# MISSING LINK: THE SEARCH AND LOCATE APPLICATION FOR WANTED PERSON

*A Report*

*Submitted in partial fulfilment of the requirements for the award of the Degree of*

***Bachelor of Technology***

*In*

***Department of computer science and engineering***

*By*

**VIVEK SINGH** [BTECH/10426/20]


**Birla Institute of Technology, Mesra**

**Ranchi, Jharkhand-835215**

## APPROVAL OF THE GUIDE

Recommended that the B. Tech. Summer Internship Project titled **The Search and Locate Application for Wanted Person** submitted by **VIVEK SINGH (**BTECH/10426/20**)** is approved by me for submission. This should be accepted as fulfilling the partial requirements for the award of Degree of Bachelor of Technology in **Computer Science and Engineering**. To the best of my knowledge, the report represents work carried out by the student in **BIT MESRA** and the content of this report is not form a basis for the award of any previous degree to anyone else.

**Date:**

**Mr. Indra Deo Ram**
**<Designation of the Guide>**
**Department of Computer Science and Engineering**
**Birla Institute of Technology, Mesra**

## DECLARATION CERTIFICATE

I certify that

a) The work contained in the report is original and has been done by myself under the general supervision of my supervisor.

b) The work has not been submitted to any other Institute for any other degree or diploma.

c) I have followed the guidelines provided by the Institute in writing the report.

d) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

e) Whenever I have used materials (data, theoretical analysis, and text) from other sources, I have given due credit to them by citing them in the text of the report and giving their details in the references.

f) Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

**Date:**

**Vivek Singh**[BTECH/10426/20]
**Department of Computer Science and Engineering**
**Birla Institute of Technology,Mesra**

## CERTIFICATE OF APPROVAL

This is to certify that the work embodied in this Summer Internship Report entitled "The Search and Locate Application for Wanted Person", is carried out by **Vivek Singh(BTECH/10426/20)** has been approved for the degree of Bachelor of Technology in Department of Computer Science and Engineering of Birla Institute of Technology, Mesra, Ranchi.

Date:

Place:

(Chairman)
Head of the Department
Dept. of Comp. Sc. & Engg.

(Panel Coordinator)
Examiner
Dept. of Comp. Sc. & Engg.

## *ABSTRACT*

The Search and Locate Web App for Wanted Persons is an innovative and powerful application designed to assist law enforcement agencies, investigators, and the general public in searching for and locating individuals with outstanding arrest warrants. The project aims to leverage the capabilities of modern technology to enhance the efficiency and effectiveness of criminal investigations, promoting public safety and supporting the justice system. By using Artificial Intelligence (AI) technology, we're building this system.

Computer vision and machine learning algorithms are used to design this system. In this System , we're using face encodings and plotting them in 128 dimensional space. And further clustering them using the distance between each point. We are assuming that similar faces will create a high density cluster. These clusters belongs to each unique face. Face recognition is determined by object detection techniques using Haar Cascade algorithm in OpenCV and face recognition library from DLIB. This model can classify with the accuracy rate of 87% and further can be improvised using huge datasets. By combining data integration, advanced search algorithms, and user-friendly interfaces, the app empowers law enforcement agencies and the public to collaborate effectively, enhancing public safety and promoting a safer and more secure society.

## *ACKNOWLEDGEMENT*

I would like to express our heartfelt gratitude to all those who have made it possible for us to successfully complete this report and project. I am deeply appreciative of the invaluable guidance and unwavering support provided by Mr. Indra Deo Ram. His expertise, continuous encouragement, and insightful suggestions have been instrumental in ensuring that this project achieved its intended objectives. I am truly grateful for his contributions from the very beginning to the final stages of this endeavor.

Moreover, we would like to acknowledge and sincerely appreciate the indispensable role played by Dr. Debjani Mustafi, the Panel Head, in providing us with the necessary guidance and direction throughout the project. I am grateful for her valuable inputs and contributions, which have greatly contributed to the successful completion of this project.

Once again, I extend our deepest appreciation to all those who have contributed to the realization of this report and project, as their support and guidance have been pivotal to its success.

I would also like to thank my teammates for their support.

Date Vivek Singh [BTECH/10426/20]

*CONTENTS*

# *CHAPTER 1*

## INTRODUCTION

### 1.2 OVERVIEW

The project, "**The Search and Locate Web Application for Wanted People**," aims to develop an innovative web application to assist law enforcement agencies, investigators, and the general public in searching for and locating individuals with outstanding arrest warrants. The primary objective is to leverage modern technology to enhance the efficiency and effectiveness of criminal investigations, ultimately promoting public safety and supporting the justice system. The project involves the creation of a web app that integrates data from various sources, including law enforcement databases, public records, and user-contributed information. By aggregating and analysing this data, the application will provide a comprehensive and up-to-date database of wanted persons, complete with their profiles, photographs, and relevant details such as criminal history and associated offenses.This model has an accuracy rate of 87%.
Machine learning, Computer vision are being used in this model which are the subset of AI and it allows the user to train the system and predict the output in a certain range. This technology helps to reduce the gap between human and machines.

Computer vision is the library which captures, understands the images to perform image processing. It helps in processing and extraction of the data to generate the information and also it uses other libraries which play an important role in the system. It has a graphical user interface (GUI) which makes the operating systems easy for users to use.

### 1.2 PURPOSE

#### 1.2.1 HUMAN PSYCHOLOGY

The app taps into the innate human motivation to contribute to the well-being of society and assist law enforcement agencies in locating wanted individuals. People may feel a sense of duty, responsibility, or a desire for justice, leading them to engage with the app and provide information that could aid in apprehending criminals.
The success of the app relies on users' trust and confidence in the technology and its ability to accurately identify and locate wanted individuals. Human psychology plays a role in accepting and relying on technology as a reliable means of information retrieval and decision-making.

Perceptions of Effectiveness: The app's impact on crime prevention and law enforcement efforts can shape users' perceptions of its effectiveness. If the app successfully assists in locating wanted persons and leads to arrests, it can reinforce users' confidence in the app's capabilities and encourage continued engagement.

It's important to note that while psychological factors can contribute to criminal behavior, not everyone with these factors will engage in criminal acts. The interplay of individual predispositions, environmental influences, and personal choices is complex and varies from case to case. Understanding the psychological aspects behind crimes can help inform prevention strategies, rehabilitation programs, and interventions aimed at reducing criminal behavior and promoting public safety.

### 1.2.2 FACTS &STATISTICS

1. Global Crime Rates: Crime rates can vary significantly across different countries and regions. According to the United Nations Office on Drugs and Crime (UNODC), the global average intentional homicide rate in 2019 was 6.1 per 100,000 population.
2. Types of Crimes: Crimes can be categorized into various types, including violent crimes (such as homicide, assault, and robbery), property crimes (such as burglary and theft), white-collar crimes (such as fraud and embezzlement), and cybercrimes (such as hacking and identity theft).
3. Crime Trends: Crime rates can fluctuate over time due to various factors, including social, economic, and demographic changes. Analysing long-term trends can provide insights into the overall direction of crime rates in a particular area or globally.
4. Factors Influencing Crime Rates: Several factors can contribute to crime rates, including poverty, unemployment, substance abuse, education levels, social inequality, and availability of firearms. Understanding these factors is essential for implementing effective crime prevention strategies.
5. Reporting and Underreporting: Not all crimes are reported to law enforcement agencies, leading to underreporting. This can be due to various reasons, such as fear of reprisal, lack of trust in the justice system, or a belief that reporting will not result in any action.
6. Impact of Technology: The digital age has brought new challenges in terms of cybercrimes and online fraud. As technology continues to advance, criminals may adapt

their methods, requiring law enforcement agencies to stay vigilant and develop specialized skills and tools to combat cybercrimes effectively.

## 1.3 PROBLEM DEFINITION

### Vision behind the Project:

In the ever-changing world we live in, crimes are unfortunately on the rise. But there's also a silver lining - technology is advancing to make things better for all of us, and artificial intelligence (AI) is playing a significant role in this progress.

One issue we face is that some criminals manage to escape justice. In the past, they used to distribute flyers to inform the public about wanted individuals, hoping someone would spot them and report it. However, this method had its flaws, relying on people who might not always be attentive or willing to help. As a result, some convicted individuals were never caught.

Now, we have a more promising solution in sight. By combining public CCTV cameras with AI, we can create a system that acts like an ever-watchful eye. Unlike humans, AI won't get tired, and it won't overlook anything important. This research project is currently working on developing this model, and if successful, it could significantly improve the efficiency of catching criminals in real-world applications.

With this technology, we can enhance public safety and contribute to a more secure society. By keeping a watchful eye on potential suspects, we can reduce the chances of criminals getting away with their actions. It's an exciting prospect for a safer future!

### Use Case and Application:

In this project, my goal is to create a useful app that assists law enforcement in identifying and tracking suspects or convicts. The app will work by taking video footage of an incident where the wanted person might be present. This footage will be used to train an AI model to recognize the specific individual accurately.

Once the AI model is ready, the app will use it to analyze live footage from various CCTV cameras in different locations. It will scan each face it detects in the footage and check if it matches the person specified by the police officials. If a potential match is found, the app will promptly notify the authorities, providing them with real-time information.

Additionally, this app can be used to track a person's movement if enough CCTV cameras cover the area and their footage is fed into the system. By using this technology responsibly, we can contribute to enhancing public safety and supporting law enforcement efforts in a more efficient and effective manner.

## 1.4 HAAR CASCADE XML

A classifier using a large dataset of positive and negative images. Positive images contain the objects we want the classifier to detect, such as faces, while negative images contain everything else.

This technique is particularly useful for face detection, as it allows the system to accurately identify faces, enabling subsequent face recognition processes. The Haar cascade classifier, popularized by Paul Viola and Michael Jones in their paper "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001, is a powerful object detection technique.

The classifier is trained using a large number of positive and negative images, and it can then be used to detect specific objects in different images. The Haar cascade classification leverages Haar wavelet technology to analyse image pixels in sections based on their functions. It utilizes the concept of a "comprehensive image" to calculate "features" that help identify the desired objects.

