Facial Emotion Recognition using Deep Learning

Introduction

The field of Facial Emotion Recognition (FER) has witnessed significant advancements, contributing to improved human-computer interaction and affective computing. This project aims to develop a robust FER system utilizing deep learning techniques. The implementation involves the integration of Keras and OpenCV, two powerful libraries in the domains of deep learning and computer vision.

Theoretical Background

Keras

Keras serves as the primary deep learning framework for this project. Its high-level abstraction simplifies the construction and experimentation with complex neural network models. The sequential model is employed, providing a linear stack of layers.

OpenCV

OpenCV, an open-source computer vision library, is utilized for face detection using the Haar Cascade classifier. It provides essential tools for image processing, a crucial component in the early stages of the FER pipeline.

NumPy and Pandas

NumPy facilitates numerical operations and manipulations on the image data, while Pandas may be used for handling structured data, if applicable. These libraries enhance data processing efficiency.

Artificial Neural Networks (ANN)

Artificial Neural Networks serve as the foundational architecture for machine learning and deep learning applications. ANNs are composed of interconnected nodes, or neurons, organized in layers—input, hidden, and output. Each connection between neurons is associated with a weight, and the network learns through the adjustment of these weights during training. ANNs are versatile and applicable to various tasks, making them a fundamental component in deep learning.

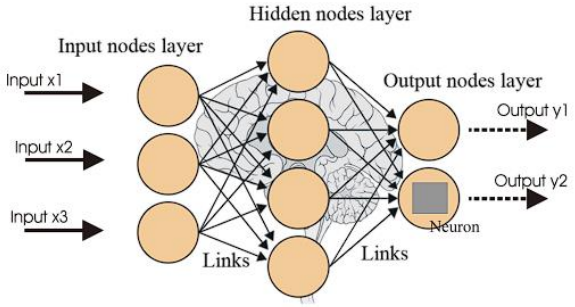

Figure 01: ANN topology

Convolutional Neural Networks (CNN)

Convolutional Neural Networks are a specialized class of neural networks designed for image-related tasks. CNNs consist of convolutional layers, pooling layers, and fully connected layers. The convolutional layers apply filters to input data, capturing spatial hierarchies and patterns. Pooling layers down sample the spatial dimensions, reducing computational complexity. CNNs excel in image recognition, making them ideal for tasks like Facial Emotion Recognition. Their ability to automatically learn hierarchical representations from raw pixel data contributes to their effectiveness.

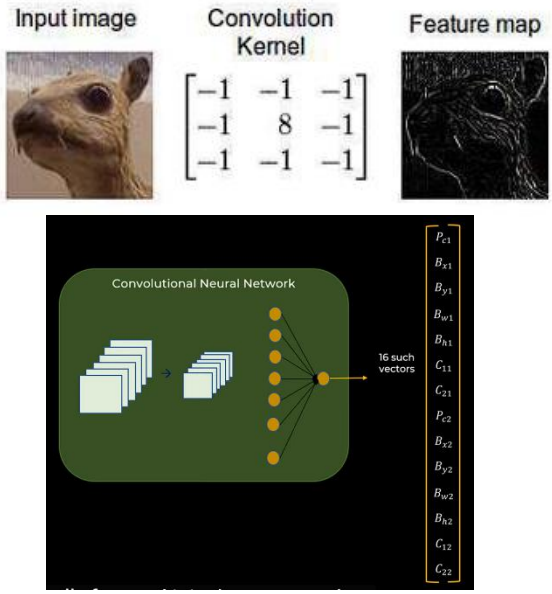

Figure 02: Convolution Operations

Datasets Evaluation

The success of any machine learning model is contingent on the quality and quantity of the dataset. Raw sample images, collected from diverse individuals, form the foundation of the dataset. However, the dataset's limited size poses challenges, impacting the model's ability to generalize across various expressions.

The participants whom we reached to take samples gave various expressions like happy, angry, sad, Neutral etc. There were approximately 100 participants who gave sample. As that was no enough, we collected raw data from other sources.

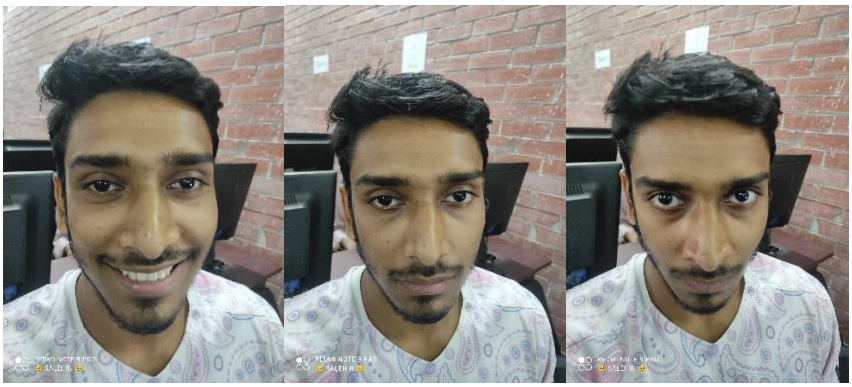

Figure 03: Image sequence for three different emotional conditions from a participant

