Facial Emotion Recognition Dashboard

SUBJECT: DATA WAREHOUSING AND DATA MINING

Project Title: Facial Emotion Recognition using Convolutional Neural Networks (CNN)

Technologies Used: Python, TensorFlow, Keras, Streamlit, Matplotlib, OpenCV.

Introduction

Objective: The objective of this project is to develop a lightweight, efficient Deep Learning model to classify human facial expressions into seven categories: **Angry, Disgust, Fear, Happy, Neutral, Sad, and Surprise.**

Significance: This technology is essential for:

- **Human-Computer Interaction (HCI):** Creating apps that respond to user moods.
- **Security:** Identifying distressed or aggressive behavior in public spaces.
- **Market Research:** Analyzing consumer reactions to products in real-time.

Abstract

This project presents the development of an efficient **Facial Emotion Recognition (FER)** system utilizing a **Custom Deep Convolutional Neural Network (CNN)**. The system is designed to categorize human facial expressions into seven distinct classes from 48×48 grayscale images. To address common computer vision challenges such as class imbalance and overfitting, techniques including **Class Weighting**, **Batch Normalization**, and **Data Augmentation** were implemented. The final model is integrated into a **Streamlit** dashboard, providing a user-friendly, real-time interface for image analysis and emotion classification without the need for high-end computational resources.

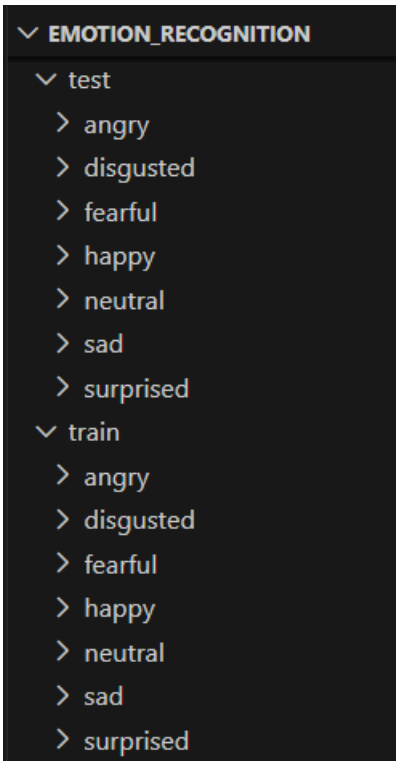
Executive Summary

The primary goal of this study was to bridge the gap between raw pixel data and actionable emotional intelligence using a lightweight AI approach. The project followed a comprehensive machine learning pipeline:

- **Data Analysis:** Exploratory Data Analysis (EDA) was performed on the FER-2013 dataset. This revealed significant class imbalances specifically an abundance of "Happy" samples which were mitigated through the calculation and application of **mathematical class weights**.
- **Model Engineering:** A custom-built **Multi-Layer CNN** was developed. By optimizing the number of convolutional filters and incorporating **Dropout layers**, the model achieved high feature extraction capabilities while maintaining a small memory footprint, allowing it to run efficiently on standard CPUs.
- **Deployment:** The integration of the trained .h5 model into a **Streamlit web application** demonstrates the practical utility of AI in sectors such as mental health monitoring, driver safety, and retail customer sentiment analysis.

Dataset Description

- **Source:** KAGGLE FER2013 FACIAL EMOTION DATASET
- **Structure:** The data is split into train and test folders, each containing 7 subfolders for each emotion.
- **Input Size:** Original images are 48*48 pixels in grayscale.



