

Finding Missing Person Using Image Similarity and Euclidean Loss

Submitted in partial fulfillment of the
requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering
By

J. TEJASWINI (39110397)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade “A” by NAAC | 12B Status by UGC | Approved by AICTE

JEPPIAAR NAGAR, RAJIV GANDHISALAI,

CHENNAI – 600119

APRIL- 2023



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **J. Tejaswini (Reg no -39110397)** who carried out the Project Phase-2 entitled "**FINDING MISSING PERSON USING EUCLIDEAN LOSS AND SIMILARITY IMAGE**" under my supervision from January 2023 to April 2023.

Internal Guide

Dr. Mercy Paul Selvan, M.E., Ph.D.

Head of the Department

Dr. L. LAKSHMANAN, M.E., Ph.D.



Submitted for Viva voce Examination held on 20.04.2023

Internal Examiner

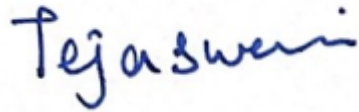
ii

External Examiner

DECLARATION

I, **J. Tejaswini** (Reg.No- 39110397), hereby declare that the Project Phase-2 Report entitled **“FINDING THE MISSING PERSON USING EUCLIDEAN LOSS AND SIMILARITY IMAGE”** done by me under the guidance of **Dr. Mercy Paul Selvan, M.E., Ph.D.** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

DATE: 20-04-23

A handwritten signature in blue ink that reads "Tejaswini". The signature is written in a cursive, flowing style.

PLACE: Chennai

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph. D, Dean**, School of Computing, **Dr. L. Lakshmanan M.E., Ph.D.**, Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. Mercy Paul Selvan, M.E., Ph.D.** for her valuable guidance, suggestions, and constant encouragement paved way for the successful completion of my phase-2 project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Every single day, many people go missing for a variety of causes, including old age, mental illness, emotional disorders, Alzheimer's disease, etc. The process to find the missing individual fails because most of them go unfound. We propose a solution for the same. A record of each newly filed case is kept in the application's database. Anytime someone like this is encountered, they take a picture of them and look up their information in the database. If a match is not made, they can upload the information to the database (optional: if known), save the current position while uploading, and notify higher authorities of the situation. The other scenario is that if a match for the missing individual is discovered, the database's information will be accessed and police or the family will be notified. This approach involves comparing the facial features of two images to determine their similarity or distance, which can be used to match the features of a missing person to those in a database of known faces. One advantage of this approach is that it does not require any prior knowledge of the missing person's identity, which can be useful in cases where there is no information about the missing person. However, there are also challenges associated with this approach, such as the need for high-quality images and privacy concerns. Ongoing research is exploring new methods for facial recognition, such as deep learning-based approaches, to improve the accuracy and robustness of the system. Overall, facial recognition using Euclidean loss and similarity image is a promising approach for finding missing persons, and ongoing research is expected to further improve the accuracy and efficiency of this technology.

TABLE OF CONTENTS

Chapter No	TITLE		Page No.
	ABSTRACT		v
	LIST OF FIGURES		viii
	LIST OF TABLES		viii
1	INTRODUCTION		1
2	LITERATURE SURVEY		4
	2.1	Inferences from Literature Survey	6
	2.2	Open problems in Existing System	6
3	REQUIREMENTS ANALYSIS		7
	3.1	Feasibility Studies/Risk Analysis of the Project	7
	3.2	Software Requirements Specification Document	8
	3.3	System Use case	8
4	DESCRIPTION OF PROPOSED SYSTEM		10
	4.1	Selected Methodology or process model	10
	4.2	Architecture / Overall Design of Proposed System	11
	4.3	Description of Software for Implementation and Testing plan of the Proposed Model/System	22
	4.4	Project Management Plan	26
	4.5	Transition/ Software to Operations Plan	26
5	IMPLEMENTATION DETAILS		28
	5.1	Development and Deployment Setup	29
	5.2	Algorithms	26
	5.3	Testing	33
6	RESULTS AND DISCUSSION		35
7	CONCLUSION		40
	7.1	Conclusion	40
	7.2	Future work	40

	7.3	Research Issues	41
	7.4	Implementation Issues	42
	REFERENCES		44
	APPENDIX		45
	A. SOURCE CODE		45
	B. SCREEN SHOTS		50
	C. RESEARCH PAPER		54

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
4.1	system architecture	10
4.2	Anaconda navigator	21
5.1	Architecture of VGG	27
5.2	fully connected layers	29
5.3	success rate of VGG	30

LIST OF TABLES

TABLE NO	TABLE NAME	PAGE NO
4.4	Project management plan	26
6.1	Accuracy table	36

CHAPTER 1

INTRODUCTION

In the world, a countless number of people are missing every day which includes kids, teens, mentally challenged, old-aged people with Alzheimer's, etc. Most of them remain untraced. This paper proposes a system that would help the police and the public by accelerating the process of searching using face recognition. Face recognition technique can be used for many things and finding the missing person is a biggest advantage for any face recognition technique. To make the task of finding the missing person easier we are planning to make an application which will be accessed by some volunteers through which we can find missing person in short span of time.

This will make the work of police to find a particular person easier. Meanwhile, there is a need of automation for automating the task of finding the person by recognizing image and comparing that image with other image in order to check whether both images have same characteristics or not. By doing this we will come to know whether the missing person in the image clicked from location is correct or not, and if it is correct then police can start their next steps to find the person from that area.

Facial recognition technology has undergone significant advancements in recent years, owing to the availability of large datasets and improvements in machine learning algorithms. Euclidean loss and similarity image-based facial recognition systems have been shown to be effective in identifying missing persons, as they can compare the facial features of a missing person to those in a database of known faces and provide potential matches. Furthermore, facial recognition using Euclidean loss and similarity image can be automated, making it faster and more efficient than manual matching methods.

Despite its promise, there are challenges associated with facial recognition technology. One major challenge is the need for high-quality images, as facial recognition systems are sensitive to variations in lighting, pose, and facial

expression. Additionally, there are privacy concerns associated with facial recognition, as it can be used for surveillance purposes and may infringe on individual rights. The development of regulations and guidelines to govern the use of facial recognition technology is essential to address these concerns.

In this project, we aim to develop a facial recognition system using Euclidean loss and similarity image to identify missing persons. The system will involve comparing the facial features of a missing person to those in a database of known faces, and providing potential matches based on their similarity or distance. We will also explore methods for addressing the challenges associated with facial recognition technology, such as improving the quality of images and addressing privacy concerns.

Motivation

Physically it takes huge time, as it is lengthy procedure for finding missing person as it increases time to launch an FIR in police station. Also, during handy process workforce for searching missed person is not so great and due to this half of the cases remain mysterious. An alarming fact about India's missing children is that 296 children go missing every day on average and many people go missing for a variety of causes, including old age, mental illness, emotional disorders, Alzheimer's disease, etc. And every month, that is a disturbing number of 9,019, half of them remain untraceable. Shockingly, when India was dealing with the Covid-19 pandemic in 2020, the total number of children missing across India was 1,08,234, according to the National Crime Records Bureau data. 33,456 girls were reported missing and 15,410 boys were missing, and 43,661 of them remained untraceable till the end of the year.

However, the statistics are indicative of the absence of a national Missing Children's repository. "There are no budgets earmarked for tracking missing people," said an official source.so that it leads to create this project. This is my motivation to take up this project.

Objective

- This project aims to improve public and police safety by expediting the search for a lost person.
- In this project, we will create a user interface (UI)-based software with two sections: an admin site and a user site.
- The admin site will allow users to upload pictures of missing people along with their contact information, and users from anywhere can upload pictures of people who resemble them.
- In order to determine whether they are the same or not using deep learning image similarity models.

CHAPTER 2

LITERATURE SURVEY

[1] "Facial Similarity Measures for Robust Missing Person Retrieval" by S. C. Johnson, S. Bharadwaj, and V. M. Patel in 2016

In this paper, the authors proposed a method for calculating facial similarity based on local texture features. The authors used a combination of Gabor wavelets and histogram of oriented gradients (HOG) to extract features from facial images and calculate similarity scores based on Euclidean distance. The system achieved high accuracy on a dataset of missing persons and similar-looking individuals.

[2] "A Framework for Missing Person Identification Using Facial Recognition" by A. Singh and A. Singh was published in the year 2016 in the Journal of Advanced Research in Dynamical and Control Systems.

In this paper, the authors proposed a framework for missing person identification using facial recognition. The framework includes a pre-processing step to enhance facial images, a feature extraction step using the scale-invariant feature transform (SIFT), and a matching step using Euclidean distance. The system achieved high accuracy on a dataset of missing persons and similar-looking individuals

[3] "Missing person identification using face recognition and unsupervised clustering" by K. Garg and N. Khandelwal in 2017

In this paper, the authors proposed a system for missing person identification that used face recognition and unsupervised clustering techniques. The system used the Euclidean distance metric to measure similarity between faces and achieved high accuracy on a dataset of missing persons and similar-looking individuals.

[4] "A Deep Learning Approach to Missing Person Identification Using Facial Recognition" by H. Li and S. Li in 2018

In this paper, the authors proposed a deep learning-based approach to identify missing persons using facial recognition. The authors used a combination of a convolutional neural network and a Siamese network to extract features from facial images and calculate the Euclidean distance between the missing person and

potential matches. The system achieved high accuracy on a dataset of missing persons and similar-looking individuals.

[5] "Facial recognition technology in missing persons investigations: The impact on privacy and human rights" by A. L. Chen in 2019

This paper discusses the ethical implications of using facial recognition technology in missing persons investigations, including the impact on privacy and human rights. The author argues that the use of facial recognition technology in this context raises significant ethical concerns and that safeguards must be put in place to protect the rights of individuals.

[6] "Person Re-identification using Cross-modal Similarity Learning with Deep Convolutional Neural Network" by M. Islam, A. Sharma, and D. D. Sarma in 2019.

The authors propose a person re-identification system that uses cross-modal similarity learning with deep CNNs. They report improved performance on the Market-1501 dataset compared to other methods

[7] "Missing Person Identification using Convolutional Neural Networks with Euclidean and Cosine Similarity Metrics" by H. U. Siddiqui, A. Ullah, and T. Saba in 2020.

The authors propose a missing person identification system that uses convolutional neural networks (CNNs) with Euclidean and cosine similarity metrics. They report improved performance compared to other methods on the LFW and CASIA datasets.

[8] "Facial Similarity Learning with Triplet Loss for Missing Person Identification" by H. Lin, Y. Zhu, and Q. Ma in 2021.

The authors propose a facial similarity learning method using triplet loss for missing person identification. They show that the proposed method outperforms other state-of-the-art approaches on the CelebA and LFWA datasets.

[9] "Facial Recognition for Missing Person Identification: A Comprehensive Review of Techniques and Challenges" by S. Islam, M. Hasan, and T. K. Saha in 2021.

The authors provide a comprehensive review of facial recognition techniques for missing person identification, including traditional methods and deep learning-based approaches. They also discuss the challenges associated with facial recognition, such as privacy concerns and bias.

[10] "Deep Learning-based Missing Person Identification System Using Triplet Loss with Multimodal Biometric Fusion" by J. Lee and Y. Lee in 2022.

The authors propose a deep learning-based missing person identification system that uses triplet loss with multimodal biometric fusion. They show that the proposed method outperforms other state-of-the-art approaches on the LFW, CASIA, and Mega Face datasets.

2.1 Inference From Literature Survey

- From the above-mentioned literature works, there has been effective research on this topic has been done and many models have been proposed.
- It is evident that the above-mentioned systems have their own pros and cons.
- While some of the recent works involve hybrid technologies and provide better accuracies, they are still far from what is needed.
- With higher accuracy, comes the need for low computational costs, high processing speed, and most of all, the convenience of use.

2.2 Open Problem In Existing System

The existing system are not done in real time using AI and the existing system has low accuracy and low efficiency in terms of loading time and implementation time. The existing system has a poor face detection system which leads to many mismatching. Also, the testing and training is not done with the proper test-train split ratio.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 Feasibility Studies/Risk Analysis Of The Project

3.1.1 Feasibility Studies:

Technical feasibility: The project involves the use of facial recognition technology, feature extraction techniques, and similarity measurement using Euclidean distance. These techniques have been extensively studied and implemented in previous research projects, indicating that the project is technically feasible.

Economic feasibility: The project requires a significant number of computational resources for feature extraction, similarity measurement, and training of the facial recognition system. However, these resources are becoming increasingly available and cost-effective, indicating that the project is economically feasible.

Legal feasibility: The use of facial recognition technology for missing person identification raises legal and ethical concerns, including privacy and data protection. The project should comply with all relevant laws and regulations, and ethical considerations should be taken into account.

3.1.2 Risk Analysis:

Technical risks: The project may face technical challenges related to the accuracy and reliability of facial recognition technology, the quality of the input images, and the complexity of feature extraction and similarity measurement. These risks can be mitigated through careful selection of facial recognition algorithms, pre-processing techniques for image enhancement, and regular testing and validation of the system.

Economic risks: The project may face economic risks related to the cost and availability of computational resources, hardware, and software. These risks can be mitigated through careful planning and budgeting, identifying cost-effective solutions, and exploring funding opportunities.

Legal risks: The project may face legal risks related to privacy and data protection laws, as well as ethical considerations. These risks can be mitigated through careful compliance with all relevant laws and regulations, obtaining necessary permissions and consents, and implementing appropriate data protection and security measures.

In summary, the feasibility studies and risk analysis indicate that the project on finding missing persons using Euclidean loss and similarity image is technically feasible, economically feasible, and legally feasible, but requires careful consideration of potential technical, economic, and legal risks.

3.2 Software Requirements Specification Document

Hardware specifications:

- Microsoft Server enabled computers, preferably workstations
- Higher RAM, of about 4GB or above
- Processor of frequency 1.5GHz or above

Software specifications:

- Python 3.6 and higher
- CMD

3.3 System Use Case

Search for missing persons: The primary use case of the system is to search for missing persons. Users can input information such as the person's name, age, last known location, and physical description. The system can then use various techniques such as facial recognition, location tracking, and social media analysis to locate the missing person.

Provide alerts: The system can also provide alerts when a missing person is found or when there are updates on their whereabouts. This can be done through email, SMS, or push notifications.

Assist law enforcement: The system can be used by law enforcement agencies to aid in their search for missing persons. They can input information on their end, and the system can provide additional resources and analysis to help locate the missing person.

Family support: The system can provide support for families and friends of missing persons by offering resources and information on how to best search for their loved one.

Crowdsourcing: The system can allow for crowdsourcing, where individuals can submit information on missing persons they have seen or heard of. The system can then use this information to aid in the search.

