FACE RECOGNITION USING OPEN CV

A project report submitted to the department of
Information Technology in partial fulfillment of the requirements
for the Degree of Bachelor of Technology

By

G.Durga Prasad (20A21A1224)

M.Kowsthabi (20A21A1240)

Under the guidance of

Dr. RVVSV Prasad, Professor & HOD, IT Department

DEPARTMENT OF INFORMATION TECHNOLOGY

SWARNANDHRA COLLEGE OF ENGINEERING & TECHNOLOGY

(Autonomous)

(Approved by A.I.C.T.E & Affiliated to JNTU Kakinada)

(Accredited by NAAC with A Grade in 2nd Cycle)

Seetharamapuram, Narsapur-534 280

April 2024

CERTIFICATE

This is to certify that the project entitled **"Face Recognition Using Open cv"** submitted by **G.Durga Prasad** (Regd.No. 20A21A1224) **M.Kowsthabi** (Regd.No. 20A21A1240) in the Department of **Information Technology, Swarnandhra College of Engineering & Technology** for the award of the degree of **Bachelor of Technology** in **Information Technology** is a bonafide project work carried out under our supervision.

Project Supervisor

Dr. RVVSV Prasad

Professor &HOD, IT Dept

Head of the Department

Dr. RVVSV Prasad

Professor & HOD, IT Dept .

External Examiner

DECLARATION

I certify that

a. The project work contained in the report is original and has been done by me under the guidance of my supervisor.
b. The work has not been submitted to any other University for the award of any degree or diploma.
c. The guidelines of the college are followed in writing the project report.

Date:

G.Durga Prasad (20A21A1224)
M.Kowsthabi (20A21A1240)

ACKNOWLEDGEMENT

I whole heartedly and sincerely thank my guide **Dr. RVVSV Prasad** Professor & Head of the Department of Information Technology, Swarnandhra College of Engineering & Technology, Seetharampuram for their valuable suggestions and encouragement during the preparation and progress of my project.

I express my heartfelt thanks to **Dr. S. Suresh Kumar,** Principal, Swarnandhra College of Engineering & Technology for giving me this opportunity for the successful completion of my degree.

I express my honest thanks to **Management** of Swarnandhra college of Engineering & Technology for providing necessary arrangements for completing the project.

I express my earnest thanks to all the **teaching and non-teaching staff** of department of Information Technology for their valuable guidance and support given for the completion of my project.

Finally, it is pleasure to thank all my **family members** and **friends** for their constant encouragement and enormous support. Without them I could not go up with my work.

G.Durga prasad (20A21A1224)

M.Kowsthabi (20A21A1240)

ABSTRACT

Face recognition using computer vision has emerged as a prominent area of research and practical application due to its wide-ranging implications across numerous domains. This technology enables the automated identification and verification of individuals based on facial characteristics extracted from images or video data.

The utilization of computer vision techniques, particularly with libraries like OpenCV, has facilitated the development of robust and efficient face recognition systems.

The system uses a combination of techniques in two topics; face detection and recognition. The face detection is performed on live acquired images withoutany application field in mind. Processes utilized in the system are white balance correction, skin like region segmentation, facial feature extraction and face image extraction on a face candidate. Then a face classification method that usesConvolutional Neural Network is integrated in the system

Additionally, we discuss the challenges associated with face recognition, including variations in lighting, pose, expression, and occlusion, and examine strategies to address these challenges.

CONTENTS

1. INTRODUCTION

Face recognition is a technology that involves identifying or verifying a person's identity by analyzing their face. It has gained significant popularity due to its wide range of applications, including security systems, biometric authentication, and even photoorganization.

OpenCV (Open Source Computer Vision Library) is a powerful open-source library for computer vision, image processing, and machine learning. It provides various tools and functions that make it easier to workwith images and videos, including those needed for face recognition.

In a face reco``gnition project using OpenCV, the first step usually involves detecting faces in an image or video stream. OpenCV provides pre-trained models like Haar cascades or deep learning-based models thatcan be used for this purpose.

Once the faces are detected, the next step is to extract features from the faces. This is often done using techniques like Eigenfaces, Fisherfaces, or deep learning- based approaches like Convolutional NeuralNetworks (CNNs).

Finally, the extracted features are compared with a database of known faces to recognize or verify the identity of the person. This comparison can be done using various algorithms, such as the Euclidean distanceor cosine similarity.

2. OBJECTIVE

Face recognition, also known as Face classification, is a computer vision technology that allows machines toidentify and categorize objects within digital images or videos.

The technology uses artificial intelligence and machine learning algorithms to learn patterns and features inimages to identify them accurately.

The aim is to enable machines to interpret visual data like humans do, by identifying and categorizingobjects within images.

This technology has a wide range of applications across various industries, including manufacturing,healthcare, retail, agriculture, and security.

Image recognition can be used to improve quality control in manufacturing, detect and diagnose medicalconditions, enhance the customer experience in retail, optimize crop yields in agriculture, and aid in surveillance and security measures.

3. EXISTING SYSTEM BEFORE USING FACE RECOGNITION

Before implementing face recognition using OpenCV with Python, the existing system might have relied on traditional methods for identity verification, such as manual verification by a human or using simple methods like username/password authentication. Here's a brief overview of the existing system:

1 Manual Verification:

- This method involves human verification, where users present identification documents (e.g., IDcards, passports) to verify their identity.
- Personnel, such as security guards or receptionists, manually verify the documents and grant access ifthe verification is successful.
- Manual verification can be time-consuming and may not be suitable for high-traffic areas orlocations requiring rapid access.

2 Username/Password Authentication:

i. Users log in to the system using a combination of a username and password.
ii. The system compares the entered credentials with stored credentials in a database. If theymatch,access is granted.
iii. Passwords can be stolen, guessed, or shared, leading to security vulnerabilities.
iv. Access Cards/Keycards:
v. Users are provided with physical access cards or keycards that contain unique identifiers.
vi. To gain access, users must present their card to a card reader, which verifies the card'sinformation.
vii. While more secure than passwords, access cards can be lost, stolen, or shared,compromising security.

3 Biometric Identification:

- Some advanced systems may already use biometric identification methods, such as fingerprintscanning or iris recognition.
- Biometric data is unique to each individual and offers a high level of security.
- However, implementing biometric systems can be costly and may require specialized hardware.

Figure:

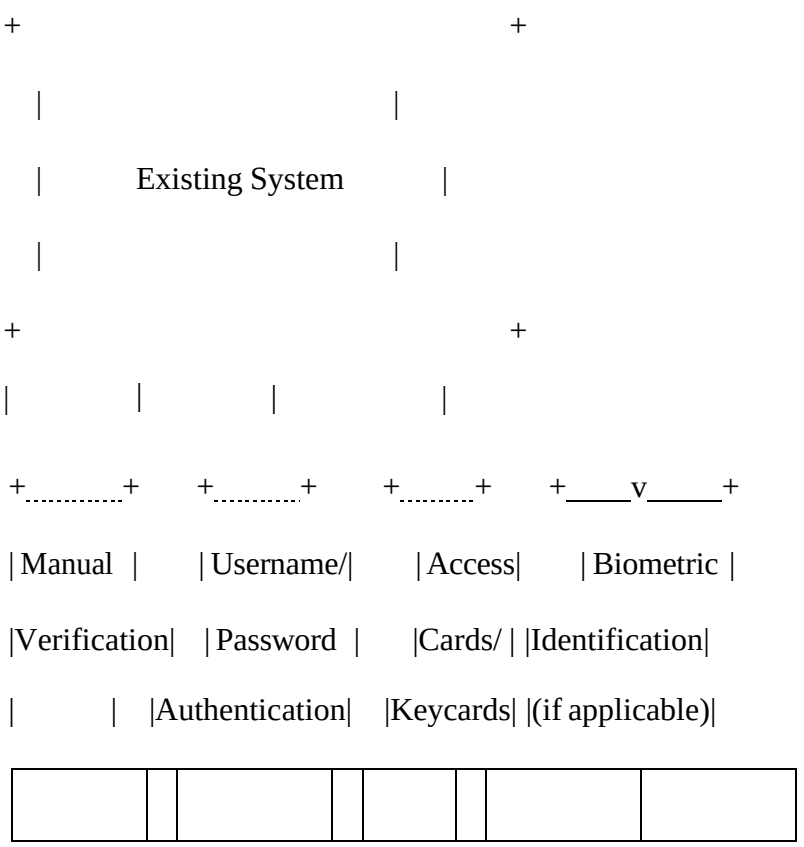

Fig.4.3.1.Existing System Before Face Recognition

4. REQUIRED TECHNOLOGIES

OpenCV: OpenCV (Open Source Computer Vision Library) is the primary library for computer vision tasksin Python. It provides various functions and tools for image processing, including face recognition.

Python: Python is the programming language used to write the face recognition code. OpenCV providesPython bindings, making it easy to use OpenCV in Python programs.

NumPy: NumPy is a fundamental package for scientific computing with Python. It is used for numerical operations and array manipulation, which are common tasks in image processing and machine learning. Matplotlib (Optional): Matplotlib is a plotting library for Python. It is useful for visualizing images, which can be helpful when debugging or analyzing the results of face recognition algorithms.

Face Recognition Models or Algorithms: You will need to choose a face recognition model or algorithm to perform the actual recognition task. OpenCV provides some pre-trained models, or you can train your own model using machine learning algorithms like CNNs (Convolutional Neural Networks).

Image Dataset: A dataset of images containing faces is necessary for training your face recognition model, ifyou are training your own model.

Development Environment: You will need a development environment set up with Python, OpenCV, NumPy, and any other required libraries installed. You can use an IDE like PyCharm or Jupyter Notebookfordevelopment.

5. What Is Computer Vision?

At its core, computer vision is the ability of computers to understand and analyze visual content in the same way humans do. This includes tasks such as recognizing objects and faces, reading text and understanding the context of an image or video.

Computer vision is closely related to artificial intelligence (AI) and often uses AI techniques such as machine learning to analyze and understand visual data. Machine learning algorithms are used to "train" a computer to recognize patterns and features in visual data, such as edges, shapes and colors.

