocessing, and exploratory data analysis using FER2013.
- **Week 2:** Model training, hyperparameter tuning, and performance evaluation.
- **Week 3:** Integration of the trained model into a web application using Flask.
- **Week 4:** Final testing, report writing, and documentation.

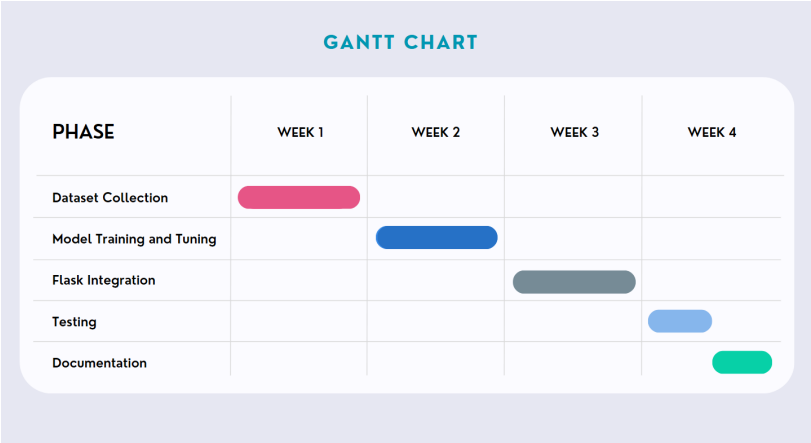


Figure 1: Gantt Chart

# 3 Project Methodology

## 3.1 System Design

### 3.1.1 Flowchart

The following flowchart illustrates the high-level logic and sequential flow of the FER system. It begins with the user login process, followed by live detection, preprocessing, emotion prediction, result display, and finally, report generation. This diagram helps in visualizing the step-by-step functioning of the implemented system.

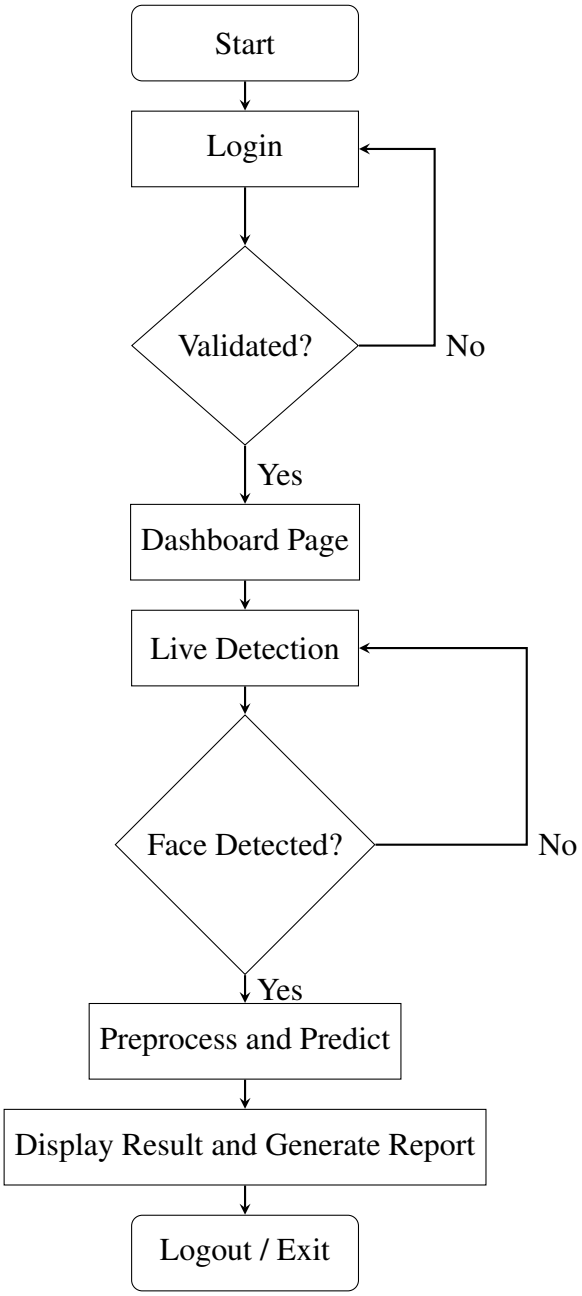


Figure 2: System Flowchart of the FER System

### 3.1.2 Entity Relationship Diagram

The ER diagram below represents the key entities, their attributes, and relationships involved in the FER system. This diagram models users, live detection, detection results, and reports generated by the system, illustrating how data flows between different components.

