Facial Expression Recognition

SML PROJECT FINAL REPORT

Abhay Dagar
2022014
abhay22014@iiitd.ac.in
Btech2026

Rohan Basugade
2022416
rohan22416@iiitd.ac.in
Btech2026

***Abstract*—Facial expressions play a crucial role in human interaction and the interpretation of emotions. Recognizing facial expressions automatically is a challenging task in computer vision due to the high variability within each expression class. In this project, we propose an architecture to classify facial images into seven basic human expressions: ANGER, DISGUST, FEAR, HAPPY, NEUTRAL, SAD, and SURPRISE. Additionally, we have implemented a system that analyzes video frames to predict the emotions of individuals.**

***Index Terms*—Facial Expression Recognition, Computer Vision, Emotion Interpretation, Human Interaction, Video Analysis, Live Web Cam Analysis**

I. Introduction

Facial Expression Recognition (FER) involves the automated detection and interpretation of facial expressions, crucial for understanding human emotions in various applications like human-computer interaction and behavioral analysis. It also finds use in Emotional Analysis in Media, Behavioral Research, Virtual Reality & Augmented Reality, and Mental Health Monitoring.

II. Literature Review

Recent research in Facial Expression Recognition (FER) has witnessed a focus on employing diverse machine learning and deep learning techniques to achieve higher accuracy and real-time processing capabilities.

A. Systematic Review

A systematic review, to [], was conducted within our project framework, delving into the methodologies employed for FER. This comprehensive review addressed challenges encountered with varied datasets and proposed solutions to enhance system performance across different models.

B. Micro-expression Recognition

Special attention was given to micro-expression recognition in real-time applications, akin to the focus observed in []. Here, our project introduced hybrid neural network models tailored to identify fleeting expressions, contributing to the growing body of research in this domain.

C. Automatic Micro-Expression Detection

Furthermore, our project also made strides in addressing the challenge of detecting subtle facial expressions, akin to the developments outlined in the Automatic Micro-Expression Detection System (AutoMEDSys) as described in [].

D. Evaluation of Multiple Models

In line with platforms like Papers With Code, our project provided insights into recent advancements through the implementation and evaluation of multiple models. Notably, the Decision Tree, Random Forest, AdaBoost, and CNN model (with an accuracy exceeding 92%) were developed and rigorously evaluated within our project scope. Each model's performance was meticulously analyzed, contributing to the ongoing efforts aimed at enhancing the accuracy and robustness of FER systems.

III. Dataset Details

We utilized two datasets from Kaggle: the fer2013 dataset [1] and the FED Dataset [2]. These datasets consist of images categorized into seven emotion categories: Angry, Disgust, Fear, Happy, Sad, Surprise, and Neutral. The training set comprises 50,262 images, while the validation set contains 16,557 images. Each image has dimensions of 48x48x1. This dataset was prepared by Pierre-Luc Carrier and Aaron Courville.

The distribution of data across emotion categories is as follows:

- Total Angry images: 8607 (label-1)
- Total Disgust images: 799 (label-2)
- Total Fear images: 8796 (label-3)
- Total Happy images: 12243 (label-4)
- Total Neutral images: 4575 (label-5)
- Total Sad images: 9304 (label-6)
- Total Surprise images: 5938 (label-7)

The total number of images in both datasets combined is 66,819.

- Due to the small size of the Disgust image class (799 out of 50,262), we explored three options: concatenating Disgust with Angry data, using SMOTE for class balancing, or leaving the data as is.
- Researching online, we found similar discussions where practitioners debated the efficacy of different approaches

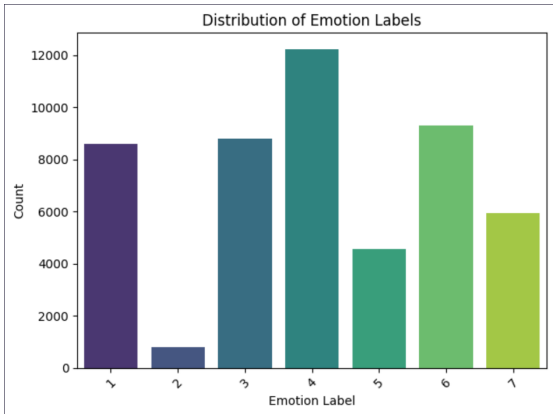

Fig. 1. Train Images

Fig. 2. 5 Images From Each Class

for handling imbalanced datasets in facial expression recognition tasks.

- All three approaches were tested, yielding similar results. Given the precision of deep neural networks and the desire to maintain the dataset's sparsity, we decided to leave the data unchanged.

IV. Model 1: Decision Tree

The decision tree classifier is a non-linear supervised learning algorithm used for classification tasks.

The architecture of the decision tree model is determined by its structure, which consists of a series of decision nodes and leaf nodes. Each decision node represents a decision based on a feature value, and each leaf node represents a class label. During training, the decision tree algorithm recursively splits the feature space into subsets based on the feature values, optimizing the splits to maximize information gain or minimize impurity.

Model Workflow:

The decision tree classifier is integrated into the workflow as follows:

- **Images:** Raw images are used as input data.
- **Create Batches:** The images are divided into batches for processing efficiency.
- **Extract Features:** Features are extracted from the images.
- **Train Classifier:** The decision tree classifier is trained using the extracted features.
- **Predict Validation Label:** The trained classifier is used to predict the labels of validation data.

This workflow enables the decision tree classifier to be seamlessly integrated into the process of feature extraction and classification for image data.

