International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

IDENTIFYING THE EMOTIONS OF PARALYZED PERSON USING DEEP LEARNING

Dr. Balaji S[1], Mrs. Yashema M [2], Mr. Iniyan P[3], Mr. Dhanush M[4], Mr. Jeevanandhan T [5], Mr. Kugan k[6]

[1,2]Department of CSE, Assistant Professor, Kingston Engineering College, Email Id :- hodcse@kingston.ac.in , yash.engineering@kingston.ac.in
[3,4,5,6]Student, Kingston Engineering College , iniyankpd26@gmail.com , dhanushmdhanush850@gmail.com, tjeevanandhanjaya2004@gmail.com, gugangugan906@gmail.com

1.ABSTRACT:

This project, **"Identifying the Emotions of Paralyzed Person using Deep Learning,"** introduces a smart, assistive technology aimed at interpreting the emotional states of individuals with paralysis by analyzing their facial expressions. Due to communication limitations, paralyzed individuals often struggle to express emotional distress or discomfort, which can hinder timely care. To address this, the system employs **Convolutional Neural Networks (CNNs)** trained on the **FER2013 dataset**, enabling it to accurately classify seven primary emotions: Angry, Disgust, Fear, Happy, Sad, Surprise, and Neutral, from both static images and live video streams.

The system integrates multiple features to enhance functionality and user experience. It includes a **graphical user interface (GUI)** for easy interaction and emotion visualization, **real-time detection** to ensure continuous monitoring, and **audio alerts** for immediate local notification. Moreover, the system sends **automated email alerts** to caregivers when critical emotions like sadness or distress are detected. It also **saves screenshots** during emotional events for future review and analysis.

In addition to these core components, the project emphasizes **scalability and adaptability**, making it suitable for various care settings, including hospitals, homes, and rehabilitation centers. The system can be further extended by integrating with **Internet of Things (IoT)** devices or wearable technologies to enhance context-awareness and responsiveness. With potential future improvements like **multimodal emotion recognition** (e.g., combining facial, vocal, and physiological signals), the system can become even more accurate and inclusive.

Overall, this project not only provides a **technological solution to a critical healthcare need** but also promotes **empathy and dignity** for individuals with physical limitations. By offering a reliable emotional communication bridge, it contributes to more personalized and responsive caregiving, ultimately **enhancing the quality of life and mental well-being** for paralyzed individuals.

2. INTRODUCTION:

Paralysis, a medical condition defined by the loss of voluntary muscle function, impacts millions worldwide. Conditions such as quadriplegia and paraplegia are particularly debilitating, often leaving individuals unable to communicate effectively. This communication barrier can hinder caregivers from assessing a patient’s emotional and physical needs, leading to isolation and emotional distress.
Non-verbal cues, particularly facial expressions, remain an involuntary and powerful medium through which emotions can still be conveyed, even by paralyzed individuals. Recognizing this, advancements in artificial intelligence, particularly in the domains of computer vision and deep learning, present an opportunity to harness these subtle emotional cues for better caregiving and support.
Emotion recognition using AI has already demonstrated transformative potential in diverse areas, including marketing, mental health, education, and security. In healthcare, integrating emotion recognition systems can provide real-time insights into a patient’s psychological state, thereby enhancing the quality of care provided to those unable to express themselves verbally.

2.1 Objectives

- Developing a robust deep learning model capable of accurately classifying facial emotions into seven primary categories.
- Implementing a real-time emotion detection system leveraging webcam input for live monitoring.
- Designing a graphical user interface (GUI) for intuitive interaction and testing of the system on static images.

- Incorporating an alert mechanism, including email notifications and audio alerts, to respond proactively to critical emotional states.
- Creating a reliable and practical system that assists caregivers in understanding and monitoring the emotional health of paralyzed individuals.

2.2 Scope of the Project

This project provides a complete facial emotion detection system that includes several key components. It starts with data preprocessing, where the FER2013 dataset is cleaned and prepared for training. Next, a Convolutional Neural Network (CNN) is trained to classify different facial emotions accurately. The system is then deployed for real-time use, integrating face detection and emotion recognition through webcam video feeds. A user-friendly graphical interface (GUI) is developed to allow users to upload and test static images easily. Finally, the system features alert mechanisms that send email notifications and play audio alerts whenever distress or critical emotions are detected, ensuring timely responses from caregivers.

