



**VIT-AP
UNIVERSITY**

Human Emotions : Enhancing facial emotions through facial expressions using Deep Learning.

Under Guidance of **Dr. Jeevana Jyothi Pujari**

Our Team :

(Team 13)

Mandadi Pavan Sai - 23BCE9626

Raya Raghavar - 23BCE9386

Shaik Rajabhaigari Mahaboob - 23BCE9426

Kommudasari Manoj Kumar - 23BCE20058

Abstract :

This research presents an effective deep learning approach for facial emotion recognition using the Kaggle "FERAC" dataset. The proposed system uses convolutional neural networks (CNNs) to accurately classify facial expressions into different emotional categories such as happiness, sadness, anger, surprise, disgust, fear, and neutral. Preprocessing techniques including image normalization and data augmentation were applied to improve model robustness and generalization.

Through extensive training and evaluation on the benchmark dataset, the model demonstrated competitive accuracy and reliability despite challenges such as class imbalance and subtle variations in expressions. The study underscores the significance of deep neural architectures and advanced training strategies in enhancing facial emotion recognition performance, with potential applications in human-computer interaction, behavioral analysis, and affective computing.

Introduction :

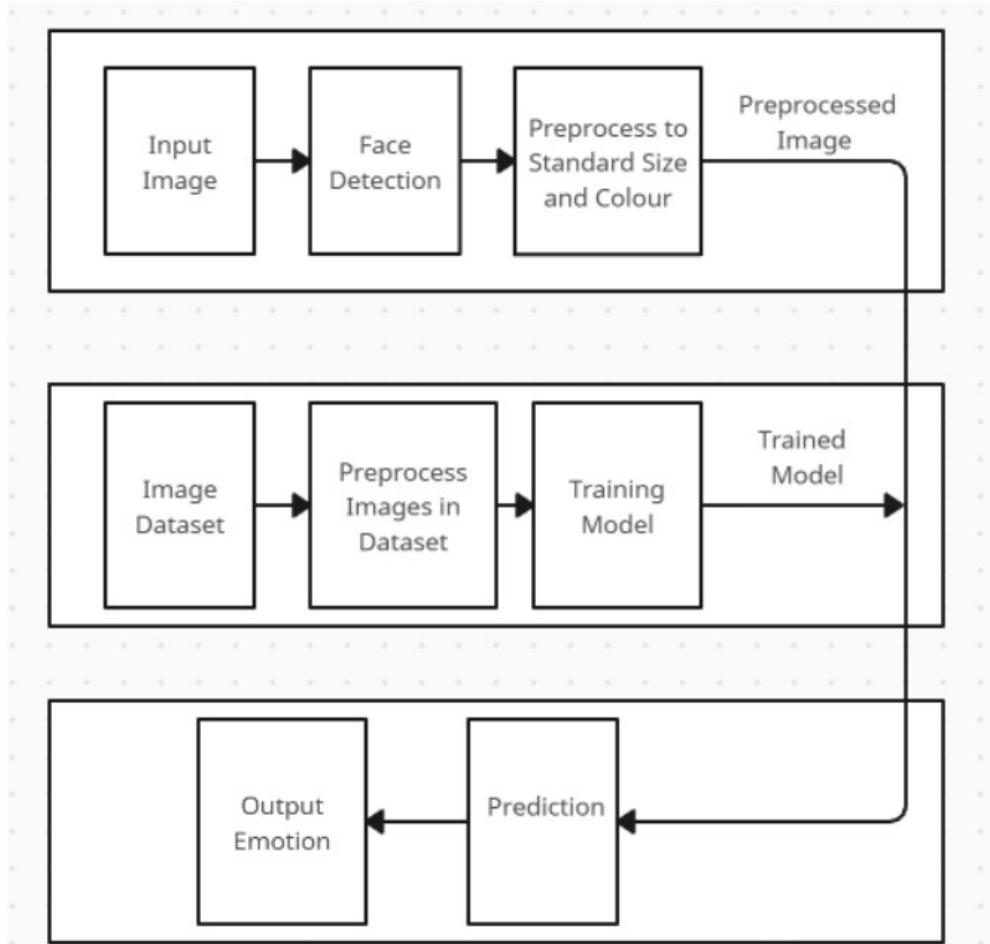
The ability of detecting human emotions is essential for creating better interactions between humans and machines. Facial expressions are globally recognized as primary indicators of human emotions and have been extensively studied in psychology and computer vision.

With the use of deep learning techniques, especially convolutional neural networks, automated facial recognition has achieved significant advancements over traditional machine learning methods in accuracy and robustness. This paper focuses on leveraging deep learning to detect facial emotions, aiming to enhance the reliability and applicability of emotion detection systems in real-time scenarios. The research investigates model architectures, preprocessing methods and dataset utilization to develop a system capable of recognizing diverse emotions effectively.