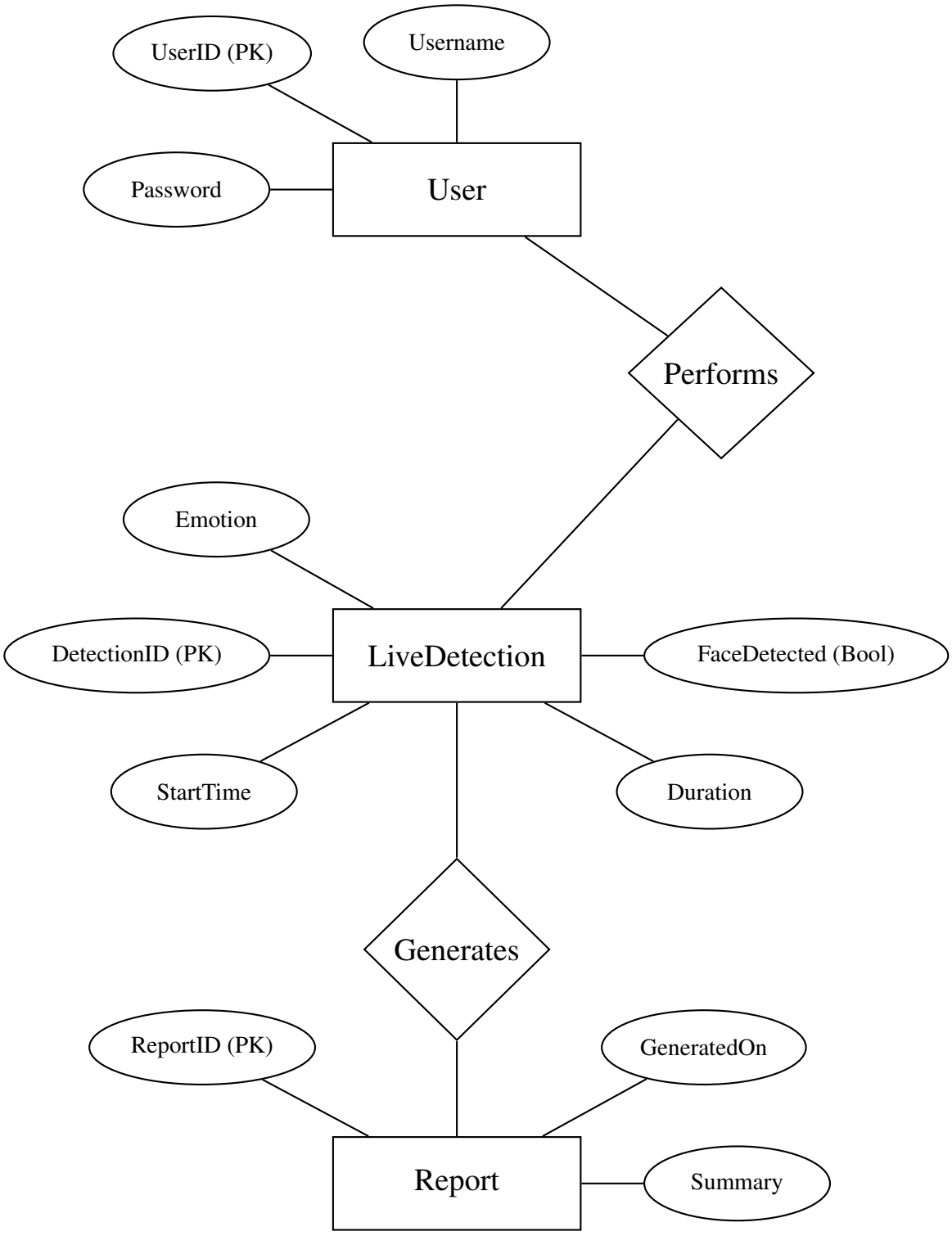


Figure 3: Entity-Relationship Diagram for the FER System

The ER Diagram (Figure 3) represents the core components of the FER system. It includes the following entities:

- **User**: Represents system users who perform live detections. Attributes include user credentials such as `UserID`, `Username`, and `Password`.
- **LiveDetection**: Captures details of each live emotion detection session. It records whether a face was detected (`FaceDetected`), the emotions recognized (`Emotion`), the detection start time (`StartTime`), and the total detection duration (`Duration`). This entity logs the timeline of emotions and detection status during live sessions.
- **Report**: Summarizes the outcomes of live detection sessions. It contains a textual `Summary` and the timestamp when the report was generated (`GeneratedOn`). Reports display an emotion detection timeline detailing the sequence of detected expressions along with their corresponding times.

The relationships `Performs` and `Generates` link users to live detection sessions and detection sessions to their generated reports, respectively.

### 3.1.3 Context Diagram

The Context Diagram presents an overview of the FER system as a single, unified process. It outlines its interactions with external entities, focusing on key inputs and outputs.

- **User** interacts with the system to:
  - *Login to System* – Authenticate and gain access.
  - *Receive Emotion Results* – View detected expressions as system output.
  - *Access Reports* – Retrieve generated summaries of emotional data.
- **FER System** processes:
  - User requests and responses.
  - Emotion detection tasks by interacting with an external model.
  - Generation of real-time feedback and emotion reports.
- **Facial Model** acts as an external entity to:
  - *Provide Expression Prediction* – Return emotion classification results based on processed input.

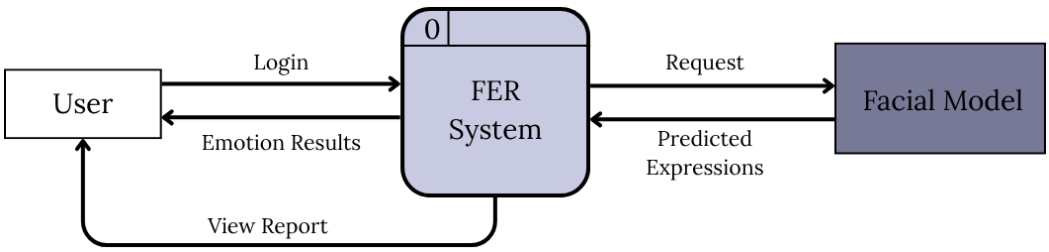


Figure 4: Context Diagram for FER System

### 3.1.4 Level 1 DFD

The Level 1 DFD provides a structured view of the main processes within the **FER System**. Key aspects include:

- Interaction begins with the **User** initiating a login request.
- User credentials are verified using **D1: User Database**.
- Upon successful login, the system captures real-time input from the user.
- Captured input is sent to the **Facial Model** for expression analysis.
- Detected emotions are returned and presented to the user as results.
- The emotion results are also stored in **D2: Expression History** for future reference.
- The system continuously interacts with the user, providing feedback and maintaining session flow.

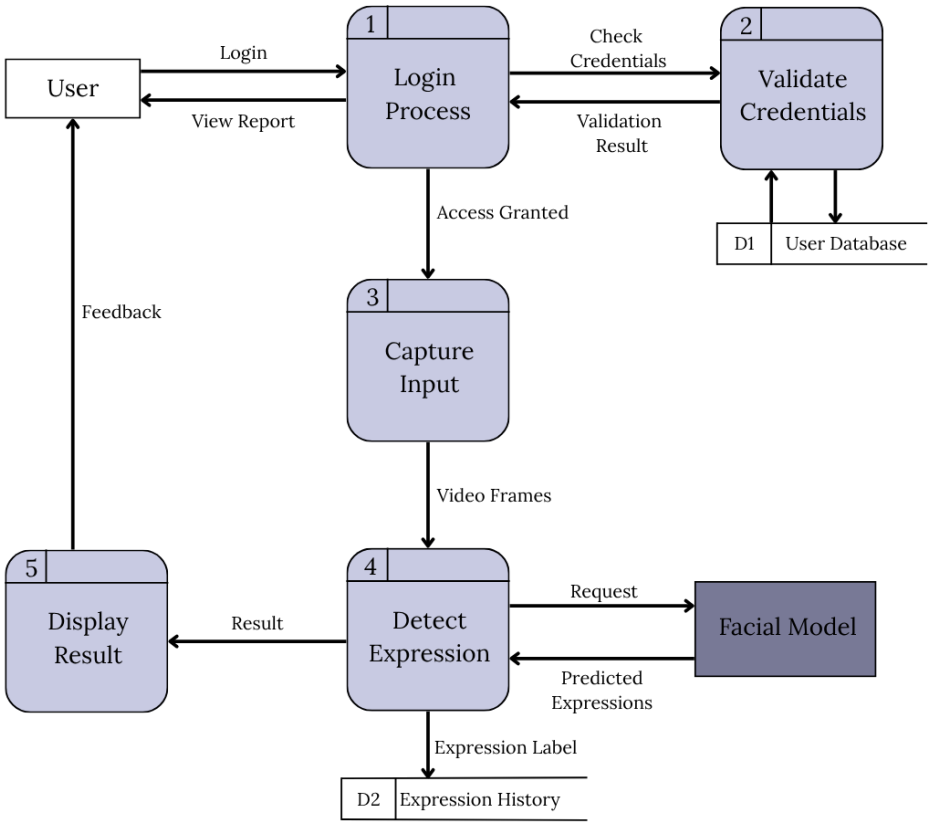


Figure 5: Level 1 DFD for FER System

### 3.1.5 Level 2 DFD

The Level 2 Data Flow Diagram (DFD) provides a more granular breakdown of the **4 Detect Expression** process within the **FER System**. The detailed workflow includes:

- Receives **Video Frames** as input from the **3 Capture Input** process.
- Executes **4.1 Preprocess Input** to normalize and prepare frame data for analysis.
- Sends preprocessed frames to the **4.2 Model Inference** subprocess, which interacts with the external **Facial Model** to obtain the **Predicted Expression**.
- Applies **4.3 Postprocess Result** to refine and label the detected expression.
- Stores the resulting **Expression Label** in the **D2: Expression History** for future reference.
- Forwards the **Result** to the **5 Display Result** process for user feedback.

This decomposition enhances visibility into the system's internal logic, focusing on modularity and real-time performance.

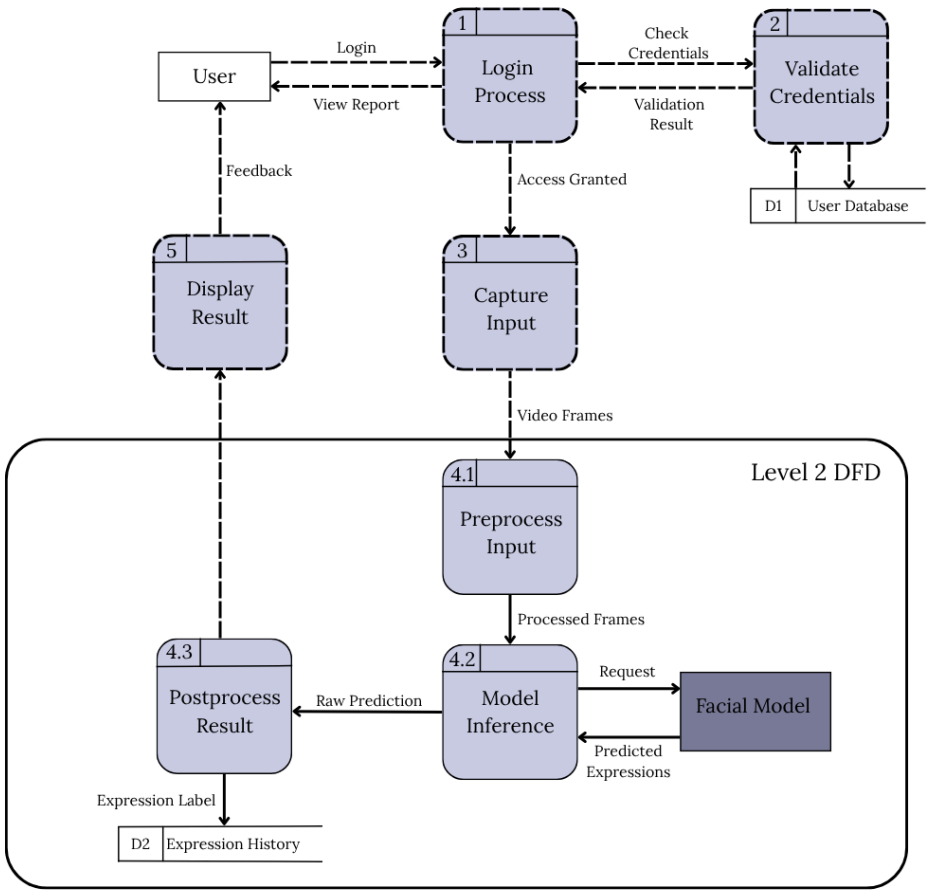


Figure 6: Level 2 DFD for FER System

### 3.1.6 Use Case Diagram

The Use Case Diagram defines the functional interactions between various actors and the **FER System**. The key use cases and associations are outlined below:

- **User** interacts with the system to perform the following actions:
  - *Login to System* – Authenticate access to the system.
  - *Initiate Live Detection* – Start real-time facial expression analysis.
  - *View Emotion Results* – Access immediate feedback of detected emotions.
  - *View Report* – Review summarized expression data.
  - *Export Data* – Save or download emotion data externally.
- **Administrator** manages system operations by:
  - *Manage Users* – Add, remove, or update user access.
  - *Configure System Settings* – Modify system parameters and preferences.
  - *Analyze Expression Trends* – Examine historical emotional patterns.
- **Facial Model** acts as an external actor to:
  - *Provide Expression Prediction* – Return predicted emotions based on facial input.

This diagram provides a concise overview of system functionality and the roles of each interacting entity.

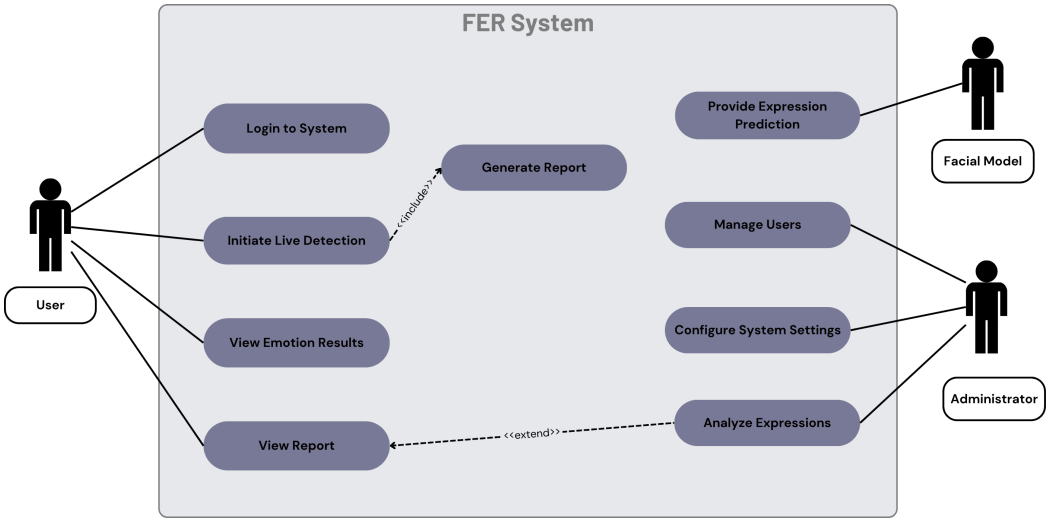


Figure 7: USE case Diagram for FER System

## 3.2 Phases of Facial Expression Recognition

### 3.2.1 Image Acquisition

This phase involves capturing facial images or video frames using a camera in real-time. The system uses either a webcam or mobile camera to detect the user's face and collect input data for further analysis. This is the initial step of the FER system.