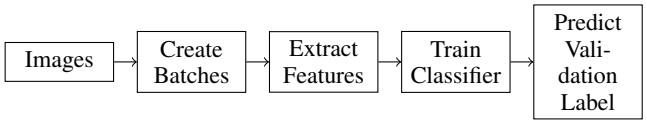

Fig. 3. Workflow Diagram

V. Model 2: Random Forest Classifier

The Random Forest classifier is an ensemble learning method that constructs a multitude of decision trees during training and outputs the class that is the mode of the classes of the individual trees.

Model Workflow:

- **Decision Trees:** Each decision tree in the forest is trained on a random subset of the training data.

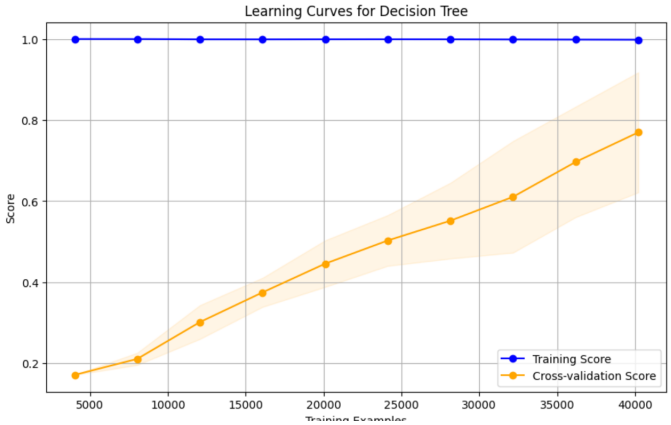

Fig. 4. Learning Curve of Decision Tree

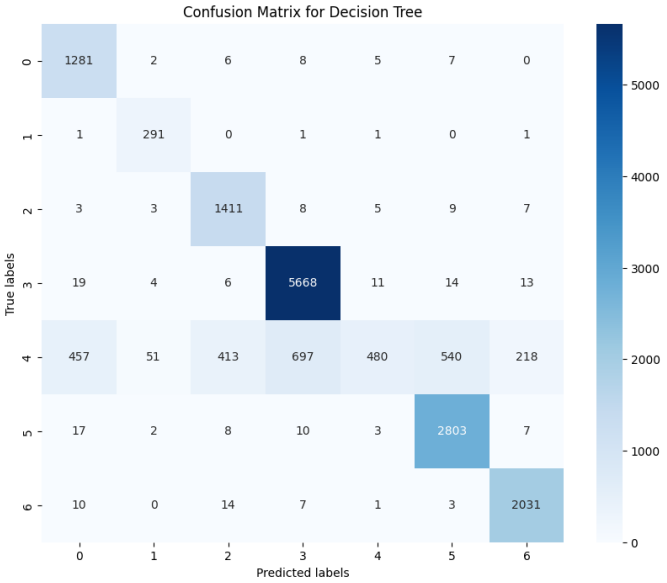

Fig. 5. Confusion Matrix Of Decision Tree

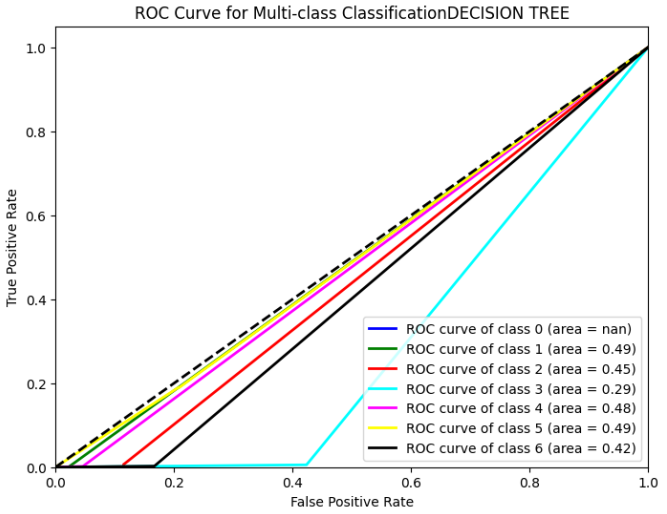

Fig. 6. ROC Curve Decision Tree

- **Bootstrap Aggregation (Bagging):** Random Forests use bootstrap sampling to create multiple subsets of the training data, ensuring diversity among the trees.
- **Random Feature Selection:** At each node of the decision tree, a random subset of features is considered for splitting, reducing correlation between trees.
- **Voting:** During classification, each tree in the forest independently predicts the class, and the class with the most votes becomes the final prediction.

The Random Forest model trained in the provided code snippet consists of a collection of decision trees, each trained on a random subset of the principal components extracted from the input data using PCA.

