Technical University of Košice
Faculty of Electrical Engineering and Informatics

Face Recognition in Camera Footage

Bachelor's Thesis

2025 **Yurii Murha**

Technical University of Košice
Faculty of Electrical Engineering and Informatics

Face Recognition in Camera Footage

Bachelor's Thesis

Study Programme:	Intelligent Systems
Field of study:	5.2.13 Electronics
Department:	Department of Cybernetics and Artificial Intelligence (KKUI)
Supervisor:	Ing. Ján Magyar, PhD.
Consultant(s):	Ing. Ján Magyar, PhD.

Košice 2025 **Yurii Murha**

Errata

Face Recognition in Camera Footage

Yurii Murha

Košice 2025

Abstract

Face recognition in surveillance systems represents a rapidly advancing field at the intersection of artificial intelligence, computer vision, and security technology. This thesis presents a comprehensive study of face detection and recognition methods, with a focus on their application in real-time camera surveillance. The work begins with an overview of the theoretical foundations, including classical approaches such as Eigenfaces, Fisherfaces, and Local Binary Patterns, as well as modern deep learning techniques like Convolutional Neural Networks (CNNs), Histogram of Oriented Gradients (HOG), and Deep Metric Learning. The thesis systematically compares open-source libraries and frameworks, including OpenCV, dlib, and TensorFlow, and evaluates their performance on custom datasets collected from webcams and security cameras. A custom dataset was created and manually annotated to reflect real-world conditions, and data augmentation was applied to improve model robustness. The practical part of the thesis details the development of a modular face recognition system, integrating detection, recognition, and database management components. Experimental results demonstrate the strengths and limitations of various algorithms in terms of accuracy, speed, and robustness to challenging conditions such as lighting, pose, and occlusion. The thesis also addresses ethical and privacy considerations relevant to the deployment of biometric systems. The findings contribute to the understanding of current capabilities and challenges in face recognition for surveillance, providing guidance for future research and practical implementation in security applications.

Keywords

TensorFlow, Python, OpenCV, Face Recognition, Camera, Surveillance

Declaration

I hereby declare that this thesis is my own work and effort. Where other sources of information have been used, they have been acknowledged.

Košice, May 30, 2025 .....

Signature

Acknowledgement

I would like to express my sincere gratitude to Ing. Ján Magyar, PhD, for his invaluable support and guidance in his dual role as both my supervisor and consultant. His constant and constructive feedback was instrumental throughout this study.
I also want to thank Kanye West for his great songs that provided much-needed motivation for my Graduation.
My special thanks also go to Arsen Markaryan for the encouragement.
The Victory is gonna be ours!

Preface

This thesis addresses the development and evaluation of face recognition systems for surveillance applications. The work was motivated by the increasing demand for intelligent, automated security solutions capable of real-time identification and monitoring. The main goal was to explore, implement, and compare state-of-the-art algorithms and practical approaches for face detection and recognition in real-world camera streams.

The thesis was created as part of the Intelligent Systems study program at the Faculty of Electrical Engineering and Informatics, Technical University of Košice. The topic was chosen due to the growing importance of artificial intelligence and computer vision in modern security and public safety. The work combines theoretical research with practical implementation, including dataset creation, model training, and system integration.

The main methods used include deep learning, data augmentation, and benchmarking of multiple face recognition libraries. The thesis also discusses ethical and privacy considerations relevant to deploying such systems in practice.

Contents

List of Figures

List of Tables

List of Symbols and Abbreviations

FRS Facial Recognition Software

AI Artificial Intelligence

IoU Intersection over Union

CSV Comma-Separated Values

JSON JavaScript Object Notation

RTSP Real Time Streaming Protocol

CD Compact Disc

GUI Graphical User Interface

API Application Programming Interface

GPU Graphics Processing Unit

CPU Central Processing Unit

GDPR General Data Protection Regulation

CCPA California Consumer Privacy Act

CPRA California Privacy Rights Act

MTCNN Multi-task Cascaded Convolutional Networks

HOG Histogram of Oriented Gradients

CNN Convolutional Neural Network

ROC Receiver Operating Characteristic

FAR False Acceptance Rate

FRR False Rejection Rate

EER Equal Error Rate

TPR True Positive Rate

FPR False Positive Rate

FN False Negative

TP True Positive

FP False Positive

TN True Negative

API Application Programming Interface

OpenCV Open Source Computer Vision Library

TensorFlow Open-source machine learning framework

Albumentations Data augmentation library for images

Labelme Image annotation tool

ResNet Residual Neural Network

FaceNet Deep learning model for face recognition

Rekognition Amazon cloud-based image/video analysis service

Introduction

This thesis addresses the problem of face recognition in camera footage. The motivation for this work stems from the increasing demand for automated attendance and security systems capable of identifying individuals in real-time video streams. The main objective is to design, implement, and evaluate a modular face recognition system that leverages both custom deep learning models and state-of-the-art libraries.