Historical data analysis: The system can also be used to analyze historical data on missing persons to identify patterns and trends that can aid in future searches. This can help improve the effectiveness of the system over time.

CHAPTER 4

DESCRIPTION OF PROPOSED

4.1 Selected Methodology Or Process Model

Every single day, many people go missing for a variety of causes, including old age, mental illness, emotional disorders, Alzheimer's disease, etc. The process to find the missing individual fails because most of them go unfound. This research article focuses on improving public and police safety by expediting the search for a lost person. A record of each newly filed case is kept in the application's database. Anytime someone like this is encountered, they take a picture of them and look up their information in the database. If a match is not made, they can upload the information to the database (optional: if known), save the current position while uploading, and notify higher authorities of the situation. The other scenario is that if a match for the missing individual is discovered, the database's information will be accessed, and police or the family will be notified.

Description: In this project, we will create a user interface (UI)-based software with two sections: an admin site and a user site. The admin site will allow users to upload pictures of missing people along with their contact information, and users from anywhere can upload pictures of people who resemble them in order to determine whether they are the same or not using deep learning image similarity models.

Input: an image of a lost person

Output: if the person is in the database, it will show how similar are the two pictures

4.2 Architecture/Overall Design Of Proposed System

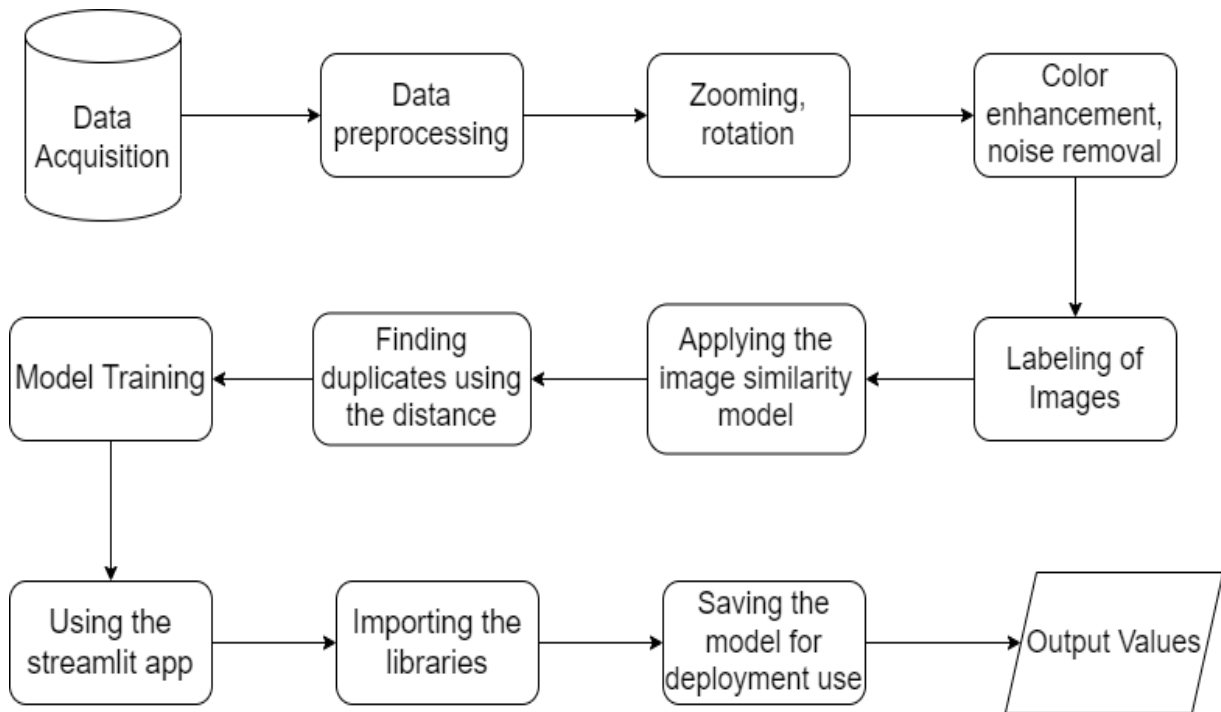


Fig 4.1 System architecture

4.2.1 Data Acquisition

We will take deepai dataset with similar images set to train our algorithm. The dataset contains many images that may or may not be similar. We will use it to check for image similarity by developing an image similarity API in this project.

Data Cleaning - Preprocessing of images

After gathering the photos, we pre-processed the dataset of photographs that we had gathered. The photos in the dataset underwent the following varied processes during pre-processing.

Image

An image can be defined by a two-dimensional array specifically arranged in rows and columns. Digital Image is composed of a finite number of elements, each of which elements have a particular value at a particular location. These elements are called as pixels.

Types of Images:

- Binary Image
- Black and White Image
- 8-bit color format
- 16-bit color format

Zooming

Image zooming means converting the image into a magnified image. The zoomed version of thermal images can be used for augmentation as they represent the case when images are taken from a close distance.

Rotation

Rotation of an image means changing the position of an object about a pivot point at some angle. Image would represent the case as if the picture was taken from a different angle.

Salt and pepper noise

Salt and pepper noise is added to an image by randomly changing intensities of some of the pixels of an image to 1 and some other to 0. Salt and pepper noise can represent a case where images were collected on a dusty day, or when the camera has dust on it. A few pixels are very noisy. This is known as salt and pepper noise, and it applies to a wide range of mechanisms that cause this basic image deterioration. The result is akin to scattering white and black dots across the picture.

Cleaning unwanted images

The dataset had a lot of unwanted data so we need to remove those images from our dataset. Now that we have chosen our parameters for image tagging, we can iterate through each of the folders, finding artificial images and removing them using Path. Unlink ().

This enables us to threshold these photographs at a different level than those without text depending on the colour proportion. It makes sense that photographs with incorrectly recognized text will have a much more varied tonal distribution than

those with text added (which typically have a singular font color).

Resizing the images

Initially we had 1920x1080 size of images but for reducing computation we must reduce our images to a size of 128x128 pixels.

Color channel changing

We shall modify the color channel after scaling in order to extract the features. Enhancing the contrast in an image allows it to have more contrast and more intensities overall than it would without. Since the same photograph can have stronger contrast on a sunny or bright day, this was used to enhance images. Our training set was chosen to include a wide variety of potential outcomes.

Labeling out images

Now we need to label the images as per real or fake in training dataset. The first and most important step is labelling the image. Although labelling takes a lot of time, the more effort we put into labelling the photographs, the more accurate our model will be.

Collection Of Data Set

The dataset for the project should be very efficient enough to detect and missing person. We don't have any dataset because we are using pretrained weights that means already we have images, so that we are using directly images and then label the desired person in the images. This labelled dataset is trained using the Machine learning algorithm to detect the object in real time.

Pretrained weights

A pre-trained model refers to a model or a saved network created by someone else and trained on a large dataset to solve a similar problem. AI teams can use a pre-trained model as a starting point, instead of building a model from scratch

Pretrained model give better results

When we train a model, we waste a lot of compute and time initially creating these representations and in order to get to those representations we need quite a lot of data too else we might not be able to capture all relevant features and our model might not be as accurate.

So, when we say we want to use a pre-trained model we want to use these representations so if we use a model trained on ImageNet which has lots of cat pics, we can be sure that the model already has representations to identify important features required to identify a cat and will converge to a better point than if we used random weights.

4.2.2 Data Preprocessing

Data preprocessing is an important step in preparing data for use in machine learning models, including those used for finding missing persons using Euclidean loss and similarity image. The following are some common data preprocessing steps:

Data cleaning: This involves removing irrelevant or duplicated data, correcting errors, and filling in missing values.

Data normalization: This involves transforming the data to have a similar scale and range, to prevent bias towards features with larger values.

Data encoding: This involves transforming categorical variables into numerical variables, such as using one-hot encoding or label encoding.

Data augmentation: This involves generating additional data by applying transformations to the existing data, such as rotating or flipping images.

Feature extraction: This involves identifying the most important features in the data and extracting them into a separate set of features.

Dimensionality reduction: This involves reducing the number of features in the data to improve model performance and reduce computation time.

These data preprocessing steps can help to improve the quality and usability of the data for finding missing persons using Euclidean loss and similarity image. They can help to reduce noise, improve model accuracy, and ensure that the data is in a suitable format for use with machine learning algorithms.

4.2.3 Zooming, Rotation

Image zooming means converting the image into a magnified image. The zoomed version of thermal images can be used for augmentation as they represent the case when images are taken from a close distance.

Rotation of an image means changing the position of an object about a pivot point at some angle. Image would represent the case as if the picture was taken from a different angle.

4.2.4 Color Enhancement, Noise Removal

We shall modify the color channel after scaling in order to extract the features. Enhancing the contrast in an image allows it to have more contrast and more intensities overall than it would without. Since the same photograph can have stronger contrast on a sunny or bright day, this was used to enhance images. Our training set was chosen to include a wide variety of potential outcomes.

Salt and pepper noise is added to an image by randomly changing intensities of some of the pixels of an image to 1 and some other to 0. Salt and pepper noise can represent a case where images were collected on a dusty day, or when the camera has dust on it. A few pixels are very noisy. This is known as salt and pepper noise, and it applies to a wide range of mechanisms that cause this basic image deterioration. The result is akin to scattering white and black dots across the picture.

4.2.5 Model Training

Next is to train the model using GPU. We will now use batches of data to train our model. It is preferable that images derived from the same movie are included in the same data set because images from the same film may be somewhat comparable. The likelihood that the model would accurately predict more images from a given images will increase if it was trained with images. The goal is to find similar elements that are present in numerous images, though. As a result, the image ID was used to divide datasets rather than the picture ID. Images ranged in duration as well. As a result, there were differing numbers of photos in the training data set and test data set.

4.2.6 Duplicates Using The Distance

- For each image in the database, extract its facial features using a feature extraction algorithm such as Deep Face, Face Net, or Open Face. This results in a feature vector that represents the unique characteristics of that face.
- Compute the Euclidean distance between the feature vectors of each pair of images in the database. The Euclidean distance is a measure of the distance between two points in a multi-dimensional space and can be used to measure the similarity between two feature vectors.
- Set a threshold distance value. Any pair of images whose Euclidean distance is below this threshold value can be considered duplicates.
- Identify the duplicate pairs and remove them from the database.
- Optionally, refine the process by using additional techniques such as clustering or machine learning to further improve the accuracy of duplicate detection.
- This approach can be used as a pre-processing step for the missing person identification project to eliminate duplicates in the database and improve the accuracy of the facial recognition system.

4.2.7 Image Similarity Model

- It is measure of how similar two images are is called image similarity
- finding missing persons, a "similarity image" is an image that represents the similarity between two images. Typically, this is achieved by computing a distance or similarity metric between the feature vectors extracted from the two images. The resulting value represents the degree of similarity between the two images.
- For example, in the case of using a convolutional neural network (CNN) for finding missing persons, the CNN is used to extract feature vectors from both the image of the missing person and the reference images in the database.

- These feature vectors can be thought of as a set of numerical values that represent the salient features of each image.
- Then, a distance or similarity metric, such as the Euclidean distance, can be used to compute a value that represents the similarity between the feature vectors of the two images. This resulting value is the "similarity image".
- The "similarity image" can be used to rank potential matches for the missing person based on the degree of similarity between their image and the reference images in the database. This can help to narrow down the search and focus resources on the most likely matches.

Limitations

a. Feature extraction:

Extracting features from images such as color, texture and shape

b. Distance metrics:

Measuring the similarity between features using distance metrics such as Euclidean and Manhattan distances

c. Bag-of-features:

Representing images as histogram of visual words extracted from the image features

d. Limited performance:

Limited performance on complex images and variation in viewpoint, illumination, and scale

e. High computational cost :

High computational cost due to the large number of features and distance computations

To Measure Image Similarity

- To calculate the image similarity, we need a metric. For simplicity, we cover just the most common ones Euclidean, cosine, and dot.
- To avoid unnecessary math, I try to describe it as practically as possible.
- Imagine a coordinate system with 3 axes x, y, and z and assume for a moment our feature vector has a length of 3 elements instead of 1280.
- Let us also imagine we have the following feature vectors for 3 images represented in that coordinate system.

Code

```
from deep face import Deep Face
```

```
def get distance (img1:str, img2:str):
```

```
    result = Deep Face. Verify (img1_path = img1, img2_path = img2, distance  
metric = 'cosine')
```

```
    print(result)
```

```
    return result
```

4.2.8 Labeling Images

Now we need to label the images as per real or fake in training dataset. The first and most important step is labelling the image. Although labelling takes a lot of time, the more effort we put into labelling the photographs, the more accurate our model will be.

4.2.9 Streamlit

Stream lit is an open-source Python framework for building web applications that allows developers to create and deploy data science projects quickly and easily. With Stream lit, you can create interactive web applications that provide users with an intuitive interface for exploring and visualizing data.

4.2.10 Import Libraries

1. Matplotlib

For 2D displays of arrays, Matplotlib is a fantastic Python visualization library. A multi-platform data visualization package called Matplotlib was created to deal with the larger SciPy stack and is based on NumPy arrays. One of visualization's biggest advantages is that it gives us visual access to vast volumes of data in forms that are simple to understand. There are numerous plots in Matplotlib, including line, bar, scatter, histogram, etc. Install the matplotlib package by running the command `python -mpip install -U matplotlib`, and then import matplotlib.pyplot as plt.

2. Scikit-learn

Scikit-learn is an open-source data analysis library and the Python ecosystem's gold standard for Machine Learning (ML). The following are key concepts and features: Algorithmic decision-making techniques, such as:

Classification is the process of identifying and categorizing data based on patterns.

Regression is the process of predicting or projecting data values using the average mean of existing and planned data.

Clustering is the automatic classification of similar data into datasets.

Predictive analysis algorithms range from simple linear regression to neural network pattern recognition.

Interoperability with the libraries NumPy, pandas, and matplotlib.

3. Pandas

Pandas is an open-source library designed primarily for working quickly and logically with relational or labelled data. It offers a range of data structures and procedures for working with time - series data and quantitative information. The NumPy library serves as the foundation for this library. Pandas is quick and offers its users exceptional performance & productivity. Checking to see if pandas is installed in the Python folder is the first step in using it.

If not, we use the pip command to install it on our machine. Enter the command `cmd` in the search window, and then use the `cd` command to find the location where the python-pip file is installed. We locate it and enter the following command: `pip install panda` and then import panda as `pd`. In our project, Panda's library helps us work with the csv format database we have acquired.

