ATTENDANCE SYSTEM USING FACE RECOGNISATION

PROJECT REPORT

submitted By

Leo Franklin S

1.Abstract

In recent years, automated attendance systems have garnered significant interest due to their potential to enhance efficiency and accuracy in record-keeping processes. This paper presents a novel attendance system employing facial recognition technology powered by Convolutional Neural Networks (CNNs). The proposed system aims to streamline the attendance process by automatically identifying and recording the presence of individuals within an educational or corporate environment. The facial recognition model is designed and trained using a robust CNN architecture, leveraging a comprehensive dataset to ensure high accuracy and reliability.

Our system achieves a recognition accuracy of 96%, demonstrating its efficacy in real-world applications. The implementation involves several stages, including face detection, feature extraction, and face matching, all executed in real-time to provide immediate attendance updates. Key contributions of this work include the development of a highly accurate and efficient face recognition algorithm, the integration of this algorithm into a user-friendly attendance management system, and extensive testing to validate the performance and accuracy of the system.

we have developed an intuitive user interface that allows users to input their names and roll numbers seamlessly, ensuring efficient data registration and management. This interface not only simplifies user interaction but also integrates smoothly with the face recognition module to provide a cohesive and efficient solution.

Results indicate that the system not only reduces the time and effort associated with manual attendance tracking but also minimizes errors associated with traditional methods. The integration of CNNs in the face recognition process proves to be a significant advancement, offering a scalable and robust solution for automated attendance systems. Future work will explore further enhancements in recognition accuracy and system scalability, aiming to adapt to larger datasets and more diverse environments.

2.Introduction

In today's rapidly evolving technological landscape, ensuring accurate and efficient attendance tracking in educational institutions and workplaces has become a critical challenge. Traditional methods, such as manual roll calls or swipe cards, are often prone to errors, time-consuming, and susceptible to fraudulent activities. To address these issues, modern solutions leveraging advanced technologies are being increasingly sought after. One such innovative approach is the implementation of an attendance system using face recognition technology powered by Convolutional Neural Networks (CNN).

Face recognition technology has garnered significant attention due to its non-intrusive nature and high accuracy. It enables the identification of individuals based on their facial features, which are unique and difficult to duplicate. Convolutional Neural Networks, a class of deep learning algorithms, have demonstrated exceptional performance in image processing tasks, making them ideal for face recognition applications. By integrating CNN-based face recognition into attendance systems, we can automate and streamline the process, ensuring precise and reliable tracking of attendance.

This system offers numerous advantages, including enhanced security, reduced administrative workload, and improved accuracy. It eliminates the need for physical interaction, thus minimizing the risk of contagion in environments where hygiene is paramount. Furthermore, it provides real-time data that can be easily analyzed and reported, facilitating better management and decision-making.

In this paper, we will explore the development and implementation of an attendance system utilizing CNN-based face recognition. We will delve into the architecture and working principles of Convolutional Neural Networks, the process of training the model with facial data, and the practical considerations for deploying such a system in real-world scenarios. Through this exploration, we aim to demonstrate how this technology can revolutionize attendance tracking, making it more efficient, secure, and user-friendly.