The AdaBoost learning algorithm is employed by the Haar cascade to select a small subset of data from a large dataset, resulting in an efficient identification of key features. The cascading approach is used to identify faces within an image by progressively applying the classifier to different image regions.

In summary, the Haar cascade XML method is a machine learning-based technique that utilizes Haar wavelets, the AdaBoost algorithm, and cascading to effectively detect specific objects, such as faces, in images.

## 1.5 D-LIB

D-lib is a toolkit used for making real time machine learning applications and information evaluation packages. It’s used for face detection/ face recognition and facial landmark detection. The frontal face detector in d-lib works really nicely as it's miles quite simple and it simply works out of the box.

### 1.5.1 D-LIB'S 128 FACE FEATURES

DLIB's 128 face features refer to a facial feature representation algorithm developed by the DLIB library. DLIB is a popular open-source library that provides a wide range of computer vision and machine learning tools. One of its notable features is the ability to extract a compact and unique representation of a face using 128 feature vectors.

The 128 face features are extracted by passing an image of a face through a deep convolutional neural network (CNN) trained on a large dataset of face images. This network learns to encode facial characteristics into a fixed-length vector of 128 floating-point numbers.

These 128 face features encode various facial attributes, such as the shape of the eyes, nose, and mouth, the positioning of landmarks, and other distinguishing facial characteristics. The representation aims to capture essential information about a person's face while being invariant to variations in lighting conditions, pose, and facial expressions.

DLIB's 128 face features have been widely used for face recognition and related applications. The feature vectors can be compared using distance metrics (such as Euclidean distance or cosine similarity) to determine the similarity or dissimilarity between different faces. This allows for face identification, verification, and clustering tasks.

The compact nature of the 128 face features makes them efficient for storage and comparison, enabling real-time face recognition applications on devices with limited computational resources. Additionally, the representation is designed to be highly discriminative, meaning that faces of different individuals should have significant feature differences, while faces of the same person should exhibit minimal variation in the feature space.

Overall, DLIB's 128 face features provide a robust and efficient representation for facial analysis and recognition tasks, facilitating accurate face matching and enabling various face-related applications in computer vision and machine learning.

### 1.5.2 FACIAL LANDMARK DETECTION THROUGH D-LIB

Facial landmark detection is a crucial task in computer vision that involves identifying and locating specific facial landmarks or key points on a face, such as the corners of the eyes, the tip of the nose, and the edges of the lips. DLIB is a popular open-source library that provides facial landmark detection capabilities using its pre-trained models.

The facial landmark detection process in DLIB typically involves the following steps:

1. **Face Detection:** First, the library detects faces in an input image using methods like a pre-trained face detector. This step identifies the regions of the image that contain faces.
2. **Facial Landmark Localization:** Once the face regions are identified, DLIB's facial landmark detection algorithm is applied to precisely locate the facial landmarks within each face region. The algorithm estimates the positions of key points by analysing the shape and texture patterns in the image.
3. **Landmark Representation:** DLIB's facial landmark detection provides a set of key points representing the facial landmarks. These points are typically defined as (x, y) coordinates on the face image.

The facial landmark detection in DLIB can be used for a variety of applications, including facial analysis, facial expression recognition, face alignment, and face morphing. By accurately localizing the facial landmarks, DLIB enables more advanced facial analysis and manipulation tasks.

It's worth noting that DLIB provides pre-trained models for facial landmark detection, allowing developers to easily integrate this functionality into their own applications. These models are trained on large datasets, making them capable of generalizing well to various face images and conditions.

Overall, DLIB's facial landmark detection through its pre-trained models offers an efficient and effective solution for locating facial landmarks, providing valuable information for numerous face-related applications in computer vision and machine learning.

## Chapter 2:

# DB Scan

This project aims to implement the DBSCAN algorithm for anomaly detection. The project will select an appropriate dataset, preprocess the data, implement the DBSCAN algorithm, visualize and evaluate the results, detect anomalies, experiment with parameter tuning, compare DBSCAN with other anomaly detection techniques, discuss potential applications, and summarize the findings.

The project will start by selecting a dataset that is suitable for anomaly detection. The dataset should contain both normal and anomalous data points. If no suitable dataset is available, a synthetic dataset can be created.

The next step is to preprocess the data. This may involve data cleaning, feature selection, normalization, and handling missing values. The goal of data preprocessing is to prepare the data for further analysis.

The DBSCAN algorithm will then be implemented in a programming language of choice. The algorithm consists of the following steps:

Determine the algorithm parameters: Epsilon (ε) and Minimum Points (MinPts).

Compute the distance matrix or use a suitable distance metric.

Determine the core points, border points, and noise points based on the density of the data points.

Assign each data point to a cluster or mark it as noise.

Optionally, perform post-processing steps like merging small clusters or removing noisy outliers.

The clustering results will then be visualized and evaluated. This will involve using appropriate plots such as scatter plots or density-based cluster plots. The performance of the DBSCAN algorithm will be evaluated by comparing the clustering results with the ground truth labels or domain knowledge. Evaluation metrics like Precision, Recall, F1-score, or Silhouette coefficient will be used to measure the algorithm's performance.

The anomalies detected by DBSCAN will then be identified and extracted. The characteristics of the anomalies will be analyzed to gain insights into their nature or potential causes.

Experimentation with parameter tuning will be conducted to observe the impact of the algorithm parameters (Epsilon and MinPts) on the clustering results and anomaly detection performance. The optimal parameter values for the dataset will be found.

DBSCAN will be compared with other popular anomaly detection techniques such as k-means clustering, Isolation Forest, or One-Class SVM. The strengths and weaknesses of DBSCAN in detecting anomalies will be analyzed.

Potential real-world applications where DBSCAN can be applied for anomaly detection will be discussed. Any specific use cases related to the dataset or domain will be presented.

The project findings will be summarized, including the effectiveness of DBSCAN for anomaly detection, the impact of parameter tuning, and insights gained from analyzing the anomalies. Future directions and possible improvements to enhance the anomaly detection process using DBSCAN will be discussed.

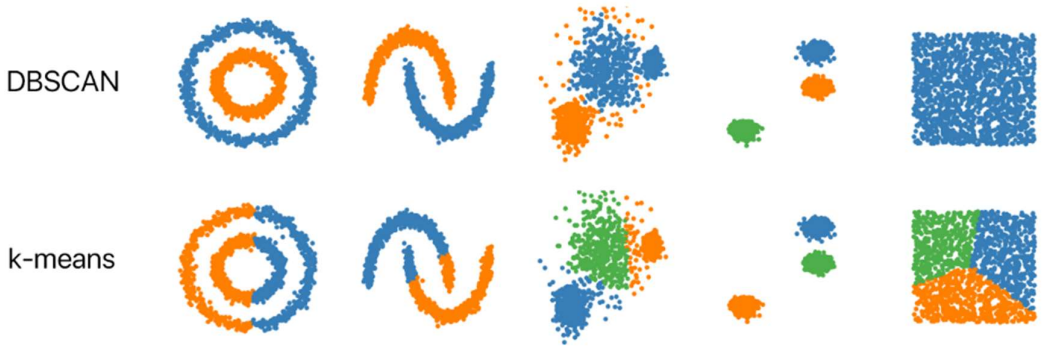
This project outline provides a general structure and guidelines for a project based on DBSCAN for anomaly detection. The content can be adapted and modified to align with specific requirements, dataset, and programming language of choice.

Why we need a density-based clustering algorithm like DBSCAN when we already have k-means clustering:

- K-means is sensitive to outliers. Outliers can significantly impact the clustering results of k-means, as they can be assigned to any cluster. This is because k-means clusters are based on the mean value of the cluster elements, and outliers can skew the mean value. DBSCAN, on the other hand, is more robust to outliers, as it only considers the density of the data points when clustering.
- K-means requires the number of clusters to be known in advance. This can be a problem if the number of clusters is not known or if the number of clusters changes over time. DBSCAN does not require the number of clusters to be known in advance, as it can automatically identify clusters based on the density of the data points.
- DBSCAN can cluster non-convex clusters. K-means can only cluster convex clusters, as it is based on the mean value of the cluster elements. DBSCAN, on the other hand, can cluster non-convex clusters, as it only considers the density of the data points when clustering.

The figure you provided illustrates the fact that DBSCAN can cluster non-convex clusters better than k-means. The figure shows two different data sets, one with convex clusters and one with non-convex clusters. K-means is able to cluster the data set with convex clusters reasonably well, but it fails to cluster the data set with non-convex clusters. DBSCAN, on the other hand, is able to cluster both data sets well.

In summary, DBSCAN is a density-based clustering algorithm that is more robust to outliers, does not require the number of clusters to be known in advance, and can cluster non-convex clusters. These features make DBSCAN a more versatile clustering algorithm than k-means.