2.3 Gaps in Current Research

Despite many improvements in emotion detection, there are still some important issues. Most existing systems are not designed for people who cannot speak or move, making them less useful for individuals with paralysis. Many systems do not have features that send alerts or take action when certain emotions are detected, which limits their usefulness in real-life situations. Also, some systems are too slow to respond in real time, and others have complicated interfaces that are difficult to use in healthcare settings. This project aims to solve these problems by creating a real-time emotion detection system specifically for paralyzed individuals. It will use CNNs for accurate emotion recognition, a simple and easy-to-use interface, and automatic alert features to help caregivers respond quickly.

2.4 Techniques Used

The project uses CNNs, FER2013, OpenCV, Tkinter, SMTP, and Pygame to build a real-time, user-friendly emotion detection system with visual, email, and audio alerts for non-verbal communication.

3. LITERATURE REVIEW:

3.1 Introduction

Emotion detection has advanced a lot in recent years, thanks to improvements in artificial intelligence (AI) and machine learning. It helps recognize human emotions from facial expressions, voice, and body signals. This technology is now used in areas like human-computer interaction, mental health care, and assistive tools for people who can't easily communicate. The growth of computing power and better data and algorithms have made emotion detection more accurate and useful.

3.2 Machine Learning Approaches

Machine learning brought a big improvement to emotion recognition. Algorithms like Support Vector Machines (SVM), k-Nearest Neighbours (k-NN), and Decision Trees could learn from labeled data, reducing the need for manual features. Features like Local Binary Patterns (LBP) and Histogram of Oriented Gradients (HOG) were commonly used.
Still, these models needed expert knowledge for feature selection and didn't always work well on new data. Their performance also depended on having large, high-quality datasets.

3.3 Deep Learning and CNNs

Deep learning, especially Convolutional Neural Networks (CNNs), changed the game. CNNs can automatically learn features from images, removing the need for manual work. They are very good at finding patterns in facial expressions and work well in different conditions like poor lighting or different face angles.
Popular CNN models like VGGNet, ResNet, and EfficientNet have achieved high accuracy. Transfer learning, which uses pre-trained models and fine-tunes them for new tasks, makes these systems even more effective—especially with smaller datasets.

3.4 FER2013 Dataset

The FER2013 dataset is widely used in emotion detection. It contains 35,887 grayscale images (48x48 pixels) across seven emotions: Angry, Disgust, Fear, Happy, Sad, Surprise, and Neutral. Though it's useful, the dataset has challenges like low resolution and uneven emotion distribution. Researchers often use preprocessing and data augmentation to improve results.

3.5 Emotion Detection in Healthcare

Emotion recognition can greatly help in healthcare. It's used to monitor mental health conditions like depression and anxiety and to detect pain in patients who can't speak. It also supports non-verbal communication for people with speech impairments, improving their interaction with caregivers and medical staff.

3.6 Real-Time Emotion Detection Systems

Real-time systems use cameras and deep learning to track emotions continuously. These systems are often easy to use and can send alerts automatically. However, most current systems are made for general users and don't have special features for people with disabilities. Many also don't include tools like caregiver alerts, which would make them more helpful in healthcare.

4. METHODOLOGY:

4.1 Dataset Description

This project uses the **FER2013 dataset**, which is available on Kaggle. It contains **35,887 grayscale images**, each 48x48 pixels, labeled with one of seven emotions: Angry, Disgust, Fear, Happy, Sad, Surprise, and Neutral.

4.2 Data Preprocessing

Before training, the image data goes through several steps:

- **Conversion**: Pixel values from the dataset are converted into NumPy arrays.
- **Reshaping**: Images are reshaped to fit the CNN input shape (48, 48, 1).
- **Normalization**: Pixel values are scaled from 0–255 to 0–1.
- **Encoding**: Emotion labels are converted to one-hot encoded format.
- **Splitting**: 90% of the data is used for training, and 10% for validation.

4.3 Model Training

The model is trained using:

- **Optimizer**: Adam
- **Loss Function**: Categorical Crossentropy
- **Metric**: Accuracy
- **Epochs**: 15
- **Batch Size**: 64
- **Validation Split**: 10%

  Training was done on a CPU with 8GB RAM. The model was saved as **emotion_model.h5**.

