# ML PROJECT REPORT

## Smart Photo Drive

***Submitted in partial fulfilment of the requirements for the degree of***

**Bachelor of Engineering**

**In**

**Computer Science and Business Systems**

***Submitted By***


| | |
|---|---|
| **Arsh Chatrath** | **102318063** |
| **Rhea Singhal** | **102318040** |


**Submitted to**

**Dr. Ashutosh Aggarwal (Assistant Professor - III, CSED)**

**Mr. Amardeep Singh (Assistant Professor, CSED)**

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)


**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY**

**PATIALA – 147004**


**December 2025**

**Table of Contents**

# PROJECT REPORT

## Smart Photo Drive – AI-BASED PHOTO MANAGEMENT AND ANALYSIS SYSTEM

# 1. Introduction

The rise of digital imaging tools, including smartphones, tablets, webcams, and DSLRs, has transformed how people collect visual information.

As a result, most users now store thousands of photos across different storage options, including local hard drives, cloud services, and portable devices. These extensive photo collections are often disorganised, with inconsistent file names, missing metadata, and no clear way to group or manage them. Finding a specific image, sorting photos by person, or removing duplicates can be a time-consuming task.

Traditional methods, like organising photos into folders by event or renaming files by date, are not efficient enough for large sets of images.

This project, called Smart Drive, aims to improve and automate the way large image collections are managed using modern artificial intelligence techniques.

It combines several functions like face detection, face encoding, similarity matching and clustering into one streamlined process that can be accessed through an easy-to-use web interface. At its heart, the system uses the face_recognition library (based on dlib) to create unique face embeddings and group images by person based on how similar the faces are.

The main goal of this system is to take over the work of organising images manually, making the process faster and more efficient.

It not only sorts images by people but also creates additional content like collages, slideshows, and resized versions of the photos. By using a database for storing information and a checkpointing system, the project ensures that the process is reliable and can work with large amounts of data.

# 2. Background

## 2.1 Growth of Digital Image Collections

The progress of smartphone technology has made it easier for people to take photos than ever before.

Global surveys show that more than a trillion images are created every year. However, most of these images are kept on personal devices without any labels or organisation. Research shows that it's hard for people to manage large numbers of photos manually, which makes automated photo management systems more important than ever.

## 2.2 Evolution of Automated Face Recognition

Face detection and recognition have gone through several stages of development:

1. Early Geometry - Based Models -
   The earliest methods focused on the relationships between facial features like the eyes, nose, and chin.

2. Eigenfaces and PCA-Based Recognition -
   These methods used principal component analysis to reduce the size of facial images, which made the process faster.

3. HOG-Based Detection and SVM Classification -
   This is a traditional method that uses the Histogram of Oriented Gradients (HOG) combined with linear SVM classifiers to find facial areas.
   It offers a good balance between how fast it works and how accurate it is.

4. Deep Learning Models-
   Modern deep CNN models like FaceNet, VGGFace2, and ArcFace have greatly improved accuracy by using high-dimensional data.
   However, they need powerful hardware and a lot of computing resources.

The current project uses the HOG + SVM model for detection because it works well on regular computers and is easy to use.

It also takes advantage of the face_recognition library, which is a widely used tool for face recognition.

### 2.3 Need for an Integrated Photo Management Solution

Although services like Google Photos offer cloud-based face clustering, they operate as black boxes and need a constant internet connection.
Most open-source tools don't combine face recognition, duplicate detection, and image editing into a single, seamless workflow. This project fills that gap by offering a modular, transparent, and open system that users can run entirely on their own machines, without giving up their privacy.

The goals of the project include:

1. Implementing an accurate and efficient face recognition system
2. Automating dataset organisation with minimal user involvement
3. Optimising the workflow for large-scale image collections
4. Ensuring high recognition accuracy even with variations in pose, lighting, and quality

## 3. Methodology

The proposed Face Recognition–Based Image Separator consists of several clearly defined modules, each responsible for executing a specific stage of the processing pipeline. The modular design ensures scalability, easy debugging, and flexibility for future extensions. The components are described in detail below.

### 3.1. Input Modules

#### a. People/ Folder - Reference Image Repository

This folder contains the **reference images** used to build the database of known individuals. Each image must contain **exactly one face**, allowing the system to extract a clean and accurate face encoding.

- The **filename (without extension)** is treated as the person's name.
- The system preprocesses these images once and stores their encodings for all future comparisons.