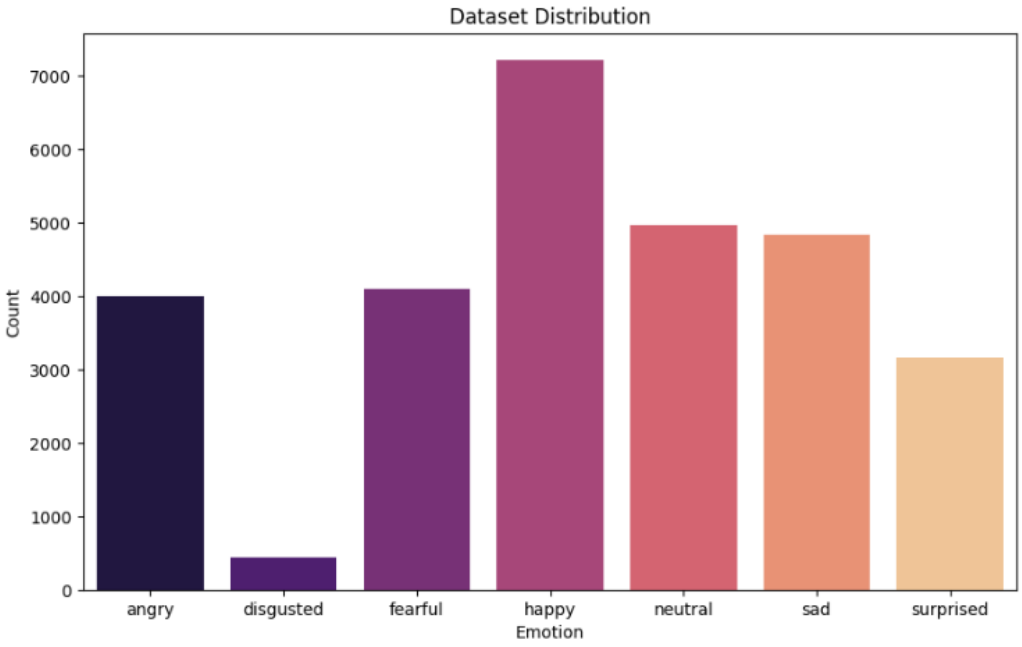
Exploratory Data Analysis (EDA)

Before training the model, a comprehensive Exploratory Data Analysis (EDA) was performed to understand the characteristics of the facial expression dataset. The primary objective of this phase was to identify class imbalance, which is a common issue where certain emotions (such as "Happy") are over-represented compared to others (such as "Disgust"). Recognizing these imbalances is crucial, as they can cause the model to develop a bias, leading it to predict the majority class more frequently.

Data Visualization

To gain a clear perspective on the dataset, two primary visualizations were generated:

- **Class Distribution Plot:** A bar chart and pie chart showing the count of images in each emotion category.



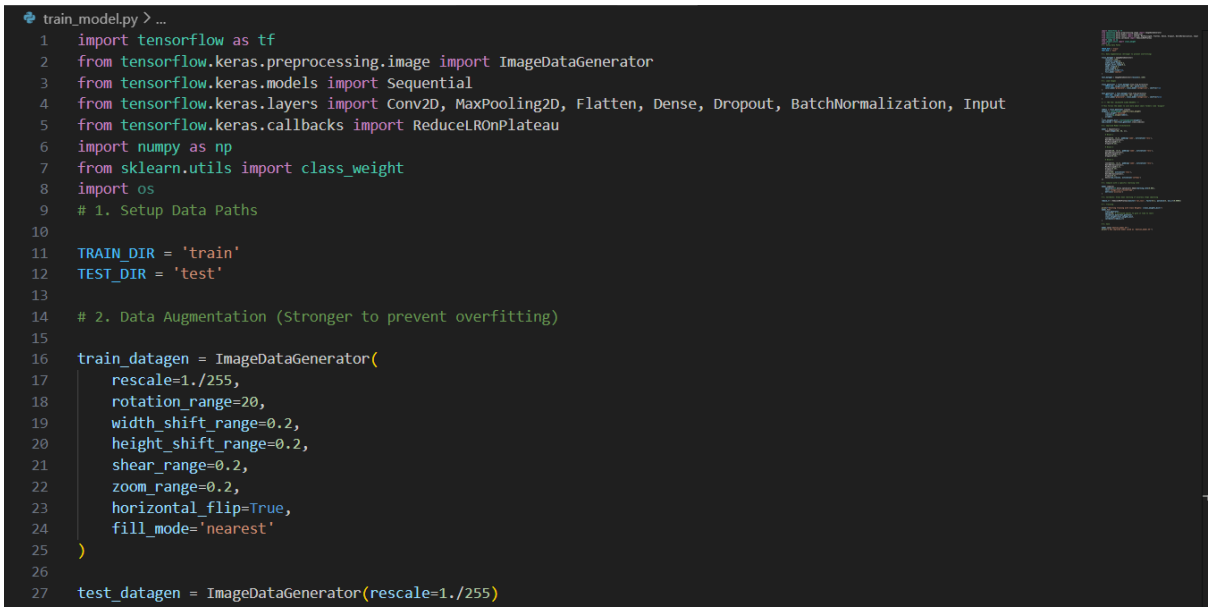
- **Sample Grid:** A selection of raw images from the dataset to verify the quality and diversity of the facial expressions.