Implementation :

The proposed system for emotion detection is built using deep learning algorithms. We used MobileNetV2 and VGG19 model architectures to train our system and after getting results, we finalize the model architecture with better results.

Algorithm:



1. Dataset :

FERAC Dataset(image dataset) from Kaggle.

<https://www.kaggle.com/datasets/rajasreechaiti/ferac-dataset>

2. Loading Dataset and Preprocessing

```
import os  
import cv2  
from zipfile import ZipFile
```

```
# 1. Unzip dataset.zip into the current working directory
```

```
with ZipFile("dataset.zip", 'r') as zip_ref:
```

```
    zip_ref.extractall()
```

```

# 2. Set dataset paths based on your structure
train_folder = "train"
test_folder = "test"
processed_data_path = "Processed_Data"

# 3. Create output directories if they don't exist
if not os.path.exists(processed_data_path):
    os.makedirs(processed_data_path)
    os.makedirs(os.path.join(processed_data_path, "train"))
    os.makedirs(os.path.join(processed_data_path, "test"))

IMG_SIZE = (224, 224)

# 4. Image preprocessing function
def preprocess_images(source_dir, target_dir, img_size=IMG_SIZE):
    emotions = os.listdir(source_dir)
    for emotion in emotions:
        emotion_path = os.path.join(source_dir, emotion)
        if not os.path.isdir(emotion_path):
            continue
        target_emotion_path = os.path.join(target_dir, emotion)
        if not os.path.exists(target_emotion_path):
            os.makedirs(target_emotion_path)
        for img_name in os.listdir(emotion_path):
            img_path = os.path.join(emotion_path, img_name)
            try:
                img = cv2.imread(img_path)
                if img is None:
                    continue
                img = cv2.resize(img, img_size)
                if len(img.shape) == 2:
                    img = cv2.cvtColor(img, cv2.COLOR_GRAY2RGB)
                elif img.shape[2] == 4:
                    img = cv2.cvtColor(img, cv2.COLOR_RGBA2RGB)
                elif img.shape[2] == 1:
                    img = cv2.cvtColor(img, cv2.COLOR_GRAY2RGB)
                target_img_path = os.path.join(target_emotion_path, img_name)
                cv2.imwrite(target_img_path, img)
            except Exception as e:
                print(f"Error processing {img_path}: {e}")
                continue

# 5. Run preprocessing on train and test
if os.path.exists(train_folder):
    print("Preprocessing training images...")
    preprocess_images(train_folder, os.path.join(processed_data_path, "train"))

```

```

else:
    print("Training folder not found!")
if os.path.exists(test_folder):
    print("Preprocessing testing images..")
    preprocess_images(test_folder, os.path.join(processed_data_path, "test"))
else:
    print("Testing folder not found!")
print("Preprocessing completed!")

```

3. Data Augmentation:

```

# Create data generators
batch_size = 20
# Data augmentation for training
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True,
    zoom_range=0.2,
    validation_split=0.2
)
# Only rescaling for validation and testing
test_datagen = ImageDataGenerator(rescale=1./255)
# Training data generator
train_generator = train_datagen.flow_from_directory(
    os.path.join(processed_data_path, "train"),
    target_size=IMG_SIZE,
    batch_size=batch_size,
    class_mode='categorical',
    subset='training')

# Validation data generator
validation_generator = train_datagen.flow_from_directory(
    os.path.join(processed_data_path, "train"),
    target_size=IMG_SIZE,
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation')
# Test data generator
test_generator = test_datagen.flow_from_directory(
    os.path.join(processed_data_path, "test"),
    target_size=IMG_SIZE,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)

```

4. MobileNetV2 Training :

```
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Dropout, GlobalAveragePooling2D,
BatchNormalization, Input
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping

# Data augmentation
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True,
    zoom_range=0.2
)

val_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    'Processed_Data/train',    # updated path!
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical'
)

val_generator = val_datagen.flow_from_directory(
    'Processed_Data/test',    # updated path!
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical'
)

# MobileNetV2 base model (pretrained on ImageNet)
mobileNet_base = tf.keras.applications.MobileNetV2(
    input_shape=(224, 224, 3),
    include_top=False,
    weights='imagenet'
)
mobileNet_base.trainable = False # Freeze base layers

# Add custom classifier on top
inputs = Input(shape=(224, 224, 3))
x = mobileNet_base(inputs, training=False)
```

```

x = GlobalAveragePooling2D()(x)
x = BatchNormalization()(x)
x = Dropout(0.5)(x)
x = Dense(128, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.5)(x)
outputs = Dense(train_generator.num_classes, activation='softmax')(x)

mobileNet_model = Model(inputs, outputs)
mobileNet_model.summary()

mobileNet_model.compile(
    optimizer=Adam(learning_rate=0.0005),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

history = mobileNet_model.fit(
    train_generator,
    validation_data=val_generator,
    epochs=150
)