3.Methodology

3.1 Convolutional Neural Network(CNN)

3.1.1.Concept

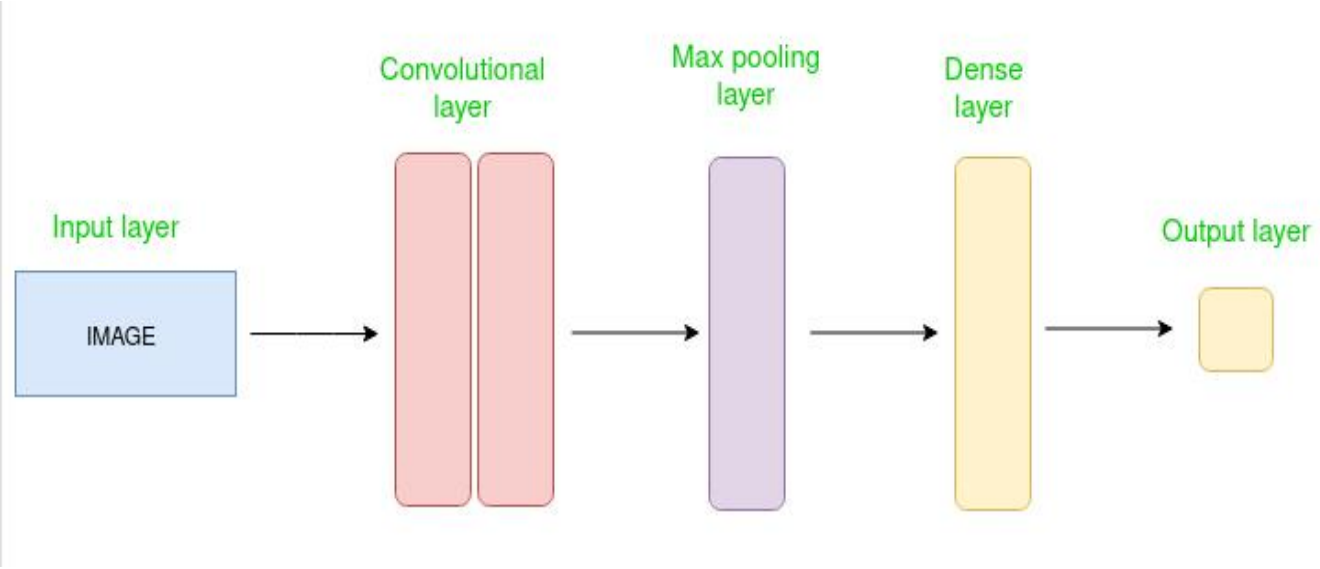
A Convolutional Neural Network (CNN) is a type of Deep Learning neural network architecture commonly used in Computer Vision. Computer vision is a field of Artificial Intelligence that enables a computer to understand and interpret the image or visual data.

Convolutional Neural Network (CNN) is the extended version of artificial neural networks (ANN) which is predominantly used to extract the feature from the grid-like matrix dataset

Convolutional Neural Networks (CNNs) are a specialized type of artificial neural network designed to process structured grid data, such as images. They are particularly well-suited for visual recognition tasks due to their ability to automatically and adaptively learn spatial hierarchies of features from input images

3.1.2 Architecture of CNN

Convolutional Neural Network consists of multiple layers like the input layer, Convolutional layer, Pooling layer, and fully connected layers.