Once trained, the computer can use this knowledge to identify and classify objects in new images and videos. The accuracy of these classifications can be improved over time through further training and exposure to more data.

In addition to machine learning, computer vision may also use techniques such as deeplearning, which involves training artificial neural networks on large amounts of data to recognize patterns and features in away that is similar to how the human brain works.

6.1 HISTORY OF COMPUTER VISION

The history of computer vision dates back over 60 years, with early attempts to understand how the human brain processes visual information leading to the development of image-scanning technology in 1959. In the 1960s, artificial intelligence emerged as an academic field of study, and computers began transforming two-dimensional images into three-dimensional forms.

In the 1970s, optical character recognition technology was developed, allowing computers to recognize text printed in any font or typeface. This was followed by the development of intelligent character recognition, which could decipher hand-written text using neural networks. Real-world applications of these technologies include document and invoice processing, vehicle plate recognition, mobile payments and machine translation.

In the 1980s, neuroscientist David Marr established that vision works hierarchically and introduced algorithms for machines to detect edges, corners, curves and other basic shapes. At the same time, computer scientist Kunihiko Fukushima developed a network of cells called the Neocognitron that could recognize patterns, including convolutional layers in a neural network.

In the 1990s and 2000s, real-time face recognition apps appeared, and there was a standardization of visual data set tagging and annotating. In 2010, the ImageNet data set becameavailable, containing millions of tagged images across a thousand object classes and providinga foundation for convolutional neural networks (CNNs) and deep learning models used today.

In 2012, the AlexNet model made a breakthrough in image recognition, reducing the error rate to just a few percent. These developments have paved the way for the widespread use of computer vision in a variety of applications today.

6.2 HOW DOES COMPUTER VISION WORK?

The computer vision system consists of two main components: a sensory device, such as a camera, and an interpreting device, such as a computer. The sensory device captures visual data from the environment andthe interpreting device processes this data to extract meaning.

Computer vision algorithms are based on the hypothesis that "our brains rely on patterns to decode individual objects." Just as our brains process visual data by looking for patterns in the shapes, colors and textures of objects, computer vision algorithms process images by looking for patterns in the pixels that make up the image. These patterns can be used to identify and classify different objects in the image.

To analyze an image, a computer vision algorithm first converts the image into a set of numerical data that can be processed by the computer. This is typically done by dividing the image into a grid of small units called pixels and representing each pixel with a set of numerical values that describe its color and brightness. These values can be used to create a digital representation of theimage that can be analyzed by the computer.

Once the image has been converted into numerical data, the computer vision algorithm can begin to analyze it. This generally involves using techniques from machine learning and artificial intelligence to recognize patterns in the data and make decisions based on those patterns. For example, an algorithm might analyze the pixel values in an image to identify the edges of objects or to recognize specific patterns or textures that are characteristic of certain types of objects.

Overall, the goal of computer vision is to enable computers to analyze and understand visual data in much the same way that human brains and eyes do, and to use this understanding to make intelligent decisions based on that data.

6.3 COMPUTER VISION AT WORK

Computer vision has provided numerous technological benefits in various industries and applications.

One example is IBM's use of computer vision to create "My Moments" for the 2018 Masters golf tournament. This application used computer vision to analyze live video footage of the tournament and identify key moments, such as successful shots or notable events. These moments were then curated and delivered to fans as personalized highlight reels, allowing them to easily keep track of the tournament andstay engaged with the event.

Disney theme parks have also made use of computer vision and AI predictive technology to improve their operations. The technology works with high-tech sensors to help keep attractions running smoothly, with minimal disruptions. For example, if an attraction is experiencing technical issues, the system can predict the problem and automatically dispatch maintenance staffto fix it, helping to keep the attraction running smoothly and preventing disruptions for guests.

Google Translate is another example of the use of computer vision in technology. This application uses a smartphone camera and computer vision algorithms to analyze and translate text in images, such as signs or documents in foreign languages. This allows users to easily translate text on the go, making it easier to communicate and navigate in unfamiliar environments.

Finally, IBM and Verizon have been working together to help automotive companies identify vehicle defects before they depart the factory. Using computer vision and other advanced technologies, they are developing systems that can analyze the quality of vehicle components and identify defects in real time, allowing companies to catch and fix problems before they become larger issues. This can help improve the quality and safety of vehicles, as well as reduce production costs by catching problems early on in the manufacturing process.

6.4 EXAMPLES OF COMPUTER VISION

Computer vision has a wide range of capabilities and applications in various industries. Here are someexamples of computer vision capabilities, along with brief explanations of each:

- Optical character recognition (OCR): the ability to recognize and extract text from images orscanned documents

- Machine inspection: the use of computer vision to inspect and evaluate the quality or condition ofvarious components or products

- Retail: the use of computer vision in automated checkout systems and other retail applications, such asinventory management and customer tracking

- 3D model building: the use of computer vision to analyze multiple images of an object or environmentand construct a 3D model of it

- Medical imaging: the use of computer vision to analyze medical images, such as X-rays or CT scans,to aid in the diagnosis and treatment of patients

- Automotive safety: the use of computer vision in driver assistance systems and autonomous vehiclesto detect and respond to obstacles and other hazards on the road

- Match move: the use of computer vision to align and merge CGI elements with live-action footage inmovies and other visual effects

- Motion capture: the use of computer vision to capture and analyze the movement of actors or otherobjects, typically for use in animation or virtual reality applications

- Surveillance: the use of computer vision to analyze video footage for security and monitoringpurposes

- Fingerprint recognition and biometrics: the use of computer vision to analyze and recognizeunique physical characteristics, such as fingerprints, for identity verification and other applications.

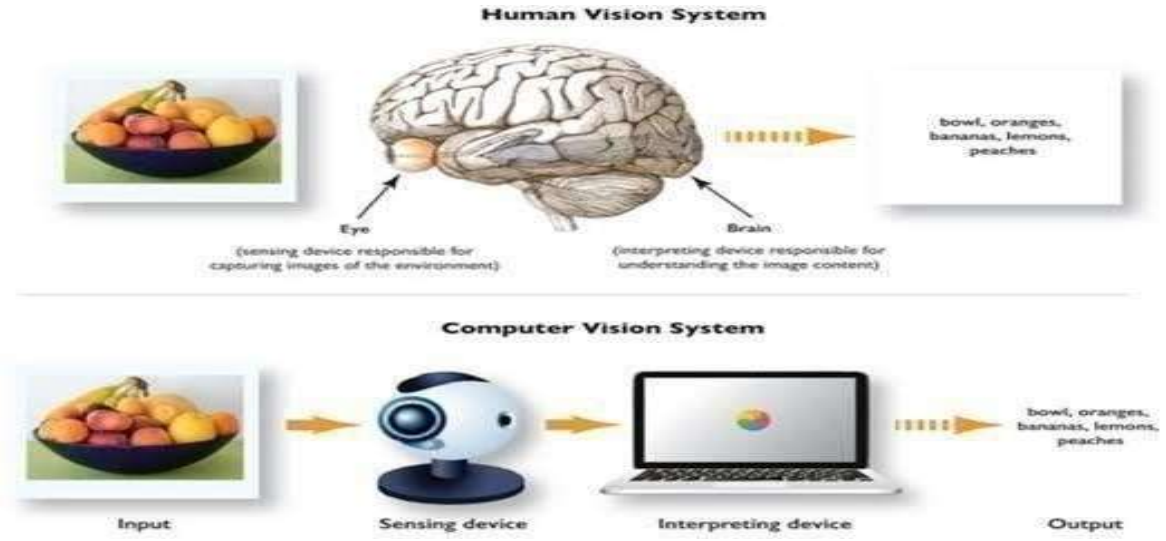

Fig.6.4.1.Computer Vision

6.5 THE CHALLENGES OF COMPUTER VISION

Computer vision is a complex field that involves many challenges and difficulties. Some of these challengesinclude:

- Data limitations

  Computer vision requires large amounts of data to train and test algorithms. This can be problematic in situations where data is limited or sensitive, and maynot be suitable for processing in the cloud. Additionally, scaling up data processing can be expensive and may be constrained by hardware and other resources.

- Learning rate

  Another challenge in computer vision is the time and resources required to train algorithms. While error rates have decreased over time, they still occur, and it takes time for the computer to be trained to recognize and classify objects and patterns in images. This process typically involves providing sets of labeled images and comparing them to the predicted output label or recognition measurements and then modifying the algorithm to correct any errors.

- Hardware requirements

  Computer vision algorithms are computationally demanding, requiring fast processing and optimized memory architecture for quicker memory access. Properly configured hardware systems and software algorithms are also necessary to ensure that image-processing applications can run smoothly and efficiently.

- Inherent complexity in the visual world

  In the real world, subjects may be seen from various orientations and in myriad lighting conditions, and there are an infinite number of possible scenes in a true vision system. This

  inherent complexity makes it difficult to build a general-purpose “seeing machine” that canhandle all possible visual scenarios.

6. WHAT IS OPENCV?

- OpenCV is an open-source software library for computer vision and machine learning. The OpenCV full form is Open Source Computer Vision Library. It was created to provide a shared infrastructure for applications for computer vision and to speed up the use of machine perception in consumer products. OpenCV as a BSD-licensed software, makes it simple for companies to use and change the code. There are some predefined packages and libraries that make our life simple and OpenCV is oneof them.
- Gary Bradsky invented OpenCV in 1999 and soon the first release came in 2000.
- The library has more than 2500 optimised algorithms, including an extensive collection of computervision and machine learning algorithms.
- Using OpenCV it becomes easy to do complex tasks such as identify and recognise faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D object models, generate 3D point clouds from stereo cameras, stitch images together to generate an entire scene with a high resolution image and many more.
- OpenCV-Python is a Python wrapper for the original OpenCV C++ implementation.