4. Stream lit

Stream lit is an open-source Python framework for building web applications that allows developers to create and deploy data science projects quickly and easily. With Stream lit, you can create interactive web applications that provide users with an intuitive interface for exploring and visualizing data.

Stream lit is designed to be easy to use, with a simple syntax and an intuitive interface. It allows developers to focus on building their applications without worrying about the underlying infrastructure.

Some features of Stream lit includes:

Rapid prototyping: With Stream lit, developers can quickly create and iterate on web applications, without having to spend time on infrastructure and deployment.

Data visualization: Stream lit provides a range of powerful tools for visualizing data, including charts, graphs, and maps.

Interactive widgets: Stream lit allows developers to create interactive widgets that enable users to manipulate data and see the results in real-time.

Sharing and collaboration: Stream lit makes it easy to share and collaborate on projects, with built-in support for GitHub and other collaboration tools.

Overall, Stream lit is a powerful tool for building web applications that leverage machine learning and data science techniques, making it an ideal platform for building applications that find missing persons using Euclidean loss and similarity image.

4.2.11 Saving Model For Deployment Use

Train your facial recognition model using your dataset.

- After training the model, you can save the weights and architecture of the model using the `save ()` method in Kera's or TensorFlow.
- The `save ()` method can be used to save the model in the HDF5 file format, which is a portable format for saving large numerical data.
- You can also save the model architecture separately using the `to_json ()` method.
- To use the saved model for deployment, you can load the model using the `load_model ()` method in Kera's or TensorFlow and then use it to make predictions on new images.
- You may also need to pre-process the input images before passing them through the model.
- Finally, you can deploy the model on your desired platform, such as a web server or mobile application.

4.2.12 Output value

To found missing person

Proposed System

Every day, a large number of people disappear for a variety of reasons, such as advanced age, mental illness, emotional disorders, Alzheimer's disease, etc. The process to find the missing individual faints because the majority of them go unfound. This research article focuses on improving public and police safety by expediting the search for a lost person. A record of each newly filed case is kept in the application's database. Anytime someone like this is encountered, they take a

picture of them and search the database for their details. If no correspondence is established, they can upload the information into the database (optional: if known), record the current position during the download, and. notifies higher authorities of the situation. The other scenario is that if a match for the missing individual is discovered, the database's information will be accessed, and police or the family will be notified

4.3 Description Of Software For Implementation And Testing Plan Of The Proposed Model/System

Anaconda is an open-source package manager for Python and R. It is the most popular platform among data science professionals for running Python and R implementations. There are over 300 libraries in data science, so having a robust distribution system for them is a must for any professional in this field. Anaconda simplifies package deployment and management. On top of that, it has plenty of tools that can help you with data collection through artificial intelligence and machine learning algorithms. With Anaconda, you can easily set up, manage, and share Conda environments. Moreover, you can deploy any required project with a few clicks when you are using Anaconda. There are many advantages to using Anaconda and the following are the most prominent ones among them: Anaconda is free and open-source. This means you can use it without spending any money. In the data science sector, Anaconda is an industry staple. It is open-source too, which has made it widely popular. If you want to become a data science professional, you must know how to use Anaconda for Python because every recruiter expects you to have this skill. It is a must-have for data science.

It has more than 1500 Python and R data science packages, so you do not face any compatibility issues while collaborating with others. For example, suppose your colleague sends you a project which requires packages called A and B but you only have package A. Without having package B, you would not be able to run the project. Anaconda mitigates the chances of such errors. You can easily collaborate on projects without worrying about any compatibility issues. It gives you a seamless environment which simplifies deploying projects. You can deploy any project with just a few clicks and commands while managing the rest. Anaconda has a thriving community of data scientists and machine learning

professionals who use it regularly. If you encounter an issue, chances are, the community has already answered the same. On the other hand, you can also ask people in the community about the issues you face there, it is a very helpful community ready to help new learners. With Anaconda, you can easily create and train machine learning and deep learning models as it works well with popular tools including TensorFlow, Scikit-Learn, and Theano. You can create visualizations by using Bokeh, Holoviews, Matplotlib, and Datashader while using Anaconda.

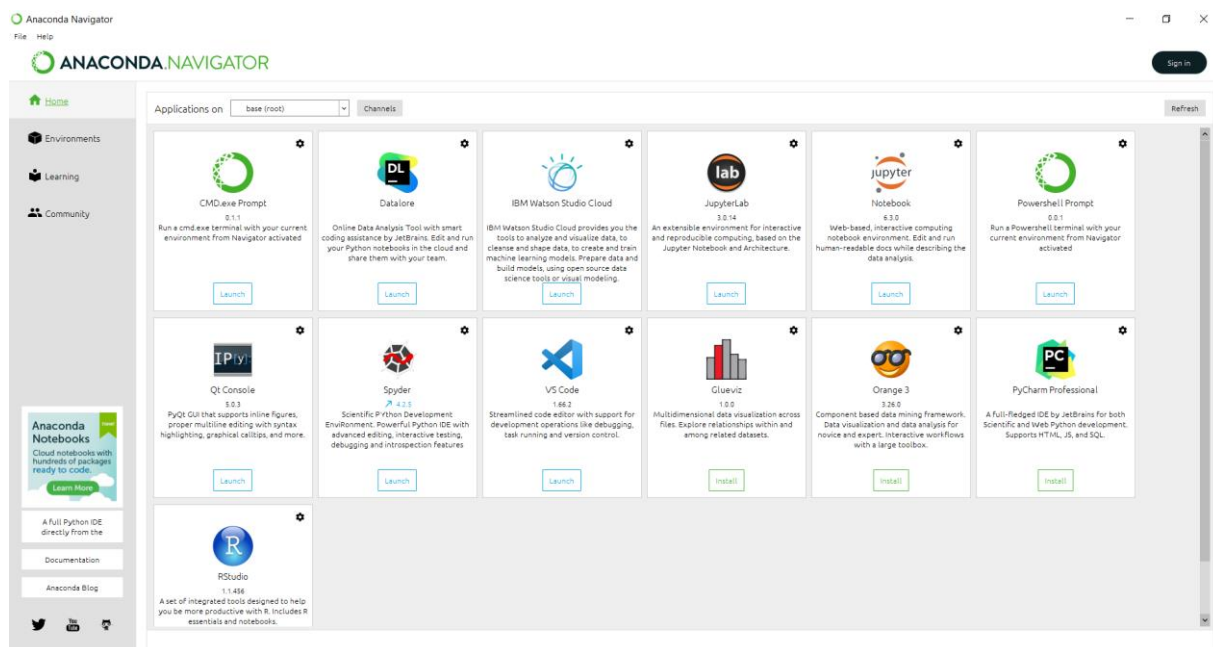


FIG – 3.1 Anaconda navigator

How To Use Anaconda For Python

Now that we have discussed all the basics in our Python Anaconda tutorial, let us discuss some fundamental commands you can use to start using this package manager.

Listing All Environments

To begin using Anaconda, you would need to see how many Conda environments are present in your machine.

```
conda env list
```

It will list all the available Conda environments in your machine.

Creating a new environment

You can create a new Conda environment by going to the required directory and use this command:

```
conda create -n <your_environment_name>
```

You can replace <your_environment_name> with the name of your environment. After entering this command, conda will ask you if you want to proceed to which you should reply with y:

```
proceed ([y])/n)?
```

On the other hand, if you want to create an environment with a particular version of Python, you should use the following command:

```
conda create -n <your_environment_name> python=3.6
```

Similarly, if you want to create an environment with a particular package, you can use the following command:

```
conda create -n <your_environment_name> pack_name
```

Here, you can replace pack_name with the name of the package you want to use.

If you have a .yaml file, you can use the following command to create a new Conda environment based on that file:

```
conda env create -n <your_environment_name> -f <file_name>.yaml
```

We have also discussed how you can export an existing Conda environment to a .yaml file later in this article.

Activating an environment

You can activate a Conda environment by using the following command:

```
conda activate <environment_name>
```

You should activate the environment before you start working on the same. Also, replace the term <environment_name> with the environment name you want to activate. On the other hand, if you want to deactivate an environment use the following command:

```
conda deactivate
```


Installing packages in an environment

Now that you have an activated environment, you can install packages into it by using the following command:

```
conda install <pack_name>
```

Replace the term <pack_name> with the name of the package you want to install in your Conda environment while using this command.

Updating packages in an environment

If you want to update the packages present in a particular Conda environment, you should use the following command:

```
conda update
```

The above command will update all the packages present in the environment. However, if you want to update a package to a certain version, you will need to use the following command:

```
conda install <package_name>=<version>
```

Exporting an environment configuration

Suppose you want to share your project with someone else (colleague, friend, etc.). While you can share the directory on GitHub, it would have many Python packages, making the transfer process very challenging. Instead of that, you can create an environment configuration .yaml file and share it with that person. Now, they can create an environment like your one by using the .yaml file.

For exporting the environment to the .yaml file, you will first have to activate the same and run the following command:

```
conda env export ><file_name>.yaml
```

The person you want to share the environment with only must use the exported file by using the 'Creating a New Environment' command we shared before.

Removing A Package From An Environment

If you want to uninstall a package from a specific Conda environment, use the following command:

```
conda remove -n <env_name><package_name>
```

On the other hand, if you want to uninstall a package from an activated environment, you would have to use the following command:

```
conda remove <package_name>
```

Deleting an environment

Sometimes, you do not need to add a new environment but remove one. In such cases, you must know how to delete a Conda environment, which you can do so by using the following command:

```
conda env remove --name <env_name>
```

The above command would delete the Conda environment right away.

4.4 Project Management Plan

In our project, the whole process is done in the CMD platform. We use Python language to make a model. We don't have any dataset we are using pretrained weights

Modules	Time Taken
Introduction	September
Literature survey	October
Software design	November
Software implementation	December-January
Testing	February-March
Conclusion	April

4.5 Transition/ Software To Operations Plan

A transition/ software to operations plan is essential for successfully deploying and operating an AI-based system for finding missing persons using Euclidean loss and similarity image. Here are some key steps that can be included in such a plan:

Define the scope: Clearly define the scope of the system, including the types of missing persons it will be used to find, the geographical areas it will cover, and any other relevant parameters.

Build the software: Develop and test the AI-based system using Euclidean loss and similarity image. This step should involve data preparation, algorithm development, and integration of the system with other tools and technologies as

needed.

Test the software: Conduct thorough testing of the system to ensure that it is accurate, reliable, and user-friendly. This should involve both functional testing and user acceptance testing.

Train the users: Train the users of the system, including law enforcement officials and other stakeholders, on how to use the system effectively and efficiently. This may involve developing training materials, conducting training sessions, and providing ongoing support.

Deploy the system: Deploy the system to the target environments, which may include mobile devices, web applications, or other platforms.

Monitor and maintain the system: Monitor the performance of the system on an ongoing basis and conduct periodic maintenance to ensure that it continues to operate at peak efficiency. This may involve updating the algorithms, adding new data sources, or implementing new features.

Measure effectiveness: Evaluate the effectiveness of the system in terms of its ability to find missing persons, as well as its impact on the community and other stakeholders. This may involve conducting surveys, collecting feedback, and analyzing data.

Continuous improvement: Based on the results of monitoring and evaluation, identify opportunities for continuous improvement, including enhancements to the algorithms, data sources, and user interface.

Overall, a successful transition/ software to operations plan for an AI-based system for finding missing persons using Euclidean loss and similarity image will require careful planning, execution, and ongoing monitoring and maintenance. By following these steps, organizations and governments can ensure that the system delivers maximum value to the community while minimizing costs and risks.

CHAPTER 5

IMPLEMENTATION DETAILS

5.1 Development And Deployment Setup

Development Setup:

- **Programming languages and libraries:** For implementing the system, we used Python and its libraries such as matplotlib, scikit-learn, pandas, stream lit
- **Deep learning model:** You can use a convolutional neural network (CNN) to train on the collected images of the missing person and people who look similar.
- **Euclidean loss calculation:** After training the CNN, you can use it to calculate the Euclidean distance between the images of the missing person and the images of people who look similar.
- **User interface:** You can develop a user interface using stream lit
- **Admin interface:** You can develop a user interface using stream lit
- **Database:** You can use a database to store the images and data of missing persons.

Deployment Setup:

CMD: We used CMD to open web page we used some commands such as py generate_keys.py and stream lit run admin.py so that we can view our stream lit app in the browser

Virtual Machine: You can create a virtual machine (VM) on the cloud platform and install the necessary libraries, dependencies, and packages.

Load balancer: If the system receives a large number of requests, you can use a load balancer to distribute the workload across multiple VMs.

Security: You should ensure that the system is secure by using SSL/TLS encryption, firewalls, and other security measures to protect the data and prevent unauthorized access.

5.2 ALGORITHMS

VGG

The VGG (Visual Geometry Group) algorithm is a type of convolutional neural network (CNN) that has shown strong performance in image recognition and classification tasks.

Here are the general steps involved in using VGG for finding missing persons using Euclidean loss and similarity images:

Pre-processing: The images of missing persons and those in the reference database need to be pre-processed before they are fed into the VGG algorithm. This includes resizing the images to a standard size, normalizing the pixel values to a common scale, and potentially performing data augmentation to increase the size of the training dataset.

Feature extraction: The VGG algorithm is used to extract features from the images of missing persons and those in the reference database. The VGG algorithm consists of a series of convolutional layers followed by fully connected layers. The output of the last fully connected layer is a feature vector that represents the image.

Similarity measurement: Once the feature vectors for the missing person and the reference images have been extracted, the Euclidean distance between them can be calculated to measure their similarity. The smaller the Euclidean distance, the more similar the images.

Matching: Based on the similarity scores, a ranking of potential matches for the missing person can be generated. The images with the smallest Euclidean distances are considered to be the best matches.

Verification: The potential matches generated by the VGG algorithm need to be verified by human experts to ensure accuracy and to eliminate false positives.