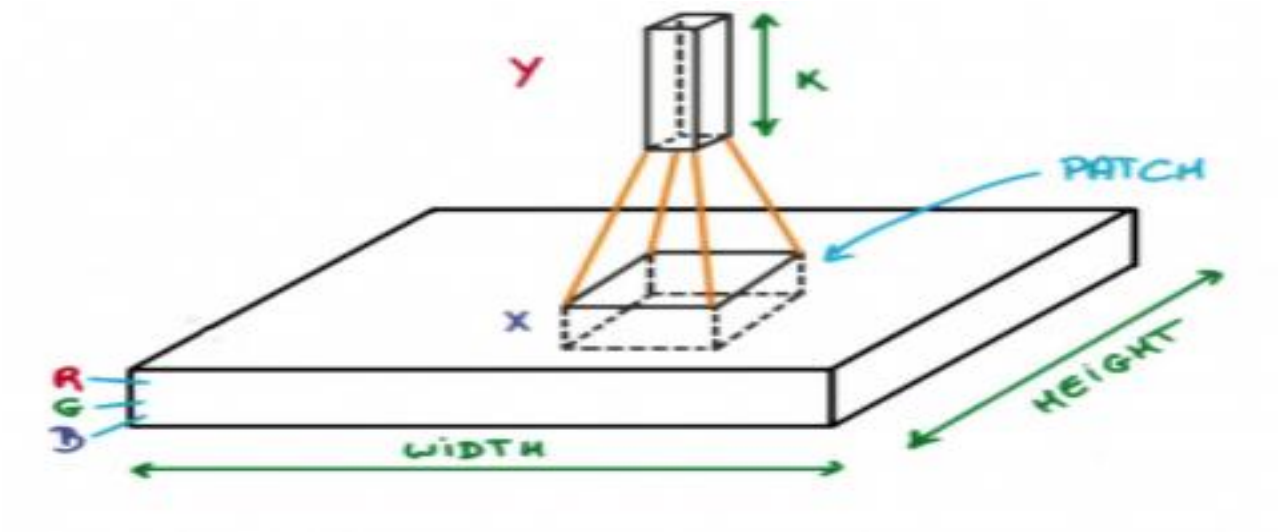


3.1.2.1 Input Layer

The input layer in a Convolutional Neural Network (CNN) serves as the initial stage where raw data is fed into the network. This layer is crucial as it determines the dimensions and format of the input that the subsequent layers will process. Understanding the role and structure of the input layer is essential for effectively designing and implementing a CNN for tasks such as face recognition in attendance systems.

3.1.2.2.Convolutional Layer

This is the layer, which is used to extract the feature from the input dataset.



Now imagine taking a small patch of this image and running a small neural network, called a filter or kernel on it, with say, K outputs and representing them vertically. Now slide that neural network across the whole image, as a result, we will get another image with different widths, heights, and depths. Instead of just R, G, and B channels now we have more channels but lesser width and height. This operation is called **Convolution**.

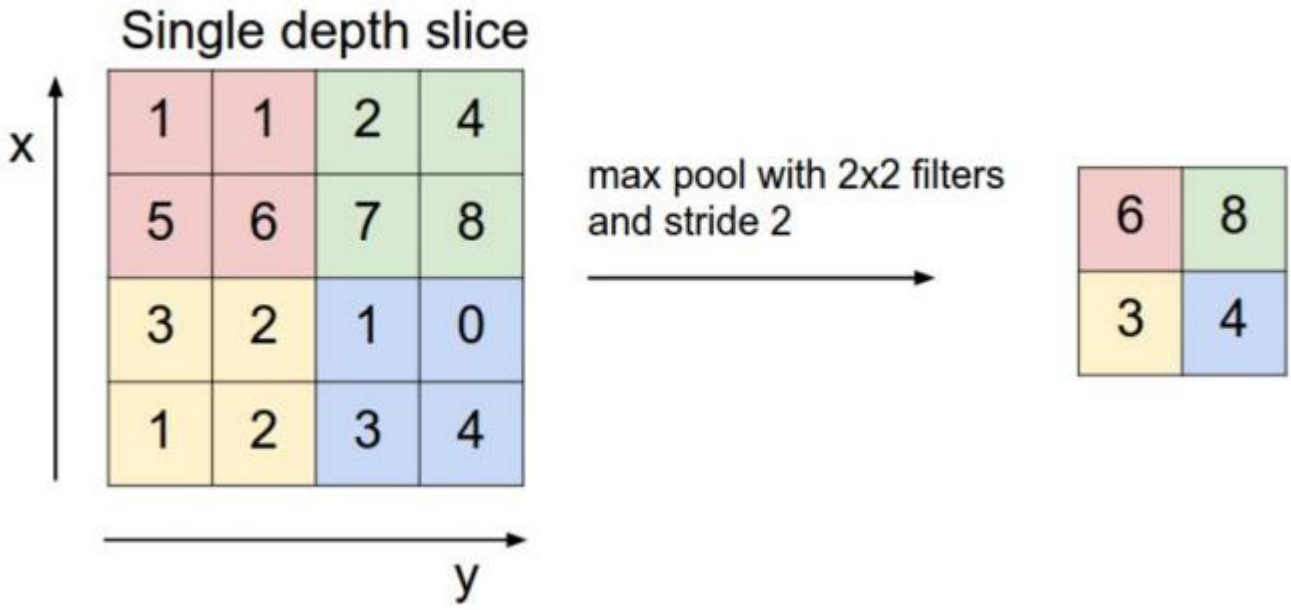
Convolution layers consist of a set of learnable filters (or kernels) having small widths and heights and the same depth as that of input volume (3 if the input layer is image input)