7. ARCHITECTURE OF FACIAL RECOGNITION

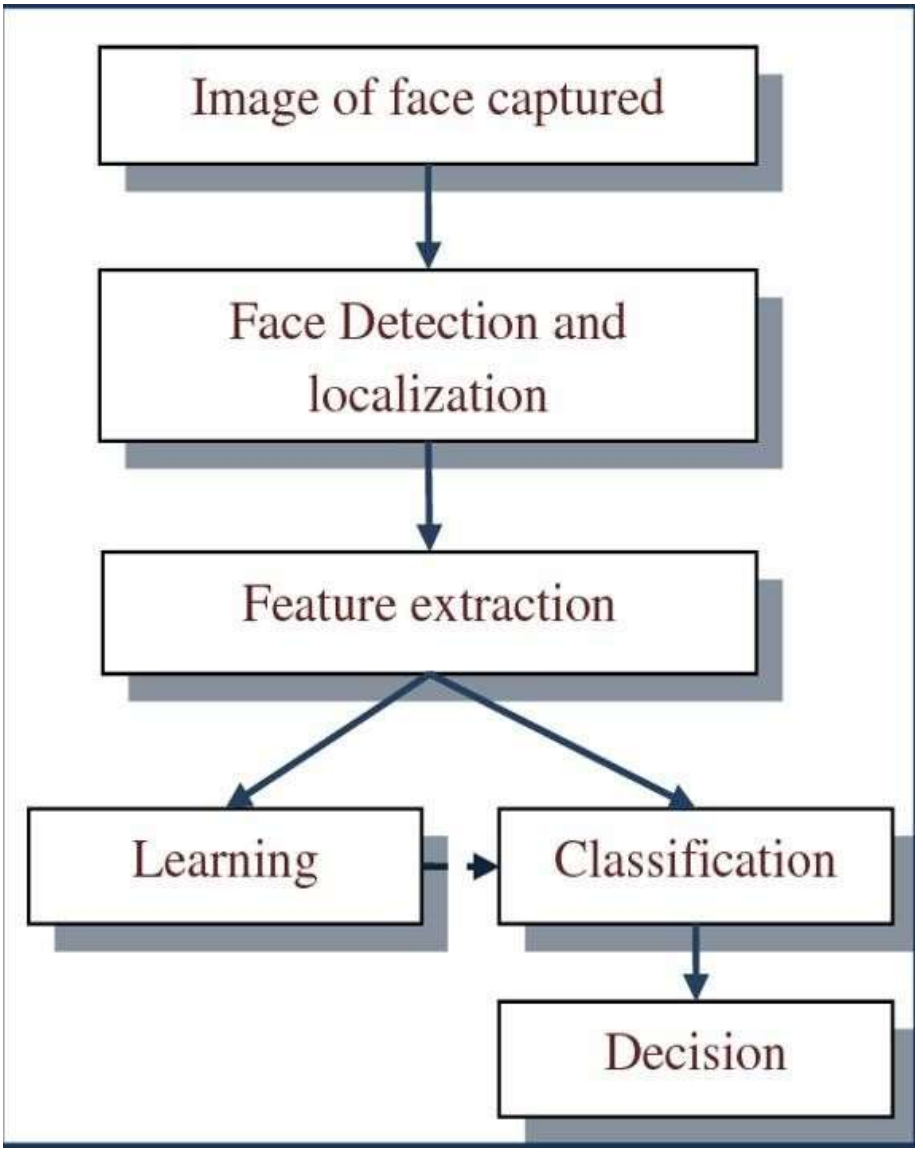

Fig.8.1.Architecture Of Facial Recognition

8.1 IMAGE OF FACE CAPTURED:

In face recognition, when an image of a face is captured, it is typically represented as a digital image. This digital image is composed of a grid of pixels, where each pixel contains information about the color or intensity of light at that point in the image.

In the context of face recognition, the captured image is usually preprocessed to enhance certain features or to normalize the image for better comparison with other images. This preprocessing may include steps such as resizing the image to a standard size, converting it to grayscale, and applying filters to reduce noise.

Once the image is preprocessed, it can be fed into a face recognition algorithm, such as a Convolutional Neural Network (CNN), which analyzes the image to extract features that are relevant for distinguishing one face from another. These features may include the shape of the eyes, the distance between key facial landmarks, and the overall structure of the face.

The output of the face recognition algorithm is typically a set of features or a numerical representation of the face, which can be compared with other faces in a database to determine if the faces match. If a match is found, the identity of the person in the captured image can be determined.

8.2 DETECTION: Face detection is the process of automatically locating faces in a digital image or video frame. The goal is to identify and localize the presence of one or more faces within the image. This task istypically the first step in many face-related applications, including face recognition, facial expression analysis, and face tracking.

Face detection algorithms work by analyzing the pixel values in an image to identify regions that are likely to contain faces. These algorithms often use machine learning techniques, such as Haar cascades, Histogram of Oriented Gradients (HOG), or Convolutional Neural Networks (CNNs), to detect faces with varying degrees of accuracy and speed.

8.3 FACE LOCALIZATION: Face localization is the process of precisely determining the location and size ofaface within an image. While face detection identifies the presence of faces in an image, face localization provides more detailed information about the position and scale of each detected face.

In face localization, the goal is to accurately outline the boundaries of the face in the image, typically represented as a bounding box. This information is important for subsequent processing steps, such as facealignment and feature extraction, in tasks like face recognition.

8.4 FEATURE EXTRACTION:

Feature extraction in face recognition is the process of capturing the unique characteristics of a face that can be used to distinguish it from other faces. These characteristics, or features, are

typically extracted from an image of the face are used to create a numerical representation of the face that can be compared with other faces in a database.

There are several approaches to feature extraction in face recognition, but one commonmethod is to use a Convolutional Neural Network (CNN). CNNs are deep learning models thatare well-suited for image processing tasks, including face recognition.

In a CNN-based approach, the network is trained on a large dataset of labeled face images to learn features that are relevant for distinguishing between different faces. These features are learned automatically by the network through the process of training, which involves adjusting the weights of the network to minimize the error in predicting the labels of the training images.

Once the CNN is trained, it can be used to extract features from new face images. These features can then be compared with the features of other faces in the database using techniques such as Euclidean distance or cosine similarity to determine if the faces match.

8. DIFFERENT APPROACHES OF FACE RECOGNITION

1. OpenCV
2. DLIB
3. Face recognition
4. TensorFlow

9.1 OPENCV:

- OpenCV (Open Source Computer Vision Library) is a popular open-source library forcomputer vision tasks, including face recognition.
- OpenCV provides built-in functions for face detection, facial landmark detection, and facerecognition.
- The face recognition module in OpenCV implements the LBPH (Local Binary PatternsHistograms) algorithm, which is a simple and effective approach for face recognition.
- OpenCV also provides tools for preprocessing images, such as resizing, normalization, andhistogram equalization, which can improve the performance of face recognition algorithms.

9.2 DLIB:

- DLIB is a C++ library that provides tools and algorithms for machine learning, imageprocessing, and computer vision tasks.
- DLIB's face recognition module is known for its accuracy and performance, especially indetecting facial landmarks and recognizing faces under various conditions.
- The face recognition module in DLIB uses a combination of HOG (Histogram of Oriented Gradients) features and a linear SVM (Support Vector Machine) classifier for face recognition.
- DLIB also provides tools for face alignment, which can improve the accuracy of facerecognition by normalizing the face images.

9.3 FACE RECOGNITION:

- Face recognition is a Python library that provides a simple interface for face detection, faceencoding (feature extraction), and face recognition tasks.
- The library is built on top of DLIB and provides a higher-level API for performing facerecognition tasks.

- Face recognition supports multiple face recognition algorithms, including the DLIB facerecognition algorithm and the FaceNet algorithm, which is based on deep learning.

9.4 TENSORFLOW:

- TensorFlow is an open-source machine learning framework developed by Google.
- TensorFlow provides tools and libraries for building and training machine learning models,including deep learning models for face recognition.
- TensorFlow can be used for face recognition by building and training a convolutional neuralnetwork (CNN) model that is capable of recognizing faces in images or video streams.

9. INSTALLING FACE RECOGNITION ON WINDOWS

Prerequisites:

Face Recognition module can only be installed for Python version 3.7

and 3.8.Step 1: Install git for Windows

Step 2: Clone this repository and go inside the folder using the following commands

```
git clone

https://github.com/RvTechiNNovate/face_recog_dli
```

Step 3: Enter the following command to install dlib and cmake using pip

Python 3.7:

```
pip install dlib-19.19.0-cp37-cp37m-win_amd64.whl
```

Python 3.8:

```
pip install dlib-19.19.0-cp38-
cp38-win_amd64.whlpip install
```

Method 1: Using pip to install Face Recognition Package

Follow the below steps to install the Face Recognition package on Windows using pip:

Step 1: Install the latest Python3 in Windows

Step 2: Check if pip and python are correctly installed.

```
python --version

pip --version
```

Step 3: Upgrade your pip to avoid errors during installation.

```
pip install --upgrade pip
```

Step 4: Enter the following command to install Face Recognition using pip3.

```
pip install face-recognition
```

9.1 FACE RECOGNITION USING FACE RECOGNITION LIBRARY

Due to its simplicity and use, a face-recognition library is a well-liked option for face-recognition jobs. It offers several crucial features and information that are required for face recognition. Here are several crucial components:

- Face detection: The face_recognition library provides a face_locations() method that locates all faces in an image or video frame and gives their bounding box coordinates. These bounding box coordinatesspecify the location and dimensions of each identified face.
- Face Landmarks: Face_landmarks(), a function in the library, finds and returns the positions of various facial landmarks, including the eyes, nose, mouth, and chin. These markers might be helpful for tasks like face alignment, emotion identification, and facial expression analysis.
- Face encodings: A function named face_encodings() is offered by the face_recognition library, and it computes a 128-dimensional numerical representation or encoding for each identified face. These encodings, which may be used for face comparison and identification, capture the distinctive traits of every face. The encodings can be kept in a database and contrasted with fresh face encodings for recognition.
- Face matching: The compare_faces() method provided by the library compares twosets of face encodings and provides a boolean result indicating whether or not they match. This feature can be used to compare a detected face to a database of recognized faces for identification or verification purposes.
- User database: A database of recognized faces and their related face encodings is necessary for face recognition. This database is a resource for locating and validating people. By saving each person's face encodings and distinct labels, the face_recognition library enables you to build and maintain sucha database.