**b. Dataset/ Folder - Target Image Collection**

This folder contains all unorganized images that the system must evaluate and classify.

- Images may include single individuals, multiple people, unknown faces, or even photos without faces.

- These images undergo the full pipeline of preprocessing, detection, encoding, and classification.

## 3.2. Preprocessing Unit

Before any face-related computation, each image is standardized to ensure consistent performance and fast processing.

**a. Image Resizing (20%)**

All images are resized to 20% of their original resolution using bilinear interpolation.

- Purpose: Reduce computational cost by ~96% while preserving essential facial structure.

- Impact: Significantly speeds up face detection and encoding without noticeably degrading accuracy.

**b. RGB Color Conversion**

The system converts all images from BGR (OpenCV default) to RGB format, ensuring compatibility with the face_recognition library, which expects RGB input.

**c. Noise Reduction (Optional Enhancement)**

Mild denoising may be applied to images with artifacts or low light, ensuring smoother gradient patterns and improving detection reliability.

## 3.3. Face Detection Unit

This module identifies and localizes faces within an image using HOG-based or CNN-based models.

**a. Bounding Box Extraction**

The system returns a list of bounding boxes that outline each detected face.
Each box follows the coordinate structure:

(top,right,bottom,left)(top, right, bottom, left)(top,right,bottom,left)

**b. Face Count Determination**

The detection step is used to classify images early:

- 0 faces → Unknown category
- >1 faces → Group category
- 1 face → Proceed to encoding and matching

This pre-classification reduces unnecessary computation for group and unknown images.

## 3.4. Face Encoding Unit

Once a face has been detected, the system converts it into a numerical representation.

128-Dimensional Embedding Generation

Each face is passed through a deep CNN (based on ResNet architecture) that converts facial features into a 128-dimensional embedding vector.

- Each dimension represents learned facial characteristics (e.g., jawline, eye spacing, skin texture).
- Encodings of the same person cluster together in the embedding space.
- These vectors form the basis for all face comparisons.

If multiple faces exist in the same image, an embedding is generated for each face individually.

## 3.5. Matching & Classification Module

This component determines whether a detected face belongs to a known person, an unknown individual, or a group.

**a. Embedding Comparison**

Using Euclidean distance, each unknown face encoding is compared against all known encodings stored in the database.

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2}$$

**b. Threshold-Based Decision**

If the minimum distance is below the tolerance value (0.75 by default), the system classifies it as a Known Person.
Otherwise, it is marked as Unknown.

**c. Classification Categories**

- Known Person:
  Exactly one face, and the best match distance < threshold.
- Group Image:
  More than one face detected, regardless of identity.
- Unknown Image:
  One or more faces but none match known individuals.

This structured classification ensures that the dataset is neatly organized based on identity and image content.

## 3.6. Output File Organizer

After classification, each image is automatically moved to the appropriate output folder.

**a. output/PersonName/**

Images of recognized individuals are stored under folders named after the person.

- These folders are created dynamically as new identities are encountered.

**b. output/Group/**

All images containing multiple people are grouped here.

- Useful for team photos, event pictures, or mixed groups.

**c. output/Unknown/**

Images with unrecognized faces—or no detectable faces—are stored in this folder.

- Helps isolate images requiring manual review or new reference samples.

The organizer ensures that end users receive a clean, structured, and intuitive directory of results.

## 3.7. Serialization Layer

This module ensures efficiency by avoiding repeated computation of known face encodings.

**a. Pickle-Based Encoding Storage**

All known person encodings are stored in a serialized file:
known_encodings.pickle

This file includes:

- List of embeddings
- Corresponding names
- Associated metadata

**b. Performance Optimization**

Loading encodings is 100× faster than recomputing them.
This is especially useful when the People/ folder grows or when processing thousands of images.

**c. Persistent Face Database**

The pickle file acts as a lightweight database, allowing the system to store and reuse identification knowledge across runs.

# 4.Edge cases and error handling

## 4.1. No Face Detected

This situation arises when the system fails to locate any face in the image.
**Possible reasons:**

- Very low image resolution
- Insufficient lighting or strong shadows
- Face turned too far sideways (profile angle)

**Handling strategy:**
If no encodings are generated (len(encs) == 0), the image is skipped and marked as *Unknown*, ensuring the pipeline continues without interruption.

## 4.2. Multiple Faces in Reference Images

Reference images in the *People/* folder are expected to contain exactly one face for accurate identity mapping.
If multiple faces appear:

- Different embeddings are created for different people.
- Embeddings are checked for similarity individually.