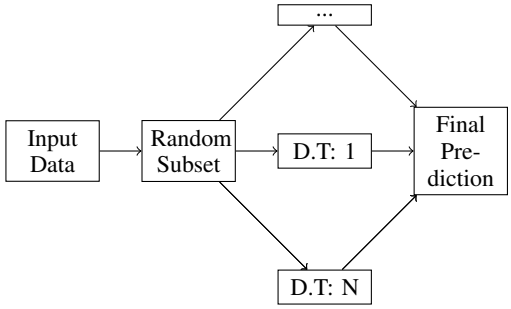

Fig. 7. Architecture of a Random Forest Classifier

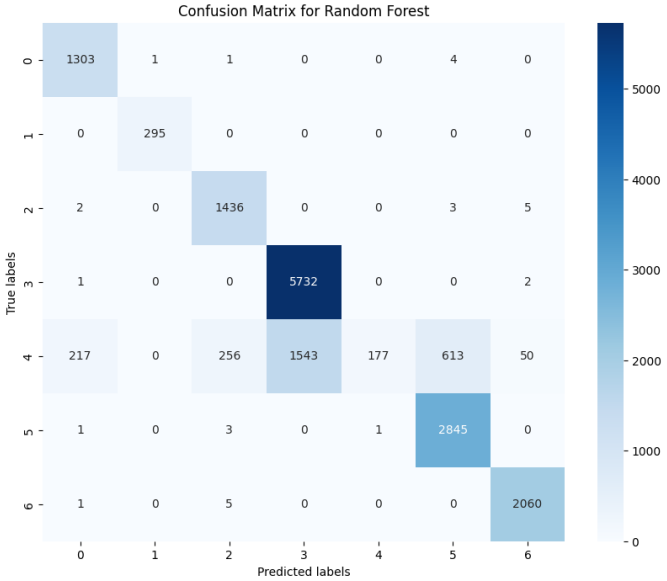

Fig. 8. Confusion Matrix Of Random Forest

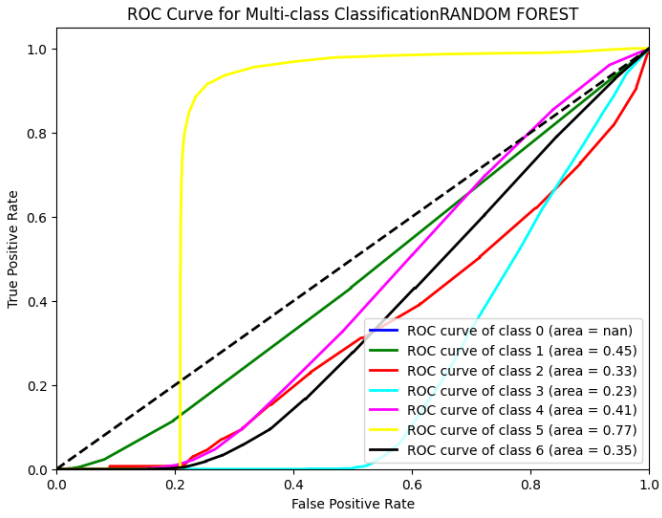

Fig. 9. ROC curve of Random Forest

VI. Model 3: AdaBoost Classifier

The AdaBoost (Adaptive Boosting) classifier is an ensemble learning method that combines multiple weak classifiers to create a strong classifier. It sequentially trains a series of weak classifiers, with each subsequent classifier focusing more on the examples that were misclassified by the previous ones.

Model Workflow:

- **Weak Classifiers:** Typically simple classifiers (e.g., decision stumps) that perform slightly better than random guessing.
- **Sample Weighting:** During training, misclassified examples are assigned higher weights to prioritize their correct classification in subsequent iterations.
- **Sequential Training:** Classifiers are trained sequentially, with each subsequent classifier focusing more on the examples that were misclassified by the previous ones.
- **Weighted Voting:** During classification, each weak classifier's prediction is weighted based on its accuracy, and the final prediction is determined by a weighted majority vote.

The AdaBoost classifier trained in the provided code snippet consists of an ensemble of weak classifiers trained sequentially on the principal components extracted from the input data using PCA.

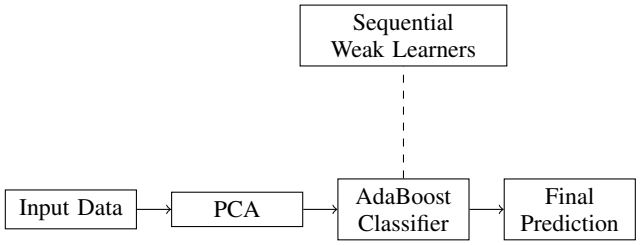

Fig. 10. Workflow Diagram of AdaBoost Classifier

Model 4: CNN(Final Model)

The architecture of the Convolutional Neural Network (CNN) can be described as follows:

1) **Input Layer**: Receives grayscale images of size 48x48 pixels with a single channel.
2) **Convolutional Layers (Conv2D)**: The model starts with two convolutional layers, each followed by a Rectified Linear Unit (ReLU) activation function. The first layer has 32 filters, and the second has 64 filters.
3) **Pooling Layer (MaxPooling2D)**: After each convolutional layer, a max-pooling layer reduces the spatial dimensions of the feature maps while retaining important information.
4) **Dropout Layer**: Two dropout layers are added to prevent overfitting by randomly deactivating a fraction of neurons during training (dropout rate of 0.25).
5) **Additional Convolutional Layers**: Two more convolutional layers are added, each followed by max-pooling and dropout layers, to extract higher-level features.

