FACIAL RECOGNITION ATTENDANCE SYSTEM USING ARTIFICIAL INTELLIGENCE AND DEEP LEARNING

Project Report submitted by

VARUN R RAO	**VIJAY RAJ V**
(4NM20IS174)	(4NM20IS176)
VINYAS S SHETTY	**RISHABH NAIK**
(4NM20IS180)	(4NM20IS199)

Under the Guidance of

DR. RAVI B
Associate Professor

In partial fulfillment of the requirements for the award of

the Degree of

Bachelor of Engineering in Information Science Engineering

from

Visvesvaraya Technological University, Belagavi

Department of Information Science Engineering
NMAM Institute of Technology, Nitte - 574110
(An Autonomous Institution affiliated to VTU, Belagavi)

APRIL 2024

N.M.A.M. INSTITUTE OF TECHNOLOGY
(An Autonomous Institution affiliated to Visvesvaraya Technological University, Belagavi)
Nitte – 574 110, Karnataka, India

NITTE EDUCATION TRUST

DEPARTMENT OF INFORMATION SCIENCE ENGINEERING

CERTIFICATE

Certified that the project work entitled

" FACIAL RECOGNITION ATTENDANCE SYSTEM USING
ARTIFICIAL INTELLIGENCE AND DEEP LEARNING "

is a bonafide work carried out by

VARUN R RAO VIJAY RAJ V
VINYAS S SHETTY RISHABH NAIK

in partial fulfilment of the requirements for the award of

Bachelor of Engineering Degree in Information Science Engineering

prescribed by ***Visvesvaraya Technological University, Belagavi***

during the year ***2023-2024****.*

It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library.

The project report has been approved as it satisfies the academic requirements in respect of the project work prescribed for the Bachelor of Engineering Degree.

Signature of the Guide	***Signature of the HOD***	***Signature of the Principal***

Semester End Viva Voce Examination

Name of the Examiners	**Signature with Date**
1.	
2.	

ACKNOWLEDGEMENT

Any achievement, be it scholastic or otherwise does not depend solely on the individual efforts but on the guidance, encouragement and cooperation of intellectuals, elders and friends. A number of personalities, in their own capacities have helped us in carrying out this major-project work. We would like to take this opportunity to thank them all.

First and foremost, we would like to thank **Dr. Niranjan N Chiplunkar**, Principal, NMAMIT, Nitte, for his moral support towards completing my mini-project work.

We would like to thank **Dr. Ashwini B**, Head of the Department, Information Science & Engineering, NMAMIT, Nitte, for her valuable suggestions and expert advice.

We also extend my cordial thanks to Major Project Coordinator **Mr. Vasudeva Pai**, and Co-coordinators: Mr. Abhishek S Rao, Dr. Manjula Gururaj Rao and Dr. Naganna Chetty for their support and guidance.

We deeply express my sincere gratitude to my guide **Dr. Ravi B**, Associate Professor**,** Department of ISE, NMAMIT, Nitte, for his able guidance, regular source of encouragement and assistance throughout this mini-project work.

We thank our Parents and all the Faculty members of Department of Information Science & Engineering for their constant support and encouragement.

Last, but not the least, we would like to thank our peers and friends who provided us with valuable suggestions to improve our major-project.

ABSTRACT

Manual attendance recording methods, such as taking roll call or swiping cards, are time-consuming, error-prone, and lack security. To address these challenges, this project proposes a facial recognition attendance system (FRAS) using Artificial Intelligence (AI) and Deep learning (DL).

The FRAS consists of three main components: webcam video capture, facial recognition, and attendance recording. Webcam video captures faces of individuals. Facial recognition extracts facial features from the captured images and compares them to the face label from pretrained model. Attendance logging marks the attendance of the individual based on the facial recognition comparison.

AI and DL play crucial roles in the FRAS. AI algorithms detect faces, extract facial features, and compare them to the stored facial templates. DL algorithms train and improve the accuracy of facial recognition.

FRAS offers several advantages over traditional methods: increased efficiency, improved accuracy, enhanced security, and reduced administrative burden. While challenges like image quality and algorithm bias exist, ongoing research and development are addressing them. FRAS is becoming adopted due adopted due to its numerous benefits.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

2.2 General Introduction

In today's fast-paced world, efficiency, accuracy, and security are paramount in various aspects of our lives, including attendance management. Traditional methods of attendance recording, such as manual roll call or card swiping, have long been criticized for their time-consuming nature, susceptibility to human error, and lack of robust security measures. To address these shortcomings, the Facial Recognition Automatic Attendance System (FRAS) has emerged as a revolutionary technology, offering a more efficient, accurate, and secure alternative for attendance management. FRAS is both cost-effective and efficient when contrasted to other biometric solutions. The cost and time saved are even larger because the data acquired from it is accurate in real-time. Because it is automated, human intervention is limited. As a result, there is no requirement for additional employees to perform this work manually. Face attendance system reduces costs while enhancing operational efficiency.

In the contemporary landscape, the demand for efficient, accurate, and secure attendance management solutions has intensified across various sectors. Traditional methods like manual roll call or card swiping have faced criticism due to their time-consuming nature, susceptibility to errors, and inadequate security measures. In response, the market has witnessed the emergence of innovative technologies, among which Facial Recognition Automatic Attendance Systems (FRAS) stand out as a groundbreaking solution. Offering unparalleled efficiency, FRAS optimizes the attendance recording process by automating it. This not only saves valuable time but also reduces the likelihood of errors inherent in manual methods. Furthermore, FRAS enhances security by leveraging advanced biometric technology to verify the identity of individuals, effectively mitigating the risks associated with unauthorized access or fraudulent activities. Its cost-effectiveness is another noteworthy aspect, as it minimizes the need for additional human resources while providing accurate real-time data. By embracing FRAS, organizations can streamline their operations, bolster security measures, and ultimately improve productivity and efficiency across the board.

1.2 Problem Statement

This project introduces an involuntary attendance marking system that uses Artificial Intelligence and Deep Learning (AI/DL), devoid of any kind of interference with the normal teaching procedure. The system can be also implemented during exam sessions or in other teaching activities where attendance is highly essential. This system eliminates classical student identification such as calling name of the student, or checking respective identification cards of the student, which can not only interfere with the ongoing teaching process, but also can be stressful for students during examination sessions. We aim to make attendance tracking more accurate and efficient for organizations. The system will enhance its ability to recognize faces under various conditions.