5. Methodology

Architecture: The project uses a **Custom Multi-Layer CNN**. Unlike heavy pre-trained models, this architecture is designed for speed and efficiency on standard hardware. It consists of:

- **Convolutional Layers:** To extract spatial features like eye shapes and mouth curves.
- **Batch Normalization:** To stabilize learning and speed up training.
- **MaxPooling:** To reduce image dimensions and focus on important features.
- **Dropout (0.3 - 0.5):** To prevent the model from memorizing training data (overfitting).

- **Preprocessing:** Using ImageDataGenerator, images were rescaled ($1/255$) and augmented with horizontal flips to make the model more robust.



```python
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization, Input
from tensorflow.keras.callbacks import ReduceLROnPlateau
import numpy as np
from sklearn.utils import class_weight
import os
# 1. Setup Data Paths

TRAIN_DIR = 'train'
TEST_DIR = 'test'

# 2. Data Augmentation (Stronger to prevent overfitting)

train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

test_datagen = ImageDataGenerator(rescale=1./255)
```

Model Training & Results

Parameters:

- **Epochs:** 30
- **Batch Size:** 32
- **Optimizer:** Adam
- **Loss Function:** Categorical Crossentropy

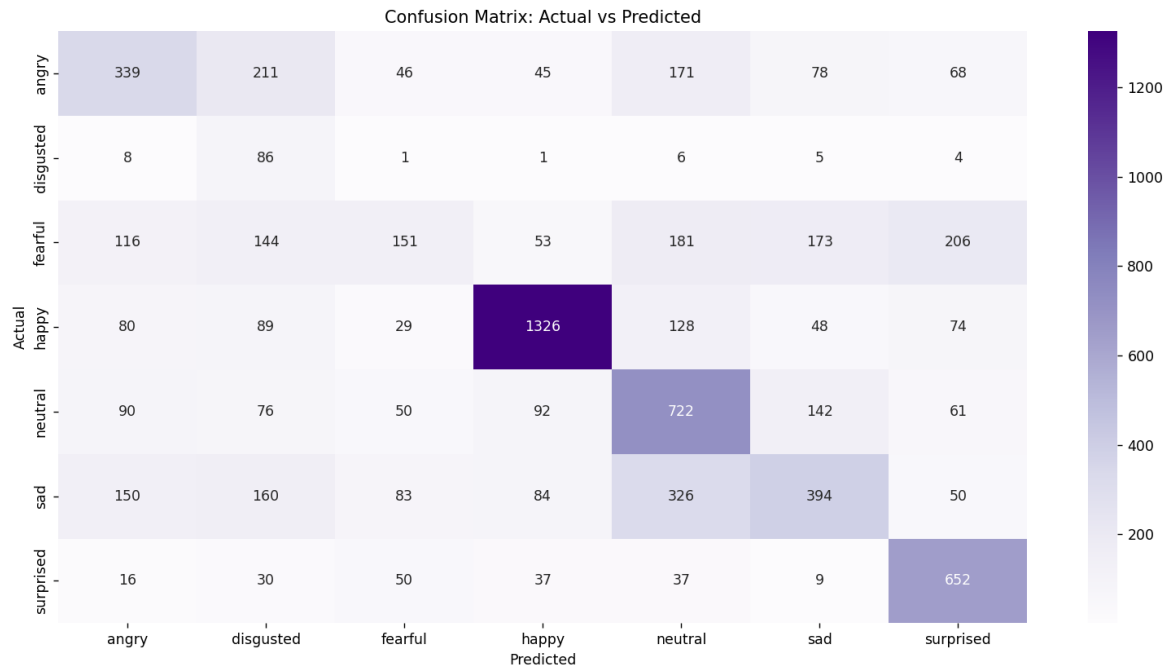
Handling Imbalance: To prevent the model from only predicting "Happy," we calculated **Class Weights** using Scikit-Learn. This forced the model to "pay more attention" to the under-represented classes like Disgust and Fear.