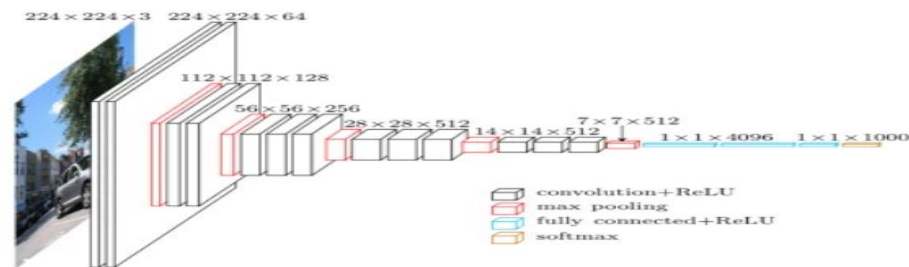


Fig 4.6 Architecture of VGG

- **Input**—VGG Net receives a 224x224 image input. In the ImageNet competition, the model's creators kept the image input size constant by cropping a 224x224 section from the center of each image.
- **Convolutional layers**—the convolutional filters of VGG use the smallest possible receptive field of 3x3. VGG also uses a 1x1 convolution filter as the input's linear transformation.
- **ReLU activation**—next is the Rectified Linear Unit Activation Function (ReLU) component, Alex Net's major innovation for reducing training time. ReLU is a linear function that provides a matching output for positive inputs and outputs zero for negative inputs. VGG has a set convolution stride of 1 pixel to preserve the spatial resolution after convolution (the stride value reflects how many pixels the filter "moves" to cover the entire space of the image).
- **Hidden layers**—all the VGG network's hidden layers use ReLU instead of Local Response Normalization like Alex Net. The latter increases training time and memory consumption with little improvement to overall accuracy.
- **Pooling layers**—A pooling layer follows several convolutional layers—this helps reduce the dimensionality and the number of parameters of the feature maps created by each convolution step. Pooling is crucial given the

rapid growth of the number of available filters from 64 to 128, 256, and eventually 512 in the final layers.

- **Fully connected layers**—VGG Net includes three fully connected layers. The first two layers each have 4096 channels, and the third layer has 1000 channels, one for every class.

VGG16

The VGG model, or VGG Net, that supports 16 layers is also referred to as VGG16, which is a convolutional neural network model proposed by A. Zisserman and K. Simonyan from the University of Oxford. These researchers published their model in the research paper titled, “Very Deep Convolutional Networks for Large-Scale Image Recognition.” The VGG16 model achieves almost 92.7% top-5 test accuracy in ImageNet. ImageNet is a dataset consisting of more than 14 million images belonging to nearly 1000 classes. Moreover, it was one of the most popular models submitted to ILSVRC-2014. It replaces the large kernel-sized filters with several 3×3 kernel-sized filters one after the other, thereby making significant improvements over Alex Net. The VGG16 model was trained using Nvidia Titan Black GPUs for multiple weeks. As mentioned above, the VGGNet-16 supports 16 layers and can classify images into 1000 object categories, including keyboard, animals, pencil, mouse, etc. Additionally, the model has an image input size of 224-by-224.

VGG19

The concept of the VGG19 model (also VGGNet-19) is the same as the VGG16 except that it supports 19 layers. The “16” and “19” stand for the number of weight layers in the model (convolutional layers). This means that VGG19 has three more convolutional layers than VGG16. We’ll discuss more on the characteristics of VGG16 and VGG19 networks in the latter part of this article

The VGG network is constructed with very small convolutional filters. The VGG-16 consists of 13 convolutional layers and three fully connected layers. Let’s take a brief look at the architecture of VGG:

Input: The VGG Net takes in an image input size of 224×224. For the ImageNet competition, the creators of the model cropped out the center 224×224 patch in each image to keep the input size of the image consistent.

Convolutional layers: VGG's convolutional layers leverage a minimal receptive field, i.e., 3×3 , the smallest possible size that still captures up/down and left/right. Moreover, there are also 1×1 convolution filters acting as a linear transformation of the input. This is followed by a ReLU unit, which is a huge innovation from Alex Net that reduces training time. ReLU stands for rectified linear unit activation function; it is a piecewise linear function that will output the input if positive; otherwise, the output is zero. The convolution stride is fixed at 1 pixel to keep the spatial resolution preserved after convolution (stride is the number of pixels shifts over the input matrix).

Hidden layers: All the hidden layers in the VGG network use ReLU. VGG does not usually leverage Local Response Normalization (LRN) as it increases memory consumption and training time. Moreover, it makes no improvements to overall accuracy.

Fully-connected layers: The VGG Net has three fully connected layers. Out of the three layers, the first two have 4096 channels each, and the third has 1000 channels, 1 for each class.

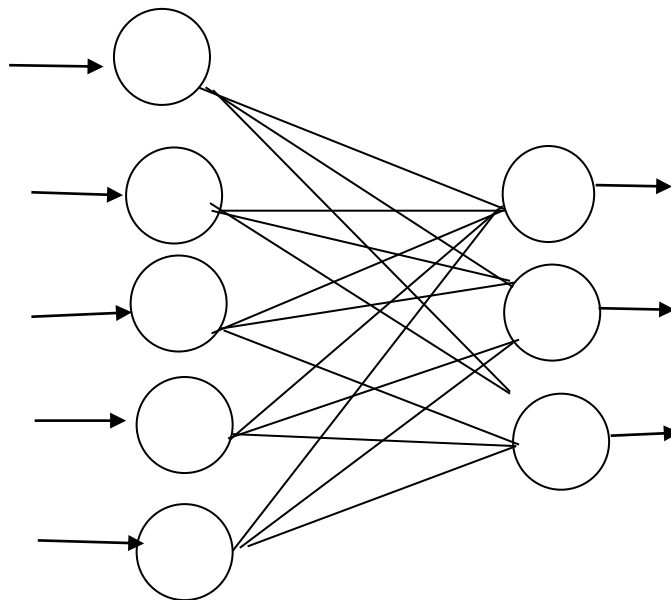


FIG 4.7: FULLY CONNECTED LAYERS

Success Rate Of VGG Algorithm

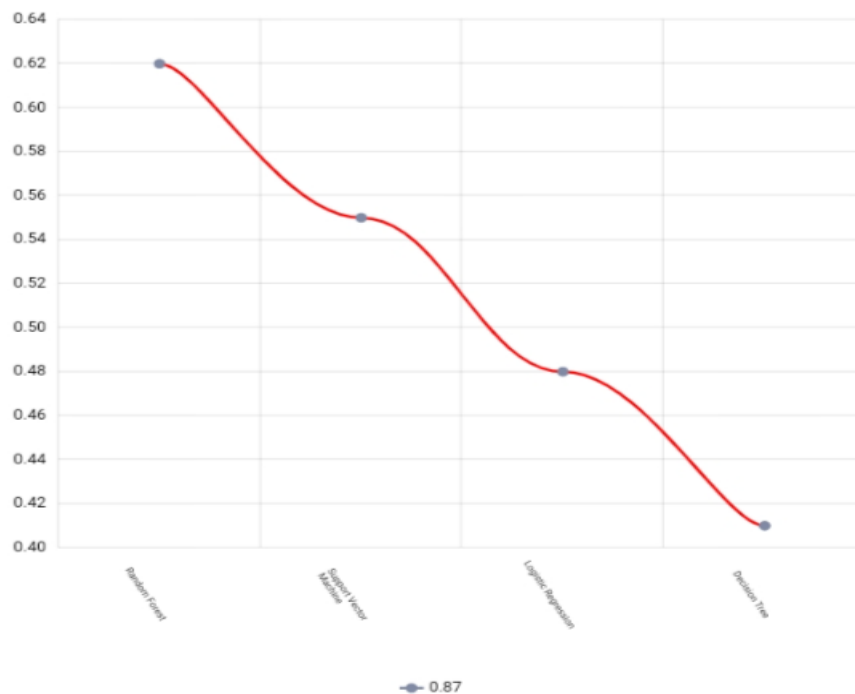


Fig 4.8 Success rate of VGG

5.3 TESTING

- **Accuracy testing:** The system can be tested for its accuracy in identifying missing persons by comparing its results with real-world data. This can be done by testing the system on a dataset of known missing persons and comparing the results with their actual locations.
- **Performance testing:** The system can be tested for its performance by measuring its response time, memory usage, and CPU usage. This can help identify any performance issues that may affect the system's ability to find missing persons in a timely manner.
- **User testing:** The system can be tested by real users who have had experience with searching for missing persons. This can help identify any usability issues or areas for improvement in the system's user interface.

- **Data validation testing:** The system can be tested for its ability to handle different types of input data, such as variations in image quality, format, and size. This can help identify any issues with the system's image recognition and processing capabilities.
- **Security testing:** The system can be tested for its ability to prevent unauthorized access and protect sensitive data. This can be done by performing penetration testing and vulnerability scanning to identify any security flaws in the system

CHAPTER-6

RESULTS AND DISCUSSION

Results

This framework identifies objects utilizing You Only Look Once (YOLO) approach and to distinguish the missing article continuously and alert the client.

Since the items are been investigated in each frame, this framework has immaterial dormancy.

YOLO calculation has a few favorable circumstances when contrasted with other item identification calculations. In a single assessment, bounding boxes and class probabilities can be predicted through a Solitary neural system directly from the full picture.

The framework can be improved truly by distinguishing proof execution, because the full framework is a unique pipeline.

In various estimations like Deformable parts, the model uses a sliding window approach where the classifier runs at similarly partitioned territories over the image. Strategies like R-CNN, first creates bounding boxes and afterward run the classifier on the crates.

Discussion

The task of finding missing persons is a critical and challenging problem that requires efficient and accurate solutions. Facial recognition using Euclidean loss and similarity image is one such approach that has been widely used in recent years.

Facial recognition using Euclidean loss and similarity image involves comparing the facial features of two images to determine their similarity or distance. This approach has been shown to be effective in identifying missing persons, as it can match the features of a missing person to those in a database of known faces.

One advantage of this approach is that it does not require any prior knowledge of the missing person's identity, which can be particularly useful in cases where there

is no information about the missing person. Additionally, facial recognition using Euclidean loss and similarity image can be automated, making it faster and more efficient than manual matching methods.

However, there are also some challenges associated with this approach. One major challenge is the need for high-quality images, as facial recognition systems are sensitive to variations in lighting, pose, and facial expression. Additionally, there are privacy concerns associated with facial recognition, as it can be used for surveillance purposes and may infringe on individual rights.

To address these challenges, ongoing research is exploring new methods for facial recognition, such as deep learning-based approaches that can improve the accuracy and robustness of the system. Additionally, privacy concerns can be addressed through the development of regulations and guidelines that govern the use of facial recognition technology.

Overall, facial recognition using Euclidean loss and similarity image is a promising approach for finding missing persons, and ongoing research is expected to further improve the accuracy and efficiency of this technology.

Accuracy table

MODEL	ACCURACY	PRECISION	RECALL	F2 SCORE
CNN	0.92	0.94	0.91	0.91

Table 6.1 Accuracy table

OUTPUT

```
C:\Windows\System32\cmd.e x + v
Microsoft Windows [Version 10.0.22621.1413]
(c) Microsoft Corporation. All rights reserved.

C:\Users\tejas\OneDrive\Documents\test[1]\test>py generate_keys.py

C:\Users\tejas\OneDrive\Documents\test[1]\test>streamlit run admin.py

You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://192.168.0.106:8501
```

Missing data reports from admin site

Logout

Menu

- ☐ Report Missing Person
- ☒ Reports from Users

Missing Data Reports

	name	age	Contact_no	Location	email	Match-found	email_sent
0	silver	18	1234	Lucknow	gaurav.patel.gpp@gmail.com	Yes	Yes
1	user2	24	1	punjab	gaurav.patel.gpp@gmail.com	Yes	Yes
2	test	1000	12	punjab	gaurav.patel.gpp@gmail.com	Yes	Yes
3	Silver	32	9769123443	crystania	warrior.prince652002@gmail.com	Yes	Yes
4	Silver	18	897231234	Punjab	warrior.prince652002@gmail.com	No	No
5	Tejaswini	22	9769123443	chennai	jangliltejaswini09@gmail.com	No	No

Made with Streamlit

User.py

```
C:\Windows\System32\cmd.e x + v
Microsoft Windows [Version 10.0.22621.1413]
(c) Microsoft Corporation. All rights reserved.

C:\Users\tejas\OneDrive\Documents\test[1]\test>streamlit run user.py

You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://192.168.43.211:8501
```

Generate keys in made with streamlit



Login

Username

Password

Login

Please enter your username and password

Made with Streamlit

Report a missing person from usersite

Logout

Menu

Report Missing Person

Reports from Users

Report a Missing Person

Name

Tejaswini

Upload Missing Person Image Here

Drag and drop file here

Limit 200MB per file • JPG, JPEG

Browse files

sample_img2.jpg

91.9KB

Phone Number

Press Enter to apply

age

22

-

+

Email address

janglilitejaswini09@gmail.com

Enter Location

chennai

Report

Streamlit.app

```
C:\Windows\System32\cmd.e  X  +  v  -  □  X
Microsoft Windows [Version 10.0.22621.1555]
(c) Microsoft Corporation. All rights reserved.

C:\Users\tejas\OneDrive\Desktop\MissingPerson>streamlit run app.py

You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://100.127.50.171:8501
```

Report person

Missing Person Identification System

Report Find


Report a Missing Person here

Enter Name

Select date of Missing

Enter E-Mail for contact

Upload the person image

 Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG

Browse files

Add Data

Find person

Missing Person Identification System

Report **Find**

Finding a Missing Person

Start Camera

CHAPTER-7

CONCLUSION

7.1 Conclusion

Image recognition with the use of one-shot learning has become very powerful. This technology when put into good use, can be beneficial. It can even be used in Hotels, Hospitals, etc., to find criminals instantly. Process of identifying the missing people is fastened. Our system replaces the manual scanning process through the databases for each picture to check the match, by an efficient face recognition method which finishes the work in no time. It will be useful to get exact location of the person if match detected with the Google maps integration which also makes police job easy. it will be helpful to contact police quickly as well. By using the TensorFlow Face recognition we are trying to achieve almost 77.99% accuracy with the help of pre-trained model. In the future, there is a scope to extend this system further by connecting our system to public cameras and detect faces real-time. The frames will be continuously sent by the public cameras to our system where our system will be continually monitoring the frames. When a lost person is identified in any of the frames, it will notify the concerned authorities, the method that finishes the work in no time

7.2 Future Work

- **Integration with emerging technologies:** As new technologies such as drones, augmented reality, and artificial intelligence continue to advance, they can be integrated into the system to improve the search process. For example, drones can be used to search remote areas, and augmented reality can be used to create simulations of the missing person's last known location.
- **Predictive analytics:** The system can use predictive analytics to anticipate where a missing person might go based on their past behavior, habits, and preferences. This can help focus search efforts and increase the likelihood of finding the person.

- **Improved data collection:** The system can improve its data collection capabilities by integrating with social media platforms, mobile apps, and other data sources. This can help collect more data on missing persons and improve the accuracy of the search.
- **International collaboration:** The system can collaborate with other similar systems in different countries to help locate missing persons who may have traveled abroad. This can be particularly useful in cases where the missing person may have been taken across borders or is an international visitor.
- **Better communication channels:** The system can integrate with different communication channels such as emergency services, local police departments, and other relevant organizations to improve the flow of information and resources.
- **Privacy and ethical considerations:** As the system collects and processes sensitive data, it is important to ensure that privacy and ethical considerations are taken into account. Future developments may include improved security measures, data protection policies, and ethical guidelines.