```

5. VGG19 Training :

```

from tensorflow.keras.applications import VGG19
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense, Dropout,
BatchNormalization, Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam

# Data augmentation
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True,
    zoom_range=0.2
)

val_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    'Processed_Data/train',  # updated path!
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical'
)

```

```

    )

val_generator = val_datagen.flow_from_directory(
    'Processed_Data/test',    # updated path!
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical'
)

# Load VGG16 pretrained on ImageNet
vgg_base = VGG19(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
vgg_base.trainable = False # Freeze the base layers

# Custom classifier
inputs = Input(shape=(224, 224, 3))
x = vgg_base(inputs, training=False)
x = GlobalAveragePooling2D()(x)
x = BatchNormalization()(x)
x = Dropout(0.5)(x)
x = Dense(256, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.5)(x)
outputs = Dense(train_generator.num_classes, activation='softmax')(x)

vgg_model = Model(inputs, outputs)
vgg_model.compile(optimizer=Adam(learning_rate=0.0005),
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])

# Train VGG16 model
vgg_history = vgg_model.fit(
    train_generator,
    validation_data=val_generator,
    epochs=150
)

6. Predictions :
import cv2
import numpy as np
import matplotlib.pyplot as plt

def predict_emotion(img_path, model, img_size=IMG_SIZE):
    # Load the image
    img = cv2.imread(img_path)
    if img is None:
        raise ValueError("Image not found or cannot be read.")

    # Preprocess image: resize and convert to RGB

```

```

img = cv2.resize(img, img_size)
if len(img.shape) == 2:
    img = cv2.cvtColor(img, cv2.COLOR_GRAY2RGB)
elif img.shape[2] == 4:
    img = cv2.cvtColor(img, cv2.COLOR_RGBA2RGB)

# Normalize pixel values
img = img.astype('float32') / 255.0

# Expand dimensions to fit model input shape
img_batch = np.expand_dims(img, axis=0) # Shape: (1, height, width, channels)

# Predict
preds = model.predict(img_batch)
pred_class = np.argmax(preds, axis=1)[0]
emotion_label = class_labels[pred_class] # Use class_labels from notebook scope

# Show result
plt.imshow(img)
plt.title(f"Predicted Emotion: {emotion_label}")
plt.axis('off')
plt.show()
print(f"Predicted Emotion: {emotion_label}")

```

#Prediction 1



```

# input image is
input_path_1 = 'Ntr.jpg'
predict_emotion(input_path_1, mobileNet_model)

```

1/1 ————— 0s 34ms/step

Predicted Emotion: anger



Predicted Emotion: anger

#Prediction 2



input image is

```
input_path_2 = 'Rohit.jpg'  
predict_emotion(input_path_2, mobileNet_model)
```

1/1 ————— 0s 33ms/step

Predicted Emotion: joy

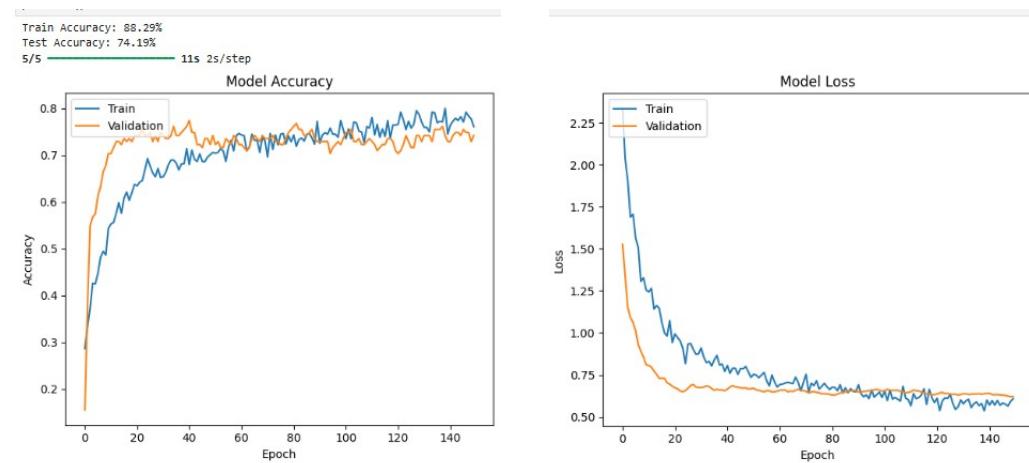


Predicted Emotion: joy

Results :

