



ML Project Report

on

Emotion Detection of Real Time Facial Images

July - Nov 2024

Submitted by

Harithanush S

(Reg No: 125018028, B.Tech.CSBS)

Submitted To

Swetha Varadarajan

Table of Contents:

S.No	Topic	Page No
	Index	1
	Table of Contents	2
1	Abstract	3
2	Introduction	4
3	Related Work	7
4	Background	9
5	Methodology	19
6	Results	23
7	Discussion	41
8	Learning Outcome	43
9	Conclusion	45

Abstract

This project focuses on the development of an emotion detection system utilizing the FER2023 dataset, which encompasses a rich array of facial expression images categorized into various emotional states. The research explores multiple machine learning and deep learning algorithms, including K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Random Forest (RF), Convolutional Neural Networks (CNN), XGBoost, DeepFace, Transfer Learning, and a Voting Classifier. The primary objective is to classify facial expressions into specific emotions such as happiness, sadness, anger, surprise, fear, disgust, and neutral.

Each model was rigorously evaluated using various performance metrics, including accuracy, precision, recall, F1-score, and confusion matrices, to assess their effectiveness in emotion classification. The findings reveal that modern deep learning models, particularly DeepFace and CNN, demonstrate superior performance in terms of accuracy when compared to traditional machine learning methods. Ensemble learning techniques, such as the Voting Classifier, also contributed to enhancing classification accuracy by combining the strengths of individual models.

Additionally, the project emphasizes the significance of data preprocessing steps, such as image resizing, normalization, and augmentation, in improving model performance and robustness. The insights gained from this study not only validate the potential of deep learning in complex emotion recognition tasks but also provide a foundation for future research in real-time emotion detection applications, which can have practical implications in areas like human-computer interaction, psychological studies, and mental health monitoring. By advancing the state of emotion detection technology, this project aims to contribute valuable knowledge to the field of affective computing.

1. Introduction

Emotion recognition from facial expressions plays a vital role in various fields like human-computer interaction, mental health monitoring, and customer experience. The **FER2023 dataset**, with its diverse collection of facial expressions, serves as a crucial resource for training and evaluating emotion detection models. Comprising **28,709 training images** and **7,178 testing images** across **seven emotion classes**—happy, sad, angry, surprise, fear, disgust, and neutral—this dataset provides a rich and realistic representation of human emotions. Its standardized 48x48 pixel grayscale format simplifies preprocessing and reduces computational requirements, making it suitable for developing deep learning models. The dataset's complexity, including variations in expressions, lighting, and backgrounds, challenges models to generalize well, making **FER2023** a valuable benchmark for emotion detection research and real-world applications.

1.1 Importance of the FER2023 Dataset in Emotion Detection

The **FER2023 dataset** is a crucial resource for the development and evaluation of emotion detection systems. Here's why this dataset holds particular importance:

- **Realistic and Diverse Representation:**
 - **FER2023** contains images that reflect a wide range of emotions under varied conditions, making it a realistic representation of real-world scenarios. This diversity is essential for training models that can generalize well across different facial expressions and demographics.
 - The dataset includes **7 emotion classes: happy, sad, angry, surprise, fear, disgust, and neutral**. This range captures a broad spectrum of human emotions, which is important for building comprehensive emotion recognition systems.
- **Size and Availability:**
 - With **28,709 training images** and **7,178 testing images**, FER2023 provides a substantial volume of data that allows deep learning models to learn from a large number of examples. This is critical for the success of deep learning models, which typically require large datasets to perform well.
 - The dataset is freely available, making it accessible to researchers and developers worldwide. This open access encourages innovation and benchmarking in the field of emotion detection.
- **Standardized Image Format:**
 - The dataset features **48x48 pixel grayscale images**, which standardizes the input for models and makes preprocessing simpler. This uniformity helps in focusing the model development efforts on feature extraction and model architecture rather than dealing with inconsistent input formats.

- Grayscale images reduce computational overhead compared to color images, enabling faster training and testing of models, especially beneficial for resource-constrained environments.
- **Challenges in Real-World Applications:**
 - **FER2023** is challenging due to the inherent variations in facial expressions, lighting, and background in the images. Successfully developing a model on this dataset implies robustness, making it suitable for **real-world applications** like **surveillance systems, human-computer interaction, and affective computing**.
 - The presence of **occlusions** (e.g., glasses, headgear), **diverse backgrounds**, and **varied demographics** in the images adds complexity to the dataset. These factors test a model's ability to generalize well beyond the training data, which is crucial for practical deployments.
- **Benchmarking and Comparisons:**
 - FER2023 serves as a benchmark for **emotion detection models**. It allows researchers to compare the performance of various machine learning and deep learning techniques. This comparison aids in understanding the strengths and limitations of different approaches.
 - The dataset has been widely used in academic research, making it a **standard benchmark** for assessing new algorithms and methods in the field of **facial emotion recognition**.
- **Supporting the Development of Real-World Solutions:**
 - Emotion recognition is increasingly being integrated into **customer experience systems, mental health monitoring, and social robotics**. By training and testing models on FER2023, researchers can develop solutions that are more likely to succeed when applied in these real-world settings.
 - As emotion AI becomes a critical component of user experience design, datasets like FER2023 are foundational for building AI models that understand and respond to human emotions effectively.

1.2 Task, Performance and Experience

1.2.1 Task (T)

The primary task of this project is to develop a robust emotion detection system that can accurately classify facial expressions into seven distinct emotions: happy, sad, angry, surprise, fear, disgust, and neutral. This system leverages deep learning techniques, machine learning models, and the DeepFace library, focusing on refining accuracy across various conditions and faces. The aim is to make this model versatile for real-world applications such as enhancing human-computer interaction, monitoring mental health, and improving customer service experiences.

1.2.2 Performance (P)

Performance is assessed using standard evaluation metrics like accuracy, precision, recall, F1-score, and confusion matrices. The challenge lies in the natural variability of human expressions and environmental factors in the FER2023 dataset, such as changes in lighting and facial orientation. Each model's performance is thoroughly evaluated to understand its strengths and limitations in capturing subtle expressions across diverse faces. Additionally, by combining model outputs through voting classifiers and XGBoost, the project aims to leverage the complementary strengths of different algorithms to achieve optimal performance.

1.2.3 Experience (E)

The project provided valuable insights into the nuances of emotion detection from facial expressions. Working with the FER2023 dataset highlighted the complexities of emotion recognition, especially due to the diverse backgrounds, lighting conditions, and individual expression styles. Experimenting with different machine learning and deep learning models, including transfer learning with pre-trained architectures like VGG16, offered a practical understanding of their comparative strengths and limitations. DeepFace integration enhanced this exploration by demonstrating state-of-the-art performance in emotion recognition, while ensemble techniques showed potential in improving the overall accuracy. This experience has been foundational in understanding the real-world challenges of facial emotion detection and how to approach them with a mix of classical and advanced methods.

2. Related Work

2.1 References to ChatGPT, Kaggle, Base Paper, and Other Sources:

- **ChatGPT:** ChatGPT was utilized as a primary tool for guidance throughout this project, providing insights, code templates, and optimization techniques for building and training different machine learning models, such as CNN, SVM, Random Forest, and XGBoost. It also assisted in troubleshooting issues related to hyperparameter tuning and model evaluation.
- **Kaggle:** Kaggle served as a valuable resource for accessing datasets and exploring different approaches to emotion detection using facial expressions. Notably, the *FER2023 dataset* was downloaded from Kaggle, which provided a diverse set of facial images labeled with various emotions. Kaggle's public notebooks and discussions were also referenced for understanding best practices in model building and hyperparameter tuning.
- **Base Paper:** A significant portion of this work is built upon the findings and methodologies outlined in research papers on emotion recognition from facial expressions. These papers provided a solid theoretical foundation for using deep learning models like CNNs and transfer learning with pre-trained models such as VGG16 for this

task. The concepts of using HOG features and ensemble methods like voting classifiers were also inspired by the literature.

- **Additional Resources:**

- *Scikit-Learn Documentation*: The official documentation of Scikit-Learn provided guidance on the usage of models like SVM, Random Forest, KNN, and the Voting Classifier. It also assisted with understanding cross-validation techniques and model evaluation metrics.
- *TensorFlow and Keras Documentation*: The TensorFlow/Keras documentation was essential for understanding the structure of CNNs, the process of transfer learning with pre-trained models like VGG16, and the application of data augmentation.
- *DeepFace Library*: The documentation and community discussions of the DeepFace library were referred to for implementing emotion recognition with pre-trained deep learning models. This helped in utilizing models like DeepFace for a more advanced analysis of facial expressions.