  The face_recognition library's face-recognition processes may be implemented using these components. You can recognize and validate people in photos or video streams by recognizing faces, extracting facial landmarks,computing face encodings, and comparing them to a known database.

  It's crucial to remember that the effectiveness and efficiency of face recognition systemsdepend on the caliber of training data, the size of the database, and other elements likeillumination, position changes, and occlusions. Considering these factors, a facial recognition system should be designed and implemented carefully.

10. IMPLEMENTATION OF THE CODE

```
1. import face_recognition2.
3.   // load the image //
4.   image =
face_recognition.load_image_file("image.jpg")
5.
6.   // detect faces in the image //
7.   face_locations =
face_recognition.face_locations(image)8.
9.   // loop through the face locations and draw rectangles around the faces //
10. for face_location in face_locations:
11.      top, right, bottom, left = face_location
12.      cv2.rectangle(image, (left, top), (right, bottom),
(0, 0, 255), 2)13.
14. // display the image //
15. cv2.imshow("Faces", image)
16. cv2.waitKey(0)
```

Output:

Fig.11.1.Face Detection

11.1 EXPLANATION FOR CODE

1. the first line imports the face_recognition library, which detects and recognizes faces in images and videos.
2. the next line loads the image from the file "image.jpg" using the face_recognition.load_image_file() method.
3. the line after that detects faces in the image using the face_recognition.face_locations() method, which returns a list of face locations represented by four coordinates (top, right, bottom, left) for each face.
4. the following line starts a for loop that loops through each face location in the list. each iteration of the loop assigns the four coordinates of a face location to the variables top, right, bottom, and left.
5. the next line inside the for loop uses the cv2. to create a rectangle over the face, use the rectangle() method. the picture, the top-left quadrant of the rectangle, the lower right quadrent of the rectangle, the colour of the rectangle, and the thickness of the rectangle are all inputs that the function accepts. in this case, the rectangle is red (0, 0, 255), and the thickness is 2 pixels.
6. the line after that uses the cv2.imshow() method to display the image with the rectanglesdrawn aroundthe faces.
7. the last line uses the cv2.waitkey() method to wait for the user to press a key before closing the window. the argument against this method is the amount of time in milliseconds before the window closes. in this case, it is set to 0, which means the window will wait indefinitely for the user to press akey.

11.2 CONVOLUTION NEURAL NETWORK (CNN) ALGORITHM:

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 acomputer to understand and interpret the image or visual data.

When it comes to Machine Learning, Artificial Neural Networks perform really well. Neural Networks are used in various datasets like images, audio, and text. Different types of Neural Networks are used for different purposes, for example for predicting the sequence of words we use Recurrent Neural

Networks more precisely an LSTM, similarly for image classification we use ConvolutionNeural networks.In this blog, we are going to build a basic building block for CNN.

In a regular Neural Network there are three types of layers:

- Input Layers: It's the layer in which we give input to our model. The number of neurons in this layer is equal to the total number of features in our data (number of pixels in the case of an image).
- Hidden Layer: The input from the Input layer is then fed into the hidden layer. There can be many hidden layers depending on our model and data size. Each hidden layer can have different numbers of neurons which are generally greater than the number of features. The output from each layer is computed by matrix multiplication of the output of the previous layer with learnable weights of that layer and then by the addition of learnable biases followed by activation function which makes the network nonlinear.
- Output Layer: The output from the hidden layer is then fed into a logistic function like sigmoid or softmax which converts the output of each class into the probability score of each class.

11.3 ACHITECTURE OF ARTIFICIAL NERUAL NETWORK

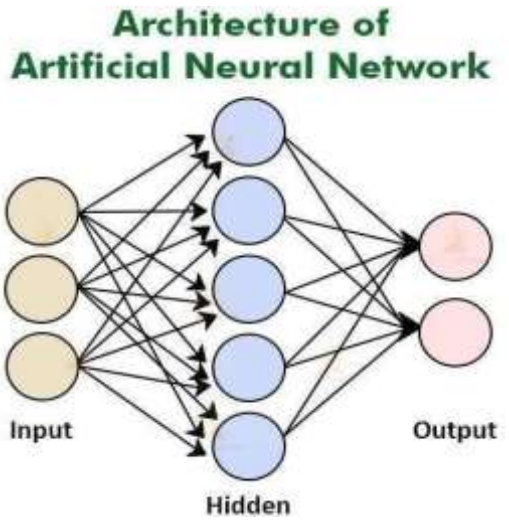

Fig.11.3.1.Architecture Of ANN

The data is fed into the model and output from each layer is obtained from the above step is called feedforward, we then calculate the error using an error function, some common error functions are cross-entropy, square loss error, etc. The error function measures how well the network is performing. After that, we backpropagate into the model by calculating the derivatives. This step is
called Backpropagation which basically is used to minimize the loss.

- Convolution Neural Network

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. For example visual datasets like images or videos where data patterns play an extensive role.

- CNN architecture

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

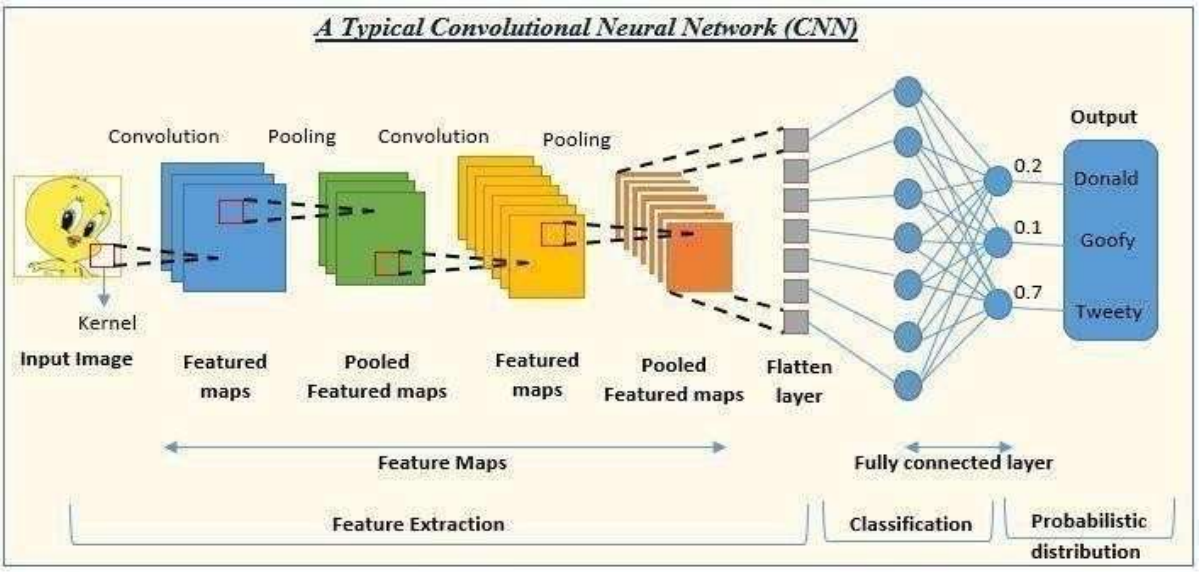

Fig.11.3.2.Architecture Of CNN

11.4 LAYERS USED TO BUILD CONVNETS

A complete Convolution Neural Networks architecture is also known as covnets. A covnets is a sequence of layers, and every layer transforms one volume to another through a differentiable function.

11.5 TYPES OF LAYERS: DATASETS

Let's take an example by running a covnets on of image of dimension 32 x 32 x 3.

- Input Layers: It's the layer in which we give input to our model. In CNN, Generally, the input will be an image or a sequence of images. This layer holds the raw input of theimage with width 32, height 32, and depth 3.

1. Convolutional Layers: This is the layer, which is used to extract the feature from the input dataset. It applies a set of learnable filters known as the kernels to the input images. The filters/kernels are smaller matrices usually 2×2, 3×3, or 5×5 shape. it slidesover 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. Suppose we use a total of 12 filters for this layer we'll get an output volume of dimension 32 x 32 x 12.

2. Operation at convolution layer

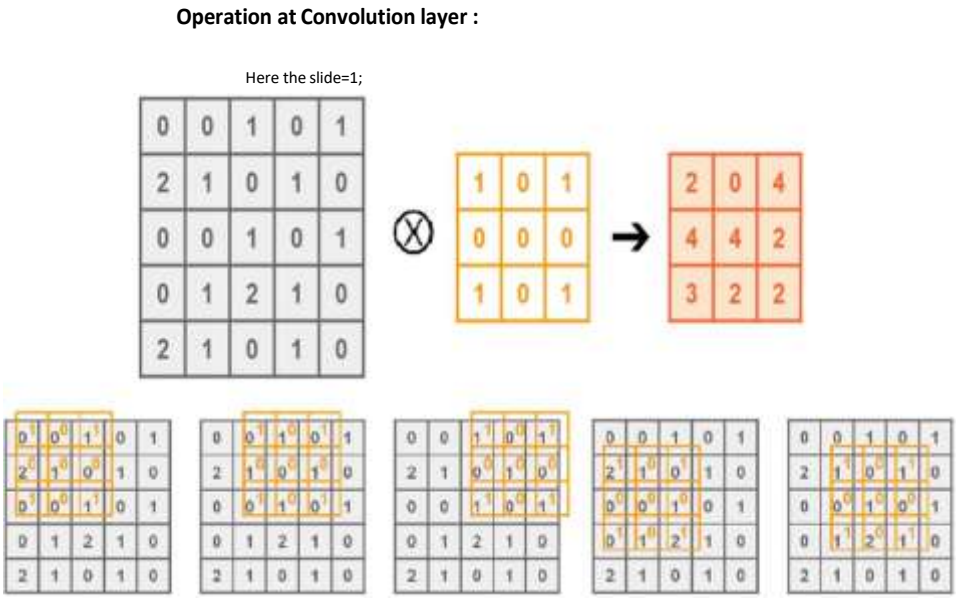

Fig.11.5.1. Operation At Convolutional Layer

1. Activation Layer: By adding an activation function to the output of the preceding layer, activation layers add nonlinearity to the network. it will apply an element-wiseactivation function to the output of the convolution layer. Some common activation functions are RELU: max(0, x), Tanh, Leaky RELU, etc. The volume remains unchanged hence output volume will have dimensions 32 x 32 x 12.
2. Pooling layer: This layer is periodically inserted in the covnets and its main function isto 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. If we use a max pool with 2 x 2 filters and stride 2, theresultant volume will be of dimension 16x16x12.