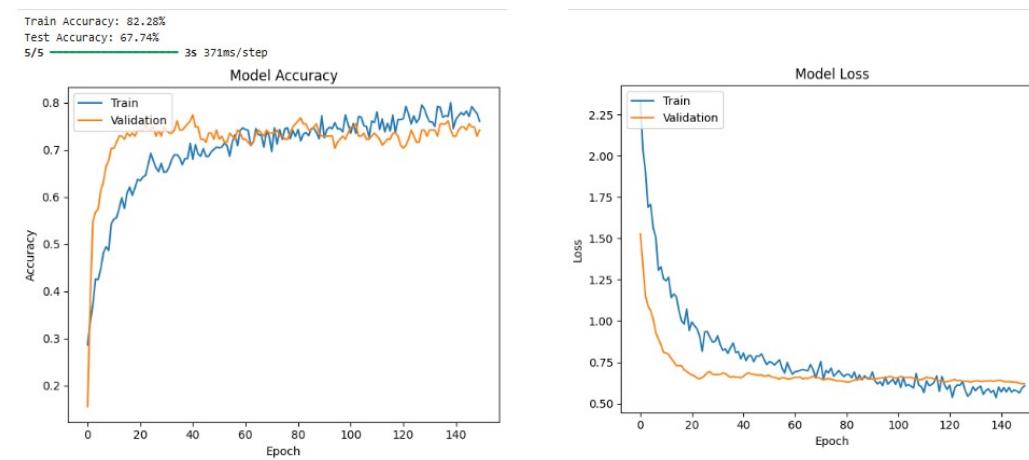
For MobileNetV2 Model :

Training Accuracy is 88.29% and Test Accuracy is 74.19%

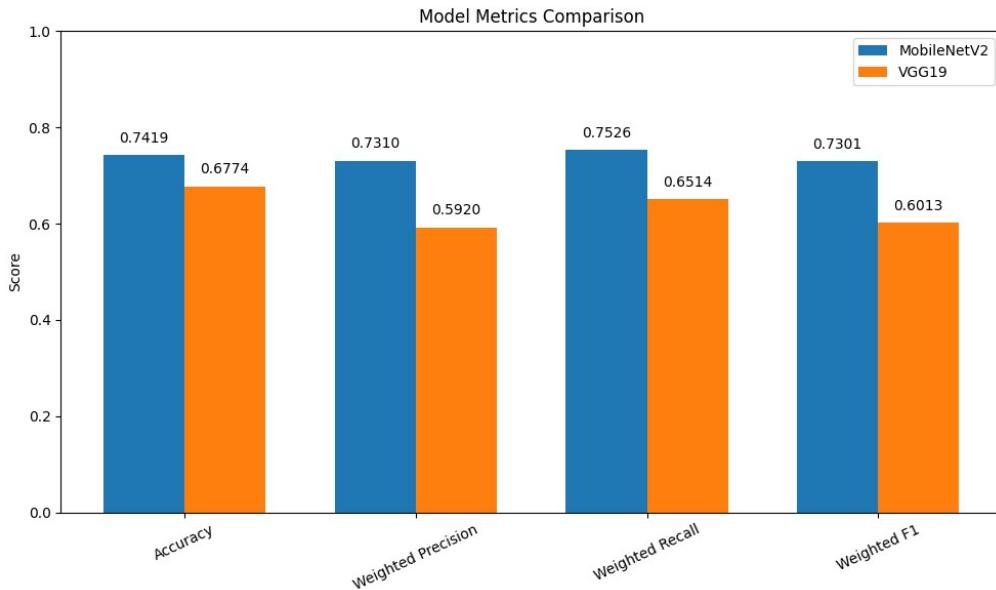


For VGG19 Model :

Training Accuracy is 82.28% and Test Accuracy is 67.74%



Comparision of both models :



From the above graph we can clearly understand that MobileNetV2 model got better results than VGG19 in all metrics calculated.

Conclusion and Future Work :

Our Emotion Detection System uses MobileNetV2 model that effectively classifies emotion of the person in Input image after preprocessing into standard size(224 x 224) pixels. We achieved 88.29% of Training accuracy and 74.19% of Validation accuracy for our MobileNetV2 model. These results prove the ability of the system to classify human emotion from an imput image. Our Future Work is to develop this model to real time detection by using web-cam of the device using OpenCV with pre-trained models like Haar Cascade Classifier.