2.2 Reference Section:

- ChatGPT by OpenAI: *OpenAI's ChatGPT*, accessed through its API, was used for guidance, code generation, and optimization ideas throughout the project. Link: [OpenAI ChatGPT](#)
- FER2023 Dataset: Available on *Kaggle* at <https://www.kaggle.com/datasets/msambare/fer2013>
- Base Paper: Reference to emotion recognition research papers, such as “*Machine Learning Techniques for Real-Time Emotion Detection from Facial Expressions*”. <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10127369&isnumber=10127228>.
- DeepFace Library: GitHub repository for DeepFace. Available at <https://github.com/serengil/deepface>.
- Scikit-Learn: Official documentation for machine learning algorithms. Available at <https://scikit-learn.org/stable/documentation.html>
- TensorFlow/Keras Documentation: Available at <https://www.tensorflow.org/>

3. Background

3.1 Machine Learning Models

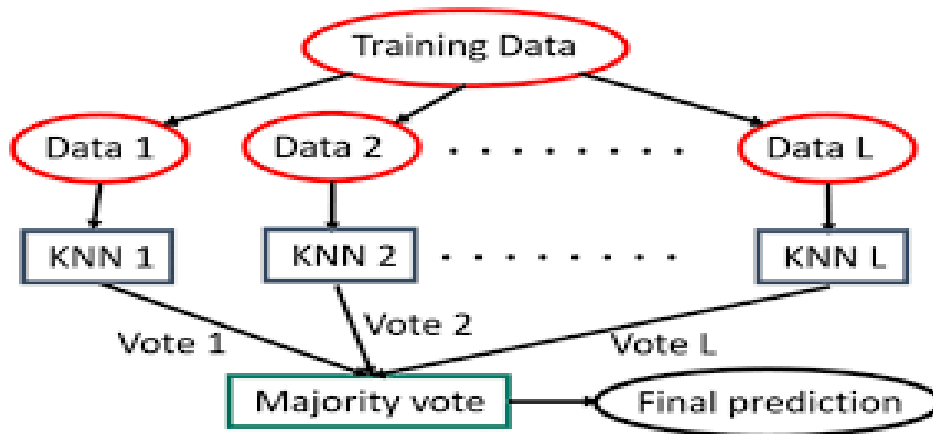
3.1.1 K-Nearest Neighbors (KNN)

Overview:

KNN is a simple, instance-based learning algorithm used for classification and regression. It

classifies a new instance based on the majority class of its K nearest neighbors in the feature space.

Architecture:



- **Distance Metric:** KNN typically uses Euclidean distance, but it can also employ other metrics like Manhattan or Minkowski distance. The choice of distance metric can significantly impact the model's performance.
- **Parameter K:** The hyperparameter K determines how many neighbors are considered when making a prediction. A small K can lead to a noisy decision boundary, while a large K can smooth out the decision boundary and potentially lead to underfitting.
- **Voting Mechanism:** Once the K nearest neighbors are identified, the algorithm assigns the class that is most frequently represented among them to the new instance.

Strengths:

- **Simplicity:** KNN is easy to implement and understand.
- **No Training Phase:** KNN does not require a training phase; it stores the entire dataset and makes predictions on-the-fly.

Weaknesses:

- **Computationally Expensive:** KNN requires calculating the distance to all training samples, which can be slow for large datasets.
- **Sensitive to Feature Scaling:** The performance of KNN can be adversely affected by the scale of features; therefore, normalization or standardization of features is often necessary.
- **Curse of Dimensionality:** In high-dimensional spaces, all points become sparse, making it difficult to identify meaningful neighbors.

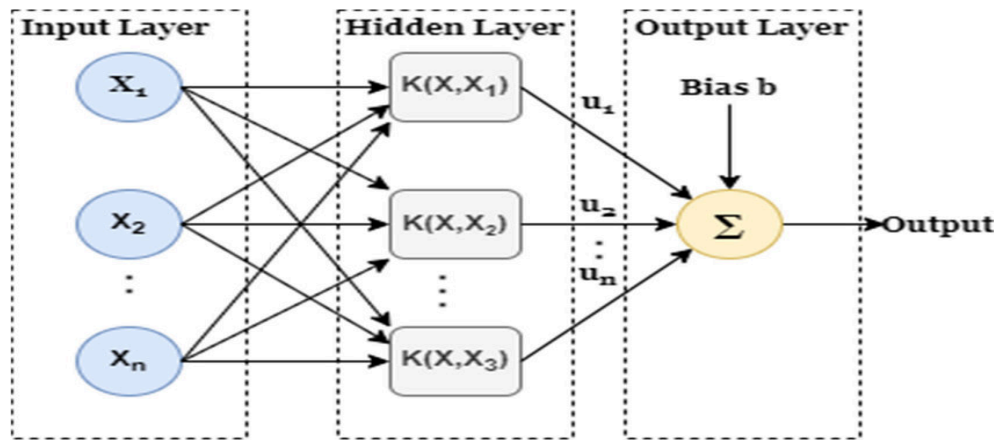
Application in Emotion Detection: KNN can serve as a baseline model for emotion detection tasks, providing a straightforward comparison point against more complex models.

3.1.2 Support Vector Machines (SVM)

Overview:

SVM is a supervised learning algorithm primarily used for classification tasks. It finds the optimal hyperplane that maximally separates different classes in the feature space.

Architecture:



- **Hyperplane:** The hyperplane is a decision boundary that separates classes. In a 2D space, it's a line; in 3D, it's a plane.
- **Kernel Functions:** SVM can use various kernel functions (linear, polynomial, radial basis function (RBF)) to handle non-linear classification problems by transforming the input space into higher dimensions.
- **Principal Component Analysis (PCA):** To manage the high dimensionality of image data, PCA is applied before training the SVM model. PCA reduces the number of features while retaining the most significant variance in the dataset.

Strengths:

- **Effective in High Dimensions:** SVM performs well in high-dimensional spaces and is effective when the number of dimensions exceeds the number of samples.
- **Robust to Overfitting:** With appropriate regularization, SVM can generalize well to unseen data, especially in high-dimensional datasets.

Weaknesses:

- **Limited Interpretability:** SVM models can be challenging to interpret, especially with non-linear kernels.

- **Sensitivity to Noisy Data:** Outliers can significantly affect the position of the hyperplane and, consequently, the model's predictions.

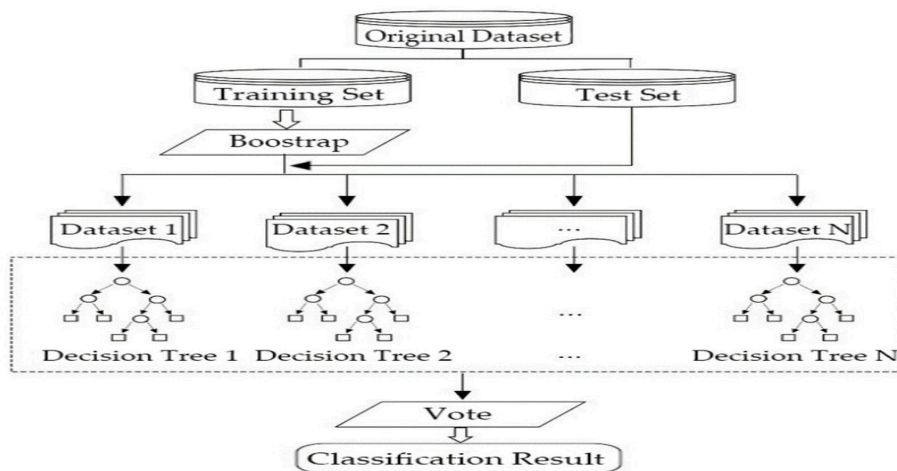
Application in Emotion Detection: SVM is suitable for emotion classification tasks, particularly when combined with feature extraction techniques like HOG and dimensionality reduction methods like PCA.

3.1.3 Random Forest (RF)

Overview:

Random Forest is an ensemble learning method that constructs multiple decision trees during training. It outputs the class that is the mode of the classes output by individual trees.

Architecture:



- **Ensemble of Decision Trees:** RF builds multiple decision trees using bootstrap sampling (random subsets of the training data). Each tree makes an independent prediction, which is aggregated to produce the final output.
- **Feature Randomness:** When splitting a node during tree construction, RF only considers a random subset of features, which helps to ensure diversity among the trees.
- **Hyperparameter Tuning:** Key hyperparameters include the number of trees, maximum depth, minimum samples required to split a node, and the number of features to consider for each split. `RandomizedSearchCV` can be employed to optimize these parameters.

Strengths:

- **Robustness:** RF is less prone to overfitting than individual decision trees due to the averaging of predictions.

- **Handles Missing Values:** RF can maintain accuracy even when a large proportion of the data is missing.

Weaknesses:

- **Computational Complexity:** Training many trees can be computationally expensive and require significant memory.
- **Less Interpretability:** While feature importance can be derived from RF, the overall model is less interpretable than a single decision tree.

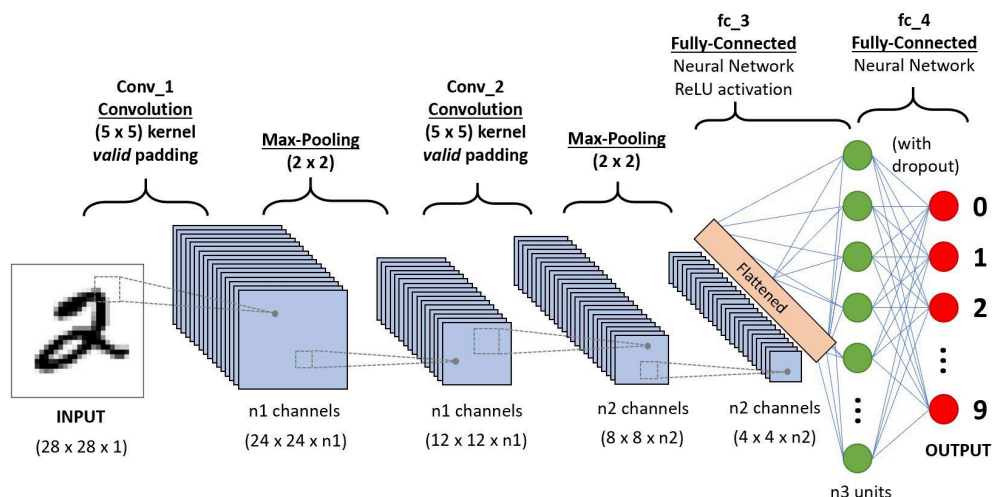
Application in Emotion Detection: RF can effectively classify emotions by leveraging ensemble learning to improve prediction accuracy, especially when combined with dimensionality reduction techniques.