Accuracy Reached: The model achieved a stable accuracy that balances speed and performance, suitable for real-time deployment.

```
train_model.py > ...
  1  import tensorflow as tf
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
(.venv) PS C:\Users\Lenovo\Desktop\Emotion_Recognition> python train_model.py
-04
Epoch 21/30
898/898 [==============================] - 260s 290ms/step - loss: 1.4653 - accuracy: 0.4347 - val_loss: 1.3626 - val_accuracy: 0.4926 - lr: 2.000
0e-04
Epoch 22/30
898/898 [==============================] - 229s 255ms/step - loss: 1.4536 - accuracy: 0.4431 - val_loss: 1.3568 - val_accuracy: 0.5006 - lr: 4.000
0e-05
Epoch 23/30
898/898 [==============================] - 233s 259ms/step - loss: 1.4487 - accuracy: 0.4410 - val_loss: 1.3364 - val_accuracy: 0.5054 - lr: 4.000
0e-05
Epoch 24/30
898/898 [==============================] - 239s 266ms/step - loss: 1.4432 - accuracy: 0.4441 - val_loss: 1.3210 - val_accuracy: 0.5081 - lr: 4.000
0e-05
Epoch 25/30
898/898 [==============================] - 239s 266ms/step - loss: 1.4513 - accuracy: 0.4433 - val_loss: 1.3059 - val_accuracy: 0.5156 - lr: 4.000
0e-05
Epoch 26/30
898/898 [==============================] - 230s 256ms/step - loss: 1.4465 - accuracy: 0.4424 - val_loss: 1.3135 - val_accuracy: 0.5117 - lr: 4.000
0e-05
Epoch 27/30
898/898 [==============================] - 227s 252ms/step - loss: 1.4431 - accuracy: 0.4483 - val_loss: 1.3286 - val_accuracy: 0.5072 - lr: 4.000
0e-05
Epoch 28/30
898/898 [==============================] - 234s 260ms/step - loss: 1.4276 - accuracy: 0.4464 - val_loss: 1.3047 - val_accuracy: 0.5187 - lr: 4.000
0e-05
Epoch 29/30
898/898 [==============================] - 245s 272ms/step - loss: 1.4397 - accuracy: 0.4443 - val_loss: 1.3240 - val_accuracy: 0.5095 - lr: 4.000
0e-05
Epoch 30/30
898/898 [==============================] - 209s 233ms/step - loss: 1.4333 - accuracy: 0.4493 - val_loss: 1.3210 - val_accuracy: 0.5113 - lr: 4.000
0e-05
C:\Users\Lenovo\Desktop\Emotion_Recognition\.venv\lib\site-packages\keras\src\engine\training.py:3103: UserWarning: You are saving your model as a
n HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_mo
del.keras')`.
  saving_api.save_model(
✅ New improved model saved as 'emotion_model.h5'
```

Evaluation

Confusion Matrix: This matrix highlights the model's performance across all 7 classes. It allows us to see if the model confuses "Sad" with "Neutral" or "Angry" with "Fear."



Classification Report:

	precision	recall	f1-score	support
angry	0.42	0.35	0.39	958
disgusted	0.11	0.77	0.19	111
fearful	0.37	0.15	0.21	1024
happy	0.81	0.75	0.78	1774
neutral	0.46	0.59	0.51	1233
sad	0.46	0.32	0.38	1247
surprised	0.58	0.78	0.67	831
accuracy			0.51	7178
macro avg	0.46	0.53	0.45	7178
weighted avg	0.54	0.51	0.51	7178

Final Application (The Dashboard)

Interface: The final product is a web-based dashboard built with Streamlit. It provides a clean, interactive interface where users can upload an image and receive an instant emotional analysis.