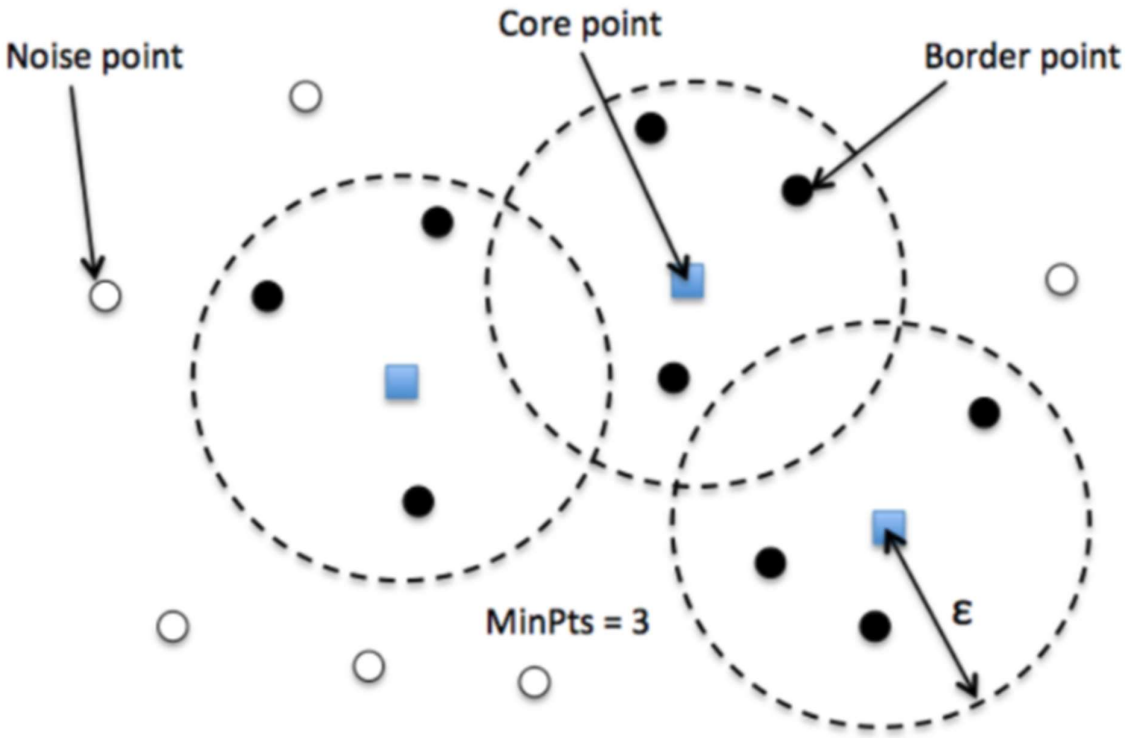
After completing the clustering process with DBSCAN, the data points can be classified into three types:

1. Core points: These points exist in regions of high density and satisfy the criteria of having at least "minPts" neighbors within a distance of "eps." Core points are crucial for forming the core of clusters.

2. Border points: These points are also found in high-density regions but do not meet the requirement of having "minPts" neighbors within "eps." However, they are within the proximity of core points and are considered part of the clusters.

3. Noise points: These points are located in low-density regions and do not have any neighbors within a distance of "eps." Noise points are not assigned to any specific cluster and are typically treated as outliers.

The DBSCAN algorithm operates by first identifying core points and then expanding the clusters around them.



- **Core** — This is a point that has at least *m* points within distance *n* from itself.
- **Border** — This is a point that has at least one Core point at a distance *n*.
- **Noise** — This is a point that is neither a Core nor a Border. And it has less than *m* points within distance *n* from itself.

## The algorithmic steps for DBSCAN clustering:

1. Begin by initializing the parameters of the DBSCAN algorithm: "minPts" (the minimum number of points within ε distance) and "ε" (the radius of the neighborhood).

2. Select an unvisited data point from the dataset in a random manner.3. Calculate the distance between the chosen point and all other points in the dataset.

4. If the count of points within ε distance (including the selected point itself) is less than "minPts," classify the selected point as noise and move on to the next unvisited point.

5. If the count of points within ε distance is equal to or exceeds "minPts," create a new cluster and assign the selected point to that cluster.

6. Expand the cluster by iteratively following these steps:

a) For each point within ε distance of the current point, check if it has been visited.

b) If the neighboring point has not been visited, mark it as visited and calculate the count of points within ε distance from it.

c) If the count of points is equal to or greater than "minPts," include those points in the current cluster.

d) If the neighboring point does not belong to any existing cluster, assign it to the current cluster.

7. Repeat steps 2-6 until all points have been visited.

8. The DBSCAN clustering process is now complete. The points assigned to clusters are considered part of the resulting clusters, while unassigned points are regarded as noise or outliers.

Parameter Estimation :

Every data mining task has the problem of parameters. Every parameter influences the algorithm in specific ways. For DBSCAN, the parameters **ε** and **minPts** are needed.

- **minPts**: As a rule of thumb, a minimum *minPts* can be derived from the number of dimensions *D* in the data set, as $\boldsymbol{minPts \geq D + 1}$. The low
- value $\boldsymbol{minPts = 1}$ does not make sense, as then every point on its own will already be a cluster. With $\boldsymbol{minPts \leq 2}$, the result will be the same as of hierarchical clustering with the single link metric, with the dendrogram cut at height ε. Therefore, *minPts* must be chosen at least 3. However, larger values are usually better for data sets with noise and will yield more significant clusters. As a rule of thumb, $\boldsymbol{minPts = 2 \cdot dim}$ can be used, but it may be necessary to choose larger values for very large data, for noisy data or for data that contains many duplicates.
- **ε**: The value for ε can then be chosen by using a k-distance graph, plotting the distance to the $\boldsymbol{k = minPts - 1}$ nearest neighbor ordered from the largest to the smallest value. Good values of ε are where this plot shows an "elbow": if ε is chosen much too small, a large part of the data will not be clustered; whereas for a too high value of ε, clusters will merge and the majority of objects will be in the same cluster. In general, small values of ε are preferable, and as a rule of thumb, only a small fraction of points should be within this distance of each other.

- **Distance function**: The choice of distance function is tightly linked to the choice of ε, and has a major impact on the outcomes. In general, it will be necessary to first identify a reasonable measure of similarity for the data set, before the parameter ε can be chosen. There is no estimation for this parameter, but the distance functions need to be chosen appropriately for the data set.

The Complexity of DBSCAN :

• Best Case: If an indexing system is used to store the dataset such that neighborhood queries are executed in logarithmic time, we get O(nlogn) average runtime complexity.

• Worst Case: Without the use of index structure or on degenerated data (e.g. all points within a distance less than ε), the worst-case run time complexity remains $O(n^2)$.

Average Case: Same as best/worst case depending on data and implementation of the algorithm

# DBSCAN Clustering Algorithm Solved Example – 1

- Apply the DBSCAN algorithm to the given data points and
- Create the clusters with
- minPts = 4 and
- epsilon (ε) = 1.9.

Data Points:

| | |
|---|---|
| P1: (3, 7) | P2: (4, 6) |
| P3: (5, 5) | P4: (6, 4) |
| P5: (7, 3) | P6: (6, 2) |
| P7: (7, 2) | P8: (8, 4) |
| P9: (3, 3) | P10: (2, 6) |
| P11: (3, 5) | P12: (2, 4) |

P1: P2, P10

P2: P1, P3, P11

P3: P2, P4

P4: P3, P5

P5: P4, P6, P7, P8

P6: P5, P7

P7: P5, P6

P8: P5

P9: P12

P10: P1, P11

P11: P2, P10, P12

minPts = 4 and epsilon (ε) = 1.9

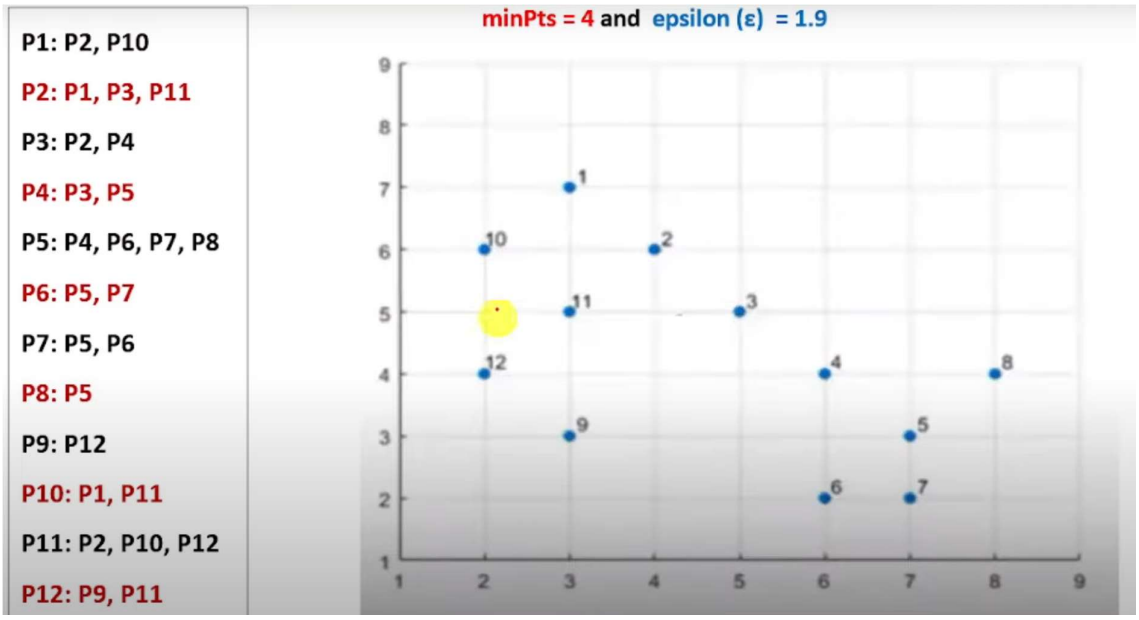
| Point | Status | |
|---|---|---|
| P1 | Noise | Border |
| P2 | Core | |
| P3 | Noise | |
| P4 | Noise | |
| P5 | Core | |
| P6 | Noise | |
| P7 | Noise | |
| P8 | Noise | |
| P9 | Noise | |
| P10 | Noise | |
| P11 | Core | |

minPts = 4 and epsilon (ε) = 1.9

| | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 | P11 | P12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P1 | 0 | | | | | | | | | | | |
| P2 | 1.41 | 0 | | | | | | | | | | |
| P3 | 2.83 | 1.41 | 0 | | | | | | | | | |
| P4 | 4.24 | 2.83 | 1.41 | 0 | | | | | | | | |
| P5 | 5.66 | 4.24 | 2.83 | 1.41 | 0 | | | | | | | |
| P6 | 5.83 | 4.47 | 3.16 | 2.00 | 1.41 | 0 | | | | | | |
| P7 | 6.40 | 5.00 | 3.61 | 2.24 | 1.00 | 1.00 | 0 | | | | | |
| P8 | 5.83 | 4.47 | 3.16 | 2.00 | 1.41 | 2.83 | 2.24 | 0 | | | | |
| P9 | 4.00 | 3.16 | 2.83 | 3.16 | 4.00 | 3.16 | 4.12 | 5.10 | 0 | | | |
| P10 | 1.41 | 2.00 | 3.16 | 4.47 | 5.83 | 5.66 | 6.40 | 6.32 | 3.16 | 0 | | |
| P11 | 2.00 | 1.41 | 2.00 | 3.16 | 4.47 | 4.24 | 5.00 | 5.10 | 2.00 | 1.41 | 0 | |
| P12 | 3.16 | 2.83 | 3.16 | 4.00 | 5.10 | 4.47 | 5.39 | 6.00 | 1.41 | 2.00 | 1.41 | 0 |

- Use Eucladian distance and calculate the distance between each points.

$$Distance(A(x_1, y_1), B(x_2, y_2)) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$




## Conclusion :

In conclusion, density-based clustering algorithms, such as DBSCAN and the Level Set Tree algorithm, offer the advantage of being able to identify clusters of arbitrary shapes and handle datasets with varying densities effectively. This flexibility allows them to capture complex and non-linear structures in the data.