1.3 Objectives

- To develop a Facial recognition attendance system using AI and DL to detect and recognize multiple faces under various lighting and brightness conditions

- To automatically log attendance data in an excel sheet

CHAPTER 2
LITERATURE SURVEY

2.1 Summary

[1] As one of the most successful applications of image analysis and understanding, face recognition has recently received significant attention, especially during the past several years. At least two reasons account for this trend: the first is the wide range of commercial and law enforcement applications, and the second is the availability of feasible technologies after 30 years of research. Even though current machine recognition systems have reached a certain level of maturity, their success is limited by the conditions imposed by many real applications. For example, recognition of face images acquired in an outdoor environment with changes in illumination and/or pose remains a largely unsolved problem. In other words, current systems are still far away from the capability of the human perception system.

[2] Authentication is one of the significant issues in the era of information system. Among other things, human face recognition (HFR) is one of known techniques which can be used for user authentication. As an important branch of biometric verification, HFR has been widely used in many applications, such as video monitoring/surveillance system, human-computer interaction, door access control system and network security. This paper proposes a method for student attendance system in classroom using face recognition technique by combining Discrete Wavelet Transforms (DWT) and Discrete Cosine Transform (DCT) to extract the features of student’s face which is followed by applying Radial Basis Function (RBF) for classifying the facial objects. From the experiments which is conducted by involving 16 students situated in classroom setting, it results in 121 out of 148 successful faces recognition.

[3] Traditionally, data mining algorithms and machine learning algorithms are engineered to approach the problems in isolation. These algorithms are employed to train the model in separation on a specific feature space and same distribution. Depending on the business case, a model is trained by applying a machine learning algorithm for a specific task.

On the contrary, in real world this assumption may not hold and thus models need to be rebuilt from the scratch if features and distribution changes. It is an arduous process to collect related training data and rebuild the models. In such cases, Transferring of Knowledge or transfer learning from disparate domains would be desirable. Transfer learning is a method of reusing a pre-trained model knowledge for another task. Transfer learning can be used for classification, regression and clustering problems.

[4] Although deep learning has achieved satisfactory performance in computer vision, a large volume of images is required. However, collecting images is often expensive and challenging. Many image augmentation algorithms have been proposed to alleviate this issue. Understanding existing algorithms is, therefore, essential for finding suitable and developing novel methods for a given task. In this study, we perform a comprehensive survey of image augmentation for deep learning using a novel informative taxonomy. To examine the basic objective of image augmentation, we introduce challenges in computer vision tasks and vicinity distribution. The algorithms are then classified among three categories: model-free, model-based, and optimizing policy-based. The model-free category employs the methods from image processing, whereas the model-based approach leverages image generation models to synthesize images.

[5] Machine learning led to the creation of a concept called deep learning which uses algorithms to create an artificial neural network and use it to develop and learn, based on which it makes intuitive decisions by itself. Image classification is a task where we classify the images into sets of different categories, which when performed using deep learning increases business productivity by saving time and manpower. In the paper, it was determined which model of the architecture of the Convoluted Neural Network (CNN) can be used to solve a real-life problem of product classification to help optimize pricing comparison. We have compared the VGG16, VGG19, and ResNet50 architectures based on their accuracy while all three of these models solve the same image classification problem. We have concluded that the VGG-16 is the best architecture based on the comparison. These models have provided accuracies of 0.9867, 0.9707, and 0.9733 for VGG16, VGG19, and ResNet50 at epoch 20.

2.2 Scope for summary

Facial Recognition Automatic Attendance System (FRAS) is a highly accurate attendance recording method. Numerous studies have demonstrated that FRAS can achieve accuracy rates of 98% or higher, making it a more reliable alternative to traditional methods such as manual roll call or card swiping. FRAS is significantly more efficient than traditional attendance recording methods. Studies have shown that FRAS can reduce the time required to record attendance by up to 80%, freeing up administrators to focus on more strategic tasks. FRAS enhances security by preventing unauthorized access. By relying on facial recognition, FRAS can ensure that only authorized individuals are marked as present, preventing impostors. And proxy attendance. FRAS is a versatile technology that can be applied in a variety of settings. FRAS has been successfully implemented in educational institutions, workplaces, and controlled access areas, demonstrating its broad applicability.

FRAS faces challenges in image quality and algorithm bias. The accuracy of FRAS is dependent on the quality of the captured images, and ML algorithms can exhibit bias if trained on datasets that are not representative of the target population. Ongoing research and development are addressing the challenges of FRAS. Researchers are working to improve image quality enhancement techniques and develop fairer ML algorithms to mitigate bias. FRAS has the potential to revolutionize attendance management. Its efficiency, accuracy, and security benefits make it a valuable tool for organizations across various sectors. FRAS is likely to play an increasingly prominent role in the future of attendance management. As research and development efforts continue to improve the technology, FRAS is poised to become the standard for attendance recording.

The implementation of FRAS should be accompanied by appropriate privacy and ethical considerations. Organizations should ensure that the use of FRAS complies with data privacy laws and respects the privacy rights of individual.