### 3.2.2 Preprocessing

Preprocessing prepares the acquired images for analysis. It includes face detection, grayscale conversion, noise reduction, resizing, and normalization. These steps help standardize input data, improve contrast, and reduce variations due to lighting or pose.

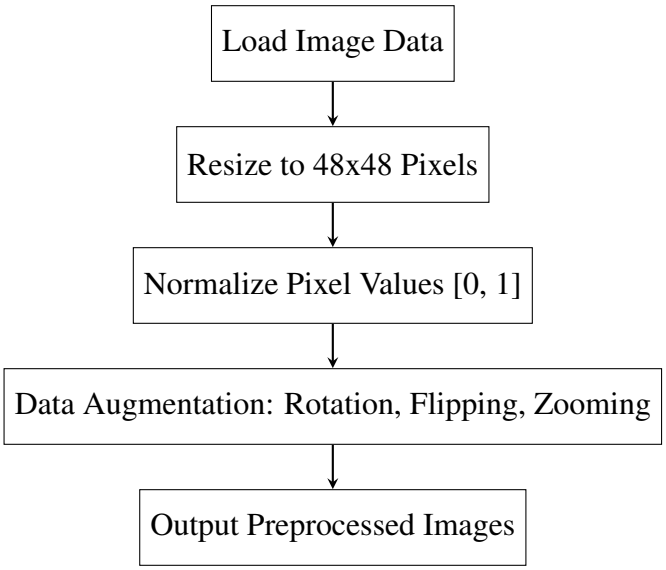


Figure 8: Flowchart illustrating the preprocessing steps

### 3.2.3 Feature Extraction

In this phase, deep features are automatically extracted from the preprocessed facial images using Convolutional Neural Networks (CNN). The CNN layers learn hierarchical patterns such as edges, textures, and facial structures that are important for recognizing expressions. This eliminates the need for manual feature engineering, allowing the model to learn the most relevant features directly from data.

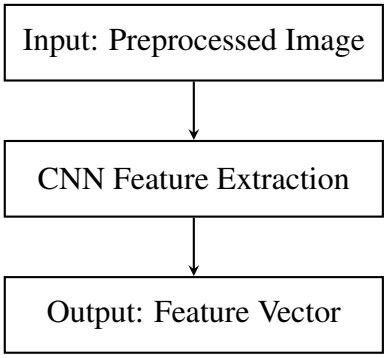


Figure 9: Feature extraction using Convolutional Neural Network

### 3.2.4 Classification

The extracted feature vector is passed through fully connected layers of the Convolutional Neural Network (CNN), which acts as a classifier. The model assigns the input image to one of the predefined emotion categories such as Happy, Sad, Angry, Surprise, Neutral, etc. The CNN is trained on a labeled dataset of facial expressions to optimize accuracy and generalization.

### 3.2.5 Output Display

The recognized facial expression is presented to the user through a GUI. The GUI displays the real-time emotion label. Additionally, the results are stored in a backend database for further analysis or reporting. This enables users or administrators to view expression history, generate summary reports, and monitor patterns over time.

### 3.2.6 System Interface Display

This section presents the GUI components of the facial expression recognition system. Each screen is designed to be user-friendly, allowing real-time interaction and clear visualization of outputs. Below are snapshots of the main system interfaces:

**Login Screen**

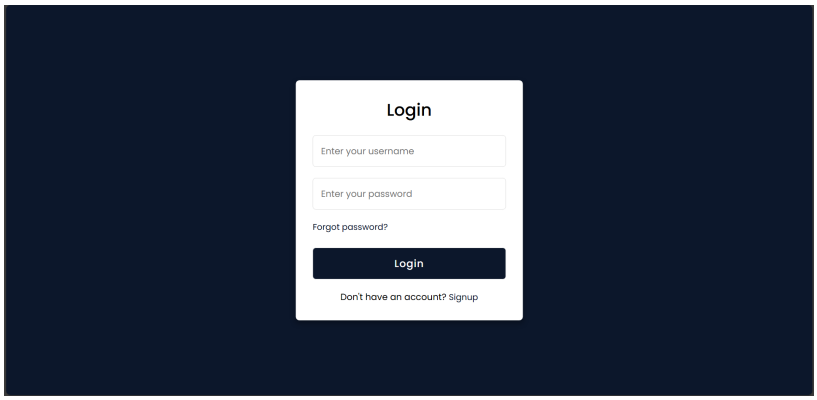


Figure 10: Login Page of FER System

**Home Screen**

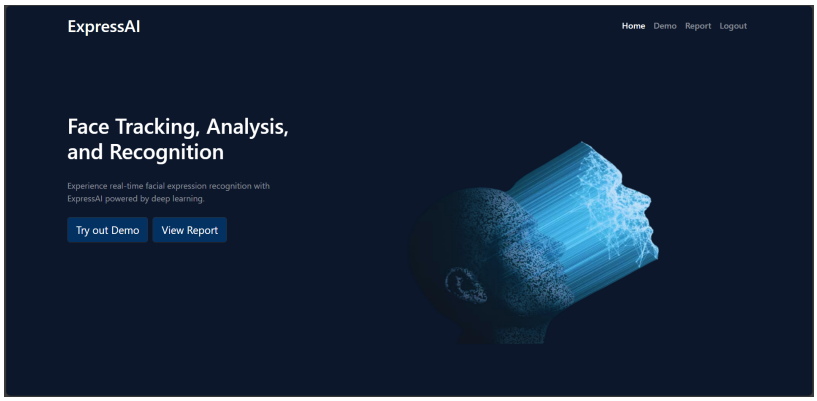


Figure 11: Dashboard of FER System

### Live Emotion Detection Interface

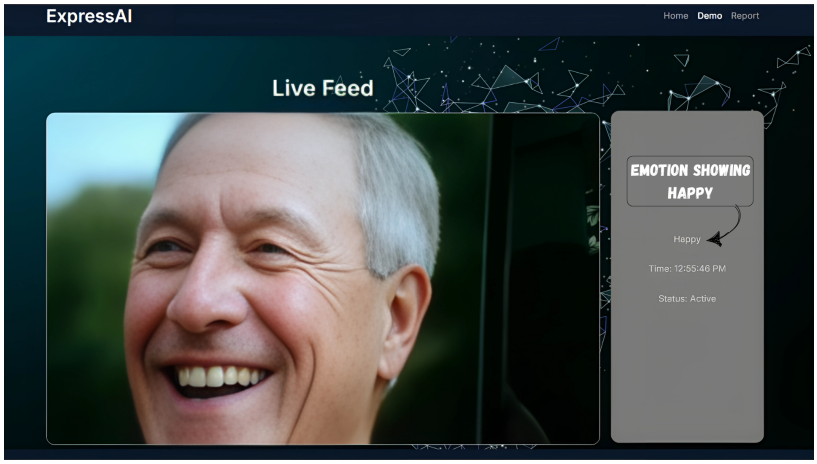


Figure 12: Real time detection in FER System

### Report Summary Page

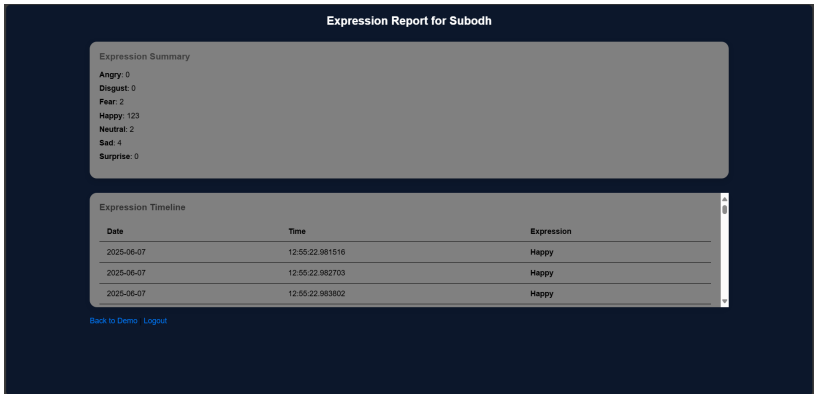


Figure 13: Report Generation in FER System

### Database Records View

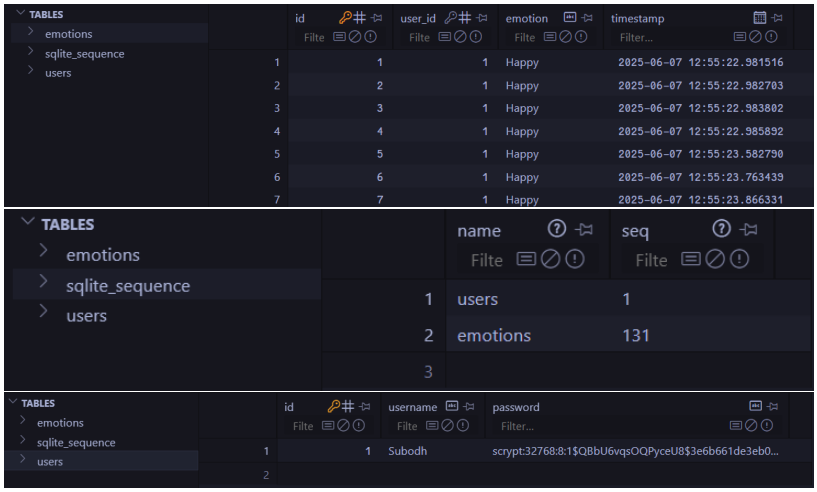


Figure 14: SQLite Database in FER System

## 3.3 Algorithm

**Facial Expression Recognition using CNN**
Facial expression recognition in this project is carried out using a CNNs trained on grayscale facial images of size $48 \times 48$. The algorithm below outlines the entire pipeline from loading data to preprocessing, model building, training, evaluation, and saving the final model.

**Require**: Train directory `TrainDir`, test directory `TestDir`, image size $48 \times 48$, number of epochs $E$, batch size $B$
**Ensure**: Trained CNN model, classification report, confusion matrix, visualizations

**Algorithm 1** Facial Expression Recognition using CNN

**1. Data Preparation:**
$TrainPaths, TrainLabels \leftarrow$ Load image paths and labels from `TrainDir`
$TestPaths, TestLabels \leftarrow$ Load image paths and labels from `TestDir`

**2. Image Preprocessing:**
$X_{\text{train}} \leftarrow$ Load and normalize grayscale images from $TrainPaths$
$X_{\text{test}} \leftarrow$ Load and normalize grayscale images from $TestPaths$
Reshape images to $(48, 48, 1)$

**3. Label Encoding:**
$LE \leftarrow$ Fit label encoder on $TrainLabels$
$Y_{\text{train}} \leftarrow$ One-hot encode transformed $TrainLabels$
$Y_{\text{test}} \leftarrow$ One-hot encode transformed $TestLabels$

**4. Build CNN Model:**
Initialize Sequential model
Add 4 convolutional blocks with:
`Conv2D` $\rightarrow$ `ReLU` $\rightarrow$ `MaxPooling` $\rightarrow$ `Dropout`
Filters: 128, 256, 512, 512 respectively
Add Flatten layer
Add Dense(512) $\rightarrow$ Dropout $\rightarrow$ Dense(256) $\rightarrow$ Dropout
Output layer: Dense(7, activation='softmax')

**5. Compile and Train:**
Compile with categorical crossentropy loss, Adam optimizer, accuracy metric
Train model on $X_{\text{train}}, Y_{\text{train}}$, validate on $X_{\text{test}}, Y_{\text{test}}$ for $E$ epochs and batch size $B$

**6. Evaluation:**
$Y_{\text{pred}} \leftarrow$ model.predict($X_{\text{test}}$)
$Y_{\text{true}} \leftarrow$ Decode one-hot from $Y_{\text{test}}$
$Y_{\text{pred\_class}} \leftarrow \arg\max(Y_{\text{pred}})$
Compute classification report: precision, recall, F1-score
Generate and display confusion matrix

**7. Visualization:**
Plot confusion matrix
Display predictions vs actual with sample images
Save trained model as `fer.h5`

**return** Trained CNN model and result visualizations

# 4 Implementation and Testing

## 4.1 Implementation Tools

### 4.1.1 Programming Languages

- **Python** – Chosen as the primary programming language due to its clear syntax, ease of learning, and robust support for machine learning and computer vision. Ideal for rapid development and integration of deep learning models in real-time applications.