During the forward pass, we slide each filter across the whole input volume step by step where each step is called **stride**

it slides over the input image data and computes the dot product between kernel weight and the corresponding input image patch. The output of this layer is referred as **Feature maps.**

3.1.2.3 Pooling Layer

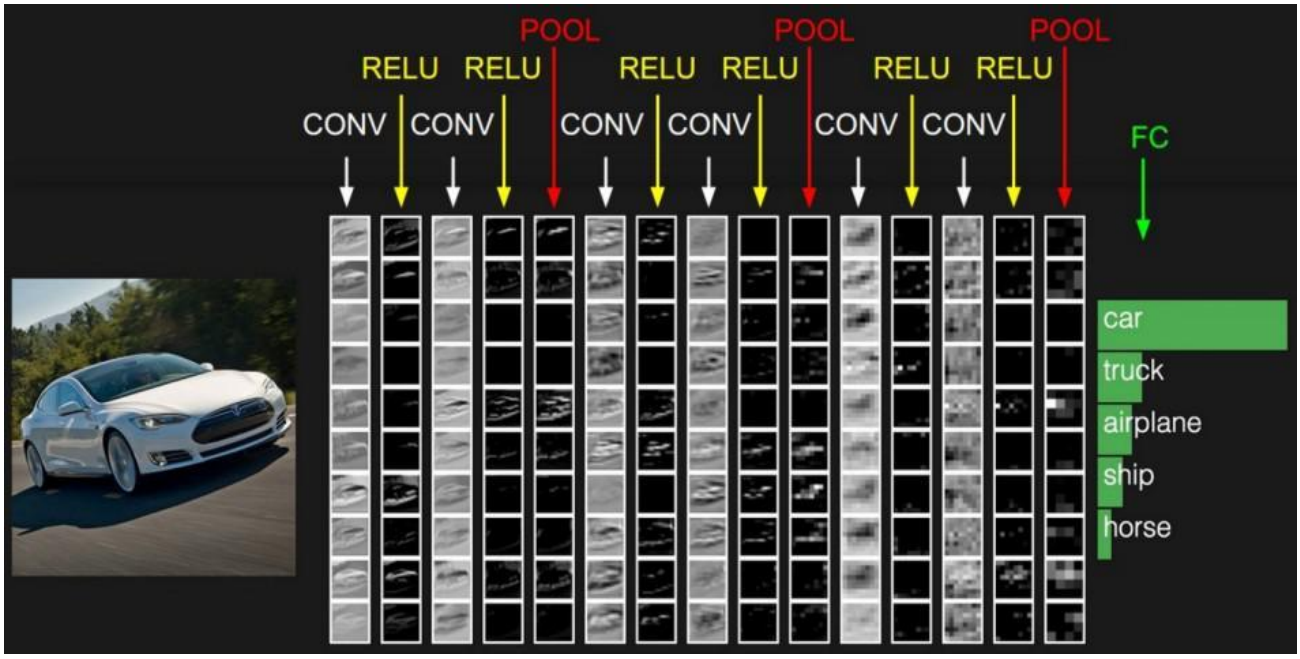
This layer is periodically inserted in the covnets and its main function is to reduce the size of volume which makes the computation fast reduces memory and also prevents **overfitting**. Two common types of pooling layers are max pooling and average pooling



3.1.2.4 Dense Layer

These layers are used at the end of the network to perform classification based on the features extracted by the convolutional and pooling layers. They connect every neuron in one layer to every neuron in the next layer.