CHAPTER 3

IMPLEMENTATION

3.1 Methodology and Approach

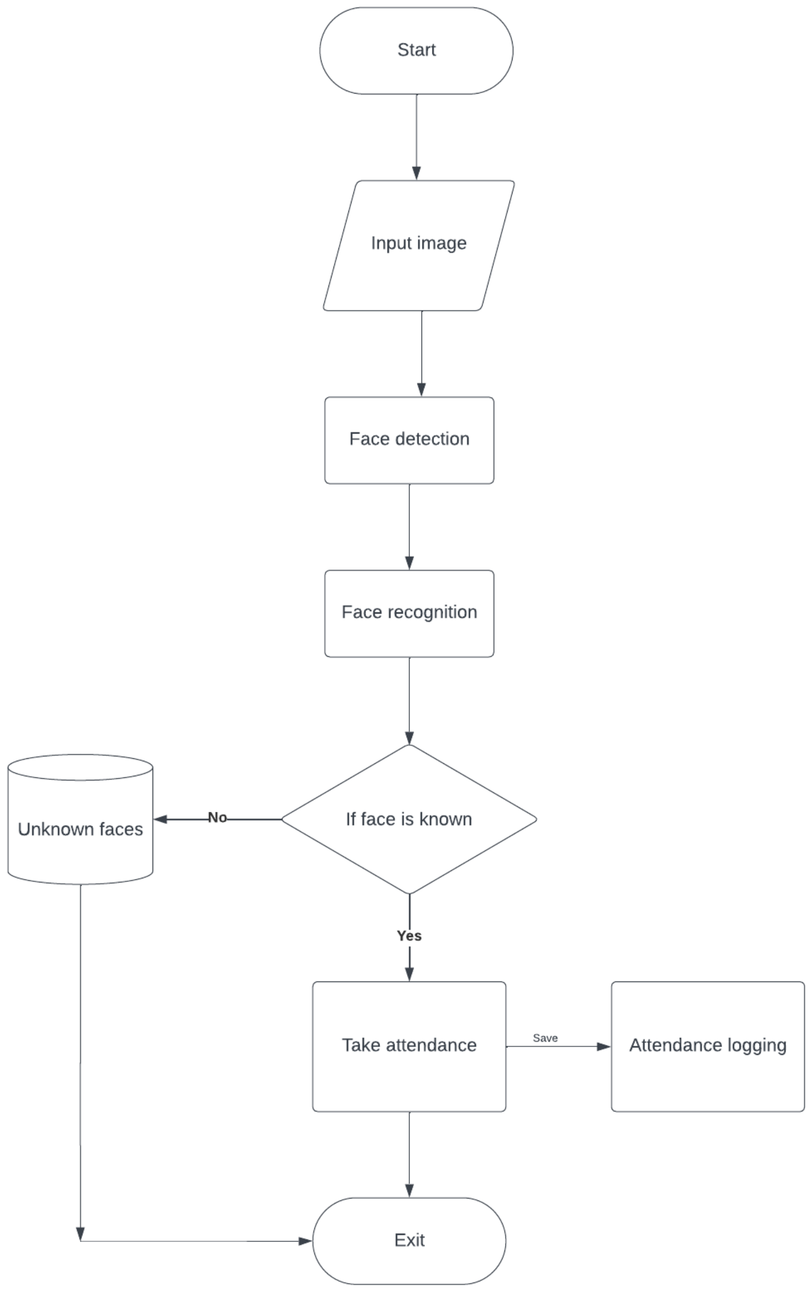


Fig 3.1.1 Flowchart of working of FRAS

• ***Dataset Collection: -*** 100 jpg images were collected where each belonging to one of the five classes, i.e Varun, Vijay, Vinyas, Rishab and Unknown Face. Each class had 20 images

• ***Data Preprocessing: -*** Each image was resized into 224X224 pixels using Open-CV and only the face part was cropped using face-recognition module

• ***Train-test split:*** - 70 % of the images were used for training, 20% of the images were used for validation and 10% of the images were using for testing

• ***Image-Augmentation: -*** Image Data generator was used for generating more samples due to which the training set increased to 96 images and testing set increased to 32 images

• ***Model Training and Testing: -*** VGG16 Model was trained and tested using the augmented samples

• ***Real time capture and attendance logging: -*** Using the webcam, we are able to detect and recognize faces using the trained model and instantly log the recognized face attendance in the excel spreadsheet.

3.2 Dataset

100 Good Quality jpeg images containing various faces were chosen and 20 images each belonging to the classes Varun, Vijay, Vinyas, Rishab and Unknown Face where all the images pertaining to the person (excluding Unknown_Face) were provided by the concerned person himself declaring his permission to use his images for training the VGG-16 model. For Unknown_Face class label, fake images generated from free online AI tool gencraft by giving various prompts. These images were pre-processed cropping the unnecessary parts and noise of the image. Each image was resized into 224x224 size so that it can be given as input to train the model. Out of the 100 images,70 images were given for training, 20 images were given for validation and 10 images were given for testing.

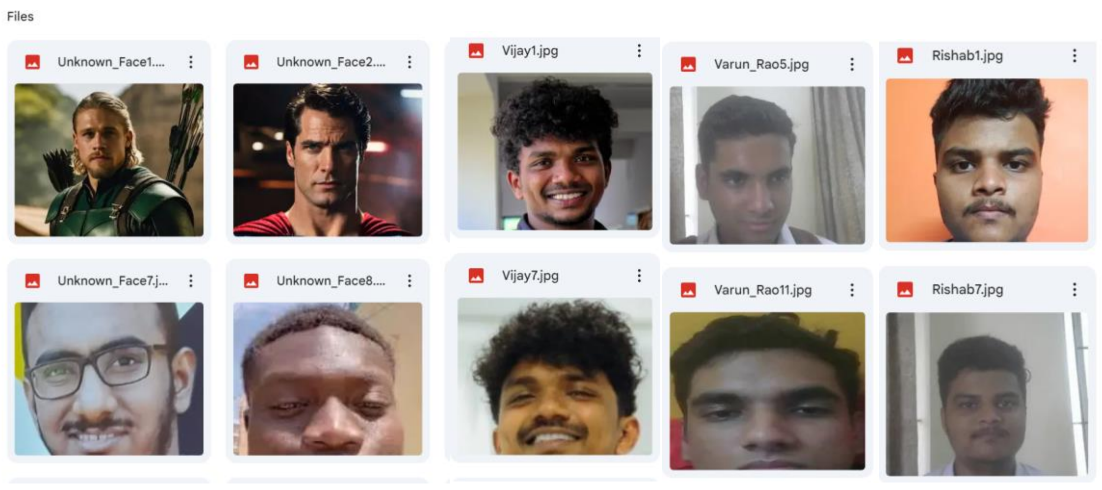


Fig 3.2.1 Snapshot of images used in dataset

3.3 Image Augmentation

Due to the lack of images in the dataset, Image augmentation was performed to train the model well with massive amounts of samples.

ImageDataGenerator is a class in the Keras deep learning library, which is commonly used for image data augmentation. Data augmentation is a technique used to artificially increase the diversity of a dataset by applying various transformations to the existing data samples. This can help improve the generalization and robustness of machine learning models, particularly in tasks like image classification, object detection, and segmentation.

ImageDataGenerator was implemented using the following steps: -

Instantiate the Generator: Create an instance of ImageDataGenerator. This object will be responsible for generating augmented images.

Configure Augmentation Parameters: Specify various augmentation parameters such as rotation range, width shift range, height shift range, shear range, zoom range, horizontal and vertical flipping, etc. These parameters determine the types and degrees of transformations that will be applied to the images.

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

Generate Augmented Images: Use the flow_from_directory, flow, or flow_from_dataframe method of the ImageDataGenerator object to generate batches of augmented images from a directory or dataframe containing the original images. The generator will automatically apply the specified augmentation transformations to the images in real-time as they are loaded.
target_size=(224,224),batch size=32,class_mode=categorical

Feed Augmented Images to Model: Feed the augmented images generated by the ImageDataGenerator to your deep learning model during training. The model learns from both the original and augmented images, thereby improving its ability to generalize to new, unseen data.