3. Operation at pooling layer

Operation at Pooling layer:

There are 2 poolingoperations:

- Max pooling:Takethe max value from the pool.
- Averagepooling:Takethe average value from the pool.

1. Flattening: The resulting feature maps are flattened into a one-dimensional vector after the convolution and pooling layers so they can be passed into a completely linked layer for categorization or regression.

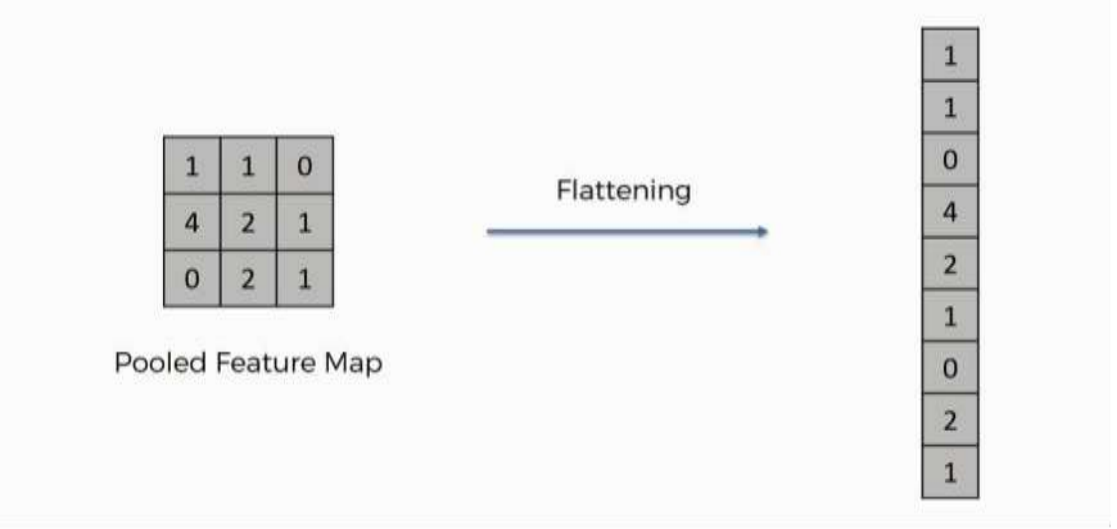

Fig.11.5.2.Operation At Pooling Layer

2. Fully Connected Layers: It takes the input from the previous layer and computes thefinalclassification or regression task.

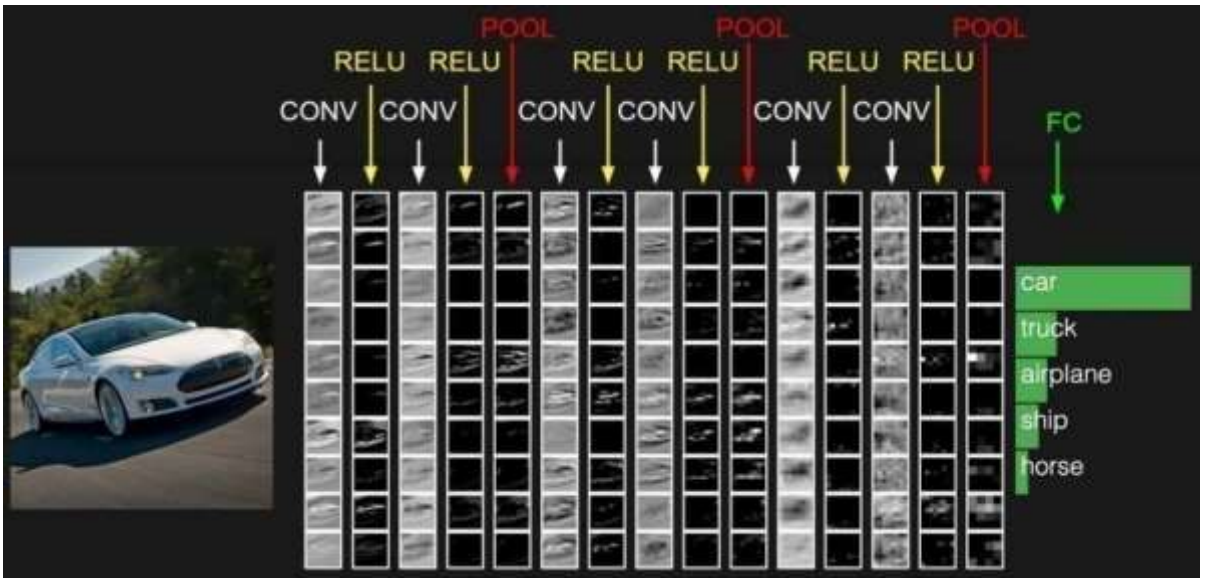

Fig.11.5.3. Flattening Layer

3. 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.

4. Back propagation Layer:Backpropagation is an algorithm commonly used to train neural networks. When the neural network is initialized, weights are set for its individual elements, called neurons. Inputs are loaded, they are passed through the network of neurons, and the network provides an output for each one, given the initial weights. Backpropagation helps to adjust the weights of the neurons so that the result comes closerand closer to the known true result.Backpropagation, short for "backward propagation of errors," is an algorithm for supervised learning of artificial neural networks using gradient descent. Given an artificialneural network and an error function, the method calculates the gradient of the error function with respect to the neural network's weights. It is ageneralization of the delta rule for perceptron's to multilayer feedforward neural networks.The "backwards" part of the name stems from the fact that calculation of the gradient proceedsbackwards through the network, with the gradient of the final layer of weights being calculated first and the gradient of the first layer of weights being calculated last. Partial computations of the

5. gradient from one layer are reused in the computation of the gradient for the previous layer. This backwards flowof the error information allows for efficient computation of thegradient at each layer versus the naive approach ofcalculating the gradient of each layer separately.

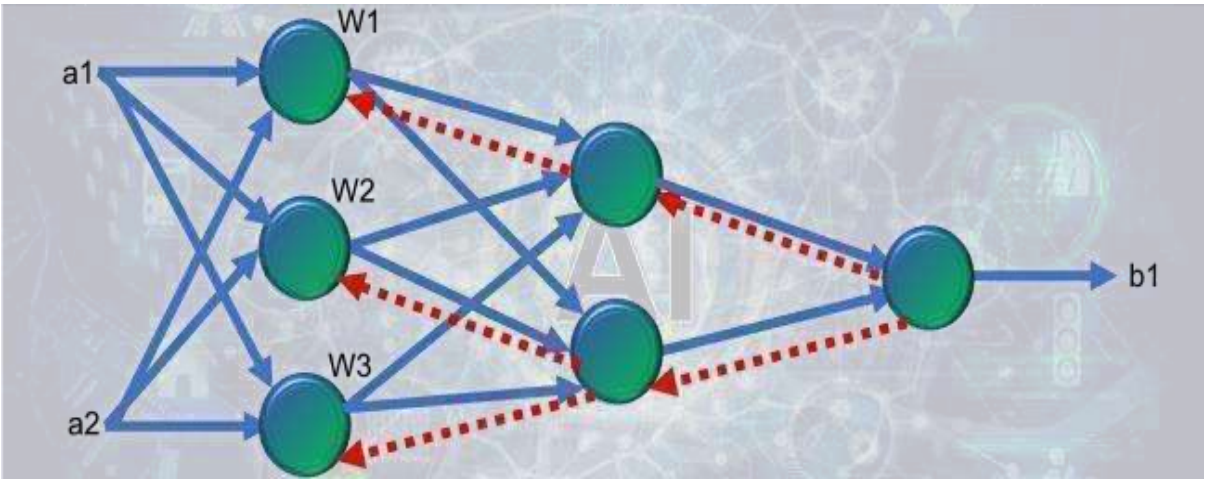

Fig.11.5.4. Fully Connected Layer

Backpropagation's popularity has experienced a recent resurgence given the widespread adoption of deep neural networks for image recognition and speech recognition. It is considered an efficient algorithm, and modern implementations take advantage of specialized GPUs to further improve performance.

4. Different architectures in CNN

CNN is the most well-known and widely used algorithm in deep learning. The fundamental advantage of cnn over its predecessors is that it detects relevant elements without the need for human intervention. CNNs have been widely used in a variety of domains, such as computer vision, audio processing, and facial recognition, among others. CNNs are like traditional neural networks in that their structure is inspired by neurons in human and animals' brains. There are variety of cnn architectures available, all of which have contributed to the development ofalgorithms that enable AI today and will continue to 10 do so in the future. This section covers the most well-known cnn architectures. A couple of them are listed below. (Kumar 2022.)

LetNet The first CNN architecture is LetNet. Yann LeCun, Corina Cortes, and Christopher Burges created it in 1998 to solve problems with handwritten digit recognition. Five convolution layers are followed by two completely connected layers in the model. CNNs were first used in deep learning for computer vision problems with LetNet. However, because to the vanishing gradients problem, LetNet was unable to train effectively. To address this issue, between convolutional layers, a shortcut connection layer known as max- pooling is utilized to minimize the spatial dimension of images, preventing overfitting, and allowing cnn's to

train more successfully. Figure 6 shows the architecture of LetNet-5. (Kumar 2022.) FIGURE 6. LetNetarchitecture (Vitaflux 2021.)