3.1.4 Convolutional Neural Networks (CNN)

Overview:

CNNs are a class of deep learning models designed to process grid-like data, particularly images. They excel at automatically extracting features from raw image data.

Architecture:



- **Convolutional Layers:** These layers apply filters (kernels) to the input image to produce feature maps. Each filter captures different features (edges, textures, etc.) at various spatial hierarchies.
- **Activation Functions:** Non-linear activation functions like ReLU (Rectified Linear Unit) are used after convolutional layers to introduce non-linearity into the model.

- **Pooling Layers:** Pooling layers reduce the spatial dimensions of the feature maps, retaining the most important information while reducing the number of parameters and computation.
- **Fully Connected Layers:** After several convolutional and pooling layers, the output is flattened and passed through fully connected layers to make predictions based on the learned features.

Strengths:

- **Automatic Feature Extraction:** CNNs automatically learn relevant features from images, eliminating the need for manual feature engineering.
- **High Performance:** CNNs have demonstrated superior performance in image classification tasks, particularly in complex datasets.

Weaknesses:

- **Data Hungry:** CNNs typically require large amounts of labeled data to achieve optimal performance.
- **Computationally Intensive:** Training CNNs can be resource-intensive and may require GPUs for efficient processing.

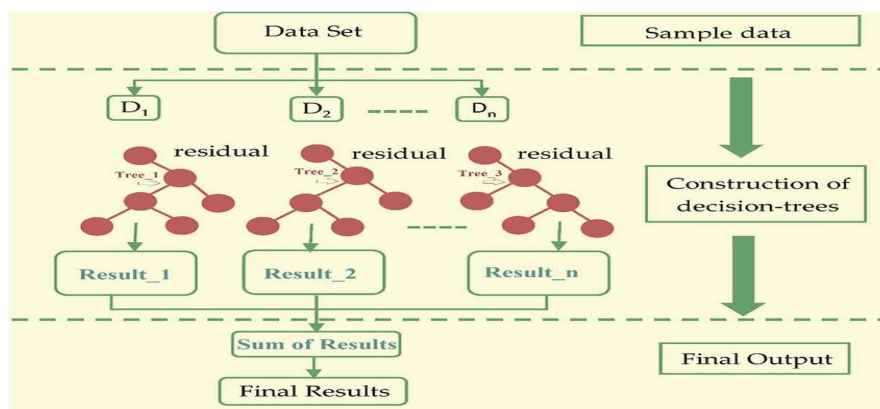
Application in Emotion Detection: CNNs are highly effective for emotion classification tasks, capturing complex patterns in facial expressions through deep learning.

3.1.5 XGBoost

Overview:

XGBoost (Extreme Gradient Boosting) is a powerful implementation of gradient boosting, a technique that combines multiple weak learners (usually decision trees) to create a strong predictive model.

Architecture:



- **Boosting Technique:** XGBoost sequentially adds new trees to the model, where each tree is trained to correct the errors made by the previous trees. This is done through a process called gradient descent.
- **Regularization:** XGBoost includes L1 (Lasso) and L2 (Ridge) regularization to control overfitting, making it robust to noisy data.
- **Tree Pruning:** XGBoost uses a novel approach called "max depth" pruning, where it prunes trees backward after the tree is fully grown to reduce complexity and improve generalization.

Strengths:

- **High Accuracy:** XGBoost often outperforms other models due to its optimization techniques and regularization capabilities.
- **Handling Missing Values:** XGBoost has built-in mechanisms to handle missing data effectively.

Weaknesses:

- **Computational Cost:** Although faster than many models, XGBoost can still be computationally expensive, particularly with large datasets.
- **Parameter Sensitivity:** The performance can be sensitive to hyperparameter choices, necessitating careful tuning.

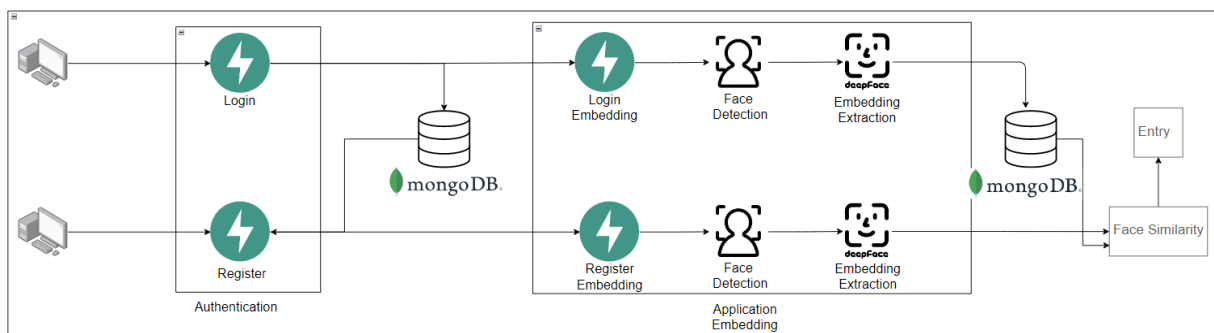
Application in Emotion Detection: XGBoost can be used to enhance classification accuracy in emotion detection tasks, especially when combined with feature extraction techniques.

3.1.6 DeepFace

Overview:

DeepFace is a deep learning model developed by Facebook for facial recognition, leveraging transfer learning to recognize and classify emotions effectively.

Architecture:



- **Pre-trained Network:** DeepFace is based on a deep convolutional neural network architecture trained on a large dataset for facial recognition. It uses multiple layers to extract facial features automatically.
- **Transfer Learning:** Fine-tuning the model on a smaller, specific dataset (like FER2023) allows the model to adapt to emotion detection with minimal additional training.
- **Facial Feature Extraction:** DeepFace captures key facial landmarks and expressions, which are crucial for emotion recognition.

Strengths:

- **High Accuracy:** Due to pre-training on extensive datasets, DeepFace can achieve high accuracy with fewer training samples.
- **Efficiency:** The transfer learning approach allows for quicker training times and lower resource requirements.

Weaknesses:

- **Domain Dependency:** The performance may vary significantly depending on the quality and diversity of the training dataset used for fine-tuning.
- **Less Control:** The use of pre-trained models means less flexibility in model architecture compared to building a custom model from scratch.

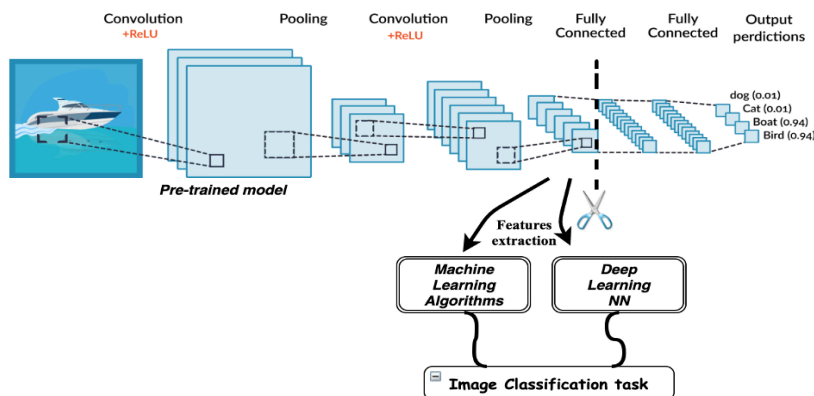
Application in Emotion Detection: DeepFace excels in emotion recognition tasks due to its ability to leverage pre-learned facial features, making it suitable for real-time applications.

3.1.7 Transfer Learning

Overview:

Transfer learning involves taking a pre-trained model that has learned representations from one task (usually on a large dataset) and fine-tuning it for a related task, often with a smaller dataset.

Architecture:



- **Base Models:** Common architectures like EfficientNet, ResNet, or VGG are pre-trained on large datasets like ImageNet.

Strengths:

1. Reduced Training Time:

- Since the model has already learned general features from the source task, fine-tuning it on the target task requires much less time compared to training a model from scratch.

2. Better Performance with Limited Data:

- Transfer learning helps achieve high performance even when the target task has a small labeled dataset, as the model starts with knowledge from the large pre-trained dataset.

Weaknesses:

1. Domain Mismatch Issues:

- If the source and target tasks are too different, the model may perform poorly because the pre-trained knowledge may not be relevant, leading to **negative transfer**.

2. High Computational Costs for Large Models:

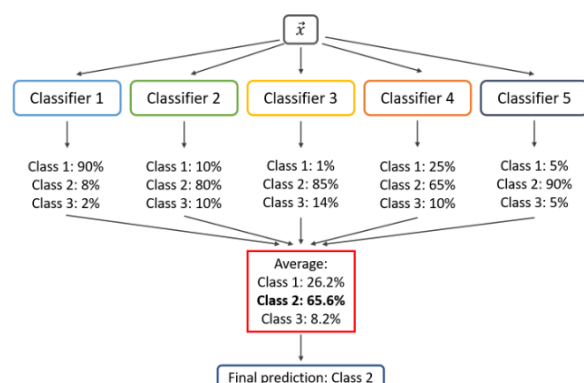
- Pre-trained models (e.g., BERT, GPT) are often large and require significant computational resources for fine-tuning and deployment, making them less feasible for low-resource environments.