After augmenting the dataset, the size increased from 100 images to 3300 images.
70 images considered for training increased to 2549 images and 20 images considered for validation increased to 777 images.

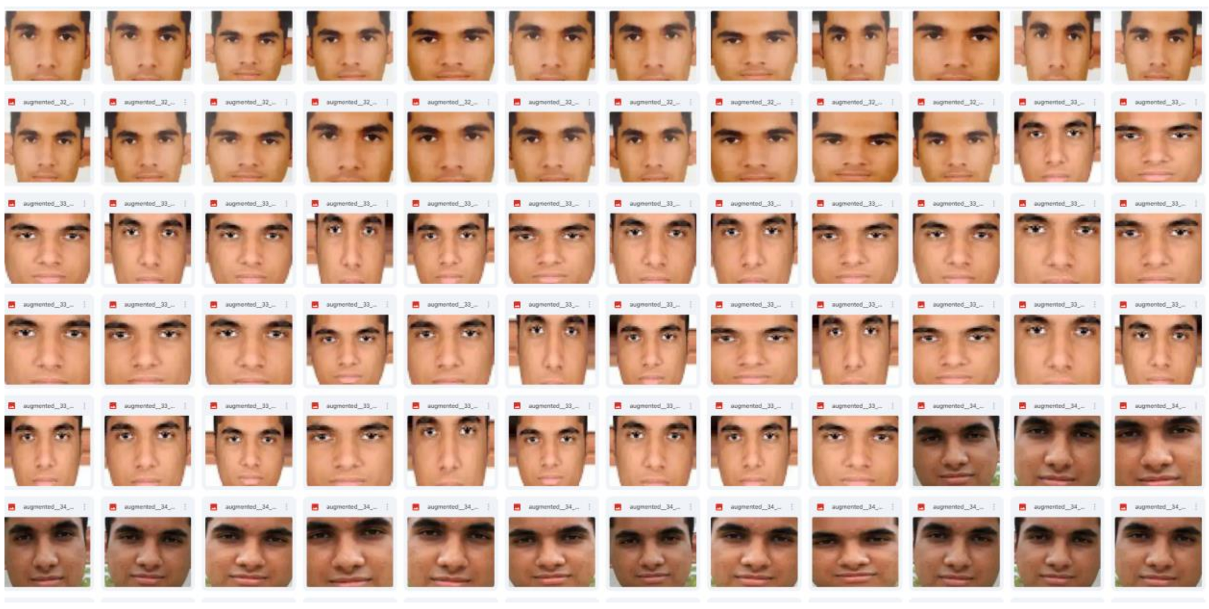


Fig 3.3.1 Snapshot of Augmented images

3.4 Model Training

VGG-16

VGGFace refers to a series of models developed for face recognition. It was developed by the Visual Geometry Group (hence its VGG name) at the University of Oxford. The models were trained on a dataset comprised mainly of celebrities, public figures, actors, and politicians. Their names were extracted from the Internet Movie Data Base (IMDB) celebrity list based on their gender, popularity, pose, illumination, ethnicity, and profession (actors, athletes, politicians). The images of these names were fetched from Google Image Search, and multiple images for each name were downloaded, vetted by humans, and then labelled for training.

There are two versions of VGGFace:

VGGFace: Developed in 2015, trained on 2.6 million images, a total of 2622 people

VGGFace2: Developed in 2017, trained on 3.31 million images, a total of 9131 people

The original VGGFace uses the VGG16 model, which is a convolutional neural network with 16 layers (see Figure 6).

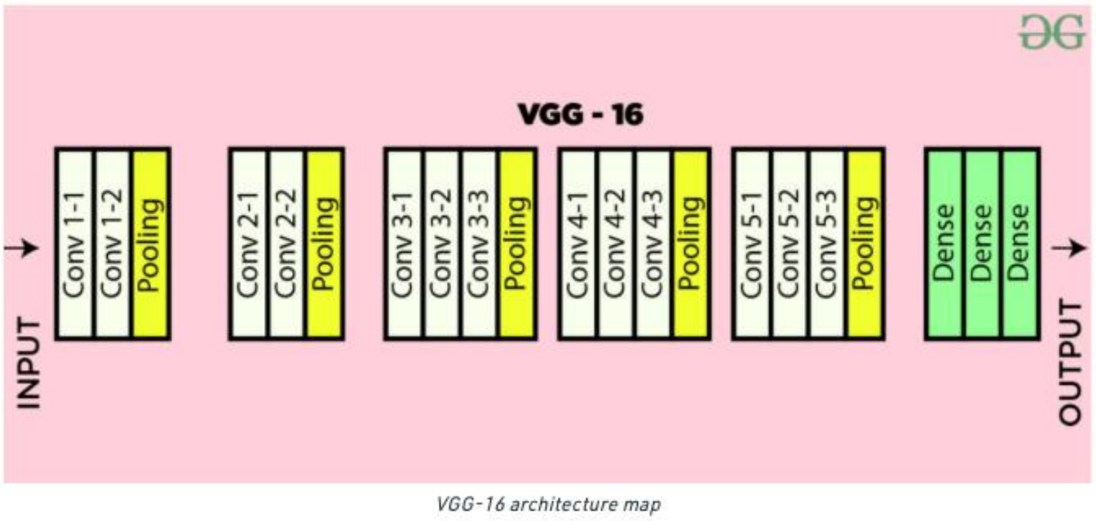


Fig 3.4.1 VGG-16 architecture

Procedure to use VGG-16

Install the libraries: -

```
!pip install keras_vggface
!pip install keras_applications
```

Load the VGG-16 model: -

```
from keras_vggface.vggface import VGGFace

base_model = VGGFace(include_top=True,
    model='vgg16',
    input_shape=(224, 224, 3))
base_model.summary()
print(len(base_model.layers))
# 26 layers in the original VGG-Face
base_model = VGGFace(include_top=False,
model='vgg16',
input_shape=(224, 224, 3))
base_model.summary()
print(len(base_model.layers))
```

Add the custom layers: -

```
x = base_model.output
x = GlobalAveragePooling2D()(x)

x = Dense(1024, activation='relu')(x)
x = Dense(1024, activation='relu')(x)
x = Dense(512, activation='relu')(x)

# final layer with softmax activation
preds = Dense(NO_CLASSES, activation='softmax')(x)
```

Set the first 19 layers to be non-trainable and the rest of the layers to be trainable:-

```
# create a new model with the base model's original input and
the new model's output
model = Model(inputs = base_model.input, outputs = preds)
model.summary()

# don't train the first 19 layers - 0..18
for layer in model.layers[:19]:
    layer.trainable = False

# train the rest of the layers - 19 onwards
for layer in model.layers[19:]:
    layer.trainable = True
```