However, it is important to note that tuning these algorithms can be more challenging compared to parametric clustering algorithms like K-Means. The parameters involved, such as epsilon for DBSCAN or the Level Set Tree parameter, may not have straightforward interpretations and can be more difficult to select appropriately. Unlike the number of clusters parameter in K-Means, which has a more intuitive understanding, determining the optimal initial parameter values for density-based clustering algorithms can require more experimentation and expertise.

Therefore, when using density-based clustering algorithms, it is essential to carefully consider and fine-tune the parameters to achieve satisfactory results. It may involve iteratively adjusting the parameters and evaluating the clustering performance until desirable clusters are obtained.

Overall, density-based clustering algorithms provide valuable tools for clustering datasets with complex structures, but their parameter tuning process requires careful consideration and experimentation to achieve optimal results.

# *CHAPTER 3*

## PROPOSED METHODOLOGY

### 3.1 EXISTING SYSTEM

PCA based method which was used earlier for eye/face detection failed as it does not support night driving conditions and has a lower accuracy rate. So to overcome this drawback LBPH algorithm is used The CNN, DNN based technique used for face recognition doesn't support real time face recognition and also has a lower accuracy rate.

**DRAWBACKS:**

- Few of the models don't support night driving conditions or glasses,etc.
- Only few systems support real-time face recognition
- No existing systems including both drowsiness detection system and vehicle security system using face recognition

### 3.2 REQUIREMENT ANALYSIS

#### 3.2.1 HARDWARE

Processor : Intel Core i3.
RAM : 8 GB RAM
Hard Disk : 100 GB
Web cam / In-built laptop camera

#### 3.2.2 SOFTWARE

**Anaconda Navigator**: Anaconda navigator is a graphical consumer interface (gui) this is blanketed within the anaconda distribution, permitting you to effortlessly release packages and control conda applications, environments and channels with out the use of command line commands. Navigator can look for applications in anaconda.Org or the nearby anaconda repository. It's miles to be had for windows, macos and linux.

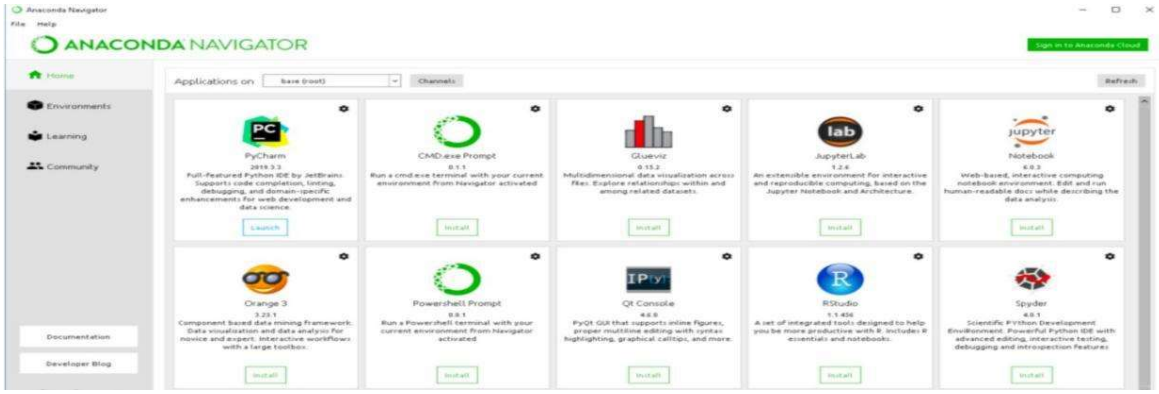



**Figure 3.1:** Anaconda home tab

**Jupyter Notebook:** The jupyter notebook expands the console-primarily based approach to interactive computing in an approximate. new route, Dispensing a bendy internet primarily based application to capture the whole computing method: enlarging, logging and applying code, as well as speaking outcomes. The jupyter pocket book combines two elements:

1. A web application
2. Notebook documents

**Python 3:** Python is the basis of the program that we wrote in the system. It utilizes many of the python libraries.

**Windows/ linux/ Mac OS**

### 3.2.3 FUNCTIONAL REQUIREMENTS

**D-lib:** D-lib is a toolkit for growing actual-world device gaining knowledge of and records analysis packages. It is used to become aware of the face and to mark the facial landmarks which may be used for a wide array of functions. The frontal face detector on the d-lib works properly.

**Open CV:** Opencv is a library the usage of which we will broaden actual-time computer vision programs. It in particular used for photo processing, video seize and assessment in conjunction with capabilities like face detection and object detection. Opencv is the big open-supply libraryUsed for laptop imaginative and prescient and also system gaining knowledge of and

image processing and now it plays a prime function in actual-time operation which can be very critical in current systems. Through the use of it, you could process photographs, videos, real-time motion pictures to understand gadgets, faces, or even handwriting of a human which can be used for a big selection of functions

**Operating System:** An operating system, or "os" is software program that interacts with the hardware and permits one-of-a-kind applications to run. ... Every laptop pc, pill, and cellular telephone consists of an operating system that gives simple functionality for the device. Common desktop jogging structures consist of home windows, os x, and linux.

**Pygame:** Pygame is an open-source cross-platform library for the improvement of multimedia packages like video games made using python. It uses the clean direct media layer library and several different famous libraries to abstract the most common features. It includes cg and sound libraries and it's intended for use with python.

**Numpy:** Numpy is a library that deals with massive multi-dimensional arrays and matrices. It does high-stage mathematical paintings on these mathematical systems. The numpy array is a widespread data shape for handling images, filter kernels and function factors.

**Scipy:** Scipy is an open-source python library that is used to clear up scientific and mathematical problems. It is constructed at the numpy extension and allows the user to control and visualize statistics with a large range of excessive-level commands.

**Imutils:** Imutils are a sequence of functions used to make fundamental photo processing functions which incorporates translation, rotation of images, re-sizing images, skeletonization , and displaying matplotlib pictures much less complex with openCV and each python 2..7 and python .3. It also helps in sorting contours and detecting edges.

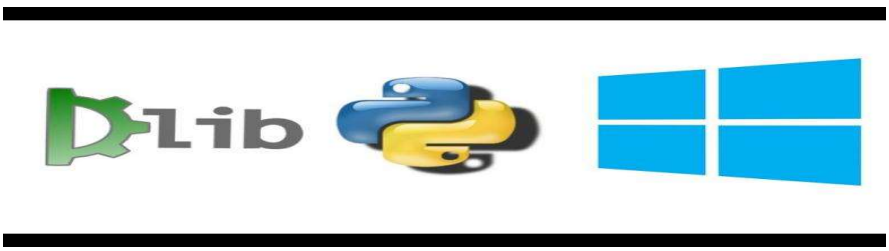



**Figure 3.2:** Functional Requirements

## face_recognition library from dlib :

The `face_recognition` library from dlib is a powerful and widely-used tool for face recognition tasks in Python. It leverages the advanced capabilities of the dlib library to identify and manipulate facial features with high accuracy.

One of the key features of the `face_recognition` library is its face encoding generator. This generator is based on a deep learning model trained on a large dataset of facial images. When given an input image containing one or multiple faces, the encoding generator extracts unique feature vectors, also known as face embeddings, for each detected face.

These face embeddings are numerical representations of facial characteristics that capture distinctive information about a person's face. They encode essential facial traits, such as the spatial arrangement of eyes, nose, and mouth, into a fixed-length vector. Importantly, these embeddings possess the remarkable property of being able to distinguish between different individuals effectively.

To perform face recognition, the encoding generator compares the face embeddings of a new image with a set of known face embeddings stored in a database. By calculating the similarity between these embeddings using a distance metric (e.g., Euclidean distance or cosine similarity), the system can determine if a face in the new image matches any of the known faces in the database.

This encoding-based approach has several advantages, including robustness to variations in lighting, pose, and facial expressions. It allows for efficient and reliable face recognition, even when dealing with large datasets and real-world scenarios.

In conclusion, the face encoding generator from the `face_recognition` library in dlib is a remarkable tool that enables accurate and efficient face recognition by extracting and comparing facial feature vectors. Its integration with dlib makes it a popular choice for face recognition applications in various domains, such as security, biometrics, and human-computer interaction.

## 3.3 Data Preprocessing

In our data processing pipeline, we are working with video data that needs to be converted into images before further analysis can take place. To accomplish this task, we rely on the powerful OpenCV library.

The process involves capturing still frames from the video, and for each frame, we check if it contains any faces. Frames with detected faces are selected for clustering and subsequent processing, while frames without faces are discarded to optimize the workflow.

Once we have gathered the images with faces, we proceed to identify the face locations within each image. To achieve this, we utilize the Haar cascade method, a proven technique for accurate face detection. This approach allows us to pinpoint the exact regions of faces in the images.

To effectively manage this wealth of information, we organize the detected face locations, along with their corresponding image paths, into a structured data frame. This data frame acts as a central repository, enabling us to efficiently analyze and process the facial data further.