The thesis is structured as follows:

- An overview of the tools and libraries used for face recognition, including both custom and pre-built solutions.
- A detailed description of the dataset creation, annotation, and augmentation process.
- The development and evaluation of deep learning models for face detection and recognition.
- Integration of the system components into a real-time attendance application.
- Comparative analysis of different face detection and recognition methods.

The following chapters provide a comprehensive account of the methods, implementation, and results achieved in this work.

Background

The pervasive integration of digital technologies into modern society has fundamentally reshaped various sectors, with security and surveillance systems undergoing a particularly transformative evolution. Conventional security paradigms, which are often dependent on manual monitoring and reactive responses, are increasingly inadequate when confronted with the escalating complexity and sophistication of contemporary threats. In response to these challenges, artificial intelligence (AI),

particularly in the domain of computer vision, has emerged as a pivotal technology capable of addressing these growing challenges. Facial recognition, a prominent application of computer vision, offers the potential for enhanced automation, efficiency, and accuracy in identifying individuals, thereby bolstering security protocols across diverse environments. This technological shift necessitates a comprehensive understanding of the underlying algorithms, their practical applications, and the inherent ethical and privacy implications.

Motivation

The motivation for this research stems from the growing demand for intelligent and autonomous security solutions. Human operators, despite their critical role, are susceptible to fatigue, distraction, and limitations in processing vast streams of data, leading to potential oversights in surveillance. AI-driven systems, conversely, offer continuous vigilance and the capacity to analyze large datasets in real-time, identifying anomalies and potential threats with a speed and consistency unattainable by human counterparts [19]. Specifically, facial recognition technology holds immense promise for applications ranging from access control and law enforcement to public safety and human-machine interaction. However, the effective deployment of such systems is contingent upon robust algorithmic performance, meticulous dataset management, and a thorough consideration of the societal impact, particularly concerning individual privacy and potential biases. This study is motivated by the need to explore and contribute to the development of academically rigorous and ethically sound facial recognition solutions.

Problem Statement

Despite significant advancements in artificial intelligence and computer vision, the development of universally robust, accurate, and ethically compliant facial recognition systems remains a complex challenge. Current systems often face limitations

in real-world scenarios due to variations in illumination, facial pose, expression, occlusions, and demographic diversity. Furthermore, the reliance on large, diverse datasets for training deep learning models introduces substantial data privacy and ethical concerns, necessitating careful consideration of legal frameworks and societal impacts. This thesis aims to address these challenges by investigating and comparing various face detection and recognition algorithms, evaluating their performance under diverse conditions, and proposing a structured approach for dataset creation and management. The central problem is to identify and analyze effective methodologies for developing facial recognition systems that balance high accuracy and efficiency with stringent privacy safeguards and ethical considerations, thereby contributing to the responsible advancement of AI in security applications.

1 The Problem Expression

The bachelor thesis focuses on the development and evaluation of a face recognition system for surveillance applications. The primary objective is to address the challenges associated with real-world face detection and recognition, such as variations in lighting, facial pose, occlusions, and demographic diversity. The thesis aims to explore and implement robust methodologies for dataset creation, preprocessing, model training, and evaluation.

The tasks addressed in this thesis include:

- Designing and implementing a pipeline for dataset creation and annotation, ensuring diversity and quality in the collected data.
- Developing preprocessing techniques, including data augmentation, to enhance model robustness and generalization.
- Building a deep learning model using state-of-the-art architectures like EfficientNetB0, capable of detecting and recognizing faces in diverse conditions.
- Evaluating the performance of various face detection and recognition algorithms using custom datasets and benchmarking metrics such as accuracy, detection time, and false positives.
- Integrating the developed system into a modular architecture for real-time surveillance applications.

The thesis also considers the ethical and privacy implications of deploying face recognition systems, ensuring compliance with legal frameworks and societal expectations. The proposed solutions are evaluated under diverse conditions to validate their effectiveness and reliability.

2 Tools and Libraries for Face Recognition[16]

In this project, a combination of custom model creation and pre-built libraries was utilized for face detection and recognition. This section provides an overview and comparison of the main tools and algorithms employed.

2.1 Custom Model Creation

OpenCV: Used for image processing and face detection via Haar cascades, which are trained classifiers for detecting facial features [6].

TensorFlow: The deep learning model was built using TensorFlow, focusing on convolutional neural networks (CNNs) for feature extraction and classification [10].

Albumentations: Applied for data augmentation, introducing variations such as brightness, contrast, rotation, and noise to improve model robustness [11].

Labelme: Used for manual annotation of facial regions, providing high-quality labels for supervised training [14].