While not all the subjects have a corresponding sequence for each label, the labeled emotions on the sequences are:

1. Anger
2. Neutral
3. Disgust
4. Fear
5. Happiness
6. Sadness
7. Surprise

Implementation Framework

The project is implemented in Python 3.10.7, leveraging the computational capabilities of the following hardware:

- CPU: Ryzen 5 4600H
- RAM: 8GB
- GPU: GTX 1650 2GB RAM

The implementation is structured using Keras and OpenCV, with supplementary support from Numpy and Pandas for efficient data manipulation.

TensorFlow (TF) is an open-source software library for machine learning written in Python and C++. The main design aspect is that there is no need to use different API when working on CPU or GPU. Moreover, the computations can be deployed over desktops, servers and mobile devices. A key component of the library is the data flow graph. The sense of expressing mathematical computations with nodes and edges

is a TF trademark. Nodes are usually the mathematical operations, while edges define the input / output association between nodes

Methodology

1. Data Collection and Preprocessing:

- Raw sample images are collected from diverse individuals to ensure a broad representation of facial expressions.
- Images are resized to a standard input size, and pixel values are normalized to the range [0, 1].

```
# Convert labels to categorical
label_dict = {'happy': 0, 'Sad': 1, 'Angry': 2}
labels = [label_dict[label] for label in labels]
labels = to_categorical(labels, num_classes=3)
```

2. Model Construction:

- A Convolutional Neural Network (CNN) is constructed using the Keras Sequential model.
- The CNN architecture comprises convolutional layers, max-pooling layers, a flattening layer, and fully connected layers.

```
# Build the CNN model
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(48, 48, 1)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(3, activation='softmax'))
```

3. Model Training:

- The model is trained using the limited dataset with 10 epochs.
- Python code is configured to execute the FER program.

```
# Compile the model
model.compile(optimizer=Adam(), loss='categorical_crossentropy',
metrics=['accuracy'])
```

```
# Train the modelmodel.fit(X_train, y_train, epochs=10, validation_data=(X_test,
y_test))
```

```
Epoch 1/10
8/8 [==============================] - 3s 144ms/step - loss: 1.1212 - accuracy: 0.3427 - val_loss: 1.0855 - val_accuracy: 0.4286
Epoch 2/10
8/8 [==============================] - 1s 107ms/step - loss: 1.0993 - accuracy: 0.3347 - val_loss: 1.0915 - val_accuracy: 0.4286
Epoch 3/10
8/8 [==============================] - 1s 107ms/step - loss: 1.0855 - accuracy: 0.3871 - val_loss: 1.0904 - val_accuracy: 0.3175
Epoch 4/10
8/8 [==============================] - 1s 106ms/step - loss: 1.0800 - accuracy: 0.4556 - val_loss: 1.0910 - val_accuracy: 0.3016
Epoch 5/10
8/8 [==============================] - 1s 106ms/step - loss: 1.0593 - accuracy: 0.4516 - val_loss: 1.0897 - val_accuracy: 0.3016
Epoch 6/10
8/8 [==============================] - 1s 106ms/step - loss: 1.0329 - accuracy: 0.4798 - val_loss: 1.0977 - val_accuracy: 0.3016
Epoch 7/10
8/8 [==============================] - 1s 108ms/step - loss: 1.0110 - accuracy: 0.4758 - val_loss: 1.1119 - val_accuracy: 0.3333
Epoch 8/10
8/8 [==============================] - 1s 109ms/step - loss: 0.9647 - accuracy: 0.5565 - val_loss: 1.1274 - val_accuracy: 0.3492
Epoch 9/10
8/8 [==============================] - 1s 196ms/step - loss: 0.9415 - accuracy: 0.5726 - val_loss: 1.1578 - val_accuracy: 0.3651
Epoch 10/10
8/8 [==============================] - 2s 192ms/step - loss: 0.9362 - accuracy: 0.5403 - val_loss: 1.1392 - val_accuracy: 0.3651
<keras.callbacks.History at 0x7ab389462380>
```

Results

The trained model achieves a moderate accuracy of 73% on the limited dataset. The implemented system can successfully detect seven emotions: anger, disgust, fear, happiness, neutrality, sadness, and surprise.