Compile and train the model:-

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

history=model.fit(training_generator,batch_size = 1,verbose =
1,epochs = 15,validation_data=validation_generator)
```

Create the hdf5 file:-

```
model.save('transfer_learning_trained' +'_face_cnn_model.h5')
```

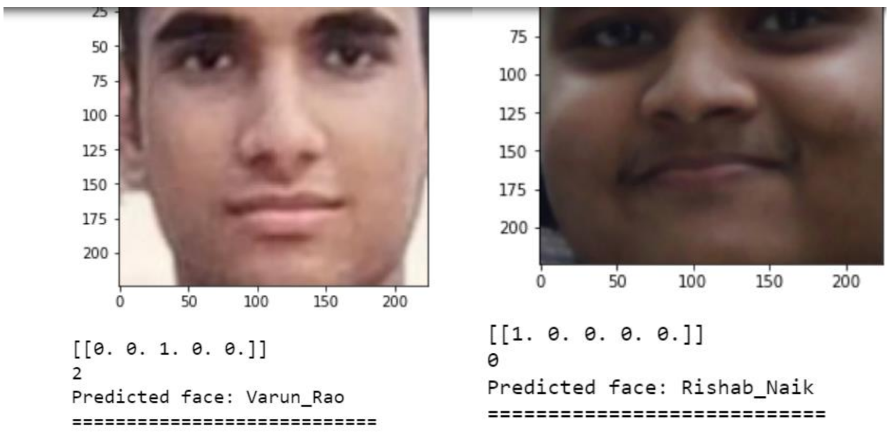
3.5 Testing

Testing the VGG-16 model is crucial to assess its performance and suitability for specific tasks or datasets. By testing the model, we can evaluate its accuracy, precision, recall, and other relevant metrics to understand how well it generalizes to new, unseen data. Additionally, testing helps identify any potential issues such as overfitting or underfitting, allowing us to fine-tune the model or adjust training strategies accordingly. Understanding the model's performance through testing is essential for making informed decisions about its deployment in real-world applications and ensuring its effectiveness in solving the intended problem.

Code to test the VGG-16 model

```
#Testing
from keras.applications.imagenet_utils import preprocess_input
from tensorflow.keras.models import load_model

# Define the image dimensions
image_width, image_height = 224, 224
y_test = []
pred12 = []

# Path to the testing folder
testing_folder = "Processed_Headshots_Testing/"

# Load the trained model outside the loop
model                                                              =
load_model('transfer_learning_trained_face_cnn_model.h5')

# Function to get label index from folder name
def get_label_index(folder_name):
    # Convert folder name to lowercase and remove spaces
    label = folder_name.replace(" ", ".").lower()
    # Check if label exists in label_ids dictionary
    if label in label_ids:
        return label_ids[label]
    else:
        return None

# Iterate over the testing folder and its subfolders
for root, dirs, files in os.walk(testing_folder):
    for file in files:
        if file.endswith(('.png', '.jpg', '.jpeg')):
            # Load image path
            test_image_filename = os.path.join(root, file)

            # Load the image using face_recognition
            imgtest                                          =
face_recognition.load_image_file(test_image_filename)
            image_array = np.array(imgtest, "uint8")

            # Get face locations in the image
            face_locations                                   =
face_recognition.face_locations(imgtest)

            for face_location in face_locations:
                # Extract face coordinates
                top, right, bottom, left = face_location

                # Resize the detected face to 224x224
                roi = image_array[top:bottom, left:right]
                resized_image  =  cv2.resize(roi,  (image_width,
image_height))

                # Prepare the image for prediction
                x = image.img_to_array(resized_image)
                x = np.expand_dims(x, axis=0)
                x = preprocess_input(x)
```

```
          # Prepare the image for prediction
                x = image.img_to_array(resized_image)
                x = np.expand_dims(x, axis=0)
                x = preprocess_input(x)

                # Making prediction
                # Load the trained model
                model                                              =
load_model('transfer_learning_trained_face_cnn_model.h5')
                predicted_prob = model.predict(x)
                plt.imshow(resized_image)
                plt.show()
                print(predicted_prob)
                print(predicted_prob[0].argmax())
                print("Predicted        face:        "         +
class_list[predicted_prob[0].argmax()])
                print("============================\n")

                 # Get label index from folder name
                label_index                                        =
get_label_index(os.path.basename(root))

                # Store ground truth label index
                y_test.append(label_index)

                # Store predicted value
                pred12.append(predicted_prob[0].argmax())

# Print results
print("Ground Truth Labels:", y_test)
print("Predicted Labels:", pred12)
```



```
[[0. 0. 1. 0. 0.]]
2
Predicted face: Varun_Rao
============================
```

```
[[1. 0. 0. 0. 0.]]
0
Predicted face: Rishab_Naik
============================
```

Fig 3.5.1 Snapshot of testing results

Most of the images were predicted correctly by the VGG-16 model. The code was designed to test a pre-trained VGG-16 model for face recognition. It begins by importing essential libraries such as Keras, OpenCV, NumPy, and Matplotlib. These libraries are crucial for image processing, deep learning operations, and visualization. Additionally, the script sets parameters for the dimensions of the input images, which are expected by the pre-trained VGG-16 model. The core functionality of the script lies in the testing loop, where it iterates through each file in the testing folder. For each image file, it utilizes the face_recognition library to load the image and detect face locations within it. It then crops and resizes the detected face regions to match the input dimensions required by the VGG-16 model. These resized face images are preprocessed using the preprocess_input function to ensure compatibility with the model's requirements**.**After preprocessing, the script loads the pre-trained VGG-16 model from the specified file (transfer_learning_trained_face_cnn_model.h5). For each face image, the model predicts the class probabilities using the predict method. The predicted class with the highest probability is extracted, and its corresponding label is printed. This process repeats for each face detected in the image. Furthermore, the script contains a function to map folder names to label indices, which is used to obtain ground truth labels from the folder structure of the testing dataset. These ground truth labels, along with the predicted labels, are stored in separate lists (y_test and pred12, respectively) for further evaluation.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Webcam Capture

Code for webcam capture and attendance logging

```
# Function to get USN from the student database CSV file based
on the predicted class label
def get_usn_from_label(label):
    with open('Student_Database.csv', mode='r') as file:
        reader = csv.DictReader(file)
        for row in reader:
            if 'NAME' in row and row['NAME'] == label:
                return row.get('USN', None)
    return None
from tensorflow.keras.models import load_model
# Load the trained model
model                                                              =
load_model('transfer_learning_trained_face_cnn_model.h5')