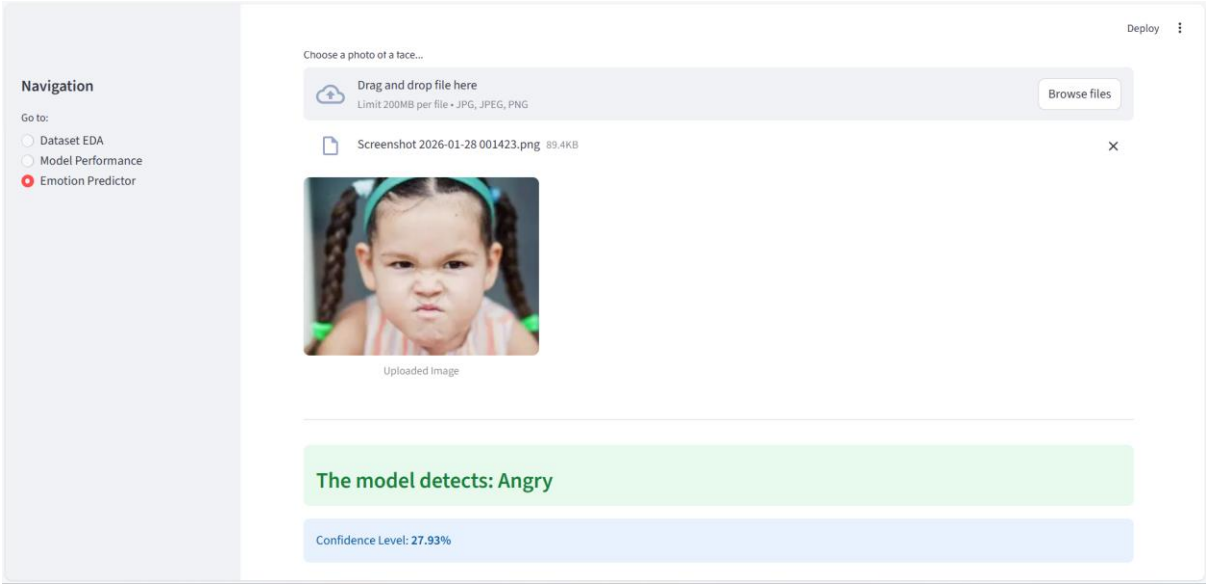
User Flow:

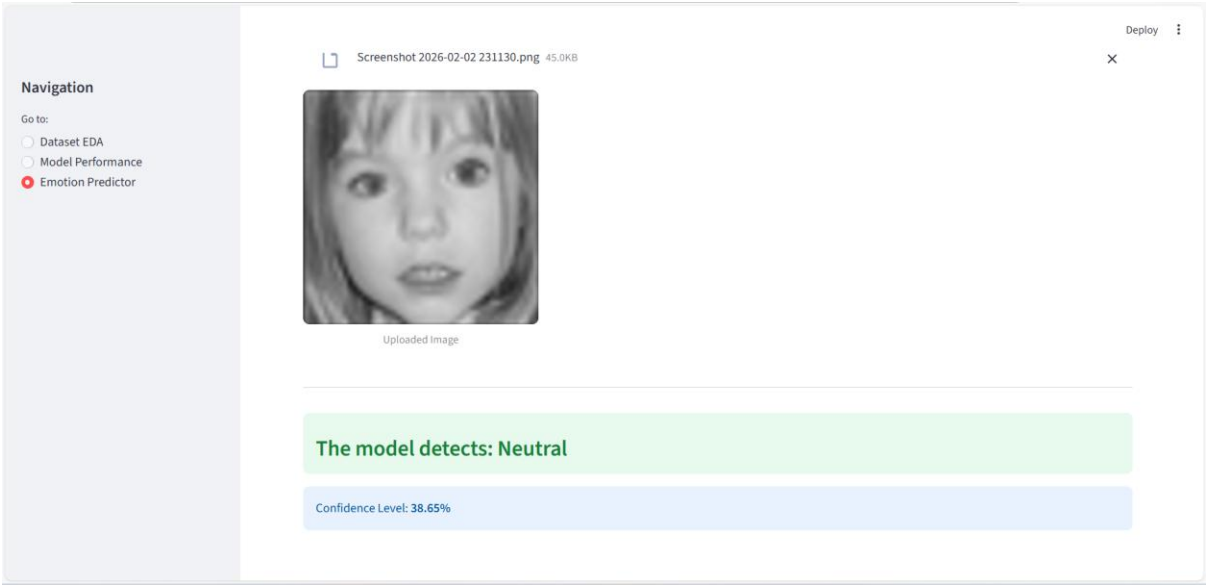
1. Upload: User selects a JPG/PNG image.