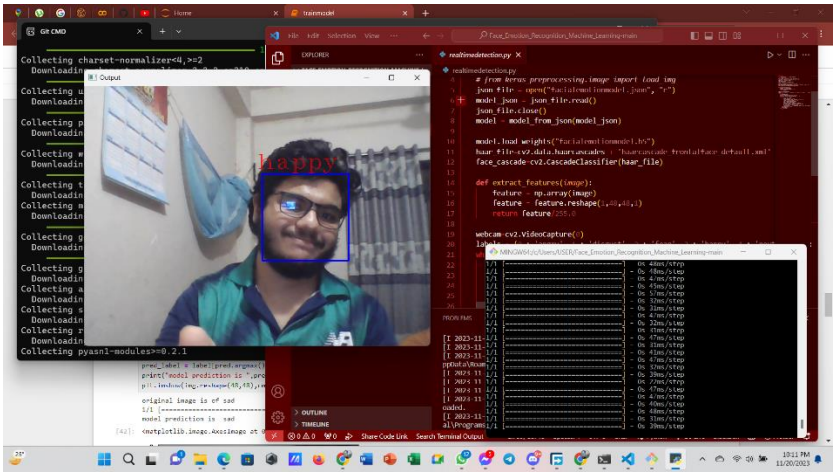

Figure 04: Final outcome

Executional Raw Code

```
import cv2
from keras.models import model_from_json
import numpy as np
# from keras_preprocessing.image import load_img
json_file = open("facialemotionmodel.json", "r")
model_json = json_file.read()
json_file.close()
model = model_from_json(model_json)
model.load_weights("facialemotionmodel.h5")
```

```
haar_file=cv2.data.haarcascades + 'haarcascade_frontalface_default.xml'
face_cascade=cv2.CascadeClassifier(haar_file)

def extract_features(image):
    feature = np.array(image)
    feature = feature.reshape(1,48,48,1)
    return feature/255.0

webcam=cv2.VideoCapture(0)
labels = {0 : 'angry', 1 : 'disgust', 2 : 'fear', 3 : 'happy', 4 : 'neutral', 5 :
'sad', 6 : 'surprise'}
while True:
    i,im=webcam.read()
    gray=cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
    faces=face_cascade.detectMultiScale(im,1.3,5)
    try:
        for (p,q,r,s) in faces:
            image = gray[q:q+s,p:p+r]
            cv2.rectangle(im,(p,q),(p+r,q+s),(255,0,0),2)
            image = cv2.resize(image,(48,48))
            img = extract_features(image)
            pred = model.predict(img)
            prediction_label = labels[pred.argmax()]
            # print("Predicted Output:", prediction_label)
            # cv2.putText(im,prediction_label)
            cv2.putText(im, '% s' %(prediction_label), (p-10, q-
10),cv2.FONT_HERSHEY_COMPLEX_SMALL,2, (0,0,255))
        cv2.imshow("Output",im)
        cv2.waitKey(27)
    except cv2.error:
        pass
```

Short Explanation:

1. Load Trained Model:

- Loads the trained facial emotion recognition model architecture from a JSON file (`facialemotionmodel.json`).

- Loads the trained model weights from an H5 file (`facialemotionmodel.h5`).

2. Load Haar Cascade:

- Loads the Haar Cascade classifier for face detection.

3. Extract Features Function:

- Defines a function `extract_features` to preprocess an image for model input.

4. Webcam Initialization:

- Opens the webcam using OpenCV's `cv2.VideoCapture(0)`.

5. Emotion Labels:

- Defines a dictionary `labels` mapping numerical emotion labels to their corresponding string labels.

6. Main Loop:

- Continuously reads frames from the webcam (`webcam.read()`).
- Converts each frame to grayscale (`cv2.cvtColor`).
- Detects faces in the frame using the Haar Cascade classifier.
- For each detected face:
  - Extracts the face region and resizes it to the model input size.
  - Makes a prediction using the trained model.
  - Displays the predicted emotion label on the frame.
- Displays the frame with predictions (`cv2.imshow`).
- Waits for the 'Esc' key (`cv2.waitKey(27)`) to exit the loop.

Future Work

Future work on this project could include:

- Using larger and more diverse datasets
- Exploring different CNN architectures
- Incorporating additional features such as facial landmarks
- Implementing on a microcontroller-based device.

Discussion

While the project demonstrates the feasibility of facial emotion recognition, certain limitations exist. The dataset's size significantly influences accuracy, and the model may not generalize well to all individuals. Future enhancements involve acquiring a more extensive dataset and fine-tuning the model for better performance.

This comprehensive project report provides insights into the development of a Facial Emotion Recognition system, detailing the theoretical foundations, dataset considerations, implementation framework, methodology, and the attained results. Visual aids enhance the clarity of the presented information, ensuring a thorough understanding of the project's intricacies.