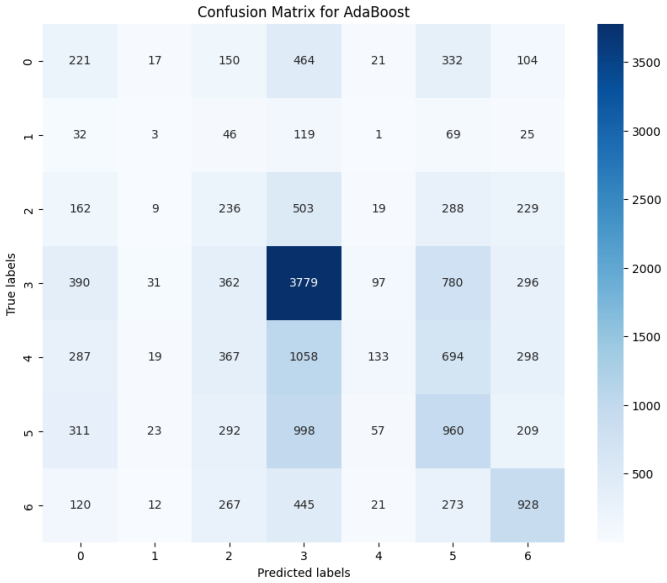

Fig. 11. Confusion Matrix Of AdaBoost

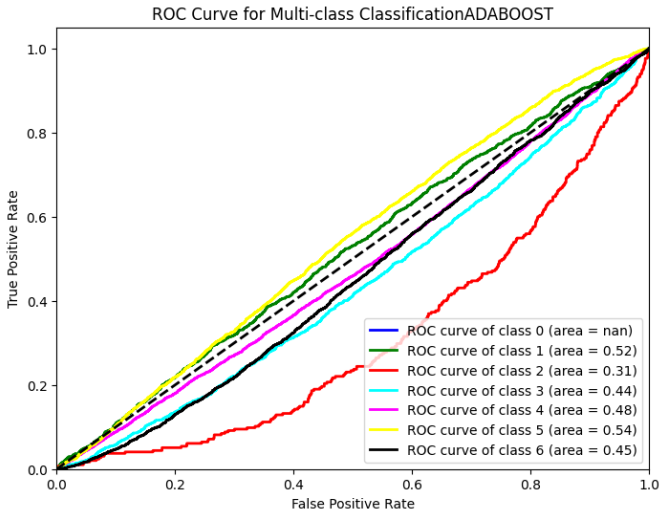

Fig. 12. ROC curve of Adaboost

6) **Flatten Layer**: Flattens the feature maps into a one-dimensional vector.
7) **Dense Layers (Fully Connected)**: Two dense layers are added for classification. The first dense layer has 1024 units with ReLU activation, and the second dense layer (output layer) has 7 units with softmax activation for class probabilities.
8) **Model Compilation**: Compiled using categorical cross-entropy loss, Adam optimizer with a learning rate of 0.0001 and decay of 1e-6, and accuracy metric.
9) **Training**: The model is trained for 100 epochs using training data generated by a data generator with a batch size of 64.

This architecture effectively captures spatial hierarchies of

features in input images for accurate emotion classification.

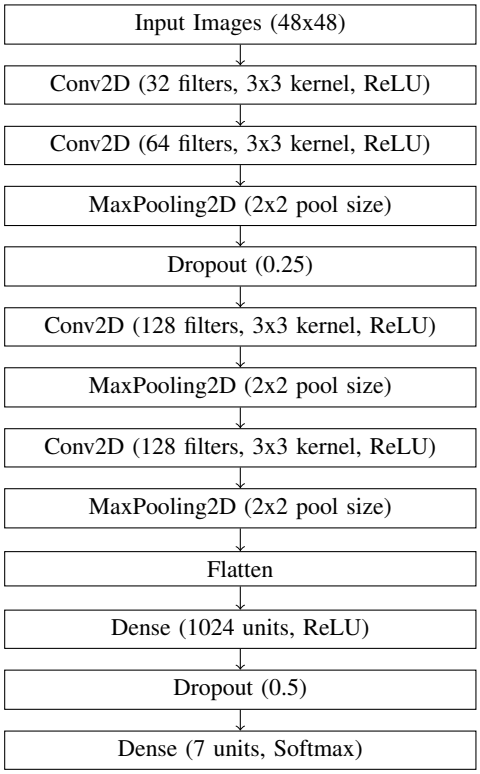

Fig. 13. Workflow Diagram of the CNN Model Architecture

VII. Result And Analysis Of Each Model

FOR 3 MODELS:

- DECISION TREE
- RANDOM FOREST
- ADABOOST

Model: "sequential"

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

Fig. 14. CNN Model Info

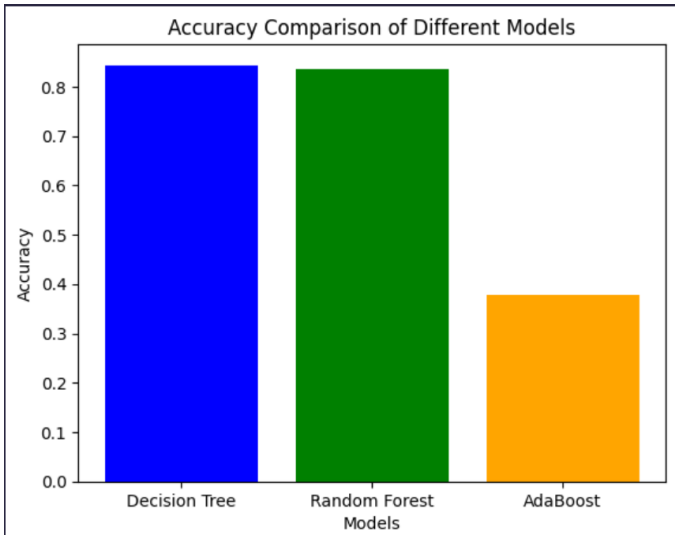

Fig. 15. Accuracy Of Models

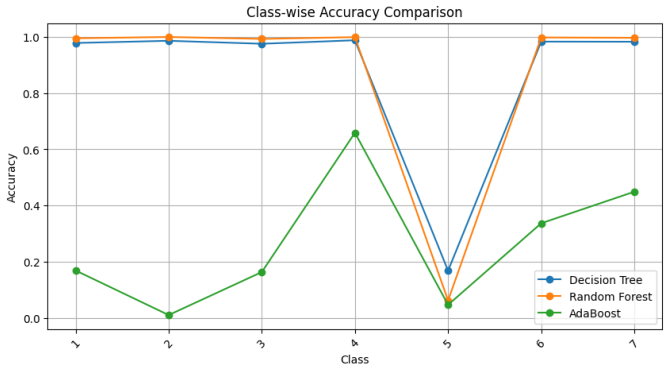

Fig. 16. ClassWise Comparison

A. Accuracy Analysis

Decision Tree:
The Decision Tree model has the highest accuracy among the three models. Its accuracy value is above 0.8, as indicated by the height of the blue bar on the chart.