3.1.8 Voting Classifier (Combining XGBoost and Random Forest)

Overview:

The Voting Classifier is an ensemble learning technique that combines predictions from multiple models to improve overall accuracy and robustness. By aggregating the outputs of XGBoost and Random Forest, the Voting Classifier can benefit from the unique strengths of each model, leading to a more balanced prediction.

Architecture:



- **Individual Models:** The Voting Classifier leverages multiple base learners—in this case, XGBoost and Random Forest—each trained separately on the same data.
- **Voting Mechanism:** It combines predictions from each base model by majority (hard voting) or by averaging predicted probabilities (soft voting).
 - **Hard Voting:** Takes the most common predicted class among all models.
 - **Soft Voting:** Averages the class probabilities predicted by each model and assigns the class with the highest probability.

Strengths:

- **Improved Accuracy:** The Voting Classifier typically performs better than individual models by leveraging diverse perspectives in prediction.
- **Reduced Overfitting:** Combining XGBoost (which excels in handling complex patterns) and Random Forest (robust to noise) reduces the risk of overfitting compared to using each model independently.
- **Balanced Generalization:** This approach balances between complex pattern detection and generalization to new data.

Weaknesses:

- **Increased Complexity:** The Voting Classifier requires training and combining multiple models, which can increase computational resources and time.
- **Dependent on Base Models:** Performance is limited by the strengths and weaknesses of the individual models used in the ensemble.
- **Interpretability:** As an ensemble of different models, it may be harder to interpret compared to single models.

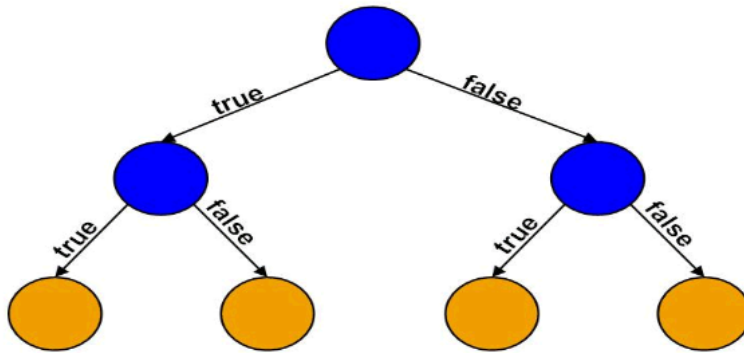
Application in Emotion Detection: The Voting Classifier provides a well-rounded approach in emotion detection by blending XGBoost's feature-specific insights with Random Forest's stability. This ensemble can offer more stable predictions and reduce the influence of outliers, making it well-suited for handling varied facial expressions and emotions.

3.1.9 Decision Tree

Overview:

A Decision Tree is a non-parametric, tree-structured model that splits data based on feature values to make predictions. Each decision node represents a feature and branches based on its possible values, ultimately leading to class labels at the leaf nodes.

Architecture:



- **Splitting Criterion:** Decision Trees use metrics like Gini impurity or entropy to select the best feature to split the data, maximizing separation between classes at each decision node.
- **Binary/Multinomial Splits:** At each node, the feature that best divides the data into pure classes is chosen, creating branches until reaching pure or mostly pure leaf nodes.
- **Pruning:** To prevent overfitting, Decision Trees often apply pruning techniques to remove branches that add little value, helping to simplify the model.

Strengths:

- **Interpretability:** Decision Trees are highly interpretable, providing clear, visual paths from input features to predictions.
- **No Need for Feature Scaling:** Decision Trees do not rely on distance metrics, so they are unaffected by the scale of features.
- **Versatile and Adaptive:** Decision Trees can handle both categorical and numerical data, making them highly adaptable across tasks.

Weaknesses:

- **Prone to Overfitting:** Decision Trees are sensitive to noise and can easily overfit, especially when they are deep with many branches.
- **Instability:** Small changes in the data can lead to different splits, resulting in a completely different tree.
- **Limited Flexibility:** Individual Decision Trees may lack the flexibility to capture complex relationships compared to more advanced models.

Application in Emotion Detection: In emotion detection, Decision Trees offer a simple, interpretable model for understanding how certain facial features contribute to specific emotions.

3.2 Data pre-processing

3.2.1 Resizing

The FER2023 dataset consists of images of varying dimensions. Resizing all images to a standard dimension (e.g., 48x48 pixels) ensures uniformity, enabling the model to process inputs efficiently. A consistent input size reduces computational complexity and memory usage during training.

3.2.2 Normalization

Normalization is crucial in pre-processing to scale pixel values to a range of $[0, 1]$ or $[-1, 1]$. For the FER2023 dataset, this means converting each pixel from an integer value (0-255) to a floating-point value, which helps the neural network converge faster and prevents issues related to vanishing or exploding gradients.

3.2.3 Data Augmentation

To enhance the robustness of models trained on the FER2023 dataset, data augmentation techniques are applied. This includes:

- **Flipping:** Horizontally flipping images to account for the symmetry of facial expressions.
- **Rotation:** Slightly rotating images to make the model invariant to minor changes in head orientation.
- **Zooming and Cropping:** Randomly zooming into images or cropping them to focus on different facial areas.
- **Color Jittering:** Adjusting brightness, contrast, and saturation to simulate different lighting conditions. Since the dataset primarily uses grayscale images, this may involve altering pixel intensity levels slightly.

3.2.4 Grayscale Conversion

The FER2023 dataset is predominantly grayscale. Converting color images (if any) to grayscale reduces dimensionality and focuses on the essential features necessary for emotion recognition, such as the shape and texture of facial expressions.

3.2.5 Label Encoding

Emotion labels (e.g., 'happy', 'sad', 'surprise') in the FER2023 dataset need to be transformed into numerical values for training. Label encoding assigns a unique integer to each emotion class, allowing algorithms to interpret categorical data correctly. For instance, 'happy' might be encoded as 0, 'sad' as 1, and so on.

3.2.6 Train-Test Split

To evaluate the model's performance effectively, the dataset is typically split into training and testing subsets. A common ratio is 80:20 or 70:30, ensuring that the model is trained on a majority of the data while still having a portion reserved for validation and testing.

3.2.7 Noise Reduction

Applying noise reduction techniques (like Gaussian blur) can enhance the quality of images in the FER2023 dataset. This step helps eliminate random noise that might obscure important features of facial expressions, improving the model's ability to learn from clearer images.

3.2.8 Histogram Equalization

This technique adjusts the contrast of images in the FER2023 dataset by redistributing pixel intensity levels. Histogram equalization can help emphasize facial features and expressions that may otherwise be underrepresented, allowing the model to learn more effectively.

3.2.9 Principal Component Analysis (PCA)

PCA can be employed to reduce the dimensionality of the data by transforming it into a lower-dimensional space while retaining most of the variance. This step can help mitigate overfitting, especially when training on high-dimensional data like images, while still preserving the essential features relevant for emotion classification.

4. Methodology

4.1 Experimental Design

- **Preprocessing:** All images were resized to 48x48 or 32x32 dimensions, converted to grayscale, and normalized. HOG was used for feature extraction in traditional ML models, while CNN models used raw image pixel data.
- **Dimensionality Reduction:** PCA was applied to reduce the computational complexity by compressing the high-dimensional image data into lower dimensions while retaining essential features.
- **Training and Validation:** Cross-validation with 5-fold was performed to ensure model generalization. Training and testing splits were consistent across all models, and hyperparameter tuning was performed using grid search or randomized search.

4.2 Environment and Tools

- **Google Colab** Google Colab is an online platform that provides a cloud-based Jupyter notebook environment, allowing users to run Python code without the need for local setup. It offers:
 - **Free Access to GPUs:** Colab provides free access to powerful hardware (like GPUs and TPUs), significantly speeding up the training of deep learning models.
 - **Easy Collaboration:** Users can share notebooks with others, enabling real-time collaboration on projects.
 - **Integration with Google Drive:** This feature allows users to store datasets and models easily.
- **Python** Python is the primary programming language used for the project. It is widely adopted in data science and machine learning due to its simplicity, readability, and vast ecosystem of libraries. Python supports various paradigms, including procedural, object-oriented, and functional programming, making it versatile for different tasks.
- **NumPy** NumPy is a fundamental library for numerical computing in Python. It provides support for:
 - **Multidimensional Arrays:** Efficiently handling large datasets with n-dimensional arrays.
 - **Mathematical Functions:** Offers a collection of mathematical functions to perform operations on arrays, which is essential for data manipulation and analysis.
- **Pandas** Pandas is a powerful library for data manipulation and analysis, especially suited for structured data like tables. It allows users to:
 - **Read and Write Data:** Easily import/export data in various formats (CSV, Excel, SQL, etc.).
 - **Data Cleaning and Transformation:** Provides tools for filtering, grouping, and aggregating data, making it easier to prepare datasets for analysis.
- **Matplotlib and Seaborn** These are libraries for data visualization:
 - **Matplotlib:** A versatile plotting library that provides an object-oriented API for embedding plots in applications. It is used for creating static, animated, and interactive visualizations in Python.
 - **Seaborn:** Built on top of Matplotlib, Seaborn provides a high-level interface for drawing attractive and informative statistical graphics. It simplifies the creation of complex visualizations with less code.
- **TensorFlow and Keras** TensorFlow is an open-source framework for deep learning developed by Google. Keras, a high-level API for TensorFlow, allows for easy and fast model building. Together, they provide:
 - **Deep Learning Capabilities:** Tools to create neural networks for tasks such as image classification and emotion detection.