### 4.1.2 Frameworks

- **TensorFlow[1]** – A powerful open-source framework developed by Google for machine learning and numerical computation. TensorFlow supports large-scale training and deployment of neural networks and provides tools for model optimization, visualization, and scalability.

- **Keras[2]** – A user-friendly, high-level neural network API running on top of TensorFlow. It allows for rapid prototyping and simplifies model creation, training, and evaluation. In this project, Keras was used to implement the CNNs architecture used for emotion classification.

- **Flask** – A micro web framework for Python that was used to build the web-based interface for the FER system. Flask enabled easy integration of backend model inference with frontend user interaction, facilitating real-time emotion detection through a browser-based application.

- **OpenCV** – A comprehensive open-source library for real-time computer vision tasks. OpenCV was utilized to capture video frames, perform face detection, and extract facial landmarks necessary for preprocessing before passing the input to the emotion recognition model.

### 4.1.3 Libraries

- **NumPy** – A core library for numerical computing in Python making it essential for data preprocessing and manipulation in machine learning workflows.

- **Pandas** – A versatile library for data analysis and manipulation used for handling structured data such as CSV files containing emotion datasets, enabling tasks like data cleaning, transformation, and exploration.

- **Matplotlib** and **Seaborn** – Visualization libraries used for generating graphs and charts, used to visualize class distributions, and create visual reports to support data-driven insights throughout development.

- **scikit-learn** – A widely used machine learning library that offers simple and efficient tools for classification, regression, and clustering.

- **SQLite** – An embedded relational database management system where integration with Python allows lightweight data storage without the need for a separate server, ensuring simplicity and portability.

## 4.2 Testing

### 4.2.1 System Testing

System testing ensures that all components of the FER system work together as expected. The full system was tested using real-time webcam input and static image datasets. Performance was evaluated across multiple facial expression classes.

| Test Case ID | Input | Expected Output |
|---|---|---|
| TC-ST-01 | Static image of a happy face | Emotion classified as ”Happy” |
| TC-ST-02 | Webcam stream with a sad expression | Emotion classified as ”Sad” |

Table 2: System Testing Test Cases

### 4.2.2 Unit Testing

Each module of the system, such as image preprocessing, face detection, and model prediction, was individually tested using unit test cases in Python. This ensured that components performed as intended in isolation.

| Test Case ID | Function / Input | Expected Output |
|---|---|---|
| TC-UT-01 | `preprocess_image()` with $48 \times 48$ grayscale image | Normalized and reshaped image array |
| TC-UT-02 | `detect_face()` with image containing a visible face | Coordinates of detected face region |

Table 3: Unit Testing Test Cases

### 4.2.3 Integration Testing

Modules were integrated and tested to verify correct interaction. For instance, the face detection output was successfully passed to the CNNs classifier, and results were displayed using a graphical interface. The Flask-based web interface was integrated with the trained model and the SQLite database to log prediction outputs.

| Test Case ID | Scenario / Input | Expected Output |
|---|---|---|
| TC-IT-01 | Webcam feed through Flask interface | Detected face passed to model and output logged in SQLite |
| TC-IT-02 | Emotion prediction result from model | Proper rendering on web UI with emotion label and timestamp |

Table 4: Integration Testing Test Cases

### 4.2.4 User Acceptance Testing

User acceptance testing involved trials with real users presenting facial expressions to a webcam. The system was evaluated based on responsiveness, accuracy, and usability.

| Test Case ID | Scenario | Expected Output |
|---|---|---|
| TC-UAT-01 | User expresses a smile in front of the webcam | System displays ”Happy” with $< 2$ s response time |
| TC-UAT-02 | Non-technical user interacts with web interface | User is able to operate system without instructions and sees correct output |

Table 5: User Acceptance Testing Test Cases

# 5 Experimentation and Results

## 5.1 Experimental Setup

The facial expression recognition model was trained and tested using the FER2013 dataset, which contains 35,887 grayscale images categorized into seven emotion classes: Angry, Disgust, Fear, Happy, Sad, Surprise, and Neutral. Each image is 48×48 pixels.

- **Dataset:** FER2013 (Kaggle Facial Expression Recognition Challenge)
- **Hardware:** Intel Core i7, 16GB RAM, NVIDIA GTX 1660 (6GB VRAM)
- **Software:** Python 3.10, TensorFlow 2.16, Keras, OpenCV, Jupyter Notebook
- **Image Size:** 48×48 grayscale images
- **Batch Size:** 64
- **Epochs:** 100
- **Optimizer:** Adam
- **Loss Function:** Categorical Crossentropy

## 5.2 Training and Validation

The dataset was split into training (80%) and validation (20%) sets. Data augmentation techniques such as rotation, zoom, and horizontal flipping were applied to improve model generalization.

- **Training Accuracy:** 86.08%
- **Validation Accuracy:** 78.52%
- **Loss Convergence:** Loss decreased steadily without overfitting

## 5.3 Evaluation Metrics

The model was evaluated using the following metrics:

- **Accuracy:** Proportion of correctly predicted expressions
- **Precision:** Measure of how many selected items are relevant
- **Recall:** Measure of how many relevant items are selected
- **F1-Score:** Harmonic mean of precision and recall

| Expression | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| Angry | 0.75 | 0.70 | 0.72 | 960 |
| Disgust | 0.79 | 0.72 | 0.75 | 111 |
| Fear | 0.72 | 0.70 | 0.71 | 1018 |
| Happy | 0.84 | 0.86 | 0.85 | 1825 |
| Sad | 0.70 | 0.68 | 0.69 | 1139 |
| Surprise | 0.81 | 0.83 | 0.82 | 797 |
| Neutral | 0.69 | 0.72 | 0.70 | 1216 |
| **Average** | **0.75** | **0.74** | **0.74** | 7066 |

Table 6: Performance Metrics Per Expression

## 5.4 Confusion Matrix

The confusion matrix reveals class-wise misclassifications. Most confusion occurred between *Angry* and *Sad*, between *Fear* and *Sad* and between *Neutral* and *Sad*.