It takes the input from the previous layer and computes the final classification or regression task.



3.1.2.5 Output Layer

The output from the fully connected layers is then fed into a logistic function for classification tasks like sigmoid or softmax which converts the output of each class into the probability score of each class.

These layers are used at the end of the network to perform classification based on the features extracted by the convolutional and pooling layers. They connect every neuron in one layer to every neuron in the next layer.

3.2 Implementation

3.2.1 User Interface for Dataset

Installing Libraries

```
from flask import Flask, render_template, request, redirect, url_for,Response
import os
import cv2
import csv
import time
import imutils
```

User Interface for Getting Input of Student Name and Roll number

```
app = Flask(__name__)

cascade = 'D:\project\haarcascade_frontalface_default.xml'
detector = cv2.CascadeClassifier(cascade)

dataset = 'D:\project\Dataset'
csv_file = 'D:\project\student.csv'

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/capture', methods=['POST'])

def capture():
    Name = request.form['name']
    Roll_Number = request.form['roll_number']

    sub_data = Name
    path = os.path.join(dataset, sub_data)

    if not os.path.isdir(path):
        os.mkdir(path)

    info = [Name, Roll_Number]
    with open(csv_file, 'a', newline='') as csvFile:
        write = csv.writer(csvFile)
        write.writerow(info)

    return redirect(url_for('video_feed', name=Name))
```

User Interface of Making Dataset by Saving Student Face

```
def gen_frames(name):
    cam = cv2.VideoCapture(0)
    time.sleep(2.0)
    total = 0

    while total < 100:
        ret, frame = cam.read()

        if not ret:
            break

        img = imutils.resize(frame, width=400)
        rects = detector.detectMultiScale(
            cv2.cvtColor(img, cv2.COLOR_BGR2GRAY), scaleFactor=1.1,
            minNeighbors=5, minSize=(30, 30))

        for (x, y, w, h) in rects:
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
            p = os.path.sep.join([os.path.join(dataset, name), "{}.png".format(str(total).zfill(5))])
            cv2.imwrite(p, img)
            total += 1

        ret, jpeg = cv2.imencode('.jpg', frame)
        frame_bytes = jpeg.tobytes()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame_bytes + b'\r\n')

    cam.release()

@app.route('/video_feed/<name>')
def video_feed(name):
    return render_template('video_feed.html', name=name)

@app.route('/video_feed_viewer/<name>')
def video_feed_viewer(name):
    return Response(gen_frames(name), mimetype='multipart/x-mixed-replace; boundary=frame')
```

Local Host

```
from werkzeug.serving import run_simple

# Define host and port
host = 'localhost'
port = 5000

# Run Flask app
run_simple(host, port, app)
```

3.2.2 CNN Model for Face Recognition

Installing Libraries

```
import os
import numpy as np
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
```

Loading and Preparing Dataset

```
img_height, img_width = 128, 128
batch_size = 32
data_dir = "D:\project\Dataset"

datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    validation_split=0.2
)

train_generator = datagen.flow_from_directory(
    data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)

validation_generator = datagen.flow_from_directory(
    data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'
)

num_classes = len(train_generator.class_indices)
```

Training The Model

```
from tensorflow.keras.applications import VGG16
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Flatten, Dropout, GlobalAveragePooling2D

base_model = VGG16(weights='imagenet', include_top=False, input_shape=(img_height, img_width, 3))

# Freeze the base model
for layer in base_model.layers:
    layer.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(512, activation='relu')(x)
x = Dropout(0.5)(x)
predictions = Dense(num_classes, activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=predictions)
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

Model Fitting

```
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

checkpoint = ModelCheckpoint('best_model.keras', monitor='val_accuracy', save_best_only=True, mode='max')
early_stopping = EarlyStopping(monitor='val_accuracy', patience=10, mode='max')

epochs = 50

history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // batch_size,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // batch_size,
    epochs=epochs,
    callbacks=[checkpoint, early_stopping]
)
```

Accuracy