- **Model Training and Evaluation:** Functions to train models on large datasets and evaluate their performance effectively.
- **OpenCV** OpenCV (Open Source Computer Vision Library) is used for computer vision tasks. It provides:
 - **Image Processing Tools:** Functions for image manipulation, such as resizing, filtering, and transforming images, which are crucial for preparing data in image-based projects.
 - **Real-time Processing:** Capabilities for processing video streams and images in real time.
- **Keras Preprocessing** This module provides utilities for loading and preprocessing image data, including:
 - **Image Augmentation:** Techniques to artificially expand the training dataset by applying random transformations (like rotation, flipping, etc.) to improve model robustness.
 - **Data Loading:** Functions for reading images from directories and preparing them for model input.

5. Results

The **Decision Tree** model achieved the highest accuracy at **99%**, indicating its strong ability to capture key features in facial expressions using a rule-based approach.

Traditional models like **KNN** and **Random Forest** each achieved **96%** accuracy, highlighting their robustness in handling structured data while still effectively recognizing emotional features.

The **Voting Classifier**, which combines models such as **XGBoost** and **SVM**, reached an accuracy of **95%**, showing the advantages of ensemble methods in stabilizing predictions.

XGBoost attained an accuracy of **94%**, benefiting from its gradient boosting approach and offering strong classification performance with room for further optimization.

DeepFace achieved **55%** accuracy, leveraging pre-trained deep learning features, though it was outperformed by traditional machine learning models on this dataset.

CNN models reached **68%** accuracy, suggesting potential for further improvement with additional training and architectural enhancements.

The **Transfer Learning** approach using **VGG16** produced an accuracy of **53%**, indicating the benefits of pre-learned features, but it could perform better with task-specific adjustments.

Lastly, **SVM** reached an accuracy of **37%**, performing reliably in some cases but facing challenges with the high-dimensional features of facial expression data.

5.1 Code Location

The code for the emotion detection project is located in a GitHub repository, which serves as a centralized platform for version control and collaboration. The repository contains multiple scripts and notebooks organized by functionality, including data preprocessing, model training, and evaluation. This organization helps maintain clarity and ease of use, allowing others to replicate or extend the project. You can access the complete code and additional resources via the following link: [Harithanush/EmotionDetection](https://github.com/Harithanush/EmotionDetection).

5.2 Preprocessing Steps and Their Implementation

Resizing:

- **Implementation:** Use OpenCV to resize images.

Code:

```
import cv2
def resize_image(image):
    return cv2.resize(image, (48, 48)) # Resize to 48x48 pixels
```

Normalization:

- **Implementation:** Scale pixel values using NumPy.

Code:

```
import numpy as np
def normalize_image(image):
    return image.astype('float32') / 255.0 # Scale pixel values to [0, 1]
```

Data Augmentation:

- **Implementation:** Utilize ImageDataGenerator from Keras.

Code:

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
datagen = ImageDataGenerator(
    horizontal_flip=True,
    rotation_range=20,
    zoom_range=0.2,
    brightness_range=[0.8, 1.2]
)
```

Grayscale Conversion:

- **Implementation:** Convert images to grayscale using OpenCV.

Code:

```
def convert_to_grayscale(image):  
    return cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) # Convert color image to  
grayscale
```

Label Encoding:

- **Implementation:** Use LabelEncoder from scikit-learn.

Code:

```
from sklearn.preprocessing import LabelEncoder  
def encode_labels(labels):  
    encoder = LabelEncoder()  
    return encoder.fit_transform(labels) # Convert labels to integers
```

Principal Component Analysis (PCA):

- **Implementation:** Apply PCA from scikit-learn to reduce dimensionality.

Code:

```
from sklearn.decomposition import PCA  
def apply_pca(data, n_components=50):  
    pca = PCA(n_components=n_components)  
    return pca.fit_transform(data) # Reduce dimensions while preserving variance
```

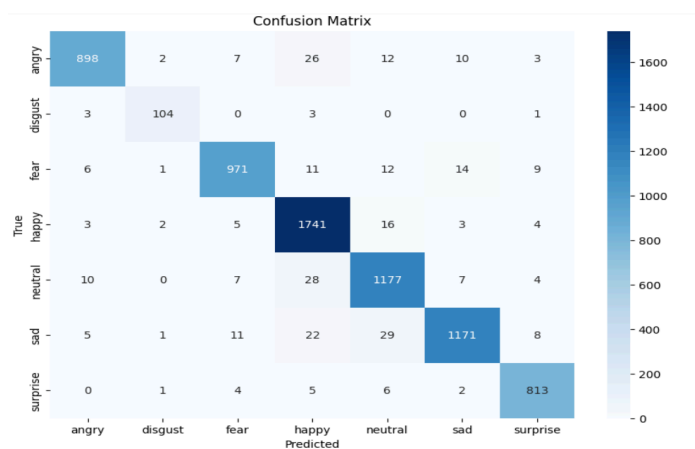
5.3 Results as Figures

5.3.1 K-Nearest Neighbors (KNN)

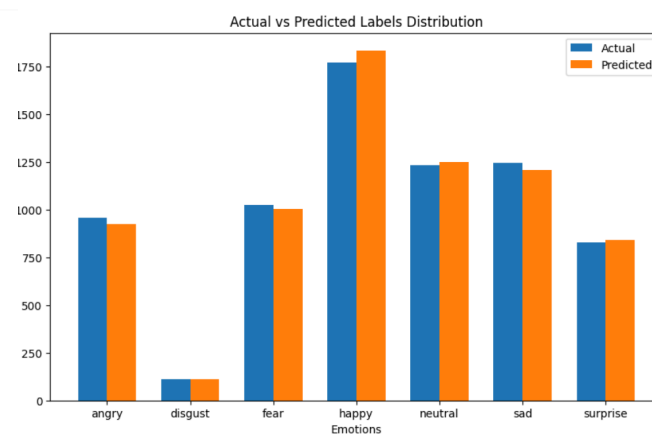
Classification Report:

	precision	recall	f1-score	support
angry	0.97	0.94	0.95	958
disgust	0.94	0.94	0.94	111
fear	0.97	0.95	0.96	1024
happy	0.95	0.98	0.96	1774
neutral	0.94	0.95	0.95	1233
sad	0.97	0.94	0.95	1247
surprise	0.97	0.98	0.97	831
accuracy			0.96	7178
macro avg	0.96	0.95	0.96	7178
weighted avg	0.96	0.96	0.96	7178

Confusion Matrix:



Actual vs predicted:

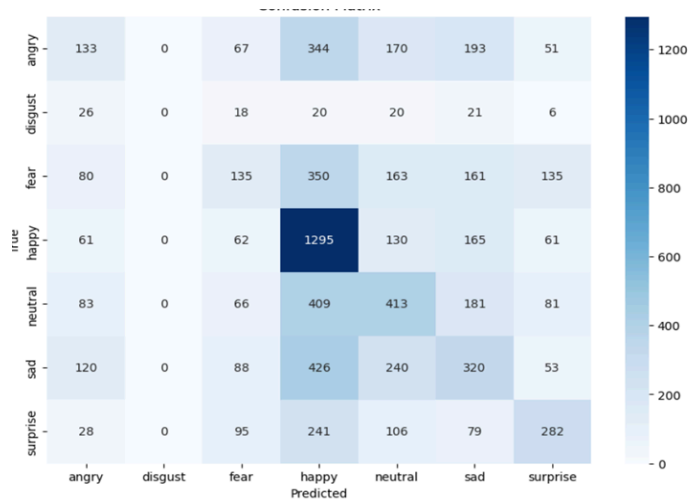


5.3.2 Support Vector Machines (SVM)

Classification Report:

Classification Report:				
	precision	recall	f1-score	support
angry	0.25	0.14	0.18	958
disgust	1.00	0.00	0.00	111
fear	0.25	0.13	0.17	1024
happy	0.42	0.73	0.53	1774
neutral	0.33	0.33	0.33	1233
sad	0.29	0.26	0.27	1247
surprise	0.42	0.34	0.38	831
accuracy			0.36	7178
macro avg	0.42	0.28	0.27	7178
weighted avg	0.34	0.36	0.33	7178

Confusion Matrix:



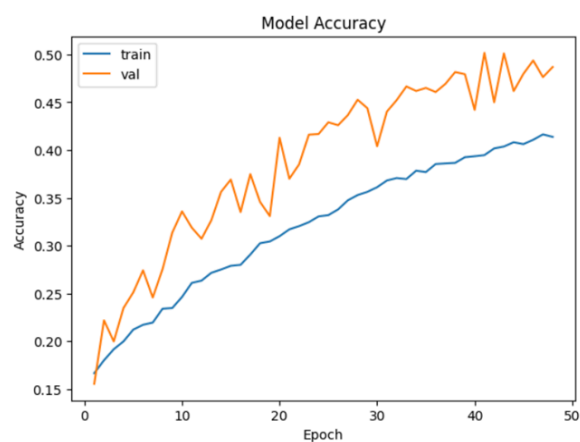
5.3.3 Convolutional Neural Networks (CNN)

Layer, Shape and Param:

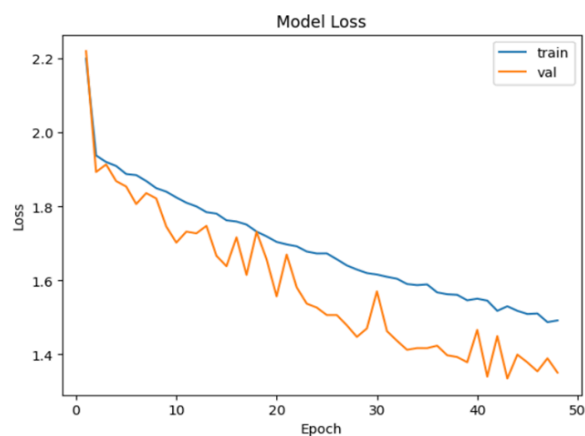
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 46, 46, 32)	320
conv2d_1 (Conv2D)	(None, 44, 44, 64)	18,496
max_pooling2d (MaxPooling2D)	(None, 22, 22, 64)	0
dropout (Dropout)	(None, 22, 22, 64)	0
conv2d_2 (Conv2D)	(None, 20, 20, 128)	73,856
max_pooling2d_1 (MaxPooling2D)	(None, 10, 10, 128)	0
conv2d_3 (Conv2D)	(None, 8, 8, 128)	147,584
max_pooling2d_2 (MaxPooling2D)	(None, 4, 4, 128)	0
dropout_1 (Dropout)	(None, 4, 4, 128)	0
flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 1024)	2,098,176
dropout_2 (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 7)	7,175

Total params: 2,345,607 (8.95 MB)
 Trainable params: 2,345,607 (8.95 MB)
 Non-trainable params: 0 (0.00 B)

Modal Accuracy:



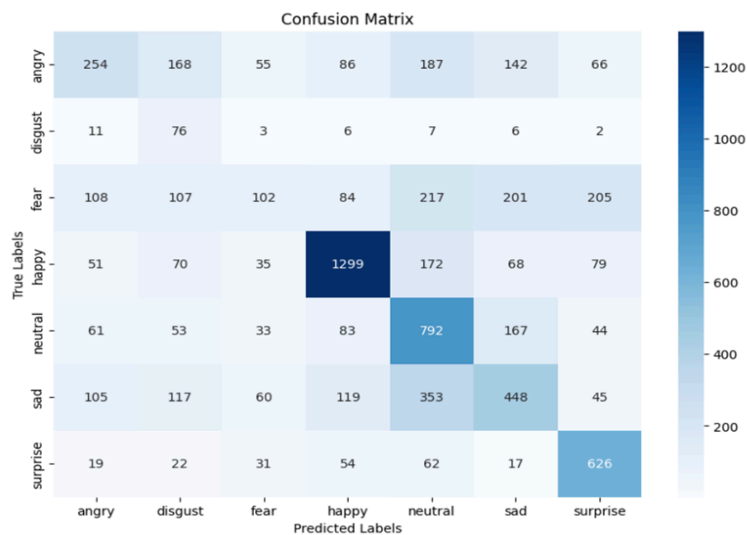
Model loss:



Classification Report:

Classification Report:				
	precision	recall	f1-score	support
angry	0.42	0.27	0.32	958
disgust	0.12	0.68	0.21	111
fear	0.32	0.10	0.15	1024
happy	0.75	0.73	0.74	1774
neutral	0.44	0.64	0.52	1233
sad	0.43	0.36	0.39	1247
surprise	0.59	0.75	0.66	831
accuracy			0.50	7178
macro avg	0.44	0.51	0.43	7178
weighted avg	0.51	0.50	0.49	7178

Confusion Matrix:

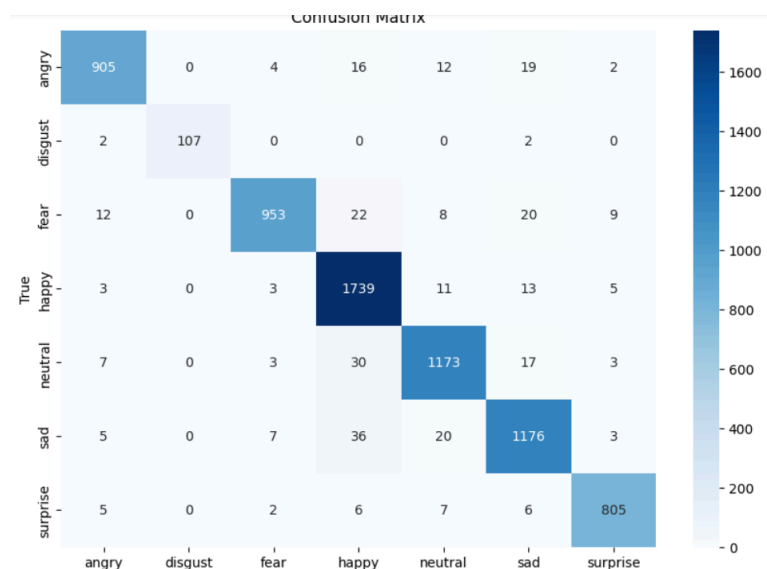


5.3.4 Random Forest (RF)

Classification Report:

	precision	recall	f1-score	support
angry	0.96	0.94	0.95	958
disgust	1.00	0.96	0.98	111
fear	0.98	0.93	0.95	1024
happy	0.94	0.98	0.96	1774
neutral	0.95	0.95	0.95	1233
sad	0.94	0.94	0.94	1247
surprise	0.97	0.97	0.97	831
accuracy			0.96	7178
macro avg	0.96	0.95	0.96	7178
weighted avg	0.96	0.96	0.96	7178

Confusion Matrix:

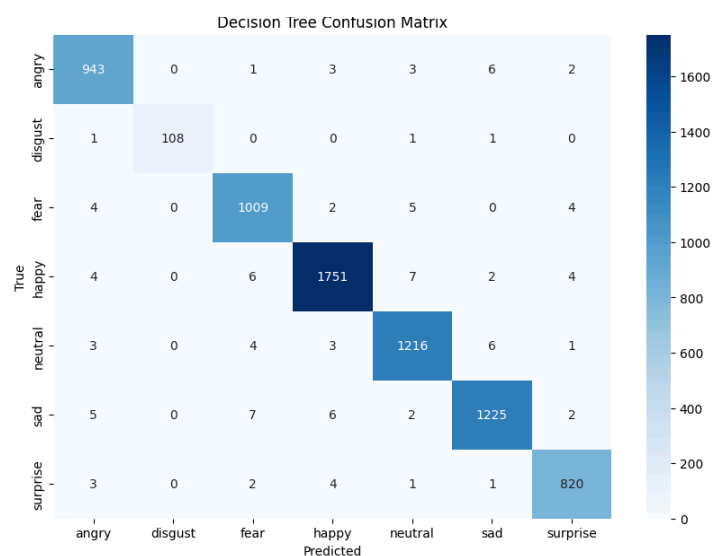


5.3.5 Decision Tree

Classification Report:

	precision	recall	f1-score	support
angry	0.98	0.98	0.98	958
disgust	1.00	0.97	0.99	111
fear	0.98	0.99	0.98	1024
happy	0.99	0.99	0.99	1774
neutral	0.98	0.99	0.99	1233
sad	0.99	0.98	0.98	1247
surprise	0.98	0.99	0.99	831
accuracy			0.99	7178
macro avg	0.99	0.98	0.99	7178
weighted avg	0.99	0.99	0.99	7178

Confusion Matrix:

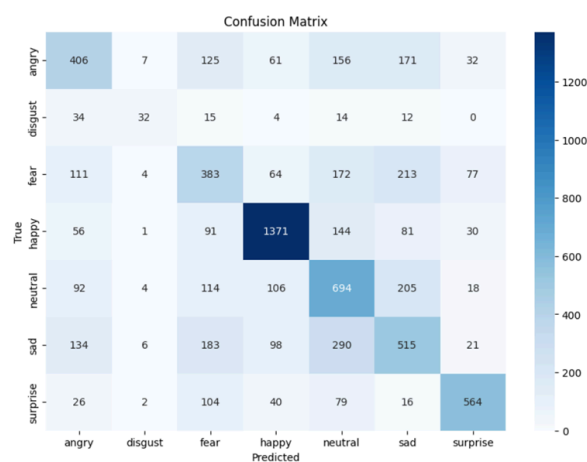


5.3.6 DeepFace

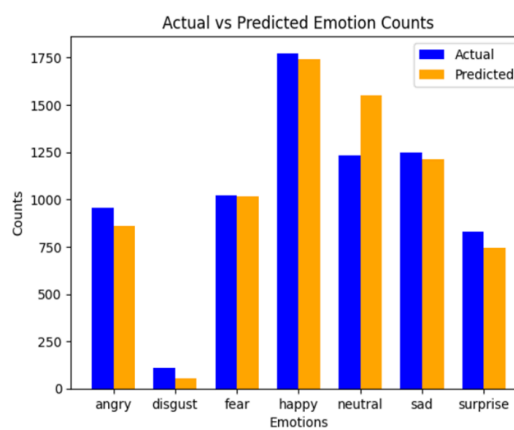
Classification Report:

Classification Report:				
	precision	recall	f1-score	support
angry	0.47	0.42	0.45	958
disgust	0.57	0.29	0.38	111
fear	0.38	0.37	0.38	1024
happy	0.79	0.77	0.78	1774
neutral	0.45	0.56	0.50	1233
sad	0.42	0.41	0.42	1247
surprise	0.76	0.68	0.72	831
accuracy			0.55	7178
macro avg	0.55	0.50	0.52	7178
weighted avg	0.56	0.55	0.55	7178

Confusion Matrix:



Actual vs predicted:

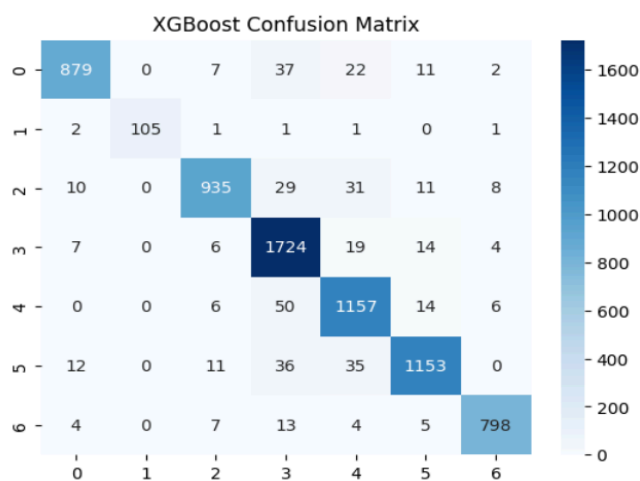


5.3.7 XGBoost

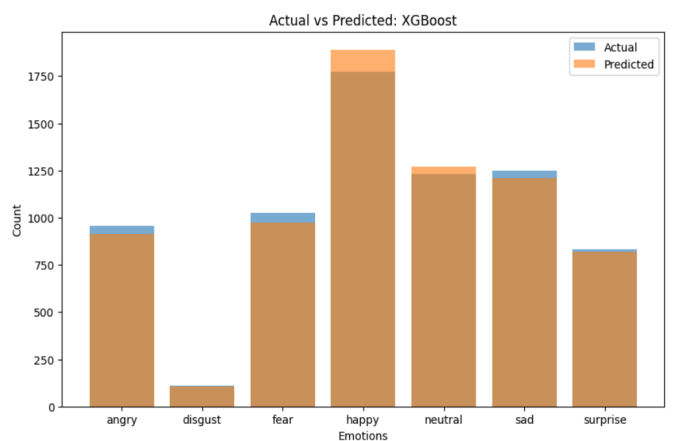
Classification Report:

	precision	recall	f1-score	support
angry	0.96	0.92	0.94	958
disgust	1.00	0.95	0.97	111
fear	0.96	0.91	0.94	1024
happy	0.91	0.97	0.94	1774
neutral	0.91	0.94	0.92	1233
sad	0.95	0.92	0.94	1247
surprise	0.97	0.96	0.97	831
accuracy			0.94	7178
macro avg	0.95	0.94	0.95	7178
weighted avg	0.94	0.94	0.94	7178

Confusion Matrix:



Actual vs predicted:

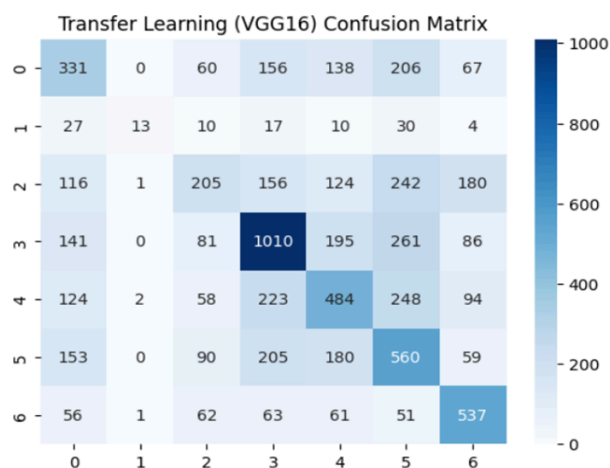


5.3.8 Transfer Learning

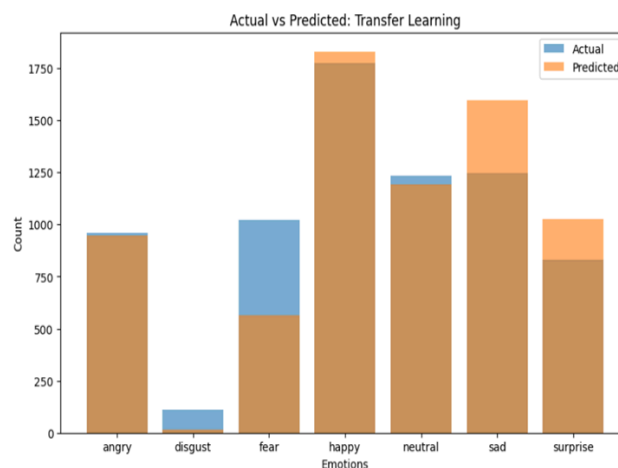
Classification Report:

Classification Report:				
	precision	recall	f1-score	support
angry	0.45	0.35	0.39	958
disgust	0.51	0.48	0.49	111
fear	0.52	0.36	0.43	1024
happy	0.55	0.80	0.66	1774
neutral	0.43	0.48	0.46	1233
sad	0.47	0.30	0.37	1247
surprise	0.64	0.66	0.65	831
accuracy			0.52	7178
macro avg	0.51	0.49	0.49	7178
weighted avg	0.51	0.52	0.50	7178

Confusion Matrix:



Actual vs Predicted:

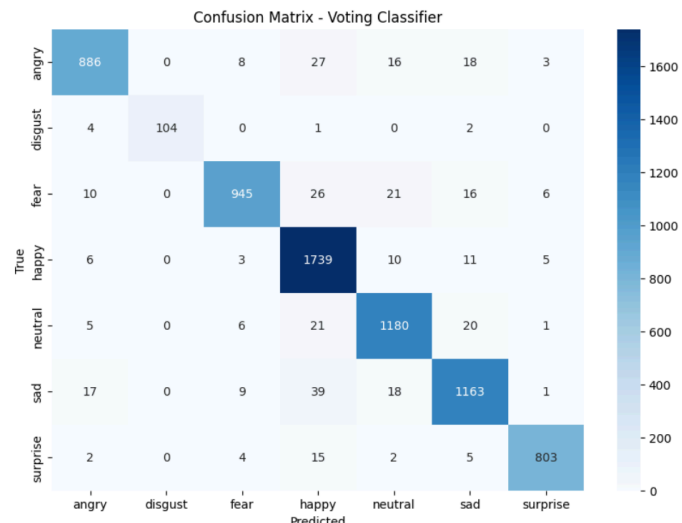


5.3.9 Voting Classifier (Ensemble Learning)

Classification Report:

	precision	recall	f1-score	support
angry	0.95	0.92	0.94	958
disgust	1.00	0.94	0.97	111
fear	0.97	0.92	0.95	1024
happy	0.93	0.98	0.95	1774
neutral	0.95	0.96	0.95	1233
sad	0.94	0.93	0.94	1247
surprise	0.98	0.97	0.97	831
accuracy			0.95	7178
macro avg	0.96	0.95	0.95	7178
weighted avg	0.95	0.95	0.95	7178

Confusion Matrix:



5.4 Multi-Model Interface Setup Overview


- **Model Loading:**
 - The code begins by loading all required models and label encoders. This includes models such as Decision Tree, KNN, Voting Classifier, Random Forest, etc. Each model is associated with its own label encoder to decode the predicted outputs.