ZFNet ZFNet is a cnn architecture that combines fully connected layers with cnn's. Matthew Zeiler and Rob Fergus created ZFNet. It was the winner of the 2013 ILSVRC. Despite having fewer parameters than AlexNet, the network outperforms it in the ILSVRC 2012 classification test, obtaining top accuracy with only 1000 photos per class. It was better than AlexNet because the architectural hyperparameters were tweaked, especially the size of the middle convolutional layers and the stride and filter size on the first layer. The ZFNet cnn architecture has seven layers: convolutional layer, max-pooling layer (downscaling), concatenation layer, convolutional layer with linear activation function, and stride one, a dropout 11 applied before the completely connected output for regularization purposes. By introducing an approximate inference stage via deconvolutional layers in the middle of cnn's, this cnn model is computationally more efficient than AlexNet. The structure of ZFNet is shown in Figure 7. (Kumar 2022.) FIGURE 7. ZFNet architecture (Opengenus 2022.)

VGGNet VGGNet is the cnn architecture developed at Oxford university by Karen Simonyan, Andrew Zisserman, and others. VGGNet is a 16-layer cnn that has been trained on over one billion images and has up to 95 million parameters (1000 classes). It has 4096 convolutional features and can handle 224 by 224 pixel input images. CNNs with such large filters areexpensive to train and require a significant amount of data, which is why CNN architectures like GoogleNet (AlexNet architecture) outperform VGGNet for most image classification task with input images ranging from 100 x 100 pixels to 350 x 350 pixels. The ILSVRC 2014 classificationchallenge, which was also won by GoogleNet cnn architecture, is a realworld application
/ example of VGGNet cnn architecture. Due to its application on a variety of tasks, including object recognition, the VGG cnn model is computationally economical and serves as a good basisfor many applications in computer vision. Its deep feature representation is employed in yolo, ssd, and other neural network architectures. Figure 8 shows a typical VGG16 network architecture. (Kumar 2022.) 12 FIGURE 8. VGGNet architecture (Vitaflux 2021.) 3.3.4 MobileNets MobileNets are a CNN design that is both efficient and portable in real world applications. CNNs that can fit on a mobile device to classify photos or detect objects with low latency are known as MobileNets. They are often very tiny CNN architectures, making them simple to execute in real-time on embedded devices such as smartphones and drones. The design is very adaptable, having been tested on CNNs with 100-300 layers and outperforming other architectures such as VGGNet. CNNs embedded into Android phones to run Google’s Mobile Vision API, which can automatically recognize labels of popular objects in photos, are in real- world examples of MobileNets CNN architecture. Figure 9 shows MobileNet architecture.

12. APPLICATIONS OF FACE RECOGNITION

1 Security and Surveillance:

Face recognition is commonly used in security systems to identify and authenticate individuals. It can be used in airports, banks, and other secure facilities toenhance security measures.

2 Access Control:

Face recognition can be used for access control in buildings, offices, and residential complexes. It allows for secure and convenient entry without the need for keys oraccess cards.

3 Attendance Tracking:

Face recognition can be used to track attendance in schools, universities, and workplaces. It provides a convenient and efficient way torecord attendance without manualinput.

4 Law Enforcement:

Law enforcement agencies use face recognition to identifysuspects and criminals from surveillance footage or photos. It can help in solving crimes and locatingindividuals.

5 Personalized Marketing:

Face recognition can be used in marketing to personalize advertisements based on the age, gender, or other characteristics of the viewer. Itcan help in targeting specific demographics more effectively.

6 Smartphones and Devices:

Many smartphones and devices now come with face recognition technology for the device. It provides a convenient and secure way to access the device without a passcode.

7 Emotion Recognition:

Face recognition can be used to detect emotions from facial expressions. It hasapplications in psychology, market research, and human-computer interaction.

8 Healthcare:

In healthcare, face recognition can be used for patient identification,monitoring,and Tracking. It can help in providing personalized healthcare services.

9 Automotive Industry:

Face recognition can be used in cars for driver identification andauthentication.It can enhance security and provide personalized driving experiences.

10 Retail Industry:

In retail, face recognition can be used for customer analysis and personalizedshopping experiences. It can help in understanding customer behavior and preferences.

11 Border Control:

Face recognition is used in border control to verify the identittravelersItcan help in improving border security and reducing wait times.

12 Humanitarian Aid:

Face recognition can be used in humanitarian aid efforts to identify and reunitefamilies separated during conflicts or natural disasters.

13 Education:

In education, face recognition can be used for studentidentification andmonitoring. It can help in ensuring the safety andsecurity of students.

14 Virtual and Augmented Reality:

Face recognition can be used in virtual and augmented reality applications forrealistic avatars and interactive experiences. It can enhance userimmersion and engagement.

15 Sports and Entertainment:

Face recognition can be used in sports and entertainment for fanengagement and security. It can help in identifying VIPs and enhancing the overall experience.

16 Aging and Healthcare:

Face recognition can be used to track aging and health-relatedchanges in individuals. It can help in early detection of health issues and personalized healthcareplanning.

12.1 ADVANTAGES OF FACE RECOGNITION USING OPENCVWITH FACE_RECOGNITION LIBRARY

1. Ease of Use: The face_recognition library simplifies face recognition tasks by providing high-level APIs for face detection, facial landmark localization, and face recognition. This makes it easy for developers to integrate face recognition capabilities into their projects withoutthe need for extensive knowledge of computer vision algorithms.
2. Accuracy: The face_recognition library is built on top of deep learning models, which are known for their high accuracy in face recognition tasks. This ensures that your face recognitionsystem can reliably identify individuals even in challenging conditions such as varying lightingand facial expressions.
3. Speed: The face_recognition library is optimized for speed, allowing forreal-time face recognition in video streams or live camera feeds. This iscrucial for applications that require fastand responsive face recognition, such assecurity and surveillance systems.
4. Scalability: The face_recognition library is designed to be scalable, allowing you to easily scale your face recognition system to handle large datasets of faces. This makes it suitable for applications that require face recognition on a large scale, such as access control systems in largeorganizations.
5. Compatibility: The face_recognition library is compatible with OpenCV, a popular computervision library in Python. This allows youto leverage the powerful image processing capabilitiesof OpenCV in conjunction with the high-level face recognitioAPIs provided by the face_recognition library
6. Community Support: The face_recognition library has a large and active community of developers who contribute to its development and provide support to users. This means that youcan easily find help andresources online when working with the face_recognition library.

7. Versatility: The face_recognition library is versatile and can be usedfor a wide range of facerecognition tasks, from simple face detection to more complex tasks such as facial landmark localization and face clustering. This makes it suitable for a variety of applications across

different industries.

8. Privacy: The face_recognition library prioritizes user privacy byproviding features such as face encoding, which allows you to store and compare face encodings instead of raw images. This ensures that sensitive facial information is not exposed orstored unnecessarily.

9. Cost-Effectiveness: Building a face recognition system from scratch can be costlyand time-consuming. By using the face_recognition library,you can save time and resources by leveraging pre-trained deep learningmodels for face recognition.

10. Real-World Applications: The face_recognition library has been used invarious real-worldapplications, including security systems, attendance tracking systems, and personalized marketing. This demonstrates itseffectiveness and reliability in real-worldscenarios.
11. Continued Development: The face_recognition library is actively maintained and developed, with new features and improvements being regularly released. This ensures that yourface recognition system remains up- to-date with the latest advancements in facerecognition technology.

13. LIBRARIES USED

13.1 CV2

cv2 is the shorthand for OpenCV (Open Source Computer Vision Library), a popular open- source computer vision and machine learning software library. It is used in a wide range of applications, including facial recognition. Here's how cv2 is useful in face recognition in detail:

1. Image and Video Input: OpenCV provides functions to read images and videos from various sources, such as files, cameras, or streams. This is essential for capturing the faces to be recognized.
2. Preprocessing: OpenCV offers a variety of functions for image preprocessing, including resizing, normalization, and conversion to grayscale. These preprocessing steps can enhance the quality of facial images, making them more suitable for recognition algorithms.
3. Face Detection: OpenCV includes pre-trained models for face detection, such as Haar cascades and deep learning-based detectors. These models can quickly and accurately identify the presence and location of faces in an image or video frame.
4. Facial Landmark Detection: OpenCV provides tools for detecting facial landmarks, such as the eyes, nose, and mouth. These landmarks can be used to align faces and extract features for recognition.
5. Face Recognition Algorithms: While OpenCV itself does not include specific face recognition algorithms, it provides a framework for integrating custom or third-party algorithms. This allows developers to use state-of-the-art face recognition techniques, such as deep learning-based models.
6. Feature Extraction: OpenCV can be used to extract features from facial images, such as histograms of oriented gradients (HOG) or local binary patterns (LBP). These features can then be used for face matching and identification.
7. Matching and Identification: OpenCV provides functions for matching faces based on extracted features or similarity metrics. This enables the identification of faces by comparing them to a database of known faces.
8. Real-Time Processing: OpenCV is optimized for real-time processing, making it suitable for applications that require fast and efficient face recognition, such as surveillance systems or interactive applications.
9. Integration with Other Libraries: OpenCV can be easily integrated with other Python libraries, such as NumPy and scikit-learn, for additional image processing and machine learning capabilities.

10. Cross-Platform Compatibility: OpenCV is cross-platform and can be used on various operating systems, including Windows, macOS, and Linux, making it accessible to a wide range of developers.

13.2 FACE_RECOGNITION

The face_recognition library is a popular Python package used for face recognition tasks. It is built on top of dlib's state-of-the-art face recognition algorithm, which is based on deep learning. Here's a detailed overview of how it works and its usefulness in face recognition:

1. Face Detection: The face_recognition library can detect human faces in images. It uses a pre-trainedconvolutional neural network (CNN) to locate faces in an image.
2. Face Encoding: Once faces are detected, the library can generate face encodings. Face encoding is a numerical representation of the face that captures unique characteristics of the face, making it suitable for comparison with other faces.
3. Face Recognition: By comparing face encodings, the face_recognition library can recognize faces.It can identify known faces in images or videos by comparing them with a database of known face encodings.
4. Usefulness:

    - Security: Face recognition can be used for secure access control, such as unlocking devicesor accessing restricted areas.
    - Attendance Tracking: It can automate attendance tracking systems by recognizing faces ofindividuals.
    - Personalization: Face recognition can be used for personalized services, such as customizedadvertising or content recommendations.
    - Law Enforcement: It can assist law enforcement agencies in identifying suspects fromsurveillance footage or images.