# Load the labels
with open("face-labels.pickle", 'rb') as f:
    og_labels = pickle.load(f)
    # Decode byte strings using UTF-8 and remove 'b' prefix
    labels  =  {key:  value.lstrip('b')  for  key,  value  in
og_labels.items()}
    print(labels)
# Webcam settings
screen_width = 1280
screen_height = 720
image_width = 224
image_height = 224

# Open default webcam
stream = cv2.VideoCapture(0)
```

```
# Set to store processed labels
processed_labels = set()

# Open Attendance CSV file in append mode
with open('Attendance1.csv', mode='a', newline='') as file:
    writer = csv.writer(file)

    while True:
        # Capture frame-by-frame
        grabbed, frame = stream.read()
        rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        # Find all face locations in the frame
        face_locations = face_recognition.face_locations(rgb)

        for (top, right, bottom, left) in face_locations:
            # Draw a rectangle around the face
            color = (255, 0, 0)  # BGR
            stroke = 2
            cv2.rectangle(frame, (left, top), (right, bottom),
color, stroke)

            # Resize and preprocess the image
            roi_rgb = rgb[top:bottom, left:right]
            resized_image = cv2.resize(roi_rgb, (image_width,
image_height))
            img = np.expand_dims(resized_image, axis=0)  # Add
batch dimension
            img = img.astype('float32') / 255.0

            # Predict the image
            predicted_prob = model.predict(img)
```

```
 # Display the label
           font = cv2.FONT_HERSHEY_SIMPLEX
           name = labels[predicted_prob[0].argmax()]
           color = (255, 0, 255)
           stroke = 2
           cv2.putText(frame, f'({name})', (left, top - 8),
                       font, 1, color, stroke, cv2.LINE_AA)

           # Get USN from student database
           usn = get_usn_from_label(name)
           if usn is not None and name not in processed_labels:
               # Get current time and date
               current_time                                  =
datetime.datetime.now().strftime('%H:%M:%S')
               current_date                                  =
datetime.datetime.now().strftime('%Y-%m-%d')

             # Write data to Attendance CSV file
               writer.writerow([name,    usn,    current_time,
current_date])
               # Add the label to the set of processed labels
               processed_labels.add(name)

       # Display the frame
       cv2.imshow("Image", frame)
# Check for the 'q' key to break out of the loop
       key = cv2.waitKey(1) & 0xFF
       if key == ord("q"):
           break

# Release the stream and close all OpenCV windows
stream.release()
cv2.destroyAllWindows()
```

Code explanation

This code is designed to perform real-time face recognition using a pre-trained convolutional neural network (CNN) model and OpenCV. Let's break down the main components and their functionalities:

Loading the Model and Labels: The code begins by loading a pre-trained CNN model (transfer_learning_trained_face_cnn_model.h5) using Keras. It also loads the labels associated with the trained model from a pickle file (face-labels.pickle). These labels are used to map predicted probabilities to actual class names.

Webcam Setup: The code sets up the webcam for capturing live video frames. It defines parameters such as screen width, screen height, image width, and image height.

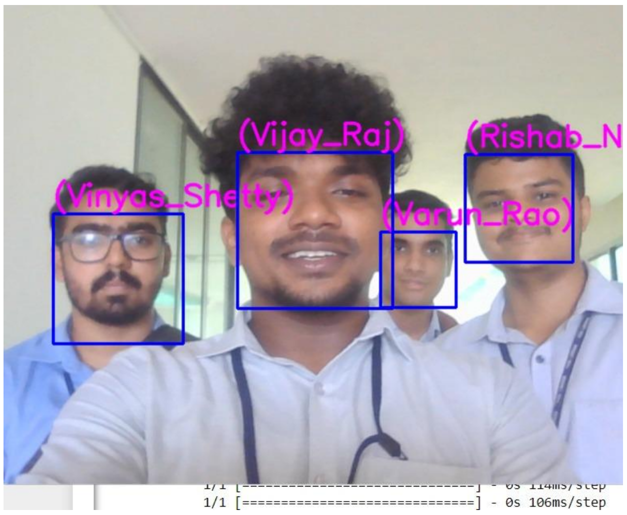
Processing Faces: Within the main loop, the code continuously captures frames from the webcam and converts them to RGB format. It then uses the face_recognition library to detect faces within the frame by identifying their locations.

Prediction and Display: For each detected face, the code resizes and preprocesses the face image to match the input dimensions of the CNN model. It then passes the preprocessed image through the model to obtain a prediction, i.e., the probability distribution over the known labels. The predicted label is then mapped to a class name using the loaded label dictionary. This name is overlayed onto the frame using OpenCV's putText function.

Attendance Tracking: The code retrieves the unique student identifier (USN) corresponding to the predicted label using the get_usn_from_label function. If the USN is found and the label has not been processed before, the current time and date are recorded, and the data (name, USN, time, and date) are appended to an attendance CSV file (Attendance1.csv). This ensures that each student is marked present only once per session.

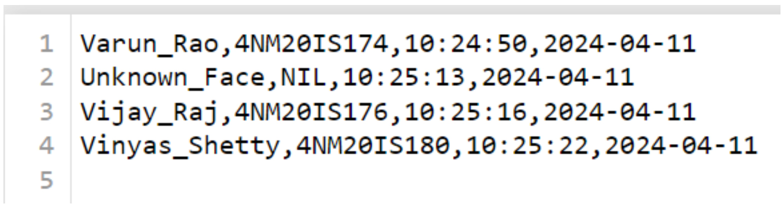
Loop Termination: The loop continues until the user presses the 'q' key, at which point the program breaks out of the loop, releasing the webcam stream and closing all OpenCV windows.

4.2 Attendance Logging