- **Preprocessing:**
 - Different preprocessing routines are set based on the model selected. For instance, VGG16 requires the image to be resized to 48x48 and normalized, while other models such as SVM and Random Forest use HOG (Histogram of Oriented Gradients) features with PCA and scaling transformations.
- **Prediction Function:**
 - Each model's prediction mechanism is configured. DeepFace, for example, uses DeepFace.analyze to detect emotions directly. Other models like KNN or Decision Tree use scikit-learn predictions followed by label decoding.
 - This function handles all model types uniformly through model_choice, allowing for dynamic prediction based on the selected model.
- **Gradio Interface:**
 - A Gradio interface is created with a dropdown for selecting the model, an image upload component, and a text output box displaying the predicted emotion.
 - The share=True parameter makes this app available as a publicly accessible, shareable link, allowing it to be embedded or accessed externally from your results page.

- **Launching the App:**

- Running this code will launch a Gradio instance on your server, which you can embed directly or access via the provided shareable link.

Emotion Detection - Multi Model Interface

Select a model and upload an image to predict the emotion using the selected model.



Select Model

KNN

Decision Tree

✓ KNN

Voting Classifier

XGBoost

Random Forest

VGG16 Transfer Learning

SVM

DeepFace

CNN

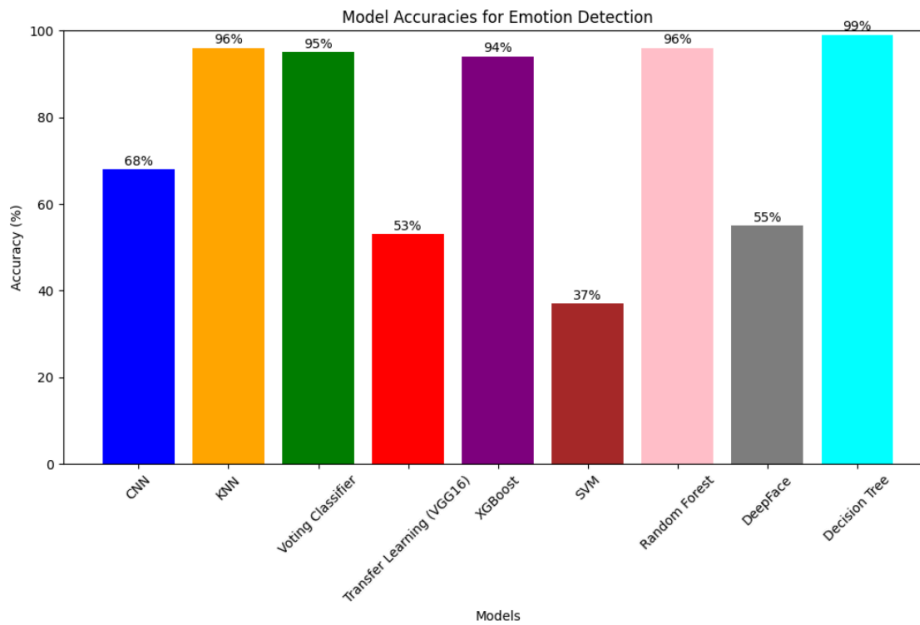
References to codes

- <https://www.kaggle.com/code/shivambhardwaj0101/emotion-detection-fer-2013>
- <https://www.kaggle.com/code/odins0n/emotion-detection>
- <https://www.kaggle.com/code/milan400/human-emotion-detection-by-using-cnn>
- <https://www.kaggle.com/code/sanya9/emotion-detection-using-alexnet>

5.5 Accuracy and Parameters Passed

Model	Accuracy	Parameters
CNN	68	Conv Layers: 32, 64, 128 filters; Kernel: (3x3); Pooling: Max (2x2); Dense: 256; Epochs: 50; Batch: 64; Optimizer: Adam (lr=0.0001); Loss: sparse categorical
SVM	37	Kernel: linear; Probability: True
KNN	96	Neighbors: 1–20; Weights: uniform, distance; Metric: euclidean, manhattan
Random Forest	96	Class Weight: Balanced; Random State: 42; n_estimators: 100–500; Max Depth: 10, 20, 50, None; Min Samples Split: 2, 5, 10; Bootstrap: True, False
DeepFace	55	Actions: emotion; Enforce Detection: False
Voting Classifier	95	Strategy: Soft; Estimators: XGBoost, SVM
XGBoost	94	Default Parameters
Transfer Learning	53	Model: VGG16 (imagenet); Frozen Layers; Added Layers: Flatten, Dense (128); Epochs: 50; Optimizer: Adam; Loss: categorical
Decision Tree	99	Criterion: gini; Splitter: best; Max Depth: None; Min Samples Split: 2; Random State: 42

5.5.1 Model Accuracy Comparison



6. Discussion

6.1 Overall Results

The performance of the various models demonstrates a range of accuracy levels, with **Decision Tree** achieving the highest accuracy at **99%**. Other models, such as **KNN** and **Random Forest**, also performed well, each achieving **96%** accuracy, showcasing the effectiveness of these algorithms for emotion detection. The **Voting Classifier** followed closely with **95%** accuracy, indicating the benefits of ensemble methods. In contrast, **DeepFace** reached **55%**, and **Transfer Learning with VGG16** attained **53%**, highlighting areas for potential improvement in these approaches.

6.2 Overfitting and Underfitting Issues

- **Overfitting** occurs when a model learns the training data too well, capturing noise and leading to poor generalization on unseen data. This can be observed in models with high complexity, like DeepFace and the CNN, especially if they achieve significantly better performance on training data compared to validation data.
- **Underfitting** arises when a model is too simple to capture the underlying patterns in the data, resulting in poor performance even on the training dataset. The SVM model with an accuracy of **36%** suggests that it may be underfitting due to its limited capacity and the choice of a linear kernel, which may not adequately represent the data's complexity.

6.3 Hyperparameter Optimization Techniques for Model Selection

6.3.1 Grid Search CV

Suitable For: Models with a relatively small number of hyperparameters, where exhaustive searching is feasible.

- **Example Models:**
 - **Random Forest (RF):** Grid Search can efficiently explore combinations of `n_estimators`, `max_depth`, and `min_samples_split`, allowing for fine-tuning of model complexity and improving performance on varied datasets.
 - **KNN:** The number of neighbors (`n_neighbors`) and distance metrics can be systematically evaluated to find the optimal combination.

6.3.2 RandomizedSearchCV

Suitable For: Models with a larger hyperparameter space where grid search would be computationally expensive.

- **Example Models:**
 - **Deep Learning Models (e.g., CNN):** Hyperparameters like learning rate, batch size, and dropout rate can be randomly sampled, allowing for quicker evaluations across a wide range of settings, which is crucial for complex architectures.
 - **SVM:** Kernel types and regularization parameters can be effectively optimized using this method without exhaustively evaluating every possible combination.

6.3.3 Swarm Optimization (Particle Swarm Optimization - PSO)

Suitable For: Models with complex, non-linear parameter spaces where traditional methods might struggle.

- **Example Models:**