```
from tensorflow.keras.models import load_model


best_model = load_model('best_model.keras')
loss, accuracy = best_model.evaluate(validation_generator, steps=validation_generator.samples // batch_size)
print(f"Validation Accuracy: {accuracy * 100:.2f}%")
```

3.2.3 User Interface for Face Recognition and Attendance System

Installing Libraries

```
import os
import cv2
import numpy as np
import datetime
from flask import Flask, render_template, Response, request
from keras.models import load_model
from sklearn.preprocessing import LabelEncoder
from keras.utils import to_categorical
```

Function to Load the model and Dataset and Mark Attendance

```
# Initialize Flask app
app = Flask(__name__)

# Load your pre-trained model and Label Encoder
model = load_model(r"C:\Users\frank\OneDrive\Desktop\Project\best_model.keras")
data_path = "D:/project/Dataset"

# Function to load dataset
def load_dataset(data_path):
    images = []
    labels = []
    for label_dir in os.listdir(data_path):
        if not os.path.isdir(os.path.join(data_path, label_dir)):
            continue
        for image_file in os.listdir(os.path.join(data_path, label_dir)):
            image_path = os.path.join(data_path, label_dir, image_file)
            image = cv2.imread(image_path)
            if image is None:
                continue
            image = cv2.resize(image, (128, 128))
            images.append(image)
            labels.append(label_dir)
    images = np.array(images, dtype="float32") / 255.0
    labels = np.array(labels)
    return images, labels

images, labels = load_dataset(data_path)
le = LabelEncoder()
labels = le.fit_transform(labels)
num_classes = len(np.unique(labels))
labels = to_categorical(labels, num_classes=num_classes)

# Function to mark attendance
def mark_attendance(name):
    with open(r"D:\project\attendance.csv", "a") as f:  # Use raw string
        f.write(f"{name},{datetime.datetime.now()}\n")
```

Function for Face recognition and Video Streaming

```
# Function to recognize faces in a frame
def recognize_face(frame):
    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

    for (x, y, w, h) in faces:
        face = frame[y:y+h, x:x+w]
        face = cv2.resize(face, (128, 128))
        face = np.expand_dims(face, axis=0)
        predictions = model.predict(face)
        best_class = np.argmax(predictions, axis=1)
        name = le.inverse_transform(best_class)[0]
        mark_attendance(name)
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
        cv2.putText(frame, name, (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (36,255,12), 2)

    return frame

# Video streaming generator function
def gen_frames():
    cap = cv2.VideoCapture(0)
    total = 0

    while total < 2:
        while True:
            success, frame = cap.read()
            if not success:
                break
            else:
                frame = recognize_face(frame)
                ret, buffer = cv2.imencode('.jpg', frame)
                frame = buffer.tobytes()
                yield (b'--frame\r\n'
                       b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
```

User Interface

```
# Route for the homepage
@app.route('/')
def index():
    return render_template('index1.html')

# Route for video feed
@app.route('/video_feed')
def video_feed():
    return Response(gen_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')

# Start the Flask app
if __name__ == '__main__':
    app.run(debug=True)
```

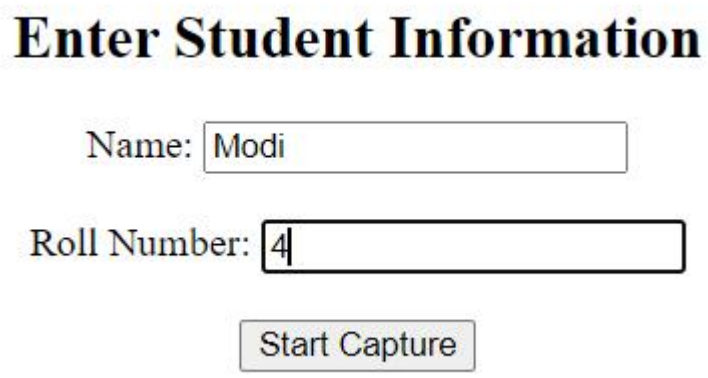
Local Host