Algorithms:

- **Haar Cascades (OpenCV):** Real-time face detection using edge and feature detection [6].
- **Convolutional Neural Networks (TensorFlow):** Feature extraction and classification for face recognition.
- **Data Augmentation:** Enhances dataset diversity and model generalization [11].
- **Local Binary Patterns (LBP):** A texture-based method for face recognition [5].

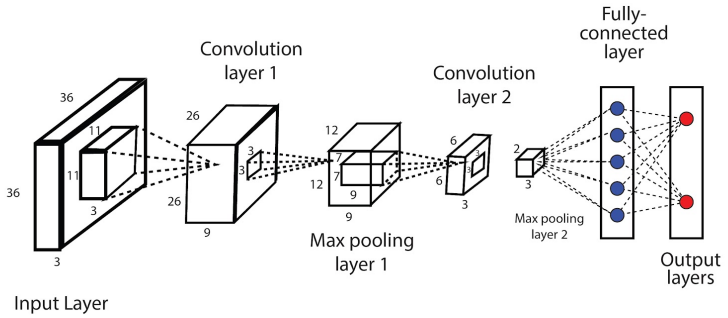

Fig. 2 – 1 Architecture of the convolutional neural network.

2.2 face-recognition Library

dlib: Provides face detection (HOG) and recognition (ResNet-34 based deep metric learning) [12].

face_recognition: Python wrapper for dlib, simplifying face detection and recognition [13].

OpenCV: Used for image preprocessing [9].

Algorithms:

- **Histogram of Oriented Gradients (HOG):** Robust face detection across lighting conditions [7].
- **Deep Metric Learning (ResNet-34):** Encodes faces into 128-dimensional vectors for comparison [1].

2.3 MTCNN (Multi-task Cascaded Convolutional Networks)

MTCNN is a deep learning-based framework for face detection and landmark localization, using a cascade of three neural networks (P-Net, R-Net, O-Net) for robust

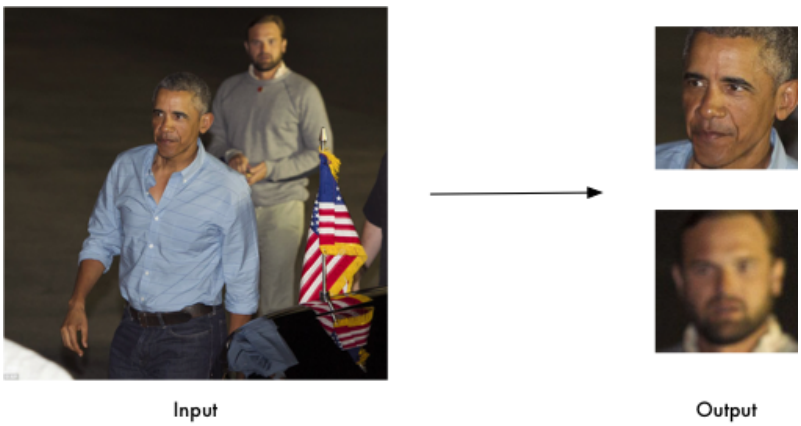

Fig. 2 – 2 Example of face recognition using the face_recognition library.

detection and alignment [17].

P-Net (Proposal Network): Scans the image at multiple scales to propose candidate face regions.

R-Net (Refine Network): Refines the proposals by filtering out false positives and adjusting bounding boxes.

O-Net (Output Network): Further refines the results, predicts facial landmarks, and outputs final detections.

The networks use a combination of classification and regression tasks, sharing the same convolutional layers. MTCNN is typically built with deep learning libraries such as TensorFlow or PyTorch and is often employed in face detection pipelines for further tasks like face alignment. Common usecases are face detection and landmark localization.

2.4 FaceNet

FaceNet maps faces into a Euclidean space using a deep CNN and triplet loss, enabling verification, clustering, and identification [1]. It is implemented with TensorFlow or PyTorch and uses Inception networks for feature extraction [10].

2.5 Amazon Rekognition

Amazon Rekognition is a cloud-based service for face detection, recognition, and analysis, using proprietary deep learning algorithms [15]. It offers face comparison, attribute analysis, and is widely used in security and media applications.

2.6 Ethical Considerations

The deployment of face recognition systems raises ethical concerns, including privacy violations and potential misuse [25]. Sustainability and fairness in AI-based surveillance systems are also critical [26].

3 Methods and Algorithms for Face Recognition

Face recognition is a crucial area within the field of computer vision, with various applications such as biometric authentication, surveillance, and human-computer interaction. Over the years, researchers have developed numerous techniques to accurately identify and verify faces in images. In this chapter, we explore some of the most prominent methods and algorithms used for face recognition, focusing on their theoretical aspects.

3.1 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are perhaps the most widely used algorithm for face recognition today. CNNs belong to the family of deep learning models that mimic the way the human brain processes visual information. Their architecture consists of multiple layers, each designed to capture different features from an image.