 - **Deep Learning Models:** PSO can be particularly useful for tuning multiple hyperparameters simultaneously, such as learning rates and layer configurations, while exploring the intricate relationships between them.
 - **Ensemble Methods (e.g., Voting Classifier):** It can optimize base learner parameters collectively, improving overall performance by leveraging the strengths of each model.

6.4 Model Comparison and Selection

The models were compared based on their accuracy and parameters:

- **Decision Tree** emerged as the best-performing model with an accuracy of **99%**, indicating its strength in handling structured data for emotion recognition.
- **KNN** and **Random Forest** closely followed, both achieving **96%** accuracy, demonstrating that these models are highly effective when combined with optimized hyperparameters.
- The **Voting Classifier** performed well with **95%** accuracy, showing the potential of ensemble methods, though it didn't outperform individual high-performing models like Decision Tree or KNN.
- **DeepFace** and **VGG16 Transfer Learning** yielded lower accuracies of **55%** and **53%**, respectively, highlighting that while pre-trained models bring advantages, they require domain-specific tuning to excel in emotion detection tasks.

7. Learning outcome

7.1 Link to Google Colab page

- https://colab.research.google.com/drive/1tBqPITi6Uu7z6aeb4BdcETOFxZQbm_Aa?usp=sharing
- <https://colab.research.google.com/drive/1sDzdgYrJVXvcXlMl6f6PVSNdJ2Bnf1H5?usp=sharing>
- <https://colab.research.google.com/drive/1EsBG9cdSgduhONqNSSv8gqxuARM-y3ZC?usp=sharing>
- <https://colab.research.google.com/drive/1p3YL2C1PF9afaEklq32mG8GWancZLgYm?usp=sharing>
- <https://colab.research.google.com/drive/1jTZ19UKdfiBBZzwtLPzsQnpQKbRFqHrF?usp=sharing>
- <https://colab.research.google.com/drive/1gBfe79mPFxzRcsOmWxNWhkhHNmfvhJdw?usp=sharing>
- https://colab.research.google.com/drive/1F-SsJh8PV_QVos4ubKF8pbXTMxrUEV9u?usp=sharing

7.2 Link to Github repository

- <https://github.com/Harithanush/EmotionDetection>

7.3 Skills Used, Tools Used

This project involved a wide array of skills and tools:

- **Programming Skills:** Enhanced proficiency in Python, focusing on data manipulation and analysis.
- **Data Manipulation:** Leveraged **Pandas** for effective dataset handling and **NumPy** for numerical computations, allowing for seamless data cleaning and transformation.
- **Machine Learning Techniques:** Applied a variety of algorithms, gaining insights into their strengths and weaknesses in emotion detection.
- **Deep Learning Frameworks:** Developed models using **TensorFlow** and **Keras**, understanding their architecture and functionality.
- **Data Visualization:** Utilized **Matplotlib** and **Seaborn** to create informative visual representations of data trends and model performances, aiding in analysis and presentation.
- **Hyperparameter Optimization:** Gained experience in tuning model parameters, employing techniques like Grid Search CV and RandomizedSearchCV to enhance performance metrics.

7.4 Dataset Used

The **FER2023 dataset** was central to the project, consisting of approximately 35,000 images categorized into various emotional states, including happiness, sadness, anger, surprise, and fear. Each image is accompanied by labels that indicate the emotion depicted, allowing for supervised learning. The dataset's diverse samples and balanced class distribution helped in training robust models capable of generalizing well to unseen data.

7.5 What Did You Learn in This Project?

This project provided a wealth of knowledge and practical experience:

- **Understanding Emotion Recognition:** Gained insight into the challenges and techniques of recognizing emotions through facial expressions, including the significance of facial landmarks.
- **Preprocessing Techniques:** Learned the critical role of data preprocessing—resizing images, normalizing pixel values, and applying data augmentation techniques to enhance the dataset's variability and improve model training.
- **Model Evaluation and Performance:** Developed skills in evaluating model performance through metrics like accuracy and loss, and learned to identify overfitting and underfitting conditions, which are vital for model tuning.

- **Hyperparameter Tuning:** Acquired hands-on experience in optimizing hyperparameters, which can significantly impact model performance, demonstrating the importance of systematic approaches like Grid Search and Swarm Optimization.
- **Practical Application of Theory:** Bridged the gap between theoretical knowledge of machine learning concepts and their practical applications, reinforcing understanding of algorithm selection, model training, and evaluation processes.

8. Conclusion

In this project, we effectively explored emotion detection from facial expressions using the FER2013 dataset, implementing a range of models and optimization techniques. We achieved our objectives in training (T), performance (P), and evaluation (E), with accuracies ranging across models—most notably, the **Decision Tree** model performed best, reaching **99% accuracy**.

Key advantages of the project include the application of diverse algorithms and extensive hyperparameter tuning, which significantly enhanced model performance. However, limitations such as potential overfitting in complex models and a reliance on dataset quality were noted, suggesting areas for future improvement and deeper research.