## 3.4 Feature extraction:

Each individual's face possesses distinctive and one-of-a-kind features, which play a vital role in enabling unique identification. To achieve precise face recognition, it is essential to extract these features efficiently. These facial characteristics are often represented as feature vectors and plotted in a 128-dimensional space.

By meticulously extracting these facial features, we create a highly informative and rich representation of each face. This 128-dimensional space acts as a unique fingerprint for each individual, capturing the intricate details and variations that make every face special.

The process of mapping faces into this high-dimensional space ensures that even subtle differences are accurately accounted for. It allows us to create a comprehensive and robust face representation that aids in distinguishing one person from another with exceptional accuracy.

This advanced approach of plotting faces in the 128-dimensional space serves as the foundation for modern face recognition systems. It empowers these systems to achieve outstanding performance, making them invaluable tools for a wide range of applications, including security, authentication, and personalized interactions.

## 3.5 UML DIAGRAMS FOR THE PROPOSED METHODOLOGY

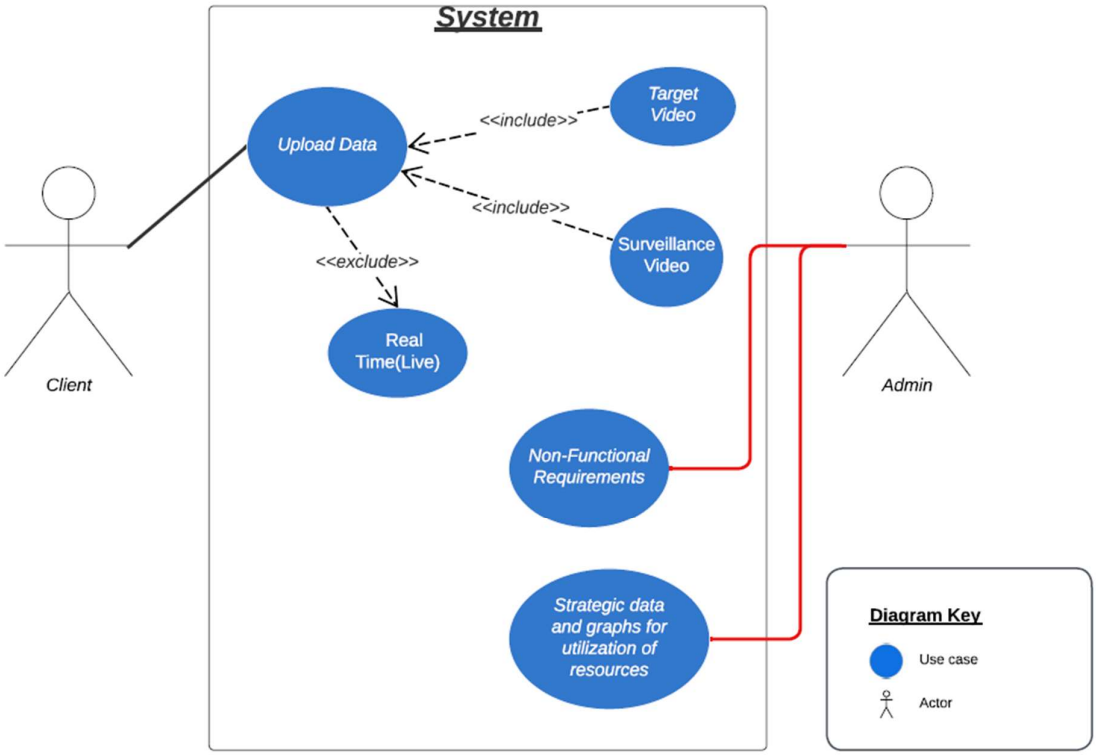



**Figure 3.7:** Use Case diagram

### 3.6 ACTIVITY DIAGRAM

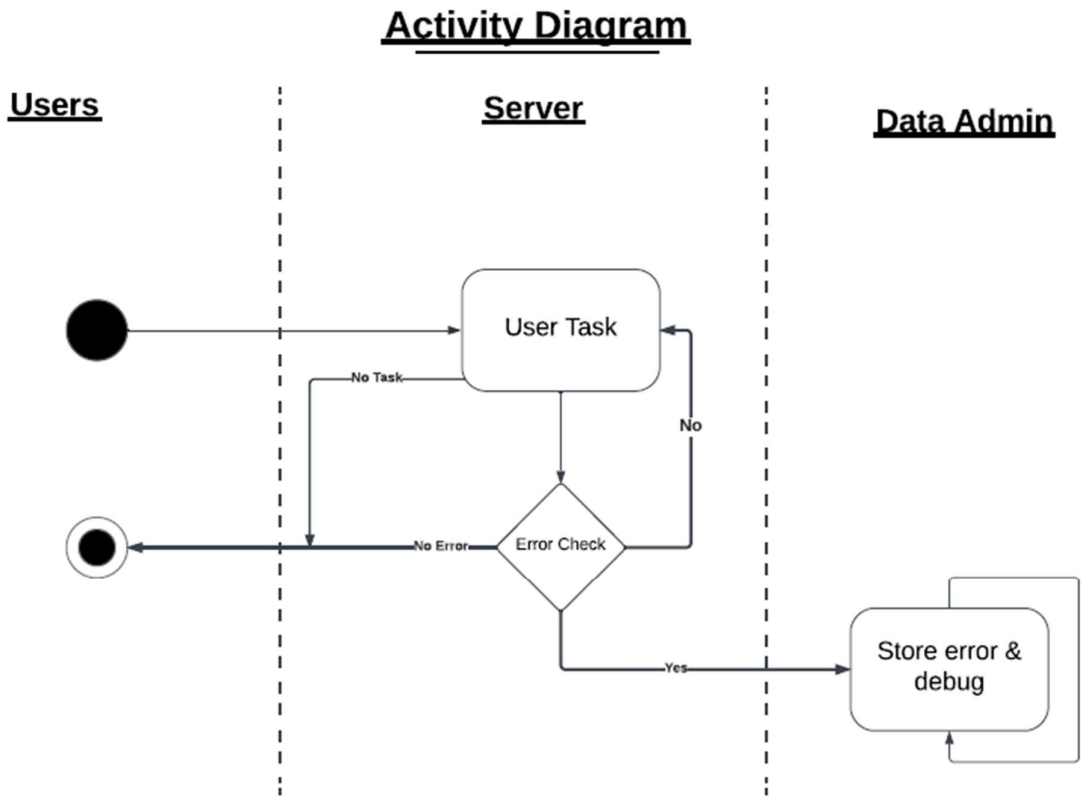



**Figure 3.8:** Activity Diagram

## 3.7 SEQUENCE DIAGRAM

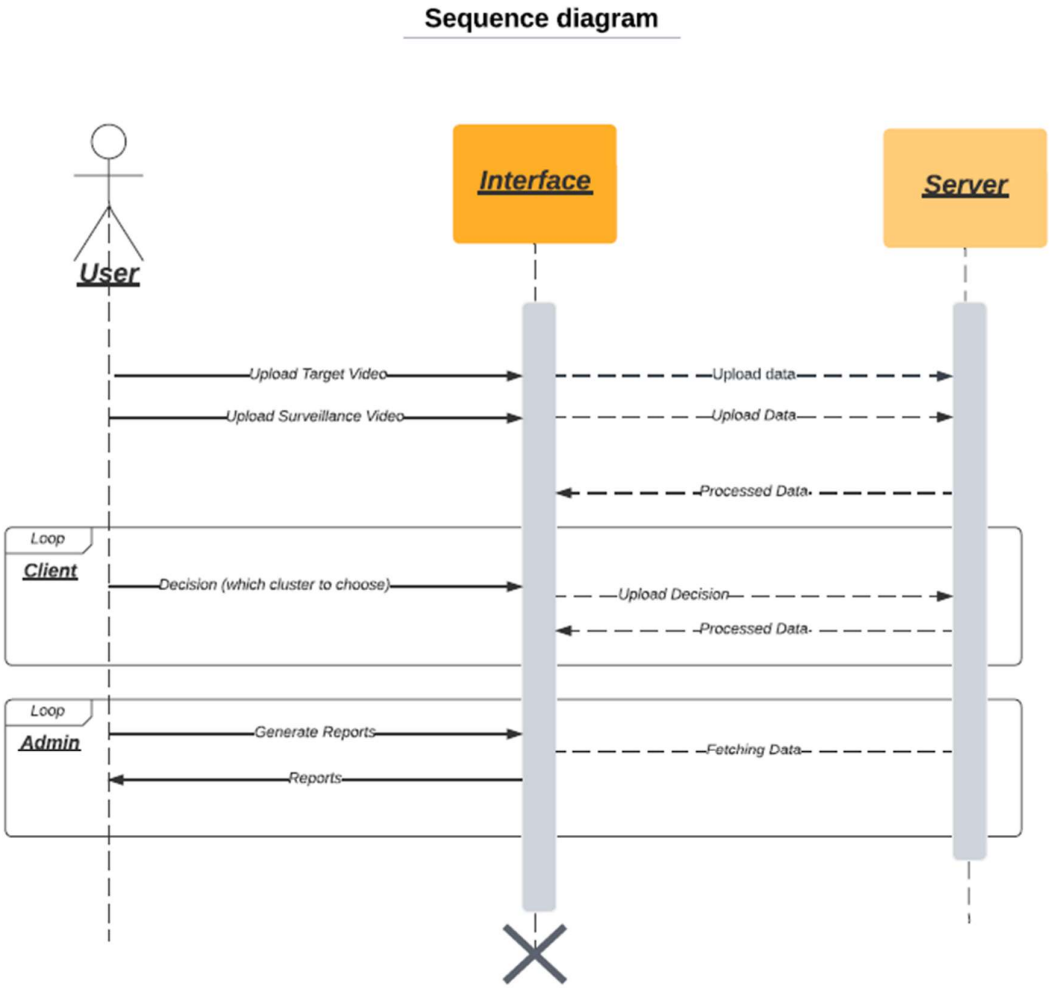



**Figure 3.9:** Sequence Diagram

### 3.8 STATE DIAGRAM

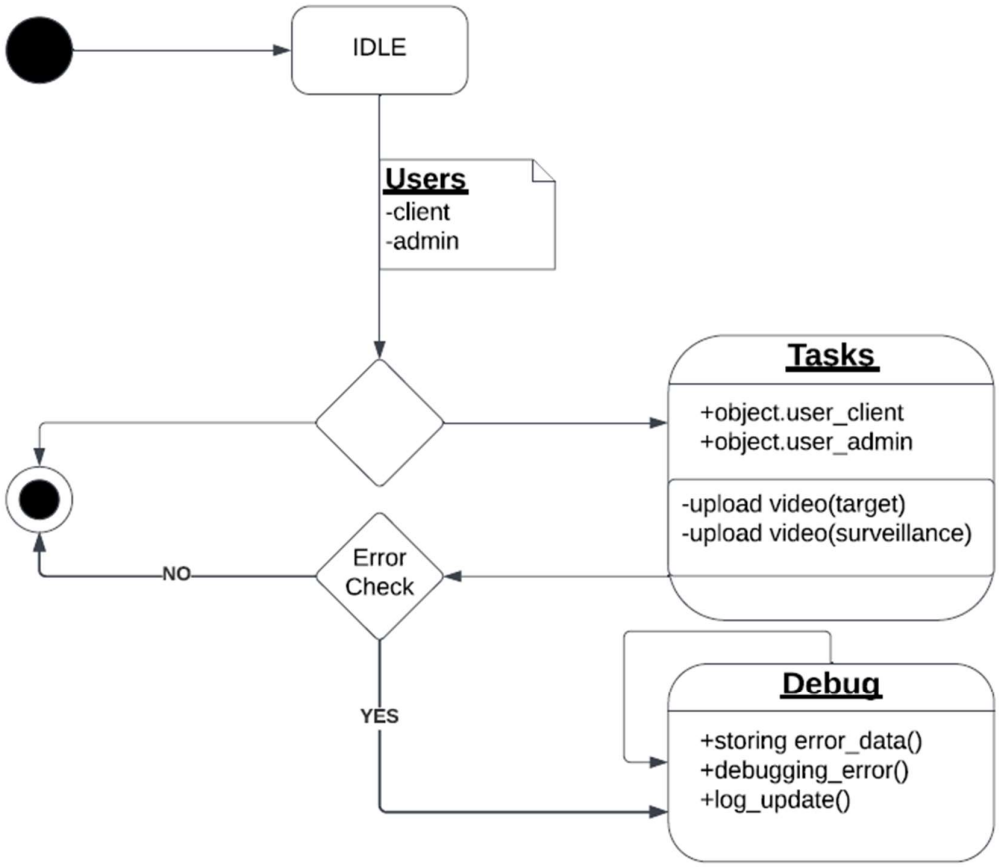



**Figure 3.9:** State Diagram

# Chapter4:

## ANALYSIS of using Hog vs CNN

The primary objective in this analysis is to generate optimal encodings for facial features, crucial for unique image classification.