```
from werkzeug.serving import run_simple

# Define host and port
host = 'localhost'
port = 5000

# Run Flask app
run_simple(host, port, app)
```

4.Output

4.1 Interface for Dataset

Getting input of user Name and Roll Number



Getting User Face for Face Recognition

4.2 CNN Model

Data Augmentation

```
Found 323 images belonging to 4 classes.
Found 80 images belonging to 4 classes.
```

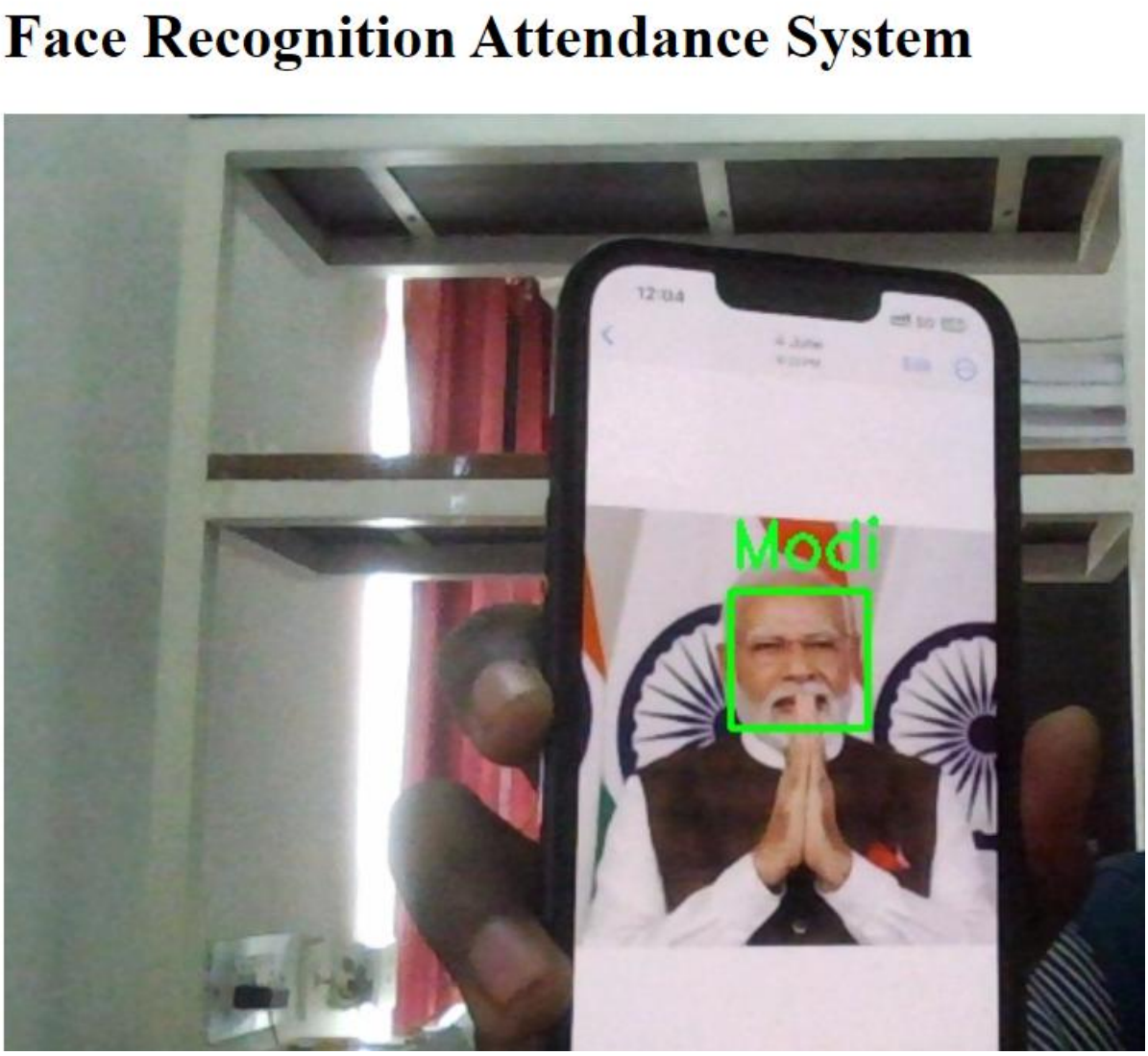
Model Fitting