Random Forest:
The Random Forest model has slightly lower accuracy than the Decision Tree. Its accuracy value is below 0.8 but higher than AdaBoost, as shown by the green bar.

AdaBoost:
The AdaBoost model has the lowest accuracy among the three. Its accuracy value is around 0.6, as represented by the orange bar.

In summary, the chart demonstrates that for facial expression detection, the Decision Tree model outperforms both Random Forest and AdaBoost in terms of accuracy. If you're working on a similar project, you might consider using the Decision Tree model for better results.

ROC CURVE: The ROC curve shows how well a classification model can distinguish between classes. It plots the true positive rate against the false positive rate, with the area

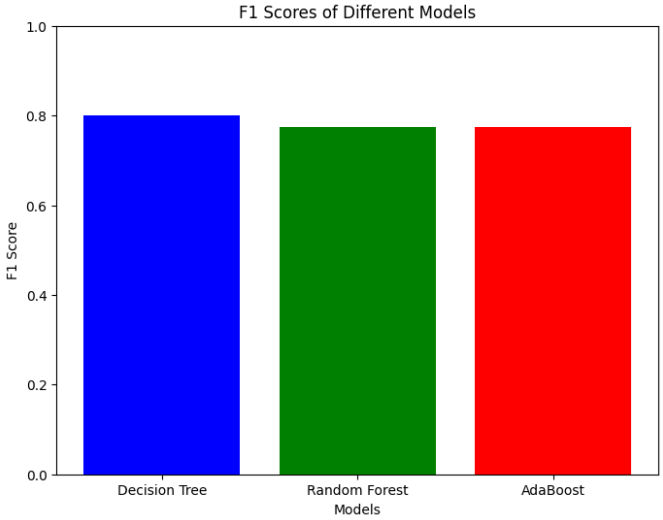

Fig. 17. F1 Score Plot

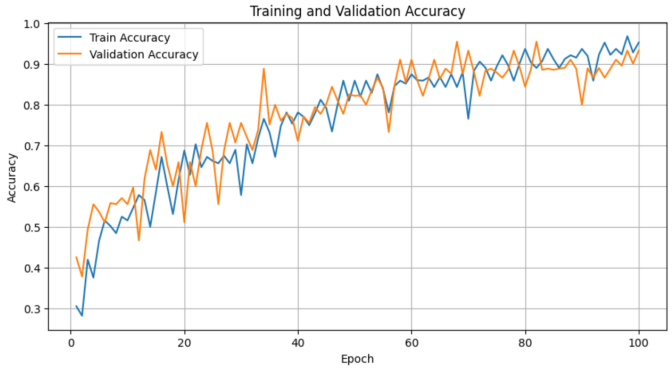

Fig. 18. Accuracy V/S Epochs

under the curve indicating the model's overall performance. A larger area means the model is better at distinguishing between classes.

B. CNN RESULT ANALYSIS

CNN MODEL

Training Metrics:

- Max Train Accuracy: 96.875%
- Max Train Loss: 1.709%
- Min Train Accuracy: 28.125%
- Min Train Loss: 0.1423%

Validation Metrics:

- Max Validation Accuracy: 95.58%
- Max Validation Loss: 1.709%
- Min Validation Accuracy: 37.77%
- Min Validation Loss: 0.14%

VIII. Conclusion And Inference

In this project, we implemented a system that can predict emotions from facial expressions in human images. An additional component that predicts facial expressions in videos and in real-time using webcams was also developed.

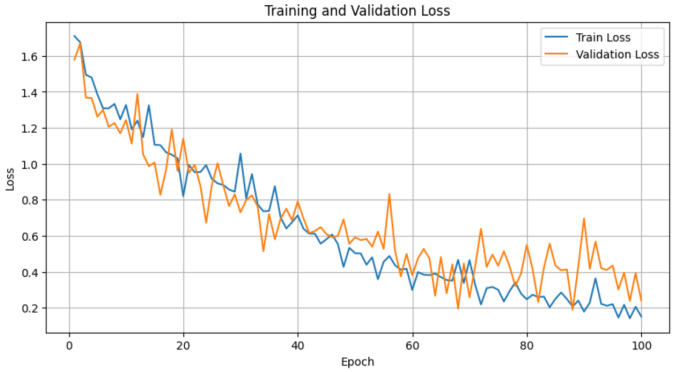

Fig. 19. Loss V/S Epochs

We explored various techniques to achieve the best results. Initially, we experimented with basic models:

1) Decision Tree
2) Random Forest
3) AdaBoost

Finally, we proposed a CNN model that performed well with our dataset and provided good results in real-time prediction. The maximum validation accuracy achieved for facial emotion recognition was approximately 95.8% with our model.

We chose to use CNN instead of conventional fully connected Multi-layer perceptron-based (MLP) networks for several reasons:

1) The number of weights in MLP grows very rapidly.
2) MLP is not transition invariant.
3) Spatial information is lost when we flatten the input.
4) CNNs preserve spatial information.

Overall, CNNs work well with image classification problems, and one can explore their architecture further to increase accuracy.

IX. Model Testing: Output Prediction

A. For Image

B. For Video

C. For Webcam

We Made an interactive front for User Friendliness using Streamlit.