2. Preprocessing: The app automatically resizes the image to 224*224 and normalizes it.

3. Prediction: The saved model (.h5) processes the pixels.

4. Display: The app displays the detected emotion with a confidence percentage.




Navigation
Go to:
Dataset EDA
Model Performance
Emotion Predictor
Deploy
Screenshot 2026-02-02 231130.png 45.0KB
Uploaded Image
The model detects: Neutral
Confidence Level: 38.65%


Navigation
Go to:
Dataset EDA
Model Performance
Emotion Predictor
Deploy
Screenshot 2026-02-02 231005.png 51.4KB
Uploaded Image
The model detects: Disgust
Confidence Level: 99.41%

Deploy

Navigation

Go to:

- Dataset EDA
- Model Performance
- Emotion Predictor

Uploaded Image

The model detects: Surprise

Confidence Level: **41.11%**

Deploy

Navigation

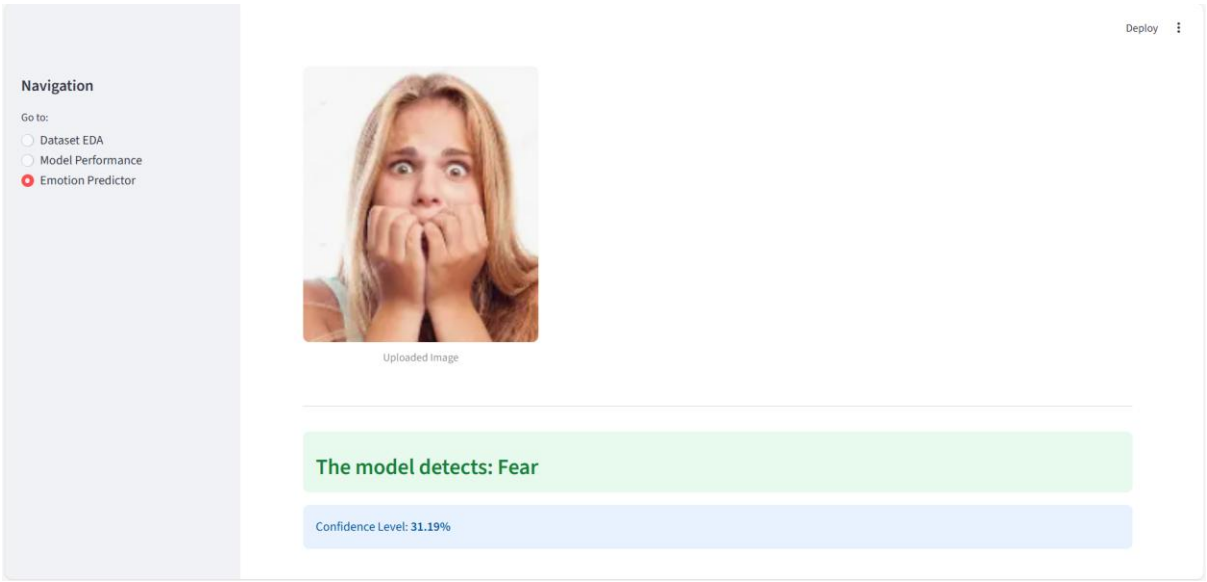
Go to:

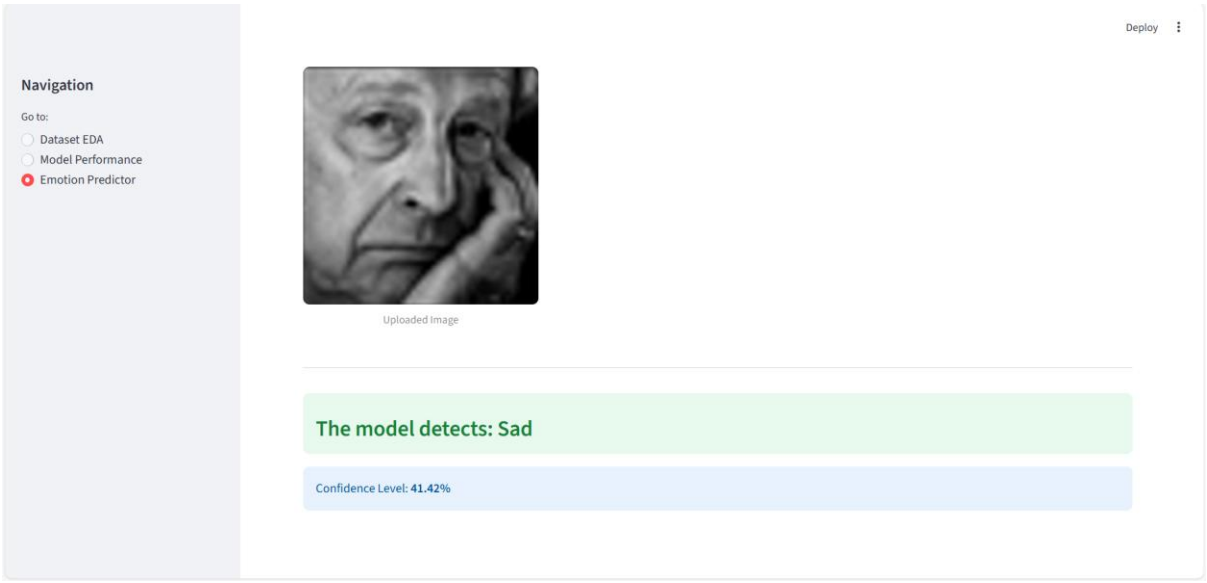
- Dataset EDA
- Model Performance
- Emotion Predictor

Uploaded Image

The model detects: Happy

Confidence Level: **92.18%**





9. Conclusion

The development of this Facial Emotion Recognition system successfully demonstrated the application of **Deep Learning** in human behavior analysis. A significant challenge encountered during the initial stages was **Class Imbalance**, where the dataset exhibited a strong bias toward the "Happy" category. Without intervention, this caused the model to over-predict the majority class.

To address this, the project implemented **Class Weights** to ensure the model learned to recognize under-represented emotions like "Disgust" and "Fear" with equal importance. Instead of using heavy, resource-intensive architectures, a **Custom Convolutional Neural Network (CNN)** was designed and optimized. This approach allowed for a balance between accuracy and computational efficiency, enabling the model to process 48*48 grayscale

images rapidly on standard hardware. The final integration into a **Streamlit** dashboard provides a lightweight, real-time tool for emotion classification, proving that effective AI solutions can be achieved through strategic architecture design and data preprocessing.

Future Scope

While the current Custom CNN provides a solid foundation for emotion recognition, several avenues exist for future enhancement:

- **Live Webcam Integration:** Utilizing OpenCV to capture and process live video streams for real-time emotional tracking.
- **Hybrid Architectures:** Experimenting with a mix of CNNs and Recurrent Neural Networks (RNNs) to analyze sequences of facial movements rather than static images.
- **Edge Deployment:** Further optimizing the model for deployment on low-power devices like smartphones or Raspberry Pi.
- **Larger Datasets:** Incorporating more diverse datasets to improve the model's robustness against variations in lighting, age, and ethnicity.

PROJECT LINK: EMOTION RECOGNITION APP

MODEL LINK: AI_MODEL