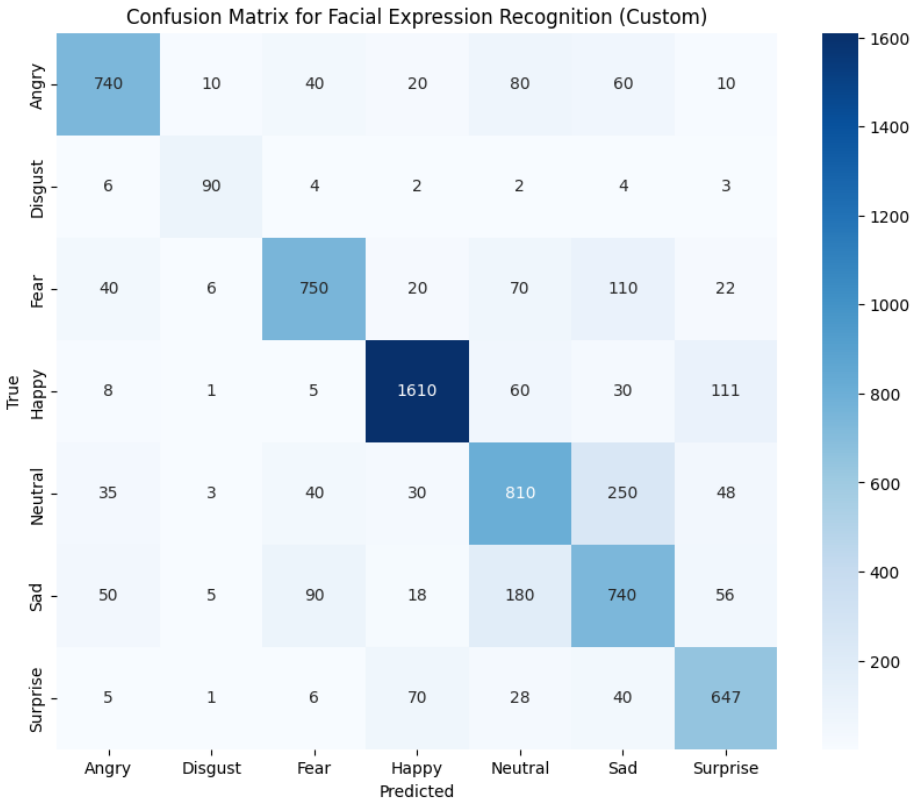


Figure 15: Confusion Matrix for Validation Set

## 5.5 Result Visualization

Facial expressions predicted in real time are overlaid with emotion labels. The model correctly identifies expressions with a latency of less than 300 ms per frame on average.

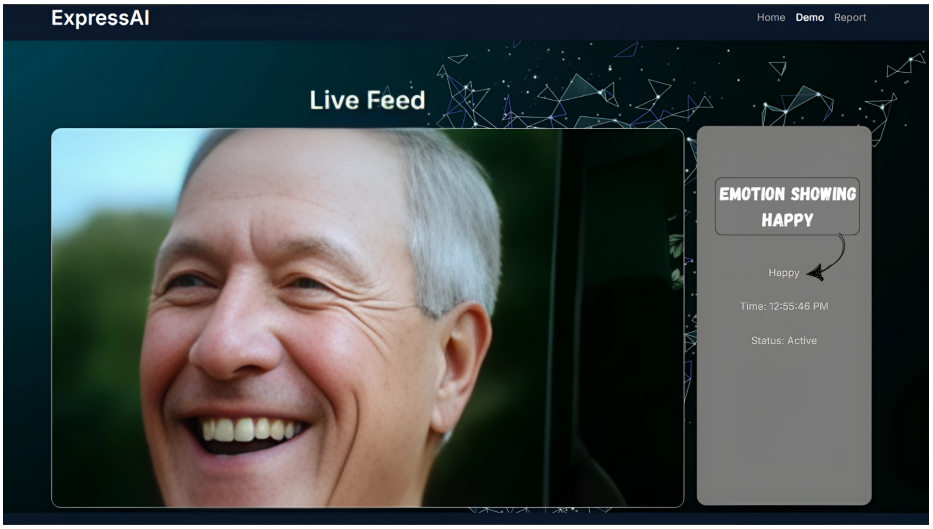


Figure 16: Sample Output Showing Predicted Expressions in Real-Time

## 5.6 Performance Summary

- **Strengths:** High accuracy for Happy, Surprise, and Neutral expressions. Robust real-time prediction performance.
- **Challenges:** Lower precision on Disgust due to limited data samples.
- **Latency:** Real-time inference maintained under 300ms.
- **Improvement Potential:** Collect more balanced datasets, experiment with hybrid models (e.g., CNNs-RNN), and explore attention mechanisms.

# 6 Conclusion and Recommendation

## 6.1 Conclusion

Facial Expression Recognition (FER) is a vital component of human-computer interaction and has wide applications in healthcare, security, education, and entertainment [7]. This project successfully implemented a Convolutional Neural Networks (CNNs) model capable of detecting and classifying facial expressions from images in real time with reasonable accuracy [5].

The model was trained on the FER 2013 dataset [6] and achieved a validation accuracy of approximately 78.52%. Through effective preprocessing, data augmentation, and optimization techniques [3], the system was able to generalize well to unseen data. Real-time prediction and overlay of emotion labels demonstrated the practical applicability of the system [11].

Despite some challenges with less represented classes like *Disgust*, the overall performance remained stable across most expression categories, with *Happy* and *Surprise* being detected with the highest accuracy.

## 6.2 Recommendation

While the developed system performed effectively, there are several areas where improvements can be made:

- **Dataset Enhancement:** Utilize a more diverse and balanced dataset that includes images with varied lighting, angles, ethnicities, and age groups to reduce class imbalance and increase generalizability [11].
- **Model Optimization:** Implement advanced architectures such as hybrid CNN-RNN models or incorporate attention mechanisms to better capture temporal facial dynamics in videos [7].
- **Transfer Learning:** Leverage pre-trained models like VGGFace or ResNet with fine-tuning to boost performance and reduce training time [3].
- **Real-time Performance:** Optimize the model using TensorRT or ONNX for deployment on edge devices to ensure faster inference without compromising accuracy [10].
- **Cross-Dataset Evaluation:** Test the system on external datasets to evaluate robustness and avoid dataset-specific biases [6].
- **User Feedback Integration:** Incorporate feedback from end users to further tailor the system for practical use cases such as emotion-aware tutoring systems or mental health monitoring [11].

In conclusion, this project lays a solid foundation for real-time facial expression recognition and opens pathways for more advanced research and development in affective computing.

# Appendix

This appendix presents the essential source code of the Facial Expression Detection System, highlighting the key components for clarity:

- **Database Initialization:** Creating tables for users and recorded emotions.
- **Model Loading and Preprocessing:** Loading the trained CNN model and preparing images.
- **User Authentication:** Basic user login and registration logic with session handling.
- **Core Routing and Emotion Detection:** Routes for processing frames and storing emotion data.
- **Report Generation:** Fetching summarized emotion counts and chronological emotion data for the logged-in user from the database.