```
1/1 [==============================] - 0s 114ms/step
1/1 [==============================] - 0s 106ms/step
```

Fig 4.2.1 Face detection and recognition from webcam



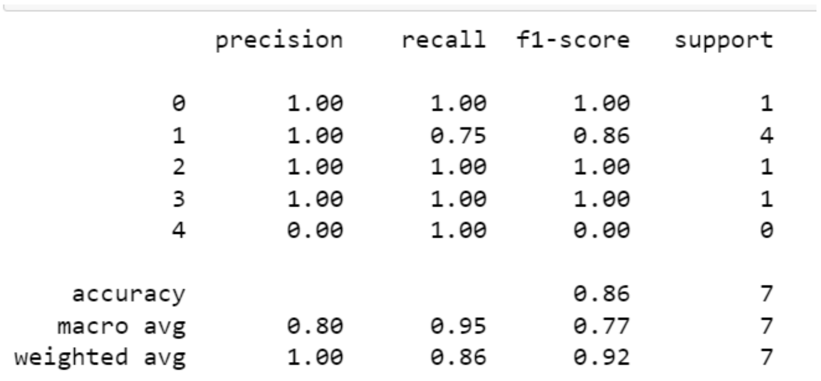
```
Varun_Rao,4NM20IS174,10:24:50,2024-04-11
Unknown_Face,NIL,10:25:13,2024-04-11
Vijay_Raj,4NM20IS176,10:25:16,2024-04-11
Vinyas_Shetty,4NM20IS180,10:25:22,2024-04-11
```

Fig 4.2.2 Attendance logging in csv file

After capturing frames from a webcam, detecting faces within each frame, and overlaying the predicted labels onto the faces, these labels are then matched with unique student identifiers (USNs) obtained from a CSV student database. Upon successful recognition, the system logs the student's name, USN, current time, and date into a CSV attendance file. By efficiently processing each frame and accurately identifying faces, this code enables seamless attendance tracking, facilitating automated and reliable monitoring of student presence without manual intervention.

4.3 Performance Metrics

The classification report was generated after testing, using the y_test and pred12 arrays.

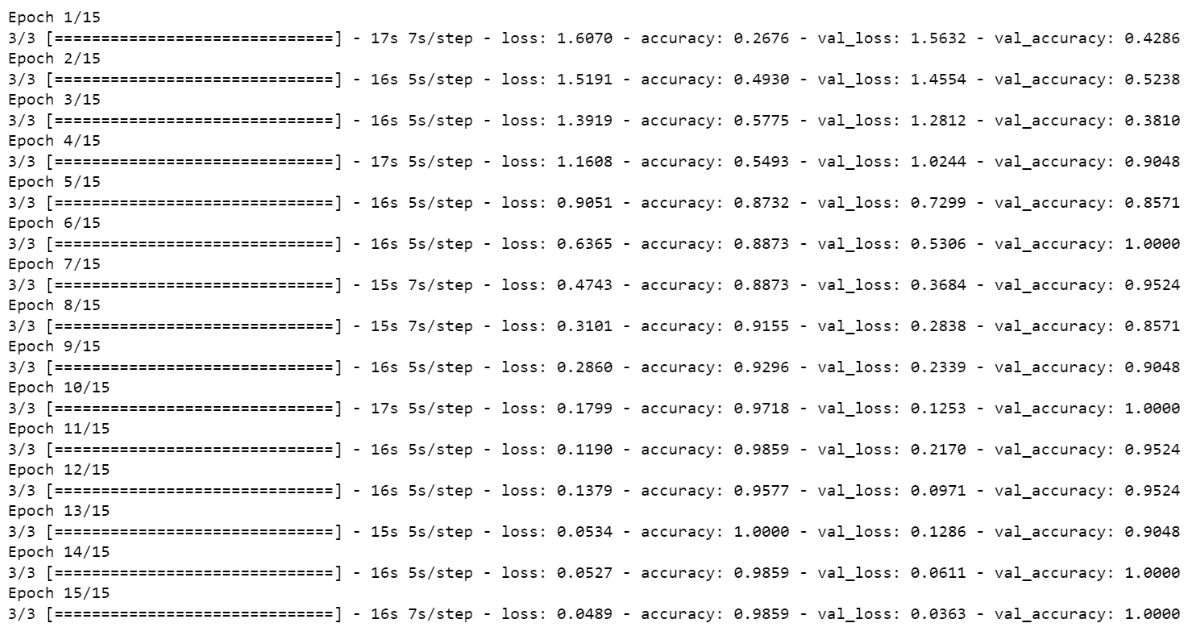


```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      0.75      0.86         4
           2       1.00      1.00      1.00         1
           3       1.00      1.00      1.00         1
           4       0.00      1.00      0.00         0

    accuracy                           0.86         7
   macro avg       0.80      0.95      0.77         7
weighted avg       1.00      0.86      0.92         7
```

Fig 4.3.1 Classification Report

Training Accuracy	Validation Accuracy	Testing Accuracy	Testing error rate
98.59 %	100 %	86 %	14 %

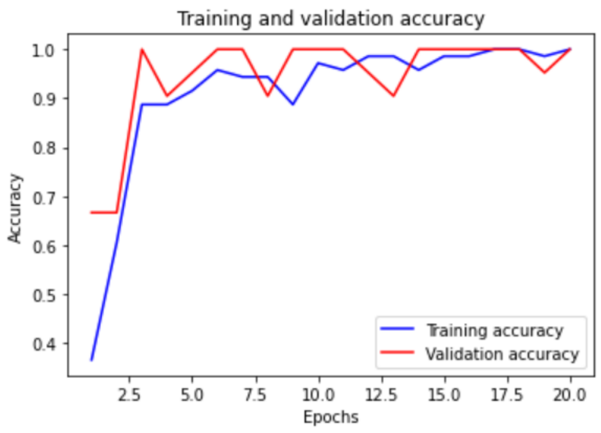
Fig 4.3.2 Performance metrics score