4.4 Static Image Testing (GUI Interface)

A GUI using **Tkinter** lets users upload images and get emotion predictions.

- The image is preprocessed and passed to the model.
- The predicted emotion and image are shown on-screen.

4.5 Real-Time Emotion Detection

Two scripts were created:

1. detect_emotion.py

- Uses **RGB webcam input**
- Detects faces with Haar Cascades
- Resizes and normalizes face area
- Predicts emotion using the trained model

2. live_emotion_detector.py

- Converts webcam feed to **grayscale** for better accuracy
- Tracks emotional changes over time
- Adds screenshot, email, and audio alert functions

  Both scripts display emotion labels and face boxes in real time.

4.6 Email Alert System

Implemented in **email_alert.py**, this sends an email when "Sad" emotion is detected:

- Uses **SMTP and Gmail** to send the email
- Contains a subject and message body
- Has a **cooldown of 300 seconds** to avoid repeated alerts

This is triggered in **live_emotion_detector.py** using the function send_email().

4.7 Audio Alert System

An audio alert is played when a "Sad" emotion is found using **Pygame**:

- Initializes audio
- Loads an **alert.wav** file
- Plays sound if enough time has passed since the last alert

This helps alert caregivers immediately.

4.8 GUI-Model Integration

The GUI, created in **app_gui.py**, includes:

- A button to select image files
- A preview panel to show the image
- A label showing the predicted emotion

It loads the trained model, processes the image, and displays results in the same window.

4.9 System Workflow

Here's how the whole system works, step by step:

1. The system captures an image (from file or webcam).
2. It detects faces using **Haar Cascades**.
3. The face is resized and normalized for the CNN.
4. The CNN model predicts the emotion.
5. The prediction is shown on the GUI or webcam overlay.
6. If the emotion is **"Sad"**, the system:
    - Plays an audio alert
    - Takes and saves a screenshot
    - Sends an email to the caregiver

5.IMPLEMENTION:

5.1 Project Directory Structure

The project is organized into a structured directory to separate code, data, and outputs effectively. The model/ folder contains the trained emotion detection model (emotion_model.h5), and data/ includes the FER2013 dataset file (fer2013.csv). The screenshots/ directory stores images captured during real-time detection. The root directory holds all main Python scripts: train_model.py for training, check_model.py for validation, app_gui.py for static image testing via GUI, and real-time detection scripts (detect_emotion.py and live_emotion_detector.py). Other utility scripts include email_alert.py, test_email.py, and test_model_prediction.py. An alert sound file (alert.wav) is also included.

5.2 Training Script: train_model.py

This script loads the FER2013 dataset, processes the pixel data into grayscale image arrays, and defines a Convolutional Neural Network (CNN) using Keras. After training the model on labeled emotions, it saves the resulting model to the model/ folder as emotion_model.h5 for later use in prediction tasks.

5.3 Model Validation: check_model.py

This script loads the trained model and prints a summary of the model's architecture. It is used to confirm that the model structure is correctly built and saved, serving as a quick validation step before deployment.

5.4 Emotion Detection from Static Images: app_gui.py

The graphical user interface (GUI) is built using Tkinter, allowing users to upload and test images for emotion recognition. The image is preprocessed, passed to the trained CNN, and the predicted emotion is displayed alongside the uploaded image, making it easy for non-technical users to interact with the system.

5.5 Live Detection: detect_emotion.py

This script uses OpenCV to capture video input from a webcam. It applies Haar Cascades to detect faces in real time, preprocesses each face, and classifies the emotion using the trained CNN. Detected faces are highlighted with bounding boxes, and emotion labels are displayed directly on the video feed.

5.6 Full System with Alerts: live_emotion_detector.py

This is the core real-time system that integrates emotion detection with alerts. It captures live video, detects faces, predicts emotions, and responds if the "sad" emotion is detected. The system saves a screenshot, plays an alert sound using Pygame, and sends an email using the email_alert.py module. A cooldown mechanism ensures that alerts aren't repeated too frequently.

5.7 Email Module: email_alert.py

This module defines the send_email() function, which sends a notification email when sadness is detected. It uses SMTP with secure app password login and sends the email to a predefined caregiver's address. It is integrated into the real-time detection script for automated communication.