References

[1] https://www.kaggle.com/datasets/msambare/fer2013
[2] https://www.kaggle.com/datasets/aadityasinghal/facial-expression-dataset
[3] M. A. Ozdemir, B. Elagoz, A. Alaybeyoglu, R. Sadighzadeh and A. Akan, "Real Time Emotion Recognition from Facial Expressions Using CNN Architecture,"
[4] Gaurav Sharma, Real Time Facial Expression Recognition, article on Medium.
[5] Pattern Recognition and Machine Learning by Christopher Bishop
[6] Introduction to CNN, GeeksforGeeks. Available at: https://www.geeksforgeeks.org/introduction-convolution-neural-network/
[7] Keras CNN, JavaTPoint. Available at: https://www.javatpoint.com/keras-convolutional-neural-network

Epoch	Train Accuracy	Validation Accuracy	Train Loss	Validation Loss
1	0.304773092	0.4250848	1.709854126	1.578348756
2	0.28125	0.377777785	1.674691439	1.669211149
3	0.419279665	0.493580431	1.494337082	1.367712975
4	0.375	0.555555582	1.479401827	1.36563766
5	0.466034502	0.536882281	1.386543036	1.262283206
6	0.515625	0.51111114	1.308107853	1.29954195
7	0.501932323	0.558624029	1.30832541	1.205394268
8	0.484375	0.555555582	1.332825422	1.226353168
9	0.524981081	0.570736408	1.246869564	1.169461846
10	0.515625	0.555555582	1.327034712	1.243700027
11	0.546655238	0.597020328	1.192016602	1.112165809
12	0.578125	0.466666669	1.240269661	1.38802588
13	0.565879107	0.618095934	1.148187876	1.051739454
14	0.5	0.688888907	1.325219631	0.986754119
15	0.584525287	0.640746117	1.105885267	1.006663084
16	0.671875	0.733333349	1.104039907	0.827749789
17	0.599764943	0.653221905	1.064158678	0.966632366
18	0.53125	0.600000024	1.050678015	1.191040158
19	0.616000652	0.658975303	1.030366778	0.960377991
20	0.6875	0.51111114	0.820915759	1.139821529
21	0.627793908	0.659217536	0.992973149	0.950944662
22	0.703125	0.600000024	0.953301609	0.994026184
23	0.646479964	0.689801335	0.954563379	0.878046632
24	0.671875	0.75555557	0.993115187	0.672986448
25	0.661978543	0.685259223	0.916791916	0.882054329
26	0.65625	0.555555582	0.891499341	1.003108025
27	0.674847603	0.68786335	0.882612348	0.880008757
28	0.65625	0.75555557	0.857685685	0.765174091
29	0.689509571	0.707243204	0.844859302	0.831291676
30	0.578125	0.75555557	1.05712676	0.730400026
31	0.703016043	0.721051335	0.80820334	0.798812985
32	0.65625	0.688888907	0.942984819	0.824988484
33	0.718016624	0.737948179	0.774552464	0.763585687
34	0.765625	0.888888896	0.736437798	0.514849961
35	0.732280195	0.751332343	0.739037156	0.721456587
36	0.671875	0.800000012	0.87552309	0.580851853
37	0.748217046	0.761082828	0.703720987	0.694693923
38	0.78125	0.777777791	0.641007602	0.751057625
39	0.753735185	0.767260194	0.675596535	0.686258614
40	0.78125	0.711111128	0.714010417	0.791776717
41	0.770449042	0.771196723	0.639924407	0.697221935
42	0.75	0.75555557	0.61255312	0.615292549
43	0.779254138	0.794331372	0.612069845	0.627309918
44	0.8125	0.777777791	0.556472838	0.648533404
45	0.791784525	0.801659405	0.581163526	0.61035496
46	0.734375	0.844444454	0.607616901	0.59235996
47	0.801406443	0.809895813	0.555095196	0.600041807
48	0.859375	0.777777791	0.427790403	0.691547096
49	0.81011194	0.825460255	0.53369689	0.555965424
50	0.859375	0.822222233	0.504022419	0.591795206

Fig. 20. History: 1 to 50 epochs

51	0.82027173	0.823280036	0.502518058	0.576679647
52	0.859375	0.800000012	0.439795017	0.583554208
53	0.829495192	0.835816383	0.480950892	0.539642036
54	0.875	0.866666675	0.359078348	0.623778999
55	0.838380039	0.841085255	0.456157923	0.527764976
56	0.78125	0.733333349	0.488246232	0.832450867
57	0.8458305	0.847262621	0.436525762	0.51537782
58	0.859375	0.911111116	0.412648082	0.375300854
59	0.852065802	0.855075121	0.417551249	0.499516696
60	0.875	0.911111116	0.299813151	0.381594688
61	0.860671759	0.860767901	0.398917794	0.474935293
62	0.859375	0.822222233	0.384474307	0.528068483
63	0.867126167	0.863493204	0.382121682	0.474591374
64	0.84375	0.911111116	0.390084416	0.267913789
65	0.867962897	0.863856614	0.371101856	0.482068956
66	0.84375	0.888888896	0.354836762	0.281323612
67	0.875453234	0.876574636	0.351428807	0.441433936
68	0.84375	0.955555558	0.466914117	0.195651814
69	0.880035043	0.876211226	0.338691682	0.447104752
70	0.765625	0.933333337	0.465519607	0.258452803
71	0.883939624	0.880147755	0.326738656	0.433215111
72	0.90625	0.822222233	0.219312027	0.639149785
73	0.891549468	0.883115292	0.309543341	0.42786935
74	0.859375	0.888888896	0.316453457	0.495753765
75	0.894258738	0.879299879	0.30094555	0.434490055
76	0.921875	0.866666675	0.235777259	0.514509082
77	0.897366405	0.885598361	0.292971134	0.431032836
78	0.859375	0.933333337	0.341484636	0.321213245
79	0.900852621	0.894561529	0.279102653	0.390987724
80	0.9375	0.844444454	0.247710735	0.549581766
81	0.904836833	0.885719478	0.27321133	0.418071598
82	0.890625	0.955555558	0.259927809	0.233234659
83	0.907486379	0.886325121	0.262831181	0.425196052
84	0.9375	0.888888896	0.203423709	0.5566715
85	0.912347078	0.886385679	0.250550389	0.436149299
86	0.890625	0.888888896	0.285589099	0.408750713
87	0.912845135	0.891109467	0.249787107	0.413591772
88	0.921875	0.911111116	0.207536981	0.189537466
89	0.916112185	0.888808131	0.240598455	0.429256171
90	0.9375	0.800000012	0.179587662	0.697007537
91	0.920176089	0.889716566	0.229009897	0.417151511
92	0.859375	0.866666675	0.364116609	0.568259656
93	0.923383415	0.890443325	0.22235395	0.420200288
94	0.953125	0.866666675	0.211925045	0.411051929
95	0.922925234	0.888323665	0.220903412	0.435141712
96	0.9375	0.911111116	0.146255016	0.301826596
97	0.924200177	0.896257281	0.217272878	0.395769656
98	0.96875	0.933333337	0.142355531	0.238885418
99	0.92874217	0.901102245	0.206511527	0.393803477
100	0.953125	0.933333337	0.152340397	0.241556421