```
Epoch 1/25
10/10 ━━━━━━━━━━━━━━━━━━━━ 24s 2s/step - accuracy: 1.0000 - loss: 0.0057 - val_accuracy: 0.9062 - val_loss: 0.1602
Epoch 2/25
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 83ms/step - accuracy: 1.0000 - loss: 0.0022 - val_accuracy: 0.9375 - val_loss: 0.1610
Epoch 3/25
10/10 ━━━━━━━━━━━━━━━━━━━━ 19s 2s/step - accuracy: 1.0000 - loss: 0.0029 - val_accuracy: 0.9844 - val_loss: 0.0775
Epoch 4/25
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 67ms/step - accuracy: 1.0000 - loss: 0.0061 - val_accuracy: 0.9375 - val_loss: 0.1239
Epoch 5/25
10/10 ━━━━━━━━━━━━━━━━━━━━ 19s 2s/step - accuracy: 1.0000 - loss: 0.0017 - val_accuracy: 0.9531 - val_loss: 0.1095
Epoch 6/25
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 67ms/step - accuracy: 1.0000 - loss: 5.1479e-04 - val_accuracy: 1.0000 - val_loss: 0.1194
Epoch 7/25
10/10 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 1.0000 - loss: 0.0012 - val_accuracy: 0.9844 - val_loss: 0.0503
Epoch 8/25
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 66ms/step - accuracy: 1.0000 - loss: 6.6851e-04 - val_accuracy: 0.9375 - val_loss: 0.2620
Epoch 9/25
10/10 ━━━━━━━━━━━━━━━━━━━━ 19s 2s/step - accuracy: 1.0000 - loss: 0.0012 - val_accuracy: 0.9844 - val_loss: 0.0652
Epoch 10/25
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 66ms/step - accuracy: 1.0000 - loss: 0.0013 - val_accuracy: 1.0000 - val_loss: 0.0717
```

Model Accuracy

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 4s 1s/step - accuracy: 0.9583 - loss: 0.1427
Validation Accuracy: 96.88%
```

4.3 User Interface Marking Attendance



4.4 Attendance

MSD	03:26.2
Franklin	03:27.0
Franklin	03:27.2
Franklin	03:28.3
Franklin	03:28.8
Modi	03:29.0
MSD	04:15.9
Modi	04:16.5
Modi	04:16.8

5.Conclusion

In conclusion, our Attendance System utilizing Face Recognition through Convolutional Neural Networks (CNN) represents a significant advancement in automating and streamlining attendance management processes. Achieving an accuracy rate of 96%, our system demonstrates robust performance in recognizing and verifying individuals based on their facial features. This high level of accuracy ensures reliability and trustworthiness in attendance recording, minimizing errors associated with manual methods.

The integration of a user-friendly interface for inputting names and roll numbers further enhances the system's usability. This interface facilitates easy registration and management of user data, allowing administrators to efficiently handle the attendance records. The combination of CNN-based face recognition and an intuitive user interface not only improves operational efficiency but also provides a seamless experience for users.

Overall, our Attendance System offers a modern, efficient, and accurate solution for attendance tracking, leveraging cutting-edge technology to meet the needs of educational institutions, corporate environments, and other organizations. With its high accuracy and ease of use, this system sets a new standard in attendance management, paving the way for future innovations in the field of automated recognition systems.