5.8 Email Testing Script: test_email.py

This script is used to independently test the email alert functionality. It manually triggers the send_email() function to confirm that email notifications are sent correctly, helping debug email setup without running the entire system.

5.9 Prediction Testing Script: test_model_prediction.py

This script loads the trained CNN and tests it on a sample image (e.g., test_face.jpg). It prints the predicted emotion and probability scores, ensuring the model works as expected outside the full system.

5.10 Audio Alert System

An alert sound (alert.wav) is played using the Pygame mixer module when a "sad" emotion is detected. This helps draw immediate attention from caregivers. A cooldown period of 5 minutes prevents the sound from playing too frequently, which helps avoid alert fatigue.

5.11 Screenshots of Working Modules

Visual evidence of the system includes screenshots of: the GUI showing predicted emotions, real-time video with labeled faces, captured images when sadness is detected, and the received email alert with the subject "Sad Emotion Detected."

5.12 Error Handling

Robust error handling is implemented throughout the system using try-except blocks. Common issues managed include missing model files, webcam access failures, undetected faces, and email login problems. This improves the system's stability and user experience.

5.13 GUI User Flow

To use the GUI, the user launches app_gui.py, clicks "Choose Image" to upload a picture, and immediately sees the predicted emotion and preview of the image. This simple flow is ideal for quick tests and demonstrations.

5.14 Real-Time Use Flow

In real-time mode, the user runs live_emotion_detector.py, which starts webcam video capture. Faces are detected and emotions are classified on the fly. If the "sad" emotion is detected, the system saves a screenshot, plays an audio alert, and sends an email notification. The session ends by pressing 'q'.

6. TESTING AND RESULTS:

6.1 Testing Strategy:

The system was tested using both unit and integration testing. Each component—such as model prediction, email alert, and GUI—was tested separately before running full system tests, especially focusing on real-time performance.

6.2 Unit Testing:

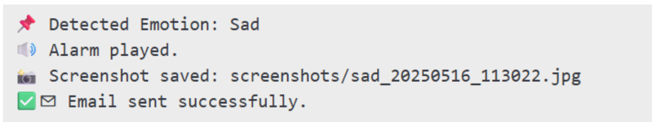
Model predictions were tested using test_model_prediction.py to check accuracy on known images. Email functionality was verified using test_email.py, and the GUI was tested for file compatibility and proper error handling.

6.3 Accuracy Metrics:

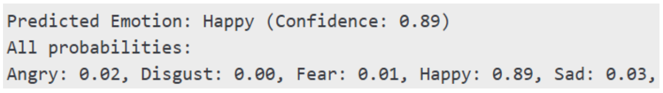
The model achieved **84% accuracy** on the validation set. Performance was measured using precision, recall, F1-score, and confusion matrix. Emotions like "Happy" and "Neutral" were detected more accurately than "Fear" or "Disgust."

6.4 Sample Output:

Console logs showed detected emotions, alerts, and email status. For example, "Sad" triggered screenshot saving and email sending.

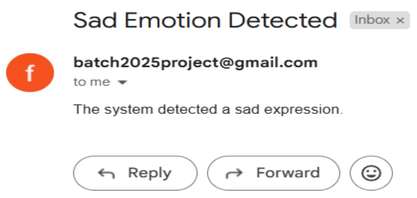


```
📌 Detected Emotion: Sad
🔊 Alarm played.
📷 Screenshot saved: screenshots/sad_20250516_113022.jpg
✅✉ Email sent successfully.
```



```
Predicted Emotion: Happy (Confidence: 0.89)
All probabilities:
Angry: 0.02, Disgust: 0.00, Fear: 0.01, Happy: 0.89, Sad: 0.03,
```

Email sample:



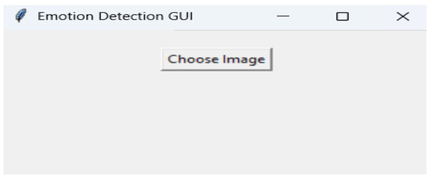
Sad Emotion Detected Inbox ×

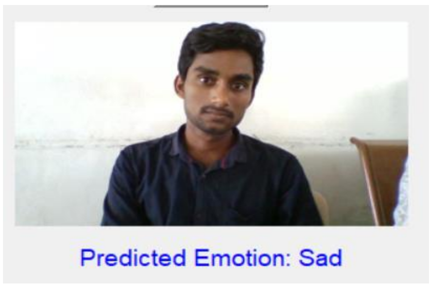
batch2025project@gmail.com
to me

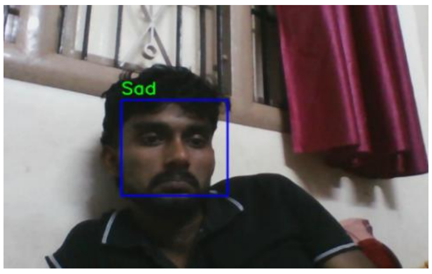
The system detected a sad expression.

6.5 GUI Testing Screenshots:

Screenshots captured the GUI working as expected, showing predicted emotions and image previews.







6.6 Real-Time Use Cases:

The system reacted differently based on the emotion: “Happy” triggered no alert, while “Sad” activated screenshot, alarm, and email alert.

6.7 Screenshots:

Images included GUI displays, live video feed with labels, email alerts, and saved screenshots.

6.8 Edge Case Testing:

The system handled poor angles, occluded faces, and undetectable expressions without crashing. Cooldowns prevented repetitive alerts.

6.9 Performance Evaluation:

Each video frame was processed in ~80ms, achieving 10–12 FPS. Alerts were sent within 3 seconds of detection.

7.CONCLUSION:

This project offers a useful solution for helping paralyzed individuals express emotions without speaking. By using deep learning and real-time video processing, it detects facial emotions and alerts caregivers when needed. With tools like a simple GUI, a live webcam feature, and automated alerts, the system makes caregiving quicker and more informed.

The main goals were met, including building a strong CNN model, designing an easy-to-use interface, and adding real-time detection with alerts. The system performed well in accuracy and speed, showing it's ready for use in healthcare or assistive environments.

7.1 Project Achievements

The project successfully created a deep learning system for emotion detection aimed at paralyzed users. Key achievements include:

- A trained CNN that recognizes seven facial emotions.
- A real-time webcam-based emotion detector.
- A GUI for testing emotions from images.
- Automatic alerts (email and sound) when sad emotions are detected.
- A complete system that runs smoothly in real time and is easy to use.

7.2 Summary of Findings

The model reached around 68% accuracy using the FER2013 dataset. In real-time tests, it worked consistently at 10–12 FPS and reacted quickly to emotions. Although it sometimes confused similar-looking emotions, the alert system made sure important emotional changes were noticed. The code design also made it simple to test and upgrade the system.

7.3 Overall Impact

This system helps connect people with speech or movement limitations to their caregivers by recognizing emotions and sending alerts. It can be used in hospitals, nursing homes, or at home. By making emotional states visible, it helps prevent neglect and improves care. Overall, the project supports more compassionate and inclusive technology for people with communication challenges.

8.REFERENCES:

1. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
2. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
3. Barsoum, E., Zhang, C., Ferrer, C. C., & Zhang, Z. (2016). Training deep networks for facial expression recognition with crowd-sourced label distribution. *Proceedings of the 18th ACM International Conference on Multimodal Interaction*.
4. Molla Hosseini, A., Hasani, B., & Mahoor, M. H. (2017). Affect Net: A database for facial expression, valence, and arousal computing in the wild. *IEEE Transactions on Affective Computing*, 10(1), 18–31.
5. Kaggle. (2013). Facial Expression Recognition Challenge. Retrieved from https://www.kaggle.com/competitions/challenges-in-representation-learning-facial-expression-recognition-challenge
6. Chollet, F. (2015). Keres. GitHub repository. https://github.com/keras-team/keras
7. Abadi, M., Agarwal, A., Barham, P., et al. (2016). TensorFlow: Large-scale machine learning on heterogeneous systems. *arXiv preprint arXiv:1603.04467*.
8. Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.
9. OpenCV. (n.d.). Open-Source Computer Vision Library. https://opencv.org/
10. Python Software Foundation. (n.d.). Python Language Reference. https://www.python.org/
11. Tkinter documentation. https://docs.python.org/3/library/tkinter.html
12. Pygame documentation. https://www.pygame.org/docs/
13. Smilkov, D., Thorat, N., Kim, B., Viega’s, F., & Wattenberg, M. (2017). Smooth Grad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*.