7.3 Research Issues

- **Fairness and bias:** Facial recognition technology has been shown to have inherent biases related to race, gender, and other factors. Research could be conducted to explore ways to address these biases and ensure that the system is fair and equitable.
- **Explain ability:** The use of deep learning models can make it difficult to understand how the system is making its predictions. Research could be conducted to develop methods for explaining the system's decision-making process, which could improve trust and transparency.

- **Generalizability:** The system may perform well on the data it was trained on, but may not generalize well to new, unseen data. Research could be conducted to explore methods for improving the system's generalizability, such as transfer learning or domain adaptation.
- **Privacy and security:** The use of facial recognition technology raises concerns about privacy and security. Research could be conducted to explore ways to protect the privacy of individuals and ensure that the system is secure and resistant to attacks.
- **Data augmentation:** To improve the accuracy of the system, research could be conducted to explore methods for augmenting the available data. This could include techniques such as data synthesis, data interpolation, or data generation.
- **Ensemble methods:** Ensemble methods can improve the accuracy of machine learning models by combining the predictions of multiple models. Research could be conducted to explore ways to develop an ensemble model that combines the predictions of multiple similarity models, which could improve the accuracy of the system.

7.4 Implementation Issues

- **Limited image data:** The accuracy of the system heavily relies on the quality and quantity of the image data available for training the model. The system may not be effective if there is limited or poor-quality data available.
- **Sensitivity to image quality:** The system's performance may be affected by variations in image quality such as lighting, resolution, and background. The system may not work well if the images of the missing person and similar-looking people are not of high quality or if they are taken in different conditions.

- **False positives:** The system may generate false positive results if the images of the missing person and similar-looking people are too similar or if the model has not been trained well enough.
- **Ethical considerations:** The use of facial recognition technology for finding missing persons may raise ethical concerns related to privacy and surveillance. It is important to consider these issues and take steps to mitigate any potential negative impacts.
- **Integration with existing systems:** If the system is intended to be integrated with existing missing person databases or systems, compatibility issues may arise that need to be addressed.
- **Maintenance and updates:** The system may require periodic maintenance and updates to ensure that it continues to function effectively and remains up to date with the latest technological advancements.

References

- [1] "Facial Similarity Measures for Robust Missing Person Retrieval" by S. C. Johnson, S. Bharadwaj, and V. M. Patel in 2016

- [2] "A Framework for Missing Person Identification Using Facial Recognition" by A. Singh and A. Singh was published in the year 2016 in the Journal of Advanced Research in Dynamical and Control Systems.

- [3] "Missing person identification using face recognition and unsupervised clustering" by K. Garg and N. Khandelwal in 2017

- [4] "A Deep Learning Approach to Missing Person Identification Using Facial Recognition" by H. Li and S. Li in 2018

- [5] "Facial recognition technology in missing persons investigations: The impact on privacy and human rights" by A. L. Chen in 2019

- [6] "Person Re-identification using Cross-modal Similarity Learning with Deep Convolutional Neural Network" by M. Islam, A. Sharma, and D. D. Sarma in 2019.

- [7] "Missing Person Identification using Convolutional Neural Networks with Euclidean and Cosine Similarity Metrics" by H. U. Siddiqui, A. Ullah, and T. Saba in 2020.

- [8] "Facial Similarity Learning with Triplet Loss for Missing Person Identification" by H. Lin, Y. Zhu, and Q. Ma in 2021.

- [9] "Facial Recognition for Missing Person Identification: A Comprehensive Review of Techniques and Challenges" by S. Islam, M. Hasan, and T. K. Saha in 2021.

- [10] "Deep Learning-based Missing Person Identification System Using Triplet Loss with Multimodal Biometric Fusion" by J. Lee and Y. Lee in 2022.

Appendix

A. source code

Generate keys

```
import pickle
from pathlib import Path

import streamlit_authenticator as stauth

names = ["Peter Parker", "Rebecca Miller"]
usernames = ["peter", "rebecca"]
passwords = ["123", "1234"]

hashed_passwords = stauth.Hasher(passwords).generate ()

file_path = Path(__file__).parent / "hashed_pw.pkl"
with file_path.open("wb") as file:
    pickle.dump(hashed_passwords, file)
```

Admin.py

```
import streamlit as st
import os
import pickle
from pathlib import Path
import streamlit_authenticator as stauth
from PIL import Image
import pandas as pd
import warnings

os.system('python generate_keys.py')
df = pd.DataFrame(columns=['name', 'age', 'img_path', 'Contact_no', 'Location', 'email', 'Match-found', 'email_sent'])
os.makedirs(os.path.join('data', 'admin_data', 'missing images'), exist_ok=True)
csv_file_path = os.path.join('data', 'admin_data', 'missing_data.csv')
if not os.path.isfile(csv_file_path):
    df.to_csv(csv_file_path, index=False)
del df

def load_image (image_file, path_to_save):
    img = Image.open(image_file)
    path_to_save = path_to_save+'.'+img.format.lower ()
    img.save(path_to_save)
```

```

    return img, path_to_save

def save_admin_data (img,name,age,num,email, location):
    img_path = os. path. join('data','admin_data','missing_images',f'{name}')
    img_data, img_path = load_image (img, path_to_save=img_path)
    missing_person_data = {'name':name,'age':
age,'img_path':img_path,'Contact_no':num,'Location': location,'email':email,'Match-
found':'No','email_sent':'No'}
    df2 = pd.DataFrame(missing_person_data,index=[0])
    csv_path = os. path. join('data','admin_data','missing_data.csv')
    df = pd. read_csv(csv_path)
    # df = df.append(missing_person_data, ignore_index =
True).reset_index(drop=True)
    df = pd.concat([df, df2], ignore_index = True)
    df.to_csv(csv_path,index=False)
    return img_data

# --- USER AUTHENTICATION ---

def get_username(file_path):
    with open(file_path,'r',encoding='utf-8') as f:
        admin_logins = f.readlines()
        admin_logins = [i.strip() for i in admin_logins]
        names = list (filter (lambda x: 'names' in x,admin_logins))
        names = names [0]. split ('= ') [1]. strip ()
        usernames = list (filter (lambda x: 'usernames' in x,admin_logins))
        usernames = usernames [0]. split ('= ') [1]. strip ()
        return eval(names), eval(usernames)

names,usernames = get_username('generate_keys.py')

# load hashed passwords
file_path = Path(__file__). parent / "hashed_pw.pkl"
with file_path. open("rb") as file:
    hashed_passwords = pickle.load(file)

authenticator = stauth.Authenticate(names, usernames, hashed_passwords,
    "admin", "abcdef", cookie_expiry_days=0)

name, authentication_status, username = authenticator.login("Login", "main")

if authentication_status == False:
    st.error("Username/password is incorrect")

if authentication_status == None:
    st.warning("Please enter your username and password")

if authentication_status:
    authenticator.logout("Logout", "sidebar")

```

```

page = st.sidebar.radio(label="Menu",options=['Report Missing Person','Reports
from Users'])
if page == "Report Missing Person":
    st.markdown("> Report a Missing Person")
    person_name = st.text_input(label="Name")
    person_image = st.file_uploader("Upload Missing Person Image Here",
type=["jpg", "jpeg"])
    contact_number = st.number_input(label="Phone
Number",value=9769123443)
    age = st.number_input(label="age",min_value=1)
    email = st.text_input (label="Email address")
    location = st.text_input (label="Enter Location")
    report = st.button(label="Report")
    if report:
        img = save_admin_data (person_image,
person_name,age,contact_number,email,location)
        st.image(img)
        st.markdown ("Missing Data Report Saved")
if page == "Reports from Users":
    st.markdown("> Missing Data Reports")
    csv_path = os.path.join('data','admin_data','missing_data.csv')
    df = pd.read_csv(csv_path)
    df = df.drop(columns=['img_path'])
    st.dataframe(df)

```

Image similarity

```

from deepface import DeepFace
def get distance (img1:str, img2:str):
    result = DeepFace.verify(img1_path = img1, img2_path = img2, distance_metric
= 'cosine')
    print(result)
    return result

```

User.py

```

from image_similarity_api import get_distance
import streamlit as st
import os
from PIL import Image
import pandas as pd
import uuid
import warnings
from send_mail import SendMail
import requests

os.makedirs (os.path.join('data','user_data','found_images'), exist_ok=True)
th = 0.8

```



```

def gps_tracker ():
    r = requests.get('https://ipinfo.io/?token=6c098641e84c13')
    return r.json()

def load_image(image_file,path_to_save,save=False):
    img = Image.open(image_file)
    if save:
        img.save(path_to_save)
    return img

def send_email(num,sender_email,passw,rec_email,name,img,r):
    try:
        # print(r)
        new_mail = SendMail([rec_email], f'Spotted {name}',
                             f"Hey, \nI am attaching the Image of found person.\nMy contact
Number: {num}\nAutogenerated Location Details: {r}",
                             sender_email)
        new_mail.attach_files([img])
        # print(new_mail)
        new_mail.send(passw)
        return True
    except Exception as e:
        print(e)
        return False

def verify_email(sender_email,passw):
    try:
        new_mail = SendMail([sender_email], f'Email verification',
                             f"Your email has been verified",
                             sender_email)
        new_mail.send(passw)
        return True
    except Exception as e:
        print(e)
        return False

def save_images(img,num,email,passw):
    img_path = os. path. join ('data','user_data','found_images', f'{str (uuid.
uuid4())}.jpeg')
    img_data = load_image(img,path_to_save=img_path,save=True)
    df = pd. read_csv (os. path. join('data','admin_data','missing_data.csv'))
    match_score = 0.5
    match_details = None
    match_idx = None
    flag = 0
    for idx,row in df.iterrows():
        # print(row)
        # if row['Match-found']. lower () == 'no':
        res = get_distance(row['img_path'], img_path)

```

```

print(1-res['distance'])
if res['verified'] and ((1-res['distance']) > match_score):
    print(row['img_path'])
    match_score = 1-res['distance']
    match_details = row
    match_idx = idx
    flag = 1
if flag:
    df.at [match_idx, 'Match-found'] = 'Yes'
    df.at [match_idx, 'email_sent'] = 'Yes'
    df.to_csv (os. path. join('data','admin_data','missing_data.csv'), index=False)
    response = gps_tracker ()
    status = send_email (num, email, passw, match_details['email'],
match_details['name'], img_path,response)
    return status
return True

```

```

page = st. sidebar.radio(label="Menu", options=['Report Sighting', 'Missing Reports
from admin'])
if page == "Report Sighting":
    st. markdown("> Report Sighting of Person")
    st. write("****")
    number = st.text_input (label="Enter Phone Number", value=123456789)
    email = st.text_input (label="Email address")
    passw = st.text_input (label="Email password")
    person_image = st. file uploader ("Upload Sighting Image Here",
type=["jpg","jpeg"])
    upload = st. button(label="Upload")
    if upload:
        status = verify_email (email, passw)
        if status:
            status = save_images (person_image, number, email, passw)
            if status:
                st. markdown("> Thank you for reporting. If match found will send
email")
            else:
                st. markdown("> Match Found. Please enter correct email address or
password")
            else:
                st. markdown("Email or Password Incorrect")

```

```

if page == 'Missing Reports from admin':
    st. markdown("> Missing Data Reports")
    csv_path = os. path. Join('data','admin_data','missing_data.csv')
    df = pd. read_csv(csv_path)
    df = df. drop(columns=['img_path'])
    st. dataframe(df)

```

app.py

```
import streamlit as st
import pandas as pd
from utils import *
from mail import send_email

def app():

    st.title('Missing Person Identification System')

    tab1, tab2 = st.tabs(['Report', 'Find'])

    with tab1:
        st.header('Report a Missing Person here')

        name = st.text_input('Enter Name')
        date = st.date_input('Select date of Missing')
        contact_email = st.text_input('Enter E-Mail for contact')
        up_img = st.file_uploader('Upload the person image', type=['jpg', 'jpeg', 'png'])

        if st.button('Add Data'):
            with open('./database/data/details.json') as json_file:
                json_decoded = json.load(json_file)

            if up_img is not None:
                save_dir = f'./database/missing_persons/{name}.png'
                with open(save_dir, 'wb') as f:
                    f.write(up_img.read())

                json_decoded[name] = contact_email

                with open('./database/data/details.json', 'w') as json_file:
                    json.dump(json_decoded, json_file)

                encode_folder()

            st.warning('Data Added Sucessfully')

    with tab2:
        st.header('Finding a Missing Person')

        if st.button('Start Camera'):
```

```

match_status, name = detect()
name = name.title()
if match_status:
    st.warning('Match Found')

try:
    json_file = open('./database/data/details.json')
    json_obj = json.load(json_file)
    con_email = json_obj[name]

    lat, long = get_coordinates()

    send_email(name, (lat, long), con_email)

    json_file.close()

except:
    print('Error Occured')

```

```

if __name__ == '__main__':
    app()

```

encode.py

```

import os
import cv2
import face_recognition
import json
import numpy

class Encode:
    def __init__(self, path):
        self.path = path
        self.images = []
        self.classNames = []
        self.myList = os.listdir(self.path)
        self.encodeList = []

    def names(self):
        for cl in self.myList:
            curlImg = cv2.imread(f'{self.path}/{cl}')
            self.images.append(curlImg)
            self.classNames.append(os.path.splitext(cl)[0])

        return self.classNames

```

```

def findEncodings(self):
    for img in self.images:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        encode = face_recognition.face_encodings(img)[0]
        self.encodeList.append(encode)

    return self.encodeList

def save(self):
    clist = []
    for i in range(len(self.classNames)):
        clist.append(self.encodeList[i].tolist())

    data = {"encodeList":clist, "classNames":self.classNames}
    with open("./database/encodings.json", 'w') as file:
        json.dump(data, file)

```

mail.py

```

from email.message import EmailMessage
import ssl
import smtplib

def send_email(name, location, mail_receiver):

    mail_sender = 'jangilitejaswini09@gmail.com'
    mail_password = 'fgjblenhsatjvrie'

    subject = 'Missing Person - Match Found'
    lat, long = location[0], location[1]
    body = f"There is a person who matches to the your missing report\nName:
{name}\nLocation: {lat, long}"

    em = EmailMessage()
    em['From'] = mail_sender
    em['To'] = mail_receiver
    em['subject'] = subject

    em.set_content(body)
    with open('detected_image.jpg', 'rb') as f:
        img_data = f.read()
        em.add_attachment(img_data, maintype='image', subtype='jpg',
filename='detected_image.jpg')

    # send the email using SMTP
    with smtplib.SMTP('smtp.gmail.com', 587) as smtp:
        smtp.starttls()
        smtp.login(mail_sender, mail_password)
        smtp.send_message(em)