```
Epoch 1/15
3/3 [==============================] - 17s 7s/step - loss: 1.6070 - accuracy: 0.2676 - val_loss: 1.5632 - val_accuracy: 0.4286
Epoch 2/15
3/3 [==============================] - 16s 5s/step - loss: 1.5191 - accuracy: 0.4930 - val_loss: 1.4554 - val_accuracy: 0.5238
Epoch 3/15
3/3 [==============================] - 16s 5s/step - loss: 1.3919 - accuracy: 0.5775 - val_loss: 1.2812 - val_accuracy: 0.3810
Epoch 4/15
3/3 [==============================] - 17s 5s/step - loss: 1.1608 - accuracy: 0.5493 - val_loss: 1.0244 - val_accuracy: 0.9048
Epoch 5/15
3/3 [==============================] - 16s 5s/step - loss: 0.9051 - accuracy: 0.8732 - val_loss: 0.7299 - val_accuracy: 0.8571
Epoch 6/15
3/3 [==============================] - 16s 5s/step - loss: 0.6365 - accuracy: 0.8873 - val_loss: 0.5306 - val_accuracy: 1.0000
Epoch 7/15
3/3 [==============================] - 15s 7s/step - loss: 0.4743 - accuracy: 0.8873 - val_loss: 0.3684 - val_accuracy: 0.9524
Epoch 8/15
3/3 [==============================] - 15s 7s/step - loss: 0.3101 - accuracy: 0.9155 - val_loss: 0.2838 - val_accuracy: 0.8571
Epoch 9/15
3/3 [==============================] - 16s 5s/step - loss: 0.2860 - accuracy: 0.9296 - val_loss: 0.2339 - val_accuracy: 0.9048
Epoch 10/15
3/3 [==============================] - 17s 5s/step - loss: 0.1799 - accuracy: 0.9718 - val_loss: 0.1253 - val_accuracy: 1.0000
Epoch 11/15
3/3 [==============================] - 16s 5s/step - loss: 0.1190 - accuracy: 0.9859 - val_loss: 0.2170 - val_accuracy: 0.9524
Epoch 12/15
3/3 [==============================] - 16s 5s/step - loss: 0.1379 - accuracy: 0.9577 - val_loss: 0.0971 - val_accuracy: 0.9524
Epoch 13/15
3/3 [==============================] - 15s 5s/step - loss: 0.0534 - accuracy: 1.0000 - val_loss: 0.1286 - val_accuracy: 0.9048
Epoch 14/15
3/3 [==============================] - 16s 5s/step - loss: 0.0527 - accuracy: 0.9859 - val_loss: 0.0611 - val_accuracy: 1.0000
Epoch 15/15
3/3 [==============================] - 16s 7s/step - loss: 0.0489 - accuracy: 0.9859 - val_loss: 0.0363 - val_accuracy: 1.0000
```

Fig 4.3.3 Snapshot of Training and Testing Accuracy

The training accuracy starts relatively low in the first epoch and gradually improves over subsequent epochs, reaching a higher value by the end of the training process. Similarly, the validation accuracy initially fluctuates but generally follows an upward trend, indicating that the model's performance on unseen data improves as training progresses. The convergence of training and validation accuracies suggests that the model learns effectively from the training data and generalizes well to new data, without overfitting.



4.3.4 Graph of training and validation accuracy

The model's performance metrics demonstrate consistent improvement and effective learning throughout the training process. Starting with relatively low accuracy in the initial epochs, both training and validation accuracies steadily increase over time, reflecting the model's ability to learn from the training data and generalize well to unseen data. The convergence of training and validation accuracies indicates that the model learns effectively without overfitting, achieving high levels of accuracy on both datasets. Additionally, the low loss values across epochs suggest that the model's predictions closely match the actual target values. Overall, the training process results in a well-performing model capable of accurately predicting target outcomes on both training and validation datasets.

CHAPTER 5

CONCLUSION

5.1 GENERAL CONCLUSION

The implementation of a face recognition attendance system presents a significant advancement in automating and streamlining the attendance tracking process. By leveraging advanced technologies such as convolutional neural networks (CNNs) and real-time image processing, this system offers several key benefits. Firstly, it provides a non-intrusive and efficient means of attendance capture, eliminating the need for manual roll calls or biometric scanning. This not only saves time but also reduces administrative burden and human error. Additionally, the system enhances security by accurately verifying the identity of individuals based on facial features, mitigating the risk of unauthorized access or proxy attendance.

Moreover, the face recognition attendance system promotes inclusivity and accessibility by accommodating a wide range of individuals, regardless of physical abilities or personal characteristics. Unlike traditional methods that may rely on physical tokens or identifiers, facial recognition technology inherently adapts to diverse populations, making it suitable for educational institutions, workplaces, and public venues alike. Furthermore, the system offers scalability and flexibility, allowing for seamless integration with existing infrastructure and adaptable deployment across various environments.

From a practical standpoint, the implementation of such a system requires careful consideration of privacy and ethical considerations. Robust data protection measures must be in place to safeguard individuals' biometric data and ensure compliance with relevant regulations such as GDPR. Additionally, transparency and user consent are essential elements to foster trust and acceptance of the technology among stakeholders. By addressing these concerns and prioritizing ethical use, the face recognition attendance system can serve as a valuable tool for enhancing efficiency, security, and accountability in attendance management processes.

5.2 Scope for future work

In the realm of face recognition technology, the potential for future advancement is substantial. One promising avenue for improvement involves integrating a feature that enables the training of the model with live pictures of new individuals. By leveraging real-time image capture and incorporating these images into the training dataset, the system can continuously expand its recognition capabilities to encompass a broader spectrum of faces. This iterative learning process not only enhances the system's inclusivity but also ensures its adaptability to evolving scenarios and demographics. Furthermore, fine-tuning the hyperparameters of the model represents another crucial aspect of future work. Adjusting parameters such as learning rate, batch size, and optimization algorithms can significantly enhance the model's accuracy and robustness, enabling it to perform effectively across various lighting and brightness conditions. By pursuing these avenues of research and development, the face recognition system can achieve greater accuracy, reliability, and versatility, thus advancing its potential applications in diverse real-world scenarios.

In addition to expanding the model's recognition capabilities and fine-tuning its hyperparameters, future work could also focus on enhancing the system's robustness to environmental factors. This could involve developing algorithms or preprocessing techniques to mitigate the impact of varying lighting conditions, shadows, and facial occlusions. By improving the model's ability to accurately detect and recognize faces under challenging environmental circumstances, the system can achieve greater reliability and performance consistency in real-world deployment scenarios. This holistic approach to system refinement will not only bolster its effectiveness but also pave the way for broader adoption and integration into everyday applications, ranging from security and surveillance to personalization and access control.

REFERENCES

[1]W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld. 2003. Face recognition: A literature survey. ACM Comput. Surv. 35, 4 (December 2003), 399–458. https://doi.org/10.1145/954339.954342

[2]S. Lukas, A. R. Mitra, R. I. Desanti and D. Krisnadi, "Student attendance system in classroom using face recognition technique," 2016 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, Korea (South), 2016, pp. 1032-1035, doi: 10.1109/ICTC.2016.7763360.

[3]Tammina, Srikanth. "Transfer learning using vgg-16 with deep convolutional neural network for classifying images." International Journal of Scientific and Research Publications (IJSRP) 9.10 (2019): 143-150.

[4]Xu, Mingle, et al. "A comprehensive survey of image augmentation techniques for deep learning." Pattern Recognition 137 (2023): 109347.

[5]S. Mascarenhas and M. Agarwal, "A comparison between VGG16, VGG19 and ResNet50 architecture frameworks for Image Classification," 2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON), Bengaluru, India, 2021, pp. 96-99, doi: 10.1109/CENTCON52345.2021.9687944.