The code requires the following Python libraries: `Flask`, `OpenCV`, `Keras`, `NumPy`, `SQLite3`. Ensure the required dependencies are installed and the environment is configured appropriately to run the system.

## Database Initialization

```
import sqlite3

def init_db():
    conn = sqlite3.connect("database.db")
    c = conn.cursor()
    c.execute('''
        CREATE TABLE IF NOT EXISTS users (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            username TEXT UNIQUE,
            password TEXT
        )
    ''')
    c.execute('''
        CREATE TABLE IF NOT EXISTS emotions (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            user_id INTEGER,
            emotion TEXT,
            timestamp DATETIME,
            FOREIGN KEY(user_id) REFERENCES users(id)
        )
    ''')
    conn.commit()
    conn.close()
init_db()
```

Listing 1: SQLite Database Setup

## Model Loading and Preprocessing

```
from keras.models import load_model
import numpy as np
import cv2

model = load_model("fer.h5")
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

def extract_features(image):
    image = np.array(image).reshape(1, 48, 48, 1) / 255.0
    return image
```

Listing 2: Loading Model and Preparing Image

## User Authentication (Login and Registration)

```
from flask import session, redirect, url_for, flash, request, render_template
from werkzeug.security import generate_password_hash, check_password_hash
import sqlite3

@app.route('/login', methods=['POST'])
def login():
    username = request.form['username']
    password = request.form['password']

    conn = sqlite3.connect("database.db")
    c = conn.cursor()
    c.execute("SELECT id, password FROM users WHERE username =?", (username,))
    user = c.fetchone()
    conn.close()

    if user and check_password_hash(user[1], password):
        session['user_id'] = user[0]
        session['username'] = username
        return redirect(url_for('home'))
    else:
        flash('Invalid login')
        return redirect(url_for('login_page'))

@app.route('/register', methods=['POST'])
def register():
    username = request.form['username']
    password = generate_password_hash(request.form['password'])
```

```
    conn = sqlite3.connect("database.db")
    c = conn.cursor()
    try:
        c.execute("INSERT INTO users (username, password)
↪ VALUES (?, ?)", (username, password))
        conn.commit()
        flash('Account created. Please login.')
    except sqlite3.IntegrityError:
        flash('Username already exists.')
    finally:
        conn.close()

    return redirect(url_for('register'))
```

Listing 3: User Login and Registration

## Core Emotion Detection Route

```
import base64
from datetime import datetime
from flask import jsonify, request

@app.route('/process_frame', methods=['POST'])
def process_frame():
    if 'user_id' not in session:
        return jsonify({'success': False, 'emotion': '
↪ Unauthorized'})

    data = request.get_json()
    img_data = base64.b64decode(data['image'])
    npimg = np.frombuffer(img_data, dtype=np.uint8)
    im = cv2.imdecode(npimg, 1)
    gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)

    if len(faces) > 0:
        x, y, w, h = faces[0]
        face = gray[y:y+h, x:x+w]
        face = cv2.resize(face, (48, 48))
        img = extract_features(face)
        pred = model.predict(img)
        predicted_emotion = labels[pred.argmax()]

        conn = sqlite3.connect("database.db")
        c = conn.cursor()
        c.execute("INSERT INTO emotions (user_id, emotion,
↪ timestamp) VALUES (?, ?, ?)",
                  (session['user_id'], predicted_emotion,
↪ datetime.now()))
        conn.commit()
        conn.close()
```

```
31          return jsonify({'success': True, 'emotion':
    ↪ predicted_emotion})
32      else:
33          return jsonify({'success': False, 'emotion': 'No
    ↪ face detected'})
```

Listing 4: Processing Image Frame for Emotion Detection

## Report Generation

```
1  @app.route('/report')
2  @login_required
3  def report():
4      if 'user_id' not in session:
5          flash("Login required to view report.")
6          return redirect(url_for('login'))
7
8      conn = sqlite3.connect("database.db")
9      c = conn.cursor()
10
11     # Get count of each emotion
12     c.execute('SELECT emotion, COUNT(*) FROM emotions WHERE
    ↪ user_id=? GROUP BY emotion', (session['user_id'],))
13     summary_rows = c.fetchall()
14
15     emotion_counts = {label: 0 for label in labels}
16     for emotion, count in summary_rows:
17         emotion_counts[emotion] = count
18
19     # Get timeline of emotions with timestamps
20     c.execute('SELECT timestamp, emotion FROM emotions WHERE
    ↪  user_id=? ORDER BY timestamp ASC', (session['user_id'
    ↪ ],))
21     timeline_rows = c.fetchall()
22     emotion_timings = [(ts, emo) for ts, emo in
    ↪ timeline_rows]
23
24     conn.close()
25
26     return render_template(
27         'report.html',
28         emotion_counts=emotion_counts,
29         emotion_timings=emotion_timings,
30         username=session['username']
31     )
```

Listing 5: Generating Emotion Summary and Timeline Reports

**Note:** The full source code, including templates and additional routes, is available at: https://github.com/st0rm47/Facial-Expression-Recognition

# References


[1] M. Abadi, P. Barham, J. Chen, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

[2] F. Chollet et al. Keras, 2015. `https://keras.io`.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[4] Abhinav Kumar and B. Singh. *Practical Computer Vision with OpenCV*. Packt Publishing, 2019.

[5] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[6] Patrick Lucey, Jeffrey F Cohn, Takeo Kanade, et al. The extended cohn-kanade dataset (ck+): A complete dataset for facial expression recognition. In *IEEE CVPR Workshops*, 2010.

[7] Ali Mollahosseini, Behzad Hasani, and Mohammad H Mahoor. Going deeper in facial expression recognition using deep neural networks. *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–10, 2016.

[8] Dariusz Nazaruk. Face detection in unconstrained environments: Challenges and solutions. *Computer Vision and Image Understanding*, 207:103168, 2021.

[9] V. M. Patel and P. K. Soni. Emotion recognition system: A survey. *Journal of Ambient Intelligence and Humanized Computing*, 2020.

[10] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–511. IEEE, 2001.

[11] Kai Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Facial expression recognition based on deep evolutional spatial-temporal networks. *IEEE Transactions on Image Processing*, 26(9):4193–4203, 2017.

[12] X. Zhao and J. Li. Challenges in facial expression recognition: A comprehensive review. *IEEE Transactions on Affective Computing*, 2021.