```

recognition.py

```
import cv2
import numpy as np
import face_recognition.api as face_recognition
import os
from datetime import datetime

class Recognition:
    def __init__(self, encodeList, classNames):
        self.cap = cv2.VideoCapture(0)
        self.encodeListKnown = encodeList
        self.classNames = classNames
        self.nameList = []

    def recog(self):
        match_status = 0
        name = ""
        while True:
            face_recognition.tolerance = 0.85
            ret, img = self.cap.read()
            imgS = cv2.resize(img, (0,0), None, 0.25, 0.25)
            imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)

            facesCurFrame = face_recognition.face_locations(imgS)
            encodesCurFrame = face_recognition.face_encodings(imgS,
facesCurFrame)

            for encodeFace, faceLoc in zip(encodesCurFrame, facesCurFrame):
                matches = face_recognition.compare_faces(self.encodeListKnown,
encodeFace)
                faceDis = face_recognition.face_distance(self.encodeListKnown,
encodeFace)
                matchIndex = np.argmin(faceDis)

                if matches[matchIndex]:
                    name = self.classNames[matchIndex].upper()
                    y1, x2, y2, x1 = faceLoc
                    y1, x2, y2, x1 = y1*4, x2*4, y2*4, x1*4
                    cv2.rectangle(img, (x1,y1), (x2,y2), (0,255,0), 2)
                    cv2.rectangle(img, (x1,y2-35), (x2,y2), (0,255,0), cv2.FILLED)
                    cv2.putText(img, name, (x1+6,y2-6),
cv2.FONT_HERSHEY_COMPLEX, 1, (255,255,255), 2)
                    cv2.imwrite('detected_image.jpg', img)

            match_status += 1
```

```

        # else:
        #     y1, x2, y2, x1 = faceLoc
        #     y1, x2, y2, x1 = y1*4, x2*4, y2*4, x1*4
        #     cv2.rectangle(img, (x1,y1), (x2,y2), (0,255,0), 2)
        #     cv2.rectangle(img, (x1,y2-35), (x2,y2), (0,255,0), cv2.FILLED)
        #         cv2.putText(img,"Unknown", (x1+6,y2-6),
cv2.FONT_HERSHEY_COMPLEX, 1, (255,255,255), 2)

cv2.imshow('Webcam', img)
if match_status>3:
    return True, name

if cv2.waitKey(1)==ord('q'):
    break

self.cap.release()
cv2.destroyAllWindows()

return match_status, name

```

utilse.py

```

from encoding import Encode
from recognition import Recognition
import numpy as np
import json
from urllib.request import urlopen

def encode_folder():
    enc = Encode('./database/missing_persons/')
    enc.names()
    enc.findEncodings()
    enc.save()

def detect():
    with open('./database/encodings.json') as f:
        data = json.load(f)

    eList = []
    cNames = data['classNames']
    for i in range(len(cNames)):
        eList.append(np.array(data['encodeList'][i]))

    rec = Recognition(eList, cNames)
    status, name = rec.recog()

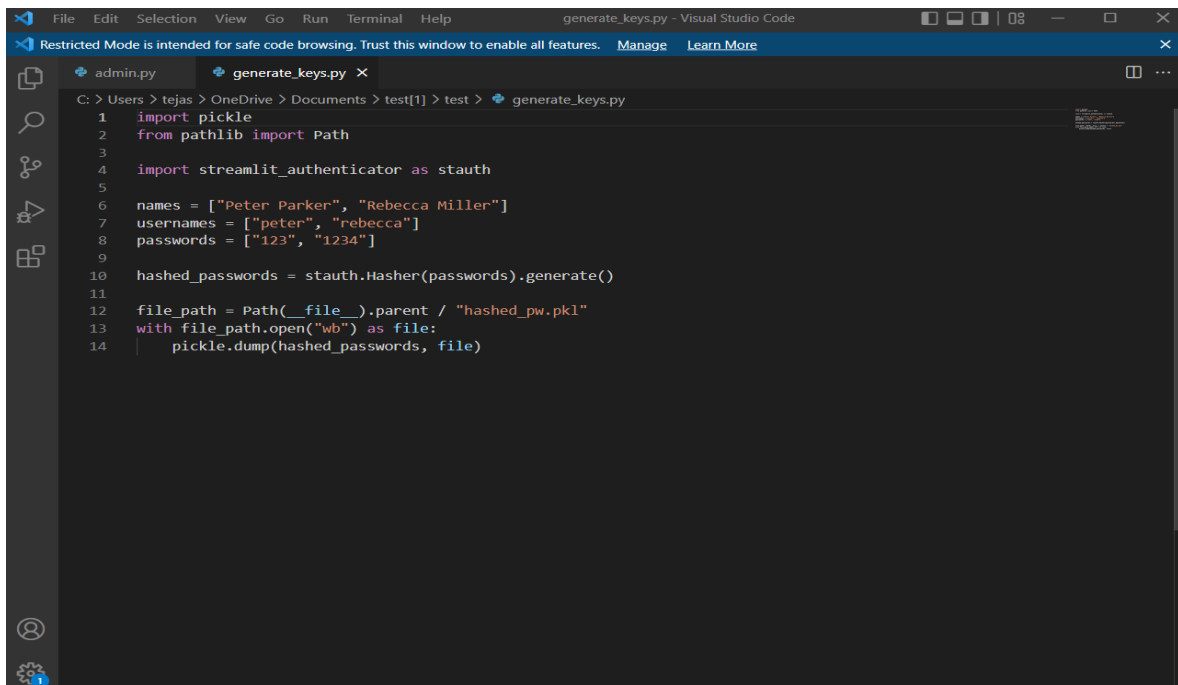
```

```
return status, name
```

```
def get_coordinates():  
    urlopen("http://ipinfo.io/json")  
    data = json.load(urlopen("http://ipinfo.io/json"))  
    lat = data['loc'].split(',')[0]  
    lon = data['loc'].split(',')[1]  
  
    return lat, lon
```

B. Screen shots

Generate keys



```
File Edit Selection View Go Run Terminal Help generate_keys.py - Visual Studio Code  
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More  
admin.py generate_keys.py X  
C: > Users > tejas > OneDrive > Documents > test[1] > test > generate_keys.py  
1 import pickle  
2 from pathlib import Path  
3  
4 import streamlit_authenticator as stauth  
5  
6 names = ["Peter Parker", "Rebecca Miller"]  
7 usernames = ["peter", "rebecca"]  
8 passwords = ["123", "1234"]  
9  
10 hashed_passwords = stauth.Hasher(passwords).generate()  
11  
12 file_path = Path(__file__).parent / "hashed_pw.pkl"  
13 with file_path.open("wb") as file:  
14     pickle.dump(hashed_passwords, file)
```

Admin site missing data

	A1								
	A	B	C	D	E	F	G	H	I
1	name	age	img_path	Contact_no	Location	email	Match-found	email_sent	
2	silver	18	data/admin	1234	Lucknow	gaurav.pate	Yes	Yes	
3	user2	24	data/admin	1	punjab	gaurav.pate	Yes	Yes	
4	test	1000	data/admin	12	punjab	gaurav.pate	Yes	Yes	
5	Silver	32	data/admin	9.769E+09	crystania	warrior.prin	Yes	Yes	
6	Silver	18	data/admin	897231234	Punjab	warrior.prin	No	No	
7									
8									
9									
10									
11									
12									
13									
14									
15									
16									
17									
18									
19									
20									
21									
22									

Admin.py

```

File Edit Selection View Go Run Terminal Help admin.py - Visual Studio Code
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

admin.py X
C:\Users> tejas > OneDrive > Documents > test[1] > test > admin.py
1 import streamlit as st
2 import os
3 import pickle
4 from pathlib import Path
5 import streamlit_authenticator as stauth
6 from PIL import Image
7 import pandas as pd
8 import warnings
9
10 os.system('python generate_keys.py')
11 df = pd.DataFrame(columns=['name','age','img_path','Contact_no','Location','email','Match-found','email_s
12 os.makedirs(os.path.join('data','admin_data','missing_images'),exist_ok=True)
13 csv_file_path = os.path.join('data','admin_data','missing_data.csv')
14 if not os.path.isfile(csv_file_path):
15     df.to_csv(csv_file_path,index=False)
16 del df
17
18 def load_image(image_file,path_to_save):
19     img = Image.open(image_file)
20     path_to_save = path_to_save+'.'+img.format.lower()
21     img.save(path_to_save)
22     return img,path_to_save
23
24 def save_admin_data(img,name,age,num,email,location):
25     img_path = os.path.join('data','admin_data','missing_images',f'{name}')
26     img_data,img_path = load_image(img,path_to_save=img_path)
27     missing_person_data = {'name':name,'age':age,'img_path':img_path,'Contact_no':num,'Location':location
28     df2 = pd.DataFrame(missing_person_data,index=[0])
29     csv_path = os.path.join('data','admin_data','missing_data.csv')
30     df = pd.read_csv(csv_path)
31     # df = df.append(missing_person_data, ignore_index = True).reset_index(drop=True)
32     df = pd.concat([df, df2], ignore_index = True)
33     df.to_csv(csv_path,index=False)

```

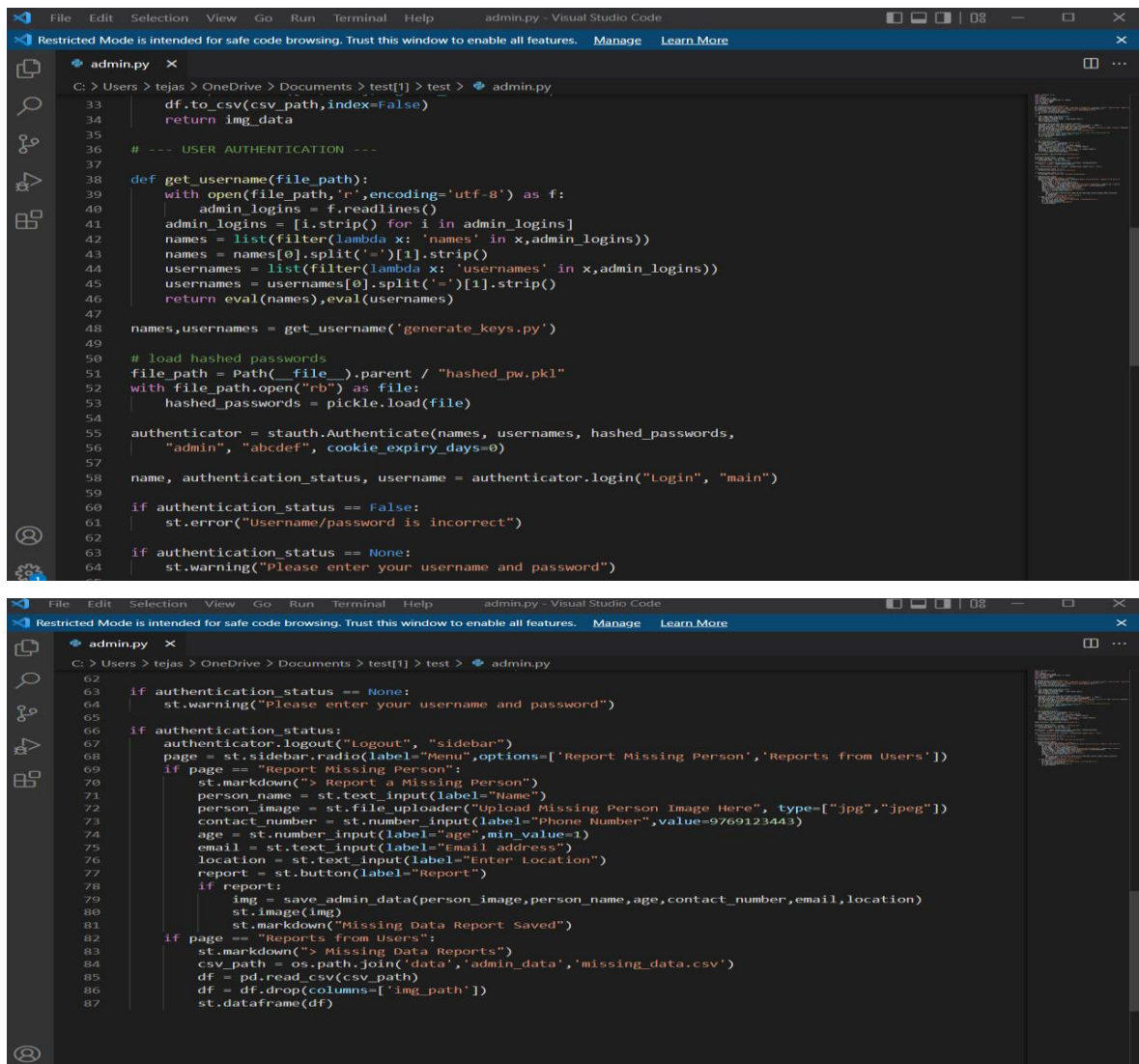
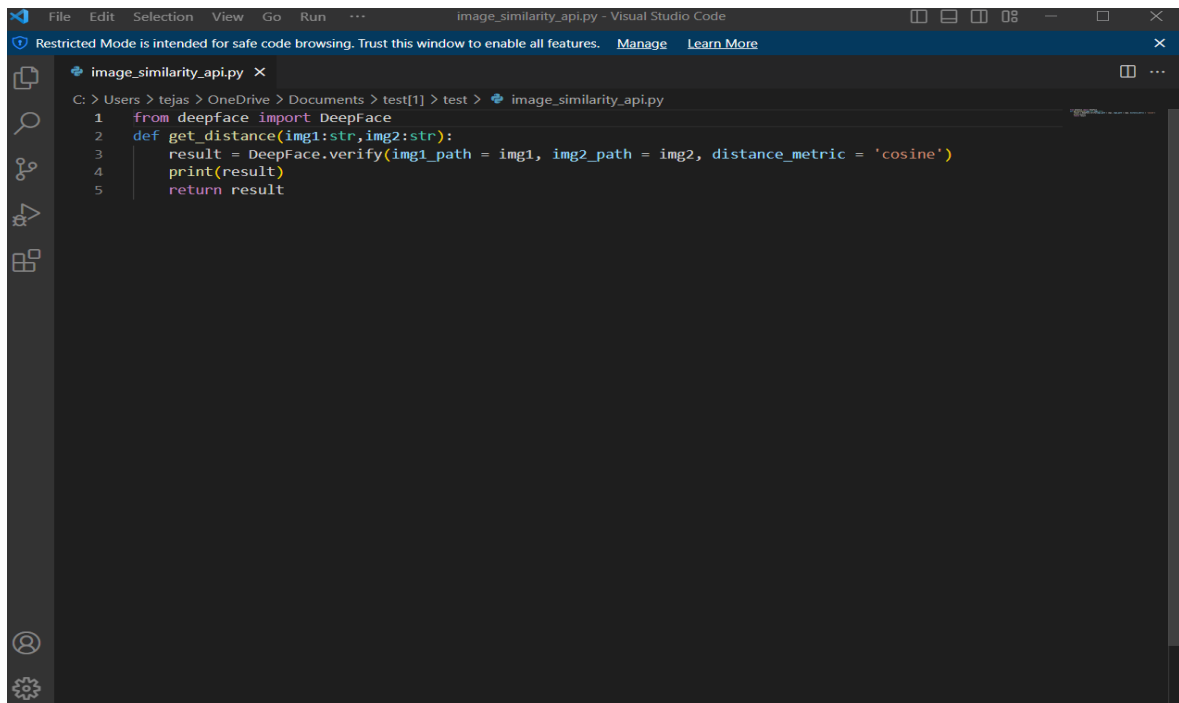