This maintains consistency and avoids contamination of the known-encodings database.

## 4.3. Low Similarity Scores

Low similarity occurs when the Euclidean distance between known and unknown encodings is high, often due to:

- Variations in lighting or angle
- Insufficient or poor-quality reference images

**Solutions:**

- Add more reference images per person (different angles/lighting)
- Adjust the tolerance value (e.g., from 0.75 → 0.8) to allow more flexible matching

This improves recognition accuracy without affecting efficiency.

## 4.4. Misclassifications

Incorrect identification may happen when:

- The face in the image is blurred
- The subject is wearing sunglasses, masks, or hats
- Facial features are partially occluded

**Improvement approach:**
Implementing **face alignment** techniques (e.g., eye-centred alignment) significantly enhances embedding quality and reduces errors.

## 4.5. Confusion Matrix & Performance Metrics

4.5 Confusion Matrix & Performance Metrics

The following confusion matrix was created using the classified results for the quantitative evaluation of the face recognition system:

| | Actually Positive (1) | Actually Negative (0) |
|---|---|---|
| Predicted Positive (1) | 17 | 3 |
| Predicted Negative (0) | 5 | 10 |

The confusion matrix includes:

**True Positives: TP = 17**, faces correctly identified as the correct person.

**False Positives: FP = 3,** Faces have been incorrectly identified as a known person.

**False Negatives: FN = 5**: known faces which were not identified by the system (marked as Unknown or incorrect).

**True Negatives: TN = 10,** Images correctly classified as not belonging to the known individual.

These values reflect both the strengths and the areas where the model can improve.

### 4.5.1 Accuracy

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{17 + 10}{17 + 10 + 3 + 5} = \frac{27}{35} \approx 0.7714$$

Accuracy ≈ 77.14%

### 4.5.2 Precision

Precision = 85%

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{17}{17 + 3} = \frac{17}{20} = 0.85$$

This indicates that misidentification is relatively low.

#### 4.5.3 Recall

Recall measures how often it successfully recognises known individuals.

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{17}{17 + 5} = \frac{17}{22} \approx 0.7727$$

Recall ≈ 77.27%

**Interpretation**

- The high precision of 85% means that the system can be reliable in identifying one's identity, with few misclassifications.
- The recall value, 77.27%, suggests that some known faces are being missed. This is probably because of changes in lighting, angle, or even limited reference images.
- An overall accuracy of 77.14% reflects the strong baseline performance of a single-embedding FR system.

These metrics, along with the confusion matrix, provide insight into where improvements-including the addition of reference images, editing of tolerance thresholds, or the use of face alignment-can be made to further improve recognition performance..

# 5. Conclusions and Future Work

The SmartDrive system shows how an integrated, AI-driven approach can be used for organizing photos automatically.

It uses classical face detection and encoding methods, applies clustering based on similarity, and includes useful tools like resizing, removing duplicates, and creating multimedia.

Combining a modern front-end framework with a scalable back-end helps keep the system user-friendly while maintaining good performance.

Possible future improvements include:

1. Switching to deep CNN-based face detectors to boost accuracy.
2. Using perceptual hashing to find near-duplicates.
3. Allowing users to manually label photos and replace the "Unknown" naming system.
4. Adding GPU support to make encoding faster.
5. Including cloud storage and access across multiple devices.
6. Introducing an "event clustering" feature to group photos by time, location, or context.

These changes would improve the system's usefulness, accuracy, and ease of use, making it a more advanced and complete photo management tool.

## 6. References

**[1]** A. Geitgey, "face_recognition: The world's simplest facial recognition api for Python and the command line," GitHub. [Online]. Available: https://github.com/ageitgey/face_recognition.

**[2]** "dlib C++ Library," dlib.net. [Online]. Available: http://dlib.net/.

**[3]** "OpenCV: Open Source Computer Vision Library," OpenCV. [Online]. Available: https://opencv.org/.

**[4]** "Face Recognition — face_recognition 1.3.0 documentation," face-recognition.readthedocs.io. [Online]. Available: https://face-recognition.readthedocs.io/.

**[5]** "Histogram of Oriented Gradients," scikit-image. [Online]. Available: https://scikit-image.org/docs/stable/auto_examples/features_detection/plot_hog.html.

**[6]** "Python Pickle Module — Python Documentation," Python.org. [Online]. Available: https://docs.python.org/3/library/pickle.html.