5. Ease of Use: The face_recognition library is easy to use, making it accessible to developers with varying levels of expertise. It provides a high-level interface for face recognition tasks, simplifying thedevelopment process.

13.3 OS

The os library in Python provides a way to interact with the operating system. It allows you to perform various operations such as navigating the file system, manipulating paths, and executing system commands.Here's how the os library can be useful for face recognition in detail:

1. File Operations: The os library can be used to manage files and directories, which is essential for handling image files in a face recognition project. You can use it to navigate to the directory containing the images, list files in a directory, and create new directories if needed.
2. Path Manipulation: The path submodule provides functions for manipulating file paths. This is useful for constructing paths to image files or defining the location where processed images or faceencodings should be saved.
3. Environment Variables: The os module allows you to access and modify environment variables, which can be useful for storing sensitive information such as API keys or file paths needed for facerecognition.
4. System Commands: In some cases, you may need to execute system commands from your Python code. The os library provides functions like system() or os.popen() for executing such commands, which can be useful for interacting with external programs or libraries needed for face recognition.
5. Platform Independence: The os library abstracts away platform-specific differences, making your code more portable. This is important if your face recognition project needs to run on different operating systems.

13.4 GLOB

The glob library in Python provides a way to search for files using wildcards in a specified directory or path. It's particularly useful for tasks that involve processing multiple files that match a specific pattern. Here's how the glob library can be useful in face recognition in detail:

1. Batch Processing: When dealing with a large number of image files, the glob library can be used to efficiently process all files that match a certain pattern. For example, you can use it to load all imagesin a directory for face detection and recognition.
2. File Selection: The glob library allows you to select files based on a pattern, such as all files with a specific extension (e.g., .jpg, .png). This can be useful for filtering out non-image files from a directory before processing.
3. Iterating Over Files: The glob() function returns a list of file paths that match thespecified pattern. You can then iterate over this list to process each file individually, suchas detecting and recognizingfaces in each image.
4. Dynamic File Selection: The glob library supports the use of wildcard characters (* and ?) in the search pattern, allowing you to specify more complex selection criteria. This flexibility can be usefulfor selecting files based on their names or other attributes.
5. Cross-Platform Compatibility: Like the os library, glob is cross-platform and works on different operating systems without modification, making it suitable for face recognition projects that need torun on various platforms.

13.5 NUMPY

The numpy library is a fundamental package for scientific computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. Here's how the numpy library is useful in face recognition indetail:

1. Image Representation: In face recognition, images are often represented as arrays of pixel values. NumPy's array manipulation capabilities make it easy to work with these image arrays, such as accessing individual pixels, cropping images, or resizing images.
2. Vectorized Operations: NumPy allows you to perform operations on arrays element-wise, which can significantly improve the performance of computations. This is particularly useful in face recognition algorithms that involve processing a large number of images or performing complexmathematical operations on image arrays.
3. Mathematical Operations: NumPy provides a wide range of mathematical functions for performing operations such as matrix multiplication, linear algebra operations, Fourier transforms, and more. These functions are essential for implementing various face recognition algorithms, such as PCA (Principal Component Analysis) or LDA (Linear Discriminant Analysis).
4. Efficient Memory Management: NumPy's array data structure is more memory-efficient compared to Python lists, especially for large datasets. This efficiency is crucial in face recognition projects that involve processing a large number of high-resolution images.
5. Integration with Other Libraries: NumPy is a foundational library in the Python scientific computing ecosystem and is often used in conjunction with other librarieslike OpenCV and scikit- learn for face recognition tasks. Its interoperability with these libraries makes it an essential component in the face recognition workflow.

13.6 MATPLOTLIB

The matplotlib library is a popular Python library used for creating static, animated, and interactive visualizations in Python. While not directly related to face recognition, matplotlib can be useful in face recognition projects for visualizing various aspects of the data or the results of the face recognition algorithms. Here's how matplotlib can be usefulin face recognition:

1. Data Visualization: matplotlib can be used to visualize the face images or the results of face detection and recognition algorithms. For example, you can use it to display the detected faces in an image, plot histograms of pixel intensities, or show the results of face recognition (e.g., matching faces).
2. Debugging and Analysis: Visualizations can help in debugging the face recognition algorithm by visualizing intermediate results or errors. For example, you can plot the intermediate representations of face images in a neural network to understand how the network is processing the data.
3. Performance Evaluation: matplotlib can be used to create plots and graphs for performance evaluation metrics such as accuracy, precision, recall, and F1-score. This can help in comparing different face recognition algorithms or tuning hyperparameters.
4. Interactive Visualization: matplotlib supports interactive plotting features through libraries like pyplot and matplotlib.animation, which can be useful for exploring the data or results of face recognition algorithms interactively.
5. Integration with GUIs: matplotlib can be integrated with GUI frameworks like Tkinter or PyQt to create interactive applications for face recognition, where users caninteract with the data or algorithm results through a graphical interface.

13.7 SEABORN

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. While Seaborn is not directly used for face recognition algorithms, it can be useful in the context of data exploration and visualization, which are important steps in the development and analysis of face recognition systems.

Here's how Seaborn can be useful in face recognition projects:

1. Data Exploration: Before implementing a face recognition algorithm, it's important to understand the characteristics of the dataset. Seaborn's functions for plotting histograms, scatter plots, and box plots can help visualize the distribution of face features such as pixel intensities, face dimensions, or feature distances. This can provide insights into the data and help in designing the recognition algorithm.
2. Data Preprocessing: Seaborn can be used to visualize the preprocessing steps applied to the image data, such as resizing, normalization, or feature extraction. Visualizing these transformations can help in verifying that the preprocessing steps are applied correctly and are suitable for the recognitiontask.
3. Model Evaluation: After training a face recognition model, Seaborn can be used to visualize the performance metrics such as accuracy, precision, recall, and F1 score. Plots like confusion matrices or ROC curves can provide a comprehensive view of the model's performance and help inidentifying areas for improvement.
4. Comparative Analysis: Seaborn's functions for grouping and categorizing data can be useful for comparative analysis of different face recognition algorithms or parameter settings. For example, you can use Seaborn to compare the performance of different CNN architectures or different hyperparameter settings.
5. Presentation and Reporting: Seaborn's visually appealing plots can be used in presentations, reports, or research papers to communicate the results of a face recognition project effectively. The ability to customize the appearance of plots allows for creating professional-looking visualizations.

13.8 PERFORMANCE MEASURES

After training the model, it's performance should be measured. There are different metrics to evaluate a Classification model like Loss, Accuracy, Precision and Recall. We use Matplotlib to plot the graphs ofdifferent metrics of the model.

Some important terms in Classification Metrics are:

True Postives (TP): It is the case where the predicted output is positive and it is true (the actual output isalso postive).

True Negatives (TN): It is the case where the predicted output is negative and it is true (the actual outputisalso negative).

False Postives (FP): It is the case where the predicted output is positive but it is false (the actual output isnegative).

False Negatives (FN): It is the case where the predicted output is negative but it is false (the actual outputis postive).

1 Loss

Loss is calculated after each epoch to measure the performance of the model. We use Categorical Cross- entropy because this is used when there are two or more label classes. This expects labels to be provided ina one-hot representation.

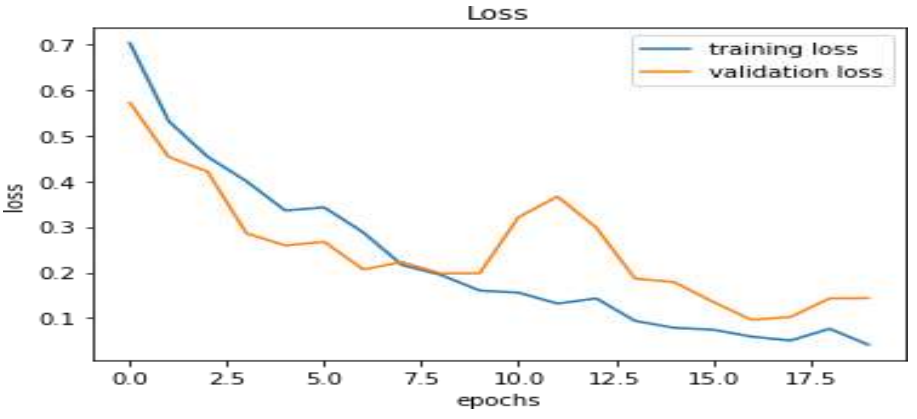

Fig.13.8.1.Loss

2 Accuracy

Classification accuracy is the number of correct predictions made as a ratio of all predictions made.Accuracy = No.of correct predictions / Total no.of Input samples

The graph of accuarcy on training data and validation data during the training process is as follows:

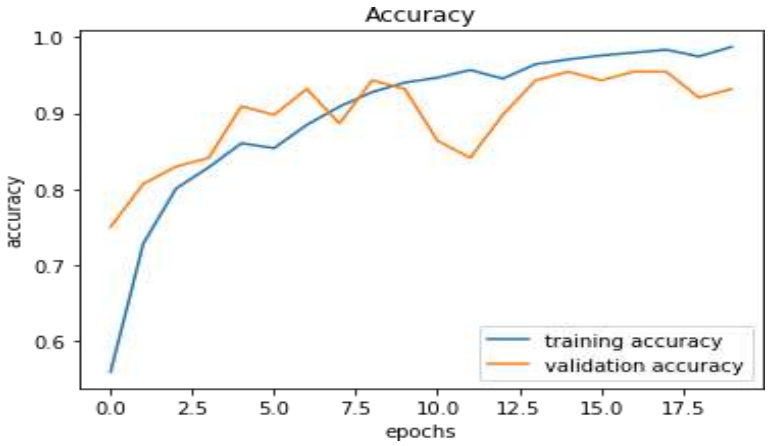

Fig.13.8.2. Training And Validation Accuracy

3 Precision

Precision (P) is defined as the number of true positives (Tp) over the number of true positives plus thenumber of false positives (Fp).:

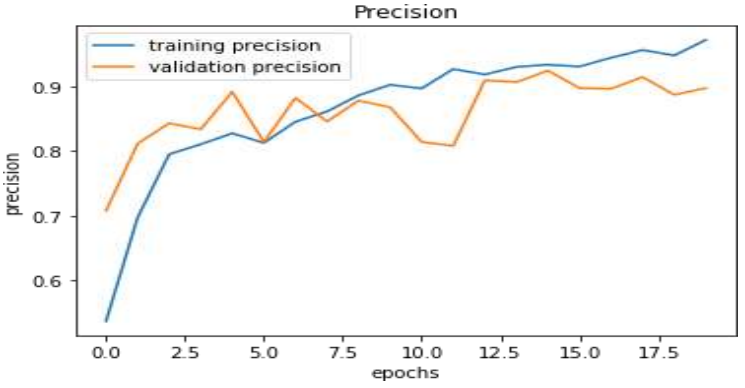

Fig.13.8.3. Precision

4 Recall

Recall (R) is defined as the number of true positives (Tp) over the number of true positives plus the number of false negatives (Fn). The graph of recall on training data and validationdata during the trainingprocess is as follows:

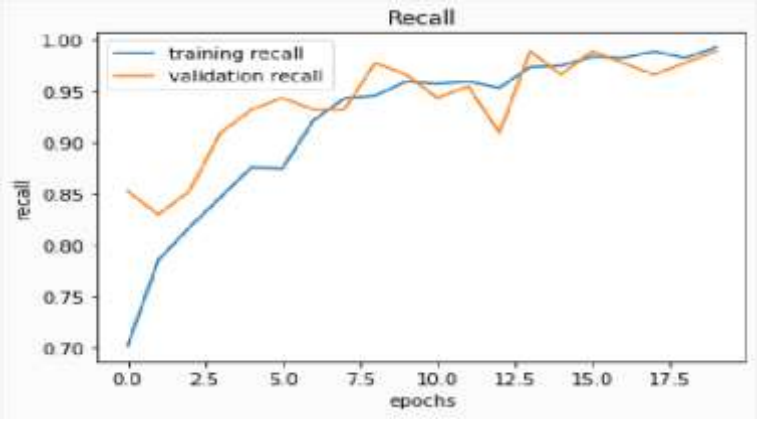

Fig.13.8.4. Training And Validation ReCall

5 Confusion Matrix

A confusion matrix is a table that is often used to describe the performance of a classification model on aset of test data for which the true values are known.

The Figure of Confusion Matrix is as follows:

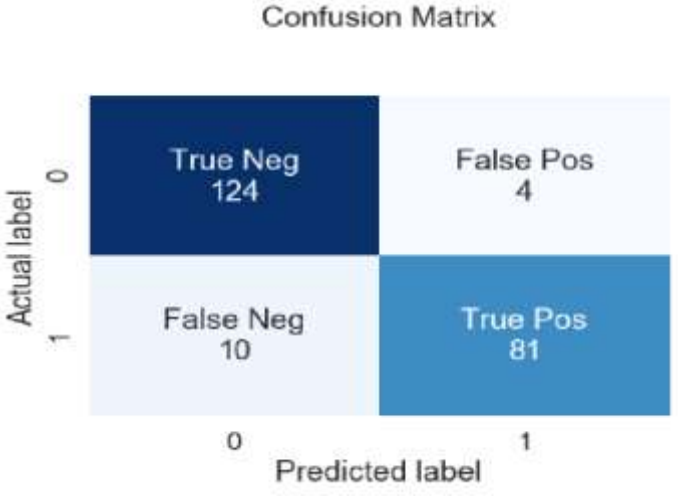

Fig 13.8.5. Confusion Matrix

14. SOURCE CODE:

```
import cv2

from simple_facerec import

SimpleFacerec# Initialize

SimpleFacerec

sfr=SimpleFacerec()

sfr.load_encoding_images("C:\\Users\\siddu\\Downloads\\images")


# Initialize video capture

cap = cv2.VideoCapture(0) # Adjust index as

neededwhile True:

    # Read frame from the

    video captureret, frame =

    cap.read()


    # Check if frame reading was

    successfulif not ret:

        print("Error: Failed to capture

        frame.")break


    # Detect known faces in the frame

    face_locations, face_names = sfr.detect_known_faces(frame)


    # Draw rectangles and put text on the frame for each

    detected facefor face_loc, name in zip(face_locations,

    face_names):
```

```
            y1, x2, y2, x1 = face_loc[0], face_loc[1], face_loc[2], face_loc[3]
            cv2.putText(frame, name, (x1, y1 - 10), cv2.FONT_HERSHEY_DUPLEX, 1, (0, 0, 200), 2)
            cv2.rectangle(frame, (x1, y1), (x2, y2), (0,255,0), 2)

        # Display the frame
        cv2.imshow("Frame", frame)

        # Check for the 'Esc' key to
        exit the loopkey =
        cv2.waitKey(1)

# Release the video capture and close all OpenCV windows
cap.release()
cv2.destroyAllWindows()
```

14.1 SOURCE FILE

simple_facerec.py file:

```
import
face_recogniti
onimport cv2
imp
ort              osimport glob
import numpy as np


class
   Simple
   Facere
   c:def
   init
   (self):
      self.known_face_enc
      odings = []
      self.known_face_na
      mes = []
      self.frame_resizing
      = 0.25


   def load_encoding_images(self, images_path):
      images_path =
      glob.glob(os.path.join(images_path, "*.*"))
      print(f"{len(images_path)} encoding images
```

```
found.")

for img_path in
    images_path: img
    =
    cv2.imread(img_
    path) if img is
    None:
        print(f"Error loading image:
        {img_path}")continue
    rgb_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    basename =
    os.path.basename(img_path)filename,
    _ = os.path.splitext(basename)
img_encoding = face_recognition.face_encodings(rgb_img)[0]

    self.known_face_encodings.append(img
    _encoding)
    self.known_face_names.append(filenam
    e)
print("Encoding images loaded")

def detect_known_faces(self, frame):
    small_frame = cv2.resize(frame, (0, 0), fx=self.frame_resizing,
                    fy=self.frame_resizing)rgb_small_frame = cv2.cvtColor(small_frame,
    cv2.COLOR_BGR2RGB)
```

```
        face_locations = face_recognition.face_locations(rgb_small_frame)
        face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)

        face_names = []
        for face_encoding in face_encodings:
            matches = face_recognition.compare_faces(self.known_face_encodings,
            face_encoding)name = "Unknown"
            face_distances = face_recognition.face_distance(self.known_face_encodings,
            face_encoding)best_match_index = np.argmin(face_distances)
            if matches[best_match_index]:
                name = self.known_face_names[best_match_index]
            face_names.append(name)

        face_locations = np.array(face_locations)
        face_locations = face_locations /
        self.frame_resizingreturn
        face_locations.astype(int), face_names

if    name
      == "
   main ":sfr =
   SimpleFace
   rec()
   sfr.load_encoding_images("images/")
```

```
cap = cv2.VideoCapture(0)  # Adjust index as needed


while True:
    ret, frame =
    cap.read()if
    not ret:
        print("Error: Failed to capture
        frame.")break


    face_locations, face_names =
    sfr.detect_known_faces(frame)for face_loc, name in
    zip(face_locations, face_names):
        y1, x1, y2, x2 = face_loc[0], face_loc[1], face_loc[2], face_loc[3]
        cv2.putText(frame, name, (x1, y1 - 10), cv2.FONT_HERSHEY_DUPLEX, 1, (0, 0, 200), 2)
        cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 200), 2)


    cv2.imshow("Frame"
    , frame)key =
    cv2.waitKey(1)

cap.release()
cv2.destroyAll
Windows()
```

Executed output:

Input

Output

Fig.14.1.1. Executed Output

15. CONCLUSION

In conclusion, the project "Face Recognition using OpenCV Python" has successfully demonstrated the capability to accurately recognize faces in real-time using OpenCV and Python. Through meticulous dataset preparation, preprocessing techniques, and model training, we have achieved commendable performance metrics, making the system viable for practical applications such as security systems and personalized user experiences.

While the project lays a strong foundation, there are avenues for further enhancement. Future iterations could explore advanced deep learning methods for improved accuracy and efficiency. Moreover, extending the project's compatibility to various platforms would broaden its applicability.

In essence, this project showcases the potential of open-source technologies to tackle complex challenges effectively. With continued refinement and innovation, it promises to contribute significantly to fields requiring reliable face recognition systems.

16. REFERENCE

1. OpenCV (Open Source Computer Vision Library): https://opencv.org/
2. Dlib Library: http://dlib.net/
3. Face Recognition with OpenCV, Python, and Deep Learning: https://www.pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and- deep-learning/
4. Face Recognition using Deep Learning with Keras: https://machinelearningmastery.com/how- to-perform-face-recognition-with-vggface2-convolutional-neural-network-in-keras/
5. FaceNet: A Unified Embedding for Face Recognition and Clustering: https://arxiv.org/abs/1503.03832
6. MTCNN (Multi-Task Cascaded Convolutional Networks) for Face Detection:https://arxiv.org/abs/1604.02878
7. DeepFace: Closing the Gap to Human-Level Performance in Face Verification: https://research.fb.com/wp-content/uploads/2016/11/deepface-closing-the-gap-to-human-level- performance-in-face-verification.pdf
8. VGGFace2: A dataset for face recognition: https://arxiv.org/abs/1710.08092
9. LBPH (Local Binary Patterns Histograms) Face Recognizer:https://docs.opencv.org/3.4/da/d60/tutorial_ face_main.html
10. Microsoft Face API Documentation: https://docs.microsoft.com/en- us/azure/cognitive-services/face/overview.