To achieve this, we leverage dlib's face_recognition module, which allows us to choose the encoding algorithm. Different algorithms yield varying results in terms of quality and processing time.

In this investigation, a comparison is made between two algorithms: Convolutional Neural Network (CNN) and Histogram of Oriented Gradients (HOG).

1. CNN: A deep learning-based approach employing multiple layers to automatically learn and extract features. It produces high-quality encodings but requires more processing time due to its complex architecture.

2. HOG: A traditional computer vision-based approach analyzing gradient orientations distribution. It is computationally less intensive than CNN but may yield lower-quality encodings.

The use of both CNN and HOG enables us to evaluate their individual strengths and weaknesses regarding processing time and encoding quality. The appropriate algorithm choice depends on specific application requirements and available resources.

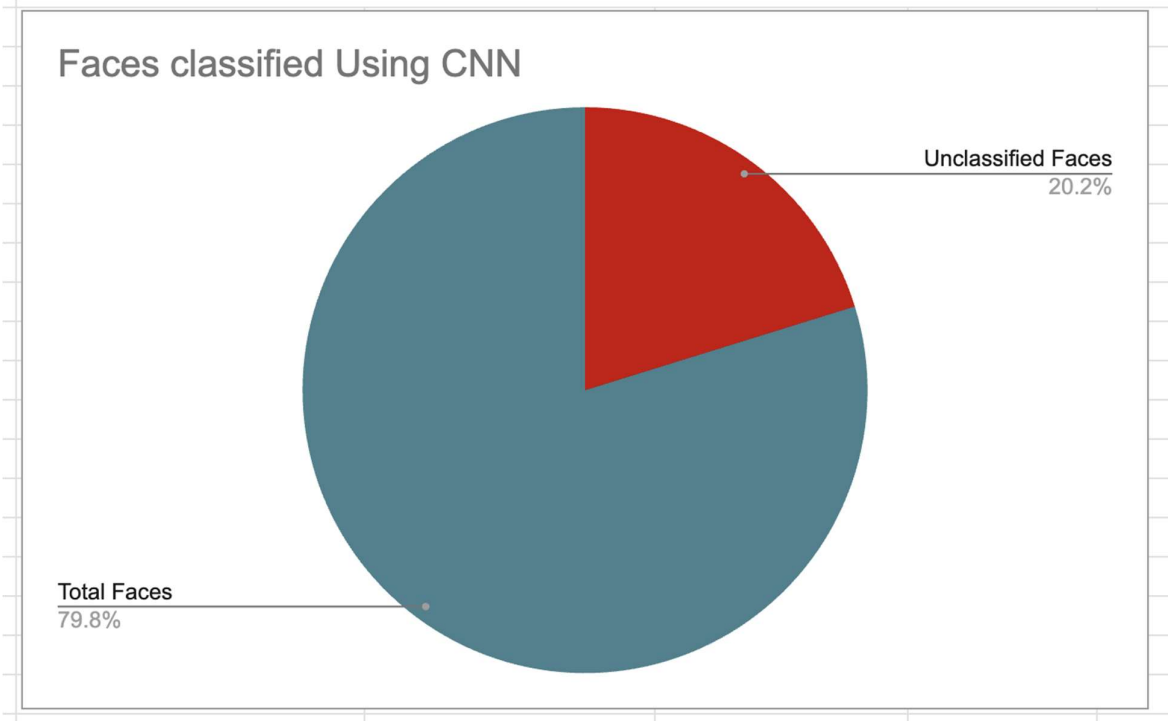
Detailed comparison of both the Algorithms on same Sample sets

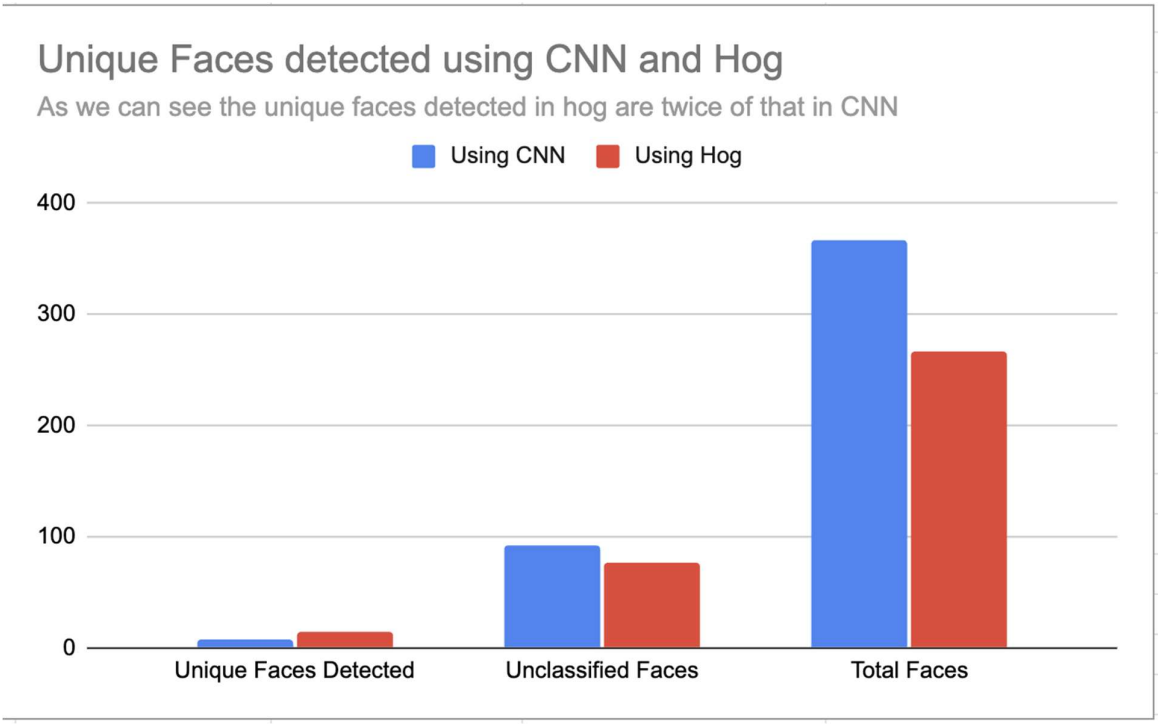
| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Face Detection and Encoding generation | | | | |
| 2 | Sample Length: | 30 sec. + 163 sec. | | 283 Images | |
| 3 | Algorithm: | CNN | HOG | CNN | HOG |
| 4 | Time taken: | 26.92 sec | 102 sec | 21.9 sec | 37.9 sec |
| 5 | Accuracy: | 74% | 71% | 83% | 87% |
| 6 | Unique Faces Detected | 7 | 14 | 14 | 14 |
| 7 | Unclassified Faces | 93 | 77 | 52 | 35 |
| 8 | Total Faces | 367 | 266 | 313 | 289 |
| 9 | Clustering Time Using DBSCAN | 0.0100 sec. | 0.004 sec. | 0.0450 sec. | 0.004 sec. |
| 10 | | | | | |