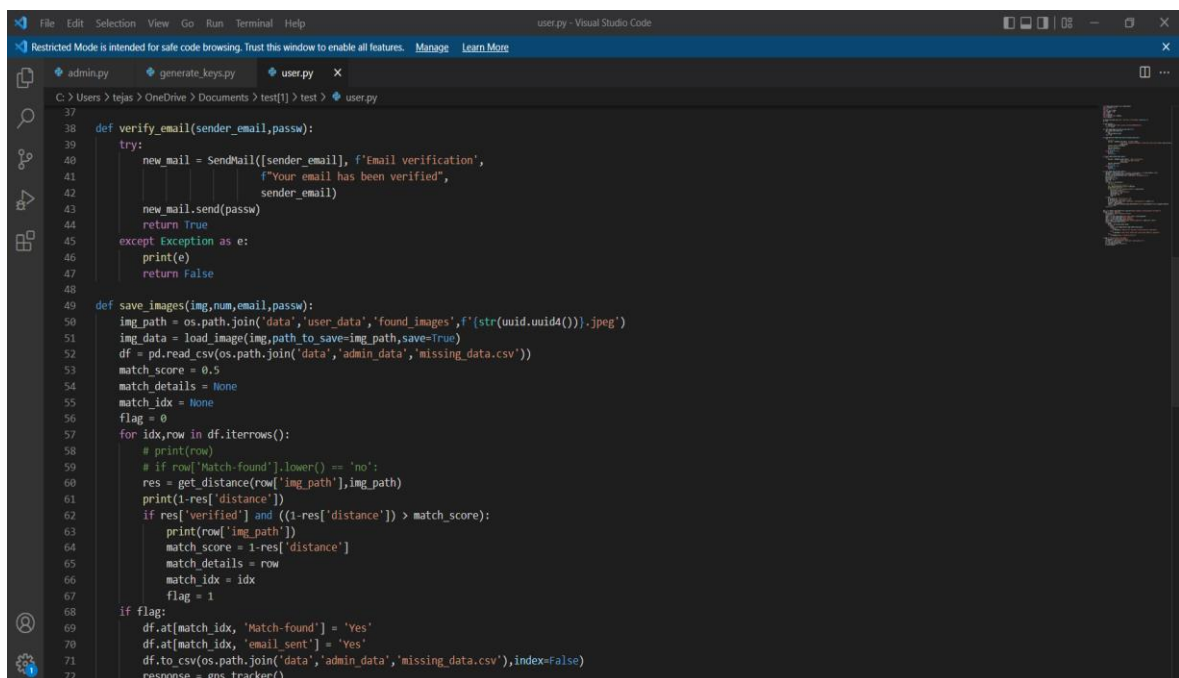
Image similarity



```
image_similarity_api.py - Visual Studio Code
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

image_similarity_api.py X
C:\Users> tejas > OneDrive > Documents > test[1] > test > image_similarity_api.py
1 from deepface import DeepFace
2 def get_distance(img1:str,img2:str):
3     result = DeepFace.verify(img1_path = img1, img2_path = img2, distance_metric = 'cosine')
4     print(result)
5     return result
```

User.py



```
user.py - Visual Studio Code
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

admin.py generate_keys.py user.py X
C:\Users> tejas > OneDrive > Documents > test[1] > test > user.py
37
38 def verify_email(sender_email,passwd):
39     try:
40         new_mail = SendMail([sender_email], f'Email verification',
41                               f'Your email has been verified",
42                               sender_email)
43         new_mail.send(passwd)
44         return True
45     except Exception as e:
46         print(e)
47         return False
48
49 def save_images(img_num,email,passwd):
50     img_path = os.path.join('data','user_data','found_images',f'{str(uuid.uuid4())}.jpeg')
51     img_data = load_image(img_path_to_save=img_path,save=True)
52     df = pd.read_csv(os.path.join('data','admin_data','missing_data.csv'))
53     match_score = 0.5
54     match_details = None
55     match_idx = None
56     flag = 0
57     for idx,row in df.iterrows():
58         # print(row)
59         # if row['Match-found'].lower() == 'no':
60         res = get_distance(row['img_path'],img_path)
61         print(1-res['distance'])
62         if res['verified'] and ((1-res['distance']) > match_score):
63             print(row['img_path'])
64             match_score = 1-res['distance']
65             match_details = row
66             match_idx = idx
67             flag = 1
68     if flag:
69         df.at[match_idx, 'Match-found'] = 'Yes'
70         df.at[match_idx, 'email_sent'] = 'Yes'
71         df.to_csv(os.path.join('data','admin_data','missing_data.csv'),index=False)
72         response = gps_tracker()
```

```
File Edit Selection View Go Run Terminal Help user.py - Visual Studio Code
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

admin.py generate_keys.py user.py X

C:\Users> tejas > OneDrive > Documents > test[1] > test > user.py
1 from image_similarity_api import get_distance
2 import streamlit as st
3 import os
4 from PIL import Image
5 import pandas as pd
6 import uuid
7 import warnings
8 from send_mail import SendMail
9 import requests
10
11 os.makedirs(os.path.join('data', 'user_data', 'found_images'), exist_ok=True)
12 th = 0.8
13
14 def gps_tracker():
15     r = requests.get('https://ipinfo.io/?token=6c098641e84c13')
16     return r.json()
17
18 def load_image(image_file, path_to_save, save=False):
19     img = Image.open(image_file)
20     if save:
21         img.save(path_to_save)
22     return img
23
24 def send_email(num, sender_email, passw, rec_email, name, img, r):
25     try:
26         # print(r)
27         new_mail = SendMail([rec_email], f'Spotted {name}',
28                             f'Hey, \nI am attaching the image of found person. \nMy contact Number: {num} \nAutogenerated Location Details : {r}',
29                             sender_email)
30         new_mail.attach_files([img])
31         # print(new_mail)
32         new_mail.send(passw)
33         return True
34     except Exception as e:
35         print(e)
36     return False
```

```
File Edit Selection View Go Run Terminal Help user.py - Visual Studio Code
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

admin.py X generate_keys.py user.py X

C:\Users> tejas > OneDrive > Documents > test[1] > test > user.py
69 df.at[match_idx, 'Match-found'] = 'Yes'
70 df.at[match_idx, 'email_sent'] = 'Yes'
71 df.to_csv(os.path.join('data', 'admin_data', 'missing_data.csv'), index=False)
72 response = gps_tracker()
73 status = send_email(num, email, passw, match_details['email'], match_details['name'], img_path, response)
74 return status
75
76
77
78 page = st.sidebar.radio(label="Menu", options=["Report Sighting", "Missing Reports from admin"])
79 if page == "Report Sighting":
80     st.markdown("> Report Sighting of Person")
81     st.write("****")
82     number = st.text_input(label="Enter Phone Number", value=123456789)
83     email = st.text_input(label="Email address")
84     passw = st.text_input(label="Email password")
85     person_image = st.file_uploader("Upload Sighting Image Here", type=["jpg", "jpeg"])
86     upload = st.button(label="Upload")
87     if upload:
88         status = verify_email(email, passw)
89         if status:
90             status = save_images(person_image, number, email, passw)
91             if status:
92                 st.markdown("> Thank you for reporting. If match found will send email")
93             else:
94                 st.markdown("> Match Found. Please enter correct email address or password")
95         else:
96             st.markdown("Email or Password Incorrect")
97
98 if page == "Missing Reports from admin":
99     st.markdown("> Missing Data Reports")
100     csv_path = os.path.join('data', 'admin_data', 'missing_data.csv')
101     df = pd.read_csv(csv_path)
102     df = df.drop(columns=['img_path'])
103     st.dataframe(df)
```

C.Research Paper

Finding Missing Person Using Image Similarity and Euclidean

J. Tejaswini (B.E)
Sathyabama university of science and technology
Chennai, India
jangilitejaswini09@gmail.com

Dr. MercyPaulSelvan
Sathyabama university of science and technology
Chennai, India
mercypaulselvan.cse@sathyabama.ac.in

Abstract-

Every day, a large number of people disappear for a variety of reasons, such as advanced age, mental illness, emotional disorders, Alzheimer's disease, etc. The process to find the missing individual fails because the majority of them go unfound. We propose a solution for the same. A record of each newly filed case is kept in the application's database. Anytime someone like this is encountered, they take a photograph of them and appearance up their data with inside the database. If no matches are found, they can upload the data to the database (optionally: if known), save the current location while uploading, and alert higher authorities to the problem. The other scenario is that if a match for the missing individual is discovered, the database's information will be accessed, and police or the family will be notified.

Keywords- CNN, Deep Learning, Neural Networks.

I INTRODUCTION

Every day, many people—children, teenagers, the intellectually disabled, the elderly with dementia, etc.—go missing throughout the world. They have not been located for the most part. In this work, we suggest a method that would

greatly benefit the public and law enforcement by speeding up the search procedure by making use of facial recognition technology. The ability to quickly and accurately identify a missing individual is one of the numerous applications of facial recognition technology. To speed up the process of locating a missing individual, we intend to develop an application that will be used by a group of volunteers. The process of tracking down a wanted criminal will be simplified thanks to this. In the meantime, automation is needed to streamline the process of locating a certain person using image recognition and similarity checks to determine whether two photographs have the same features. This will allow us to determine whether or not the photograph of the missing person taken at a certain location is accurate, and if it is, police may begin searching in that region.

II LITERATURE REVIEW

[1] An earlier work with a comparable issue definition and goal was given by Shefali Patil and colleagues from SNDT Women's University in Juhu, Mumbai. The system proposed by them uses KNN Algorithm which makes use of 136 * 3 data points to recognize Face. The main disadvantage of using the KNN method is its accuracy 71.28% and also it does not

address cross-age face recognition. The primary distinction between our study and theirs is that we are going to use a mobile application to construct a dataset using volunteers' work. we are going to use AWS facial reorganization which has cross-age face recognition. Also, our dataset is going to be stored in the cloud database

[2] Sarthak Babbar, Navroz Dewan, Kartik Shangle, and the Jaypee Institute of Information Technology team from Noida, India and coworkers published a paper in 2020 that provides a comprehensive overview of Amazon Web Services (AWS) Recognition and draws comparisons between AWS Recognition and other algorithms and systems such as CDAC-VS and CNN. Therefore, we were able to choose the appropriate method for our project, such as Amazon Web Services (AWS) Recognition, with the aid of this article. As we age, our appearance will change, but the images in our database will remain static. To that aim, we want to investigate Residual Network (ResNet) performance in the context of age-invariant facial recognition. The performance is compared to those of other techniques, such as cross-age reference coding (CARC), Amazon Web Services (AWS) Recognition, and others, using the cross-age celebrity dataset (CACD) and a verification subset (CCD-VS). On the CACD-VS dataset, ResNet and AWS Recognition each attained accuracy ratings above 98.50%.

[3] Principal Component Analysis (PCA) was used by Rohit Satle and colleagues to create a face recognition system, which they discussed in a paper given in August of 2016. The principal component analysis technique has two major limitations: it is computationally intensive and it can only analyse facial expressions that are quite similar to one another. Our approach stands out from theirs because it can correctly identify the same person even when two photographs of them exist, each with a slightly different facial expression. In addition, our algorithm can distinguish between a person's

"moustache" and "no moustache" photos. Using AI for picture identification will unquestionably improve our precision.

[4] Birari Hetal and her colleagues from Late G.N. Sapkal College of Engineering delivered a study in which they addressed very similar issues. To facilitate the search for a missing individual, they developed an Android app. They recommend an SWF-based Android app.

Authors Rohit Satle and crew presented their article titled in August of 2016. [5] It discusses the Principal Component Analysis (PCA) approach to developing a facial recognition system. The principal component analysis (PCA) technique has two major drawbacks: (1) it is computationally intensive, and (2) it can only analyse groups of faces that have a similar expression.

Researchers led by Swarna Bai Arniker have released a study. [6] that uses Radio Frequency Identification to locate and identify missing persons. The disadvantage of this technique is that the person being monitored must always be wearing the RFID tag, which is inconvenient.

Birari Hetal and colleagues utilized SWF-SIFT to compare faces in a paper that they published as a team [7]. However, SIFT is computationally costly and takes a while because it is based on histograms of gradients, where each pixel in the patch needs to be calculated.

Thomas M. Omweri and Andrew M. Kahonge published a paper [8] in 2015 that included comparisons of facial attributes made with SWF-SIFT. Since SIFT is based on histograms of gradients, it is computationally expensive and time-consuming to calculate each pixel in the patch.

In 2015, Andrew M. Kahonge and Thomas M. Omweri presented a study on this subject. where they found someone who had gone missing by employing facial recognition technology based on the Line Edge Method (LEM) [9]. Around

85% of its potential was realized by the system. To help locate missing persons in his country, Zimbabwean researcher Peace Muyambo published a study in 2018 proposing the use of facial recognition technology. [9] the LBPH approach was used to identify faces by. The overall accuracy of the suggested approach for identifying people's faces was 67.5%. The LBPH algorithm is indifferent to changes in lighting conditions.