Fig. 21. History: 51 to 100 epochs

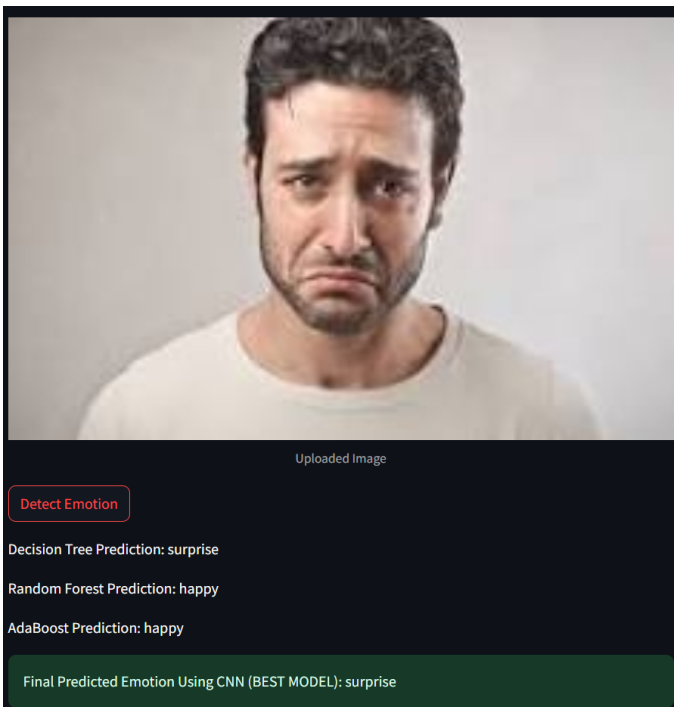

Fig. 22. Image test:1

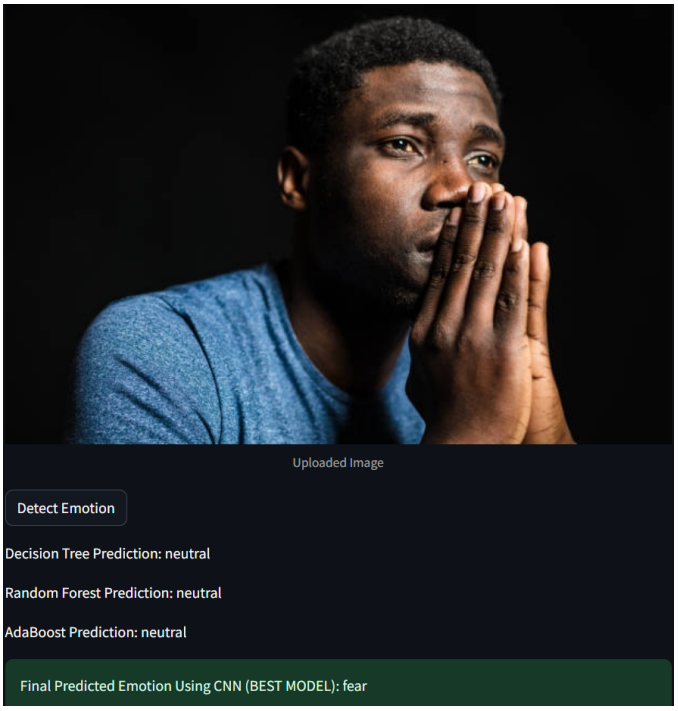

Fig. 23. Image test:2

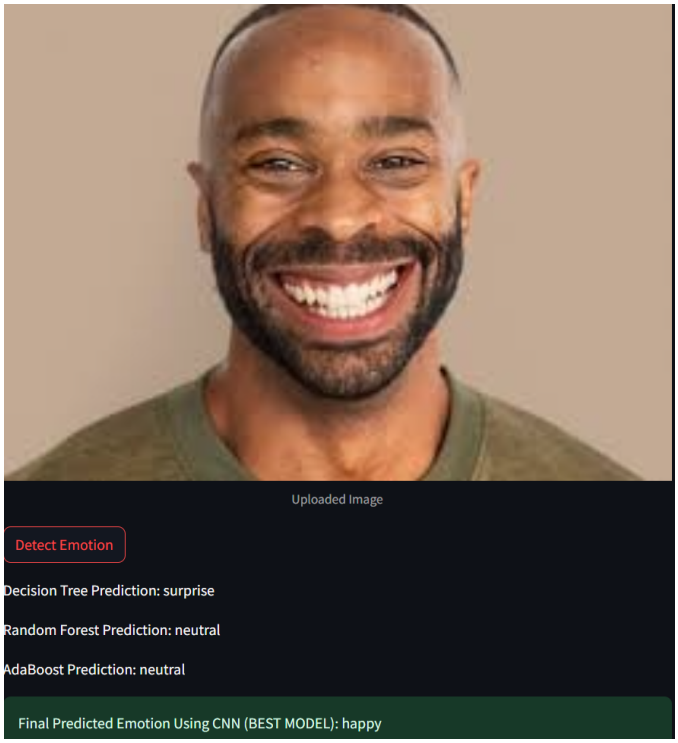

Fig. 24. Image test:3

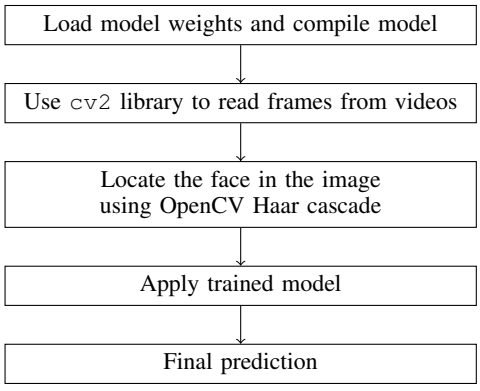