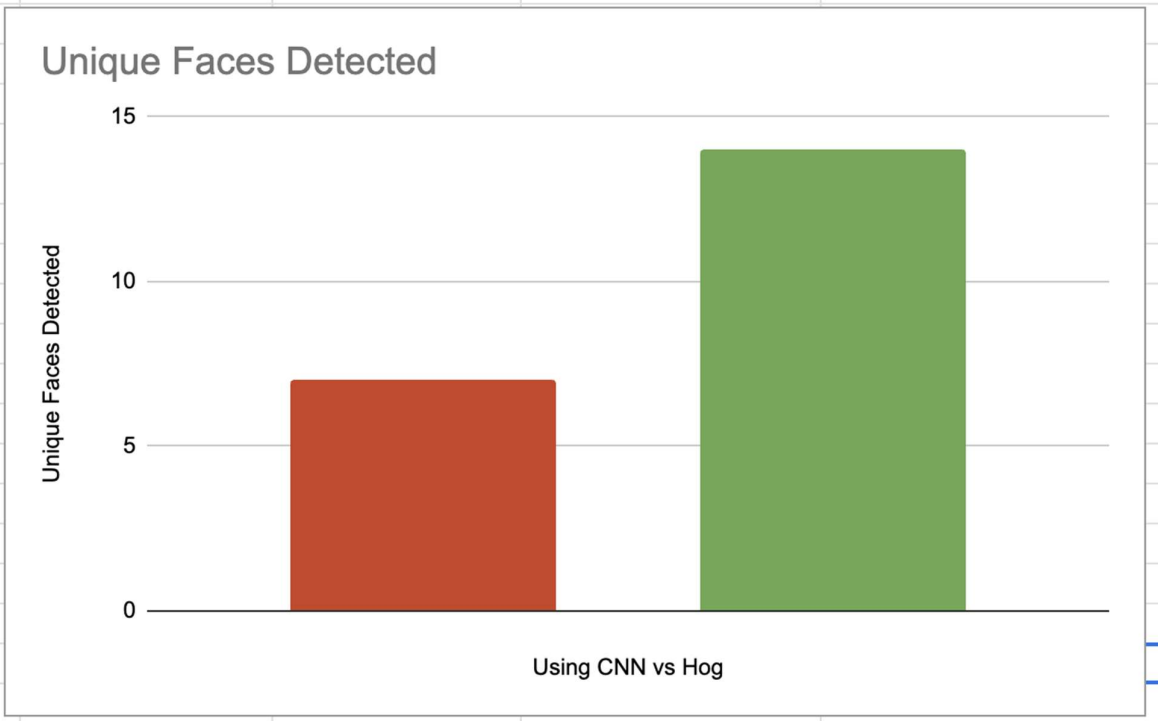
The comparison between CNN and Hog reveals distinct characteristics in terms of processing time and encoding accuracy.

CNN outperforms Hog in terms of processing speed, requiring only 29 seconds compared to Hog's 102 seconds for the same sample set. However, the trade-off is evident in the encoding quality. While CNN takes slightly more time to cluster the encodings, it produces more false positive results, leading to the detection of more faces than Hog.

Despite its faster processing time, CNN's accuracy is compromised, resulting in unclassified faces. In this scenario, there are 93 unclassified faces using CNN, compared to 77 with Hog. Furthermore, Hog demonstrates better performance in accurately classifying unique faces, correctly identifying 14 different individuals, whereas CNN manages to classify only 7 different faces with slight inaccuracies. These findings indicate that for clustering purposes, Hog's encodings perform notably better than CNN.


Faces classified Using CNN
Unclassified Faces
20.2%
Total Faces
79.8%


Unique Faces detected using CNN and Hog
As we can see the unique faces detected in hog are twice of that in CNN
Using CNN
Using Hog
400
300
200
100
0
Unique Faces Detected
Unclassified Faces
Total Faces

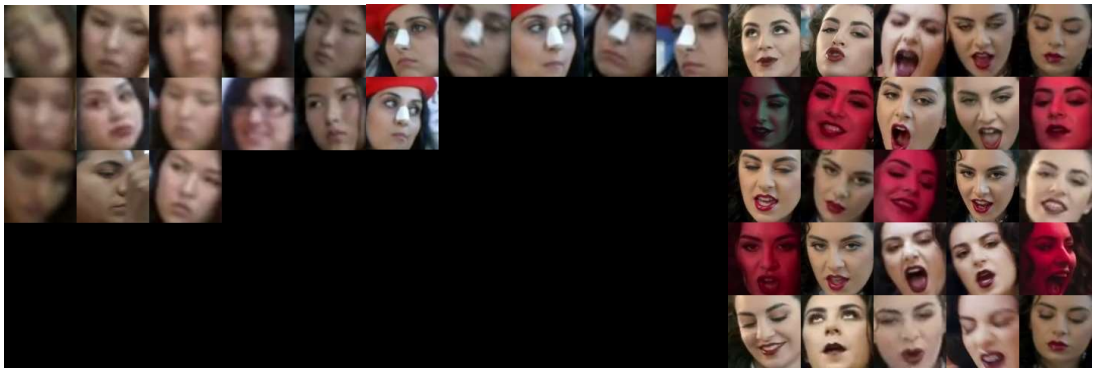

Unique Faces Detected
15
10
5
0
Unique Faces Detected
Using CNN vs Hog

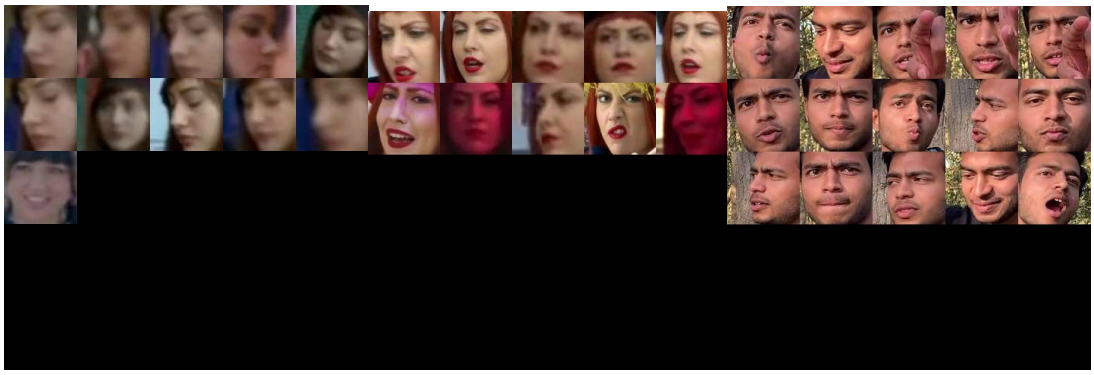
Correct Classification Using CNN

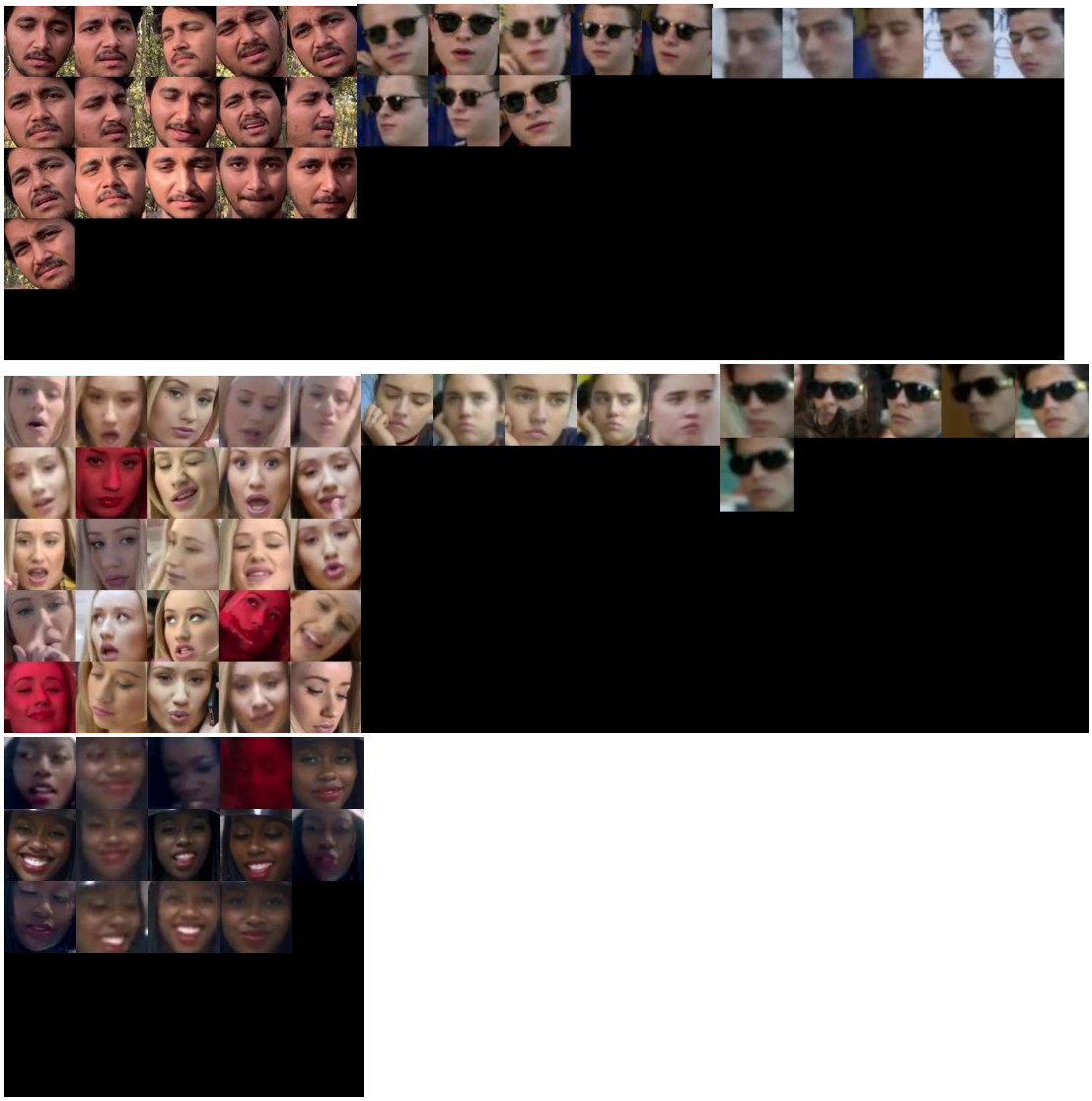
## Incorrect Classification:

## Unclassified Faces In CNN:

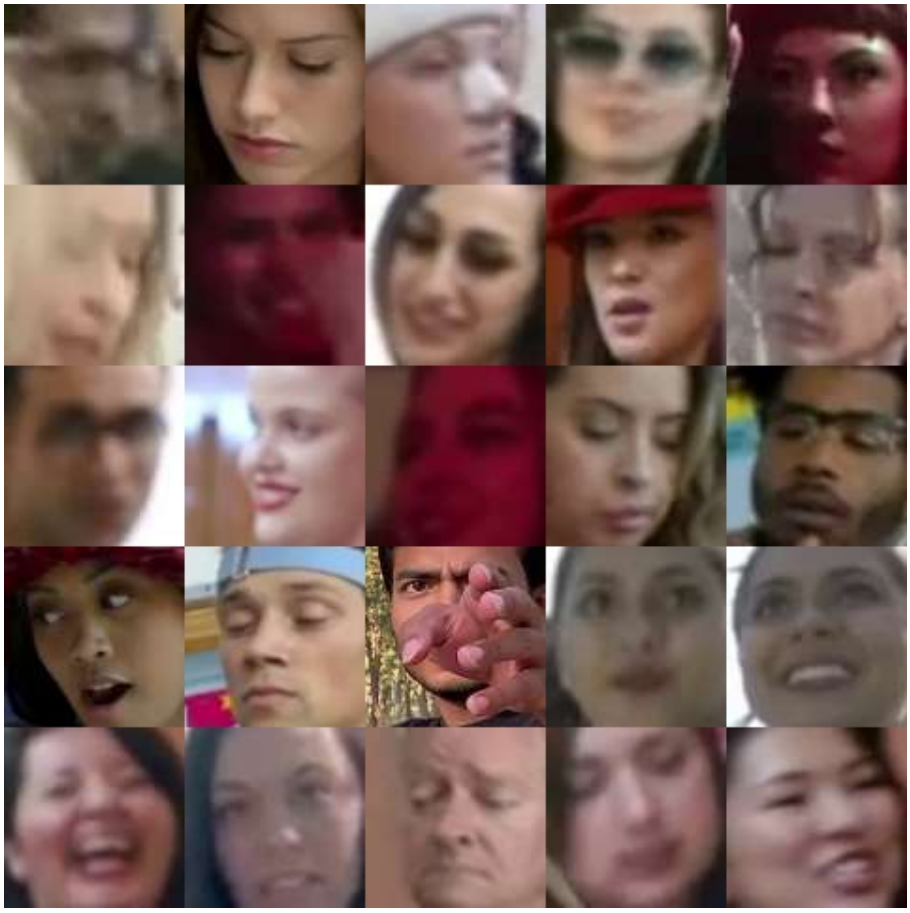
## Classification Using Hog:







## Unclassified Faces:



- **There Is no Incorrect classification in this case**

# Conclusion:

The results clearly demonstrate that Hog generates superior quality encodings compared to CNN. Despite taking more processing time, Hog's accuracy and ability to correctly classify unique faces outweigh the faster processing time of CNN.

Given these findings, Hog can be considered as the preferred choice over CNN for tasks where encoding quality and accurate classification of faces are crucial factors. The marginal increase in processing time with Hog is justified by the significantly improved results it offers, making it a more suitable option for applications where precision and reliability are of utmost importance

# REFERENCE


1) 1Su, H.,& Zheng, G. (2008). A partial least squares regression-based fusion model for predicting the trend in drowsiness. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, *38*(5), 1085-1092.

2) Friedrichs, F., & Yang, B. (2010, June). Camera-based drowsiness reference for driver state classification under real driving conditions. In *2010 IEEE Intelligent Vehicles Symposium* (pp. 101-106). IEEE.

3) Flores, M. J., Armingol, J. M., & de la Escalera, A. (2011). Driver drowsiness detection system under infrared illumination for an intelligent vehicle. *IET intelligent transport systems*, *5*(4), 241-251.

4) Zhang, W., Cheng, B., & Lin, Y. (2012). Driver drowsiness recognition based on computer vision technology. *Tsinghua Science and Technology*, *17*(3), 354-362.

5) Mbouna, R. O., Kong, S. G., & Chun, M. G. (2013). Visual analysis of eye state and head pose for driver alertness monitoring. *IEEE transactions on intelligent transportation systems*, *14*(3), 1462-1469.

6) Tadesse, E., Sheng, W., &Liu, M. (2014, May). Driver drowsiness detection through HMM based dynamic modeling. In *2014 IEEE International conference on robotics and automation (ICRA)* (pp. 4003-4008). IEEE.

7) García, F., de la Escalera, A., & Armingol, J. M. (2014). Driver monitoring based on low-cost 3-D sensors. *IEEE Transactions on Intelligent Transportation Systems*, *15*(4), 1855-1860

# MissingLink.docx

ORIGINALITY REPORT

14 %
SIMILARITY INDEX

PRIMARY SOURCES

| | | |
|---|---|---|
| 1 | www.coursehero.com<br>Internet | 206 words — 3% |
| 2 | www.ijraset.com<br>Internet | 67 words — 1% |
| 3 | webhostinggeeks.com<br>Internet | 63 words — 1% |
| 4 | Tarun Telang. "Chapter 8 MicroProfile JSON Web Tokens and Jakarta Security", Springer Science and Business Media LLC, 2023<br>Crossref | 31 words — 1% |
| 5 | dev.to<br>Internet | 31 words — 1% |
| 6 | dspace.daffodilvarsity.edu.bd:8080<br>Internet | 28 words — < 1% |
| 7 | Muhammad Zarar, Yulin Wang. "Early Stage Diabetes Prediction by Approach Using Machine Learning Techniques", Research Square Platform LLC, 2023<br>Crossref Posted Content | 26 words — < 1% |
| 8 | Zhu, Taoyuanmin. "Design of a Highly Dynamic Humanoid Robot", University of California, Los Angeles, 2023<br>ProQuest | 26 words — < 1% |

| | | |
|---|---|---|
| 9 | medium.com<br>Internet | 24 words — < 1% |
| 10 | blog.logrocket.com<br>Internet | 22 words — < 1% |
| 11 | docplayer.net<br>Internet | 22 words — < 1% |
| 12 | ijsret.com<br>Internet | 22 words — < 1% |
| 13 | ir.lib.uwo.ca<br>Internet | 19 words — < 1% |
| 14 | Devlin Basilan Duldulao, Ruby Jane Leyva Cabagnot. "Practical Enterprise React", Springer Science and Business Media LLC, 2021<br>Crossref | 17 words — < 1% |
| 15 | safespot-eu.org<br>Internet | 17 words — < 1% |
| 16 | github.com<br>Internet | 16 words — < 1% |
| 17 | www.facweb.iitkgp.ernet.in<br>Internet | 16 words — < 1% |
| 18 | doaj.org<br>Internet | 15 words — < 1% |
| 19 | develop.finki.ukim.mk<br>Internet | 14 words — < 1% |
| 20 | novapublishers.com<br>Internet | |

| | | |
|---|---|---|
| | | 14 words — < 1% |
| 21 | pages.cs.wisc.edu<br>Internet | 13 words — < 1% |
| 22 | scholarworks.lib.csusb.edu<br>Internet | 13 words — < 1% |
| 23 | stackoverflow.com<br>Internet | 13 words — < 1% |
| 24 | themeforest.net<br>Internet | 13 words — < 1% |
| 25 | websta.me<br>Internet | 13 words — < 1% |
| 26 | sleeknote.com<br>Internet | 12 words — < 1% |
| 27 | harat-zutut.balticecovillages.eu<br>Internet | 11 words — < 1% |
| 28 | isgb.otago.ac.nz<br>Internet | 11 words — < 1% |
| 29 | www.slideshare.net<br>Internet | 11 words — < 1% |
| 30 | codeclimate.com<br>Internet | 10 words — < 1% |
| 31 | www.freecodecamp.org<br>Internet | 10 words — < 1% |
| 32 | www.ijisrt.com | |

Internet

10 words — < 1%

33 www.neliti.com
Internet

10 words — < 1%

34 Bacchuwar, Sanket Shrikant. "X-Ray Computed Tomography for Analysis of Gangue Rejection in Gravity Preconcentration of Low Grade Sulfide Ores", The University of Utah, 2023
ProQuest

9 words — < 1%

35 m.mu.edu.sa
Internet

9 words — < 1%

36 Martins, Tiago Alexandre Pinheiro. "Internship at be One Solutions", Instituto Politecnico de Leiria (Portugal), 2022
ProQuest

8 words — < 1%

37 Philip Japikse, Kevin Grossnicklaus, Ben Dewey. "Building Web Applications with .NET Core 2.1 and JavaScript", Springer Science and Business Media LLC, 2020
Crossref

8 words — < 1%

38 Tapas Si, Debolina Bhattacharya, Somen Nayak, Péricles B.C. Miranda, Utpal Nandi, Saurav Mallik, Ujjwal Maulik, Hong Qin. "PCOBL: A Novel Opposition-based Learning Strategy to Improve Metaheuristics Exploration and Exploitation for Solving Global Optimization Problems", IEEE Access, 2023
Crossref

8 words — < 1%

39 alanxelsys.com
Internet

8 words — < 1%

40 pubmed.ncbi.nlm.nih.gov
Internet

8 words — < 1%

41 www.gadgetsnow.com
Internet

8 words — < 1%

| | | | |
|---|---|---|---|
| EXCLUDE QUOTES | OFF | EXCLUDE SOURCES | OFF |
| EXCLUDE BIBLIOGRAPHY | ON | EXCLUDE MATCHES | OFF |