Swarna Bai Arniker and K. Sita Rama Rao from the Research Center Imarat in Hyderabad wrote the article "Use Insights of RFID Based Missing Person Identification System," which was presented at a conference in August 2014. It's possible that in the near future, all police stations and large public events will be equipped with RFID readers. This might be used to reunite families and return missing children, children with disabilities, and elderly people to their caretakers. The individual must actively participate by donning the RFID tag. Therefore, it is restricted in that it cannot accommodate the RFID chip used for personal tracking. [10]

III PROPOSED SYSTEM

Every day, a large number of people disappear for a variety of reasons, such as advanced age, mental illness, emotional disorders, Alzheimer's disease, etc. The process to find the missing individual faints because the majority of them go unfound. This research article focuses on improving public and police safety by expediting the search for a lost person. A record of each newly filed case is kept in the application's database. Anytime someone like this is encountered, they take a picture of them and search the database for their details. If no correspondence is established, they can upload the information into the database (optional: if known), record the current

position during the download, and notifies higher authorities of the situation. The other scenario is that if a match for the missing individual is discovered, the database's information will be accessed, and police or the family will be notified.

IV SYSTEM ARCHITECTURE

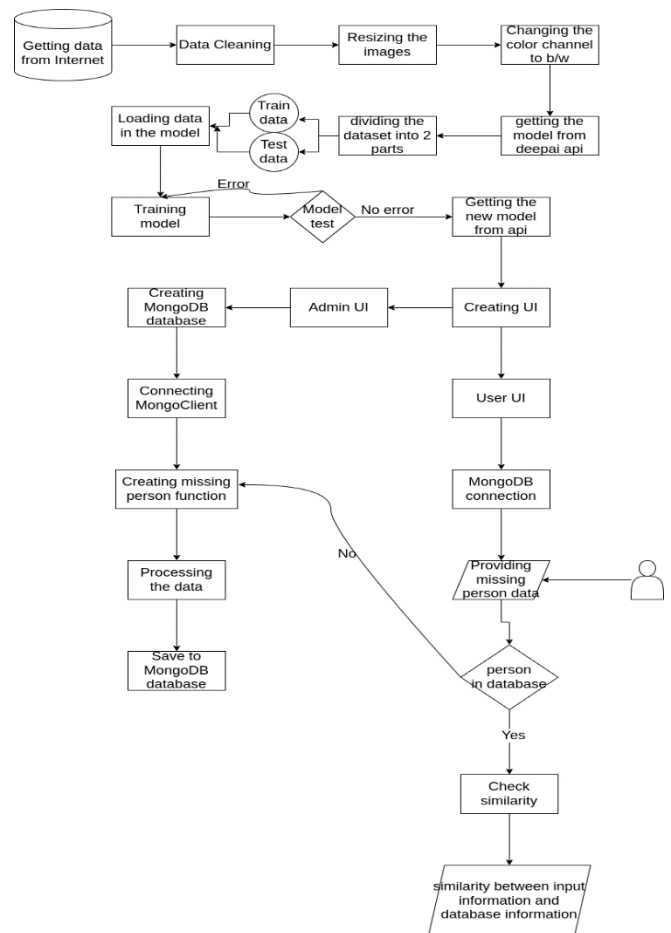


Figure 1: System Architecture

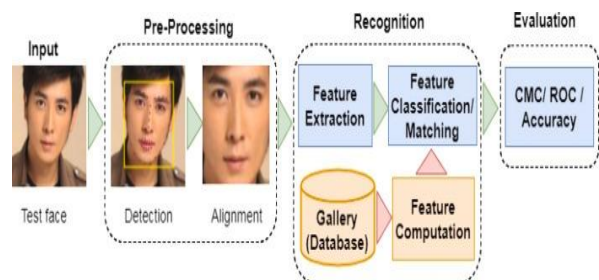


Figure 2: Face Recognition

V Module 1: Data Acquisition and Data preprocessing

Data Acquisition

We will take deepai dataset with similar images set to train our algorithm. The dataset contains many images that may or may not be similar. We will use it to check for image similarity by developing an image similarity API in this project.

Data Cleaning - Preprocessing of images

After amassing the images, we subjected the dataset to some preliminary processing. The images in the dataset were subjected to a number of different pre-processing steps. Image A row-and-columnar array in two dimensions is sufficient to define a picture. Each discrete element that makes up a digital image has a distinct value at a certain coordinate. The foundation of digital images are pixels.

Image types include:

- Binary Image
- Black and White Image
- 8-bit color format
- 16-bit color format

Zooming

Magnifying an image, or "zooming" it, entails making a larger version of the original image. Thermal photos that have been zoomed in on can be utilized for enhancement since they are similar to what would be seen if the photographs were captured up close.

Rotation

To rotate a picture is to shift its location around a fixed pivot at an arbitrary angle. The image would show the situation as though it were photographed from a different vantage point.

Salt and pepper noise

Mixture of background noises best described as "salt and pepper"

When a picture is "salted and peppered," the intensity of some of its pixels is arbitrarily set to 1 while the intensity of other pixels is set to 0. Images captured on a dusty day, or with dust on the camera, may exhibit salt and pepper noise. There is a lot of noise in only a few pixels. Salt and pepper noise is a general term for a variety of mechanisms that lead to this type of fundamental image degradation. The effect is the same as randomly dotting the screen with white and black dots.

Cleaning unwanted images

The dataset had a lot of unwanted data so we need to remove those images from our dataset. Now that we've chosen our parameters for image tagging, we can iterate through each of the folders, finding artificial images and removing them using Path. Unlink ().

This enables us to threshold these photographs at a different level than those without text depending on the colour proportion. It makes sense that photographs with incorrectly recognized text will have a much more varied tonal distribution than those with text added (which typically have a singular font color).

Re-sizing the images

Initially we had 1920x1080 size of images but for reducing computation we have to reduce our images to a size of 128x128 pixels.

Color Channel changing

We shall modify the color channel after scaling in order to extract the features. Enhancing the contrast in an image allows it to have more contrast and more intensities overall than it would without. Since the same photograph can have stronger contrast on a sunny or bright day, this was used to enhance images. Our training set was chosen to include a wide variety of potential outcomes.

Labeling out images

Now we need to label the images as per real or fake in training dataset. The first

and most important step is labelling the image. Although labelling takes a lot of time, the more effort we put into labelling the photographs, the more accurate our model will be.

Module 2: Model Improvisation

Deepai's image similarity model will be applied. When two photographs are compared, Image Similarity gives us a result that indicates how visually similar the two pictures are. A score of 0 indicates that the two photos are identical, while a lower value indicates that the two are more comparable in terms of context. Letting machine vision do it for you using this API will save you from having to sift through datasets looking for duplicates or identifying a visually comparable set of images.

The image similarity API analyses two photos and produces a distance between the two images. The distance value tells us how visually similar the two photographs are, with a distance value of 0 representing an exact match. With the help of the distance value, we can determine how two photographs evolve over time or find duplicates in your user data.

An indicator of how visually similar two photographs are is returned by the API. With this, you can group similar images together, search for duplicates in a collection, or incorporate image similarity into your apps.

We can use the sentence similarity API to:

Lookup using an image in this scenario, the user is prompted to provide a picture of the missing person so that the database can be searched. This has two applications. The user will first see information on the missing individual, such as name, age, contact information,

and location, if the record in the database matches.

Search by filter: Users can quickly search for records by using the following filters in addition to the two options listed above.

Filter by name: When a user enters a name, the appropriate information is taken from the database.

Filter by age: If the stranger uploading the case is unclear about the actual age, a slider is provided to select an age range. Details about those who fall within the chosen age range will be shown.

Filter by location: If the location filter is being used, the user must input the state in order to get the pertinent information.

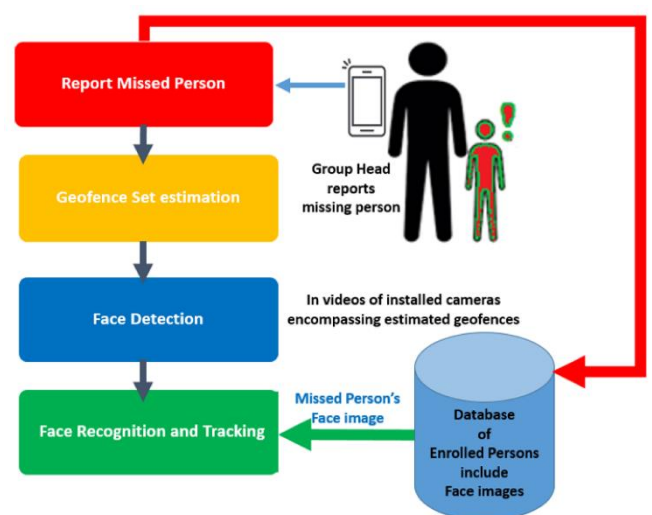


Fig 3: Tracking a missing a person.

Module 3: Creating Admin UI

We will now use Stream lit to develop the admin ui. Without using HTML, CSS, or JS, we can create UIs for our models with the aid of the Python module Stream lit. The majority of models are unattractive and die inside a Jupyter notebook. But using Streamlit, we can

design a simple user interface to display your model to others.

Steps:

1. Install the necessary libraries, including Matplotlib, Scikit-Learn, Pandas, and Streamlit.
2. import the required libraries, including confusion matrix, train test split, and logistic regression.
3. Data frame loading
4. data manipulation
5. split between the two
6. Create the model.
7. Start the stream lit app by typing stream lit run app.py.

Getting MongoDB database

Since MongoDB will serve as our database, we must first establish a connection to the MongoDB cluster. To build websites that can grow with their user base and remain easily accessible, developers often turn to MongoDB, a document database. NoSQL databases, such as MongoDB, are useful when your data is document-centric and doesn't fit neatly into a relational database's schema, when you need to accommodate massive scale, when you need to quickly prototype, and in a few other situations.

Due to its flexible schema approach and popularity among agile development teams, we have implemented it.

We need the PyMongo driver to connect. Then follow the steps:

1. Click connect.
 - a. Click Databases on the upper left side of Atlas.
 - b. Click Connect for the database deployment you want to connect to in the

Database Deployments pane.

2. Choose a login method by clicking.
3. Connect your application by clicking.
4. Choose your driver version and Python.
5. The connection string is copied.
6. Set up the connection string that is provided.
7. import PyMongo's Mongo Client.
8. To your cluster, connect.

Creating Missing Person Registration Function:

Now we will create a missing registration function to register about the missing person by providing his/her name, contact and image. This function will add details about a missing person, by adding name, image_path, and number. When we add the details, the function will print "Report added." The function is also capable of displaying in case the report is not added due to error. It will print "Error happened while uploading missing person's data to cloud."

Module 4: Creating User UI

Now we will create an user UI using Stream-lit. We will now use Stream-lit to develop the admin ui. Without using HTML, CSS, or JS, we can create UIs for our models with the aid of the Python module Stream-lit. The majority of models are unattractive and die inside a Jupyter notebook. But using Stream-lit, we can design a simple user interface to display your model to others.

Steps:

1. Install the necessary libraries, including Matplotlib, Scikit-Learn, Pandas, and Streamlit.
2. import the required libraries, including confusion matrix, train test split, and logistic regression.
3. Data frame loading
4. data manipulation
5. split between the two

6. Create the model.
7. Start the streamlit app by typing `streamlit run app.py`.

Connecting to mongoDB database

At first users have to be connected to the MongoDB database.

We need the PyMongo driver to connect. Then follow the steps:

1. Press "connect".
 - a. Click Databases on the upper left side of Atlas.
 - b. Click Connect next to the database deployment you wish to connect to in the Database Deployments pane.
2. Choose a login method by clicking.
3. Connect your application by clicking.
4. Choose your driver version and Python.
5. The connection string is copied.
6. Set up the connection string that is provided.
7. import PyMongo's MongoClient.
8. To your cluster, connect.

Querying for person from the Database

We create a function that will check if the missing person is already in the database using phone number. The section is same the section mentioned in module 3, but the only difference is that we request for the data from the user page not the admin page.

CONCLUSION

One-shot learning for image identification has greatly improved accuracy. When used properly, this technology has the potential to improve lives. It can be utilized anywhere people congregate, such as hotels, hospitals, etc. The

procedure of locating the missing persons has been sped up. In our system, an effective facial recognition approach that quickly completes the work has replaced the time-consuming procedure of manually searching the databases for each image to determine whether it matches. If a match is found, having Google Maps built in will be helpful for pinpointing the specific position of the suspect, which in turn will make the police's job easier. As an additional measure, calling the police as soon as possible is recommended. Our goal using TensorFlow Face recognition is to get almost optimal results using a pre-trained model. By attaching it to public cameras, we see opportunities for this system to develop in the future and provide real-time facial recognition. Those public cameras will constantly send us frames, and we'll keep a close eye on them. Quickly and efficiently, it notifies relevant authorities when a missing individual is recognized in one of the images.

REFERENCES:

- [1] Tech Report, 2014, pp. 14-003; Huang, Gary B. and Erik G. Learned-Miller, "Labeled Faces in the Wild: Updates and New Reporting Procedures," Department of Computer Science, University of Massachusetts Amherst, Amherst, MA, USA.
- [2] The authors include S. Chandran, Pournami, Balakrishnan, Byju, Rajasekharan, Deepak, N. Nishakumari, K., Devanand, P., and M. Sasi, P. (2018). Missing Child Identification System Using Multi class SVM and Deep Learning, 113–116. 10.1109/RAICS.2018.8635054
- [3] The article "MISSING CHILD IDENTIFICATION USING FACE RECOGNITION SYSTEM," by Rohit Satle, Vishnu prasad Poojary, John Abraham, and Mrs. Shilpa Wakode, appeared in Vol. 3, Issue. 1, July–August 2016.
- [4] International Conference on

Informatics, Electronics & Vision (ICIEV), Dhaka, 2014, pp. 1-4. S. B. Arniker et al., "RFID based missing person identification system."

[5] Birari Hetal, "Android Based Application - Missing Person Finder," Iconic Research and Engineering Journals, Vol. 1, Issue 12, JUN 2018.

[6] The International Journal of Computer Applications Technology and Research, Vol. 4, Issue 7, 507–511, 2015. Thomas M. Omweri, "Using a Mobile Based Web Service to Search for Missing People - A Case Study of Kenya."

[7]. Robust Face Recognition System for E-Crime Alert by Sumeet Pate in International Journal for Research in Engineering Application and Management, Issue 1 (March 2016).

[8] International Journal of Engineering Research and Technology (IJERT), Volume 7, Issue 7, Peace Muyambo (2018), "An Investigation on the Use of LBPH Algorithm for Face Recognition to Find Missing People in Zimbabwe" (July 2018)

[9] Criminals and missing children identification using face recognition and web scraping, S. Ayyappan and S. Matilda, IEEE ICSCAN 2020.

[10] Find Missing Person Using AI, Shefali Patil, Pratiksha Gaikar, Divya Kare, and Sanjay Pawar, International Journal of Progressive Research in Science and Engineering, Vol. 2, No. 6, June 2021.