Fig. 25. Workflow Steps For Video And WebCam

Fig. 26. Vid Frame: 01

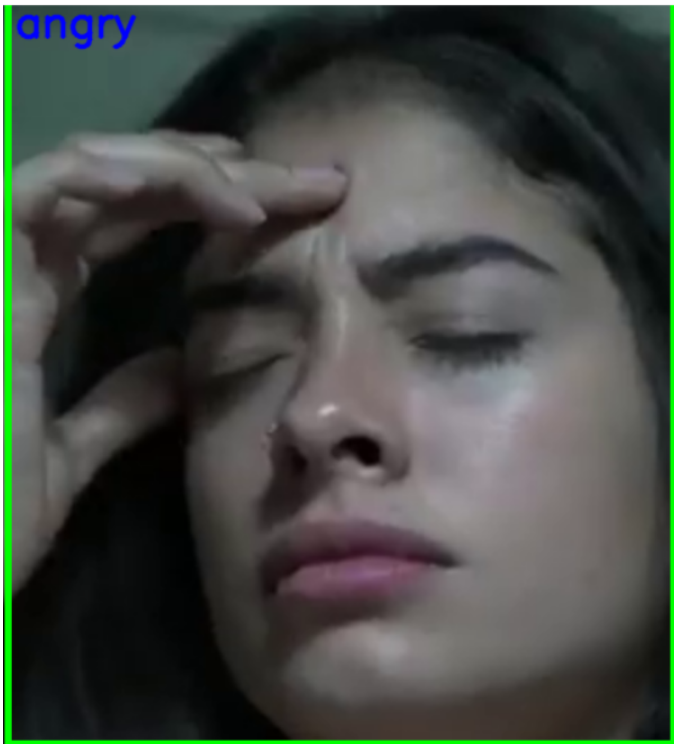

Fig. 27. Vid Frame :02

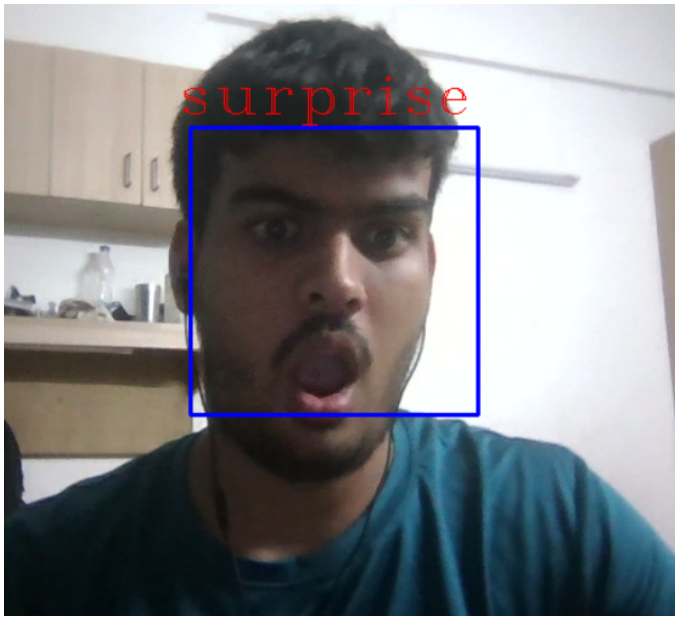

Fig. 28. Webcam:1

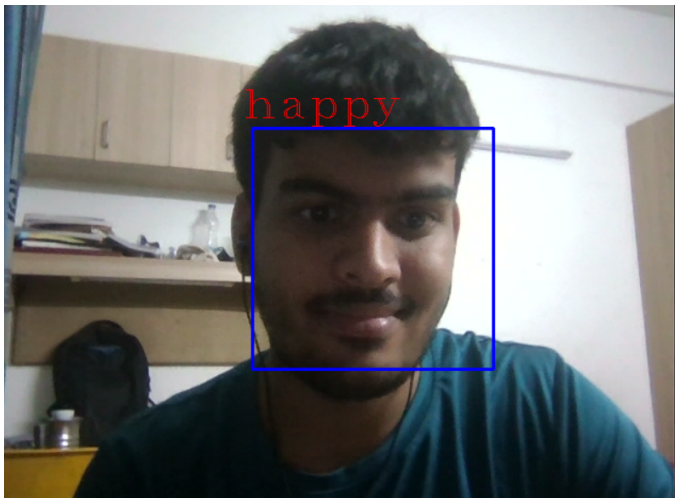

Fig. 29. Webcam:2