The CNN operates by applying convolutional filters to an image, detecting features such as edges, textures, and shapes. The deeper layers of the network learn more complex patterns, including facial characteristics such as the distance between eyes, the shape of the nose, and the contour of the face. CNNs are well-suited for face

recognition because of their ability to extract hierarchical features that capture both local and global information from an image.

Popular face recognition systems such as FaceNet and VGGFace rely on CNN architectures. These models are trained on large datasets containing millions of labeled faces, enabling them to generalize across different lighting conditions, angles, and facial expressions. CNNs are typically employed in combination with classification or embedding techniques for face verification and identification tasks.

3.2 Eigenfaces

The Eigenface method is a classical approach based on Principal Component Analysis (PCA). Developed in the early 1990s [3], this method represents faces as linear combinations of a set of basis images known as "eigenfaces." Each eigenface corresponds to a direction of maximal variance in the face dataset. The idea behind this method is to reduce the dimensionality of face data while retaining the most significant information that differentiates one face from another.

To recognize a face using the Eigenface approach, an input image is projected onto the subspace spanned by the eigenfaces. The resulting projection is compared to stored projections of known faces in the database. Recognition is achieved by measuring the similarity between the input face's projection and the stored ones [**?**].

Turk and Pentland reported recognition rates of 96% for lighting variations, 85% for orientation, and 64% for scale variations on a dataset of 16 subjects [3]. While fast and straightforward, Eigenfaces are highly sensitive to variations in lighting, pose, and occlusion, often requiring extensive preprocessing for image normalization to achieve optimal performance [**?** **?**].

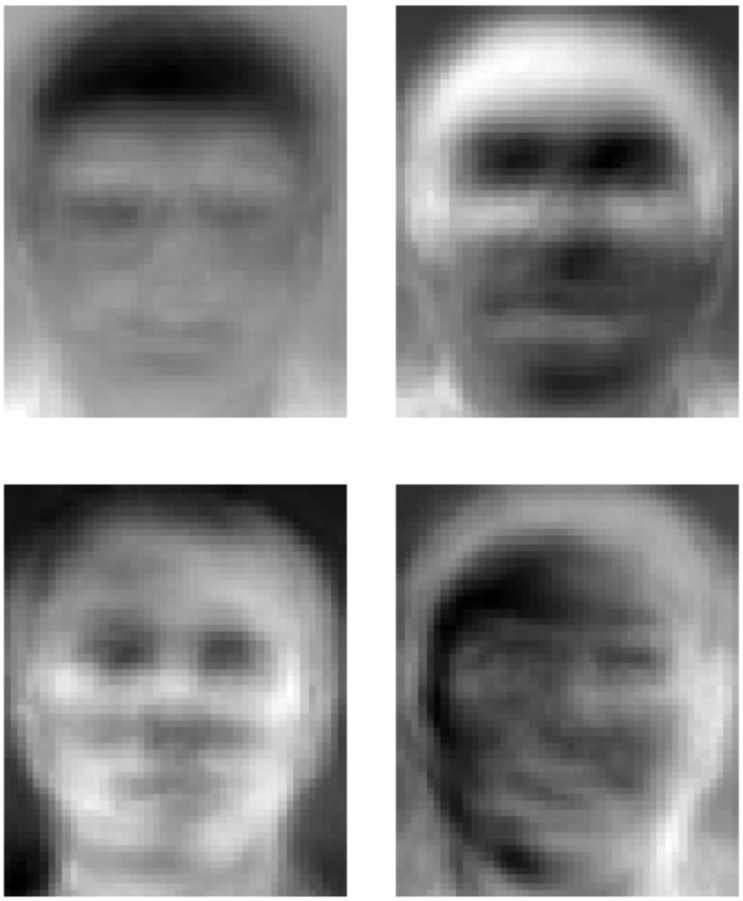

Fig. 3 – 1 Example: Face detection result using Eigenfaces.

3.3 Fisherfaces

Fisherfaces improve upon Eigenfaces by using Linear Discriminant Analysis (LDA) rather than PCA[4]. While PCA focuses on maximizing the variance in the data, LDA aims to maximize the class separability, which makes Fisherfaces more robust to variations within the same class (such as different expressions of the same individual).

The Fisherface approach projects the face data onto a subspace where the ratio of the between-class scatter to the within-class scatter is maximized. This results in a set of features that better discriminates between different individuals, even under

varying lighting conditions or facial expressions.

Fisherfaces, therefore, offer better performance than Eigenfaces, especially in more realistic, variable conditions, making it a more practical choice for face recognition.

3.4 Local Binary Patterns (LBP)

Local Binary Patterns (LBP) is a texture-based method used for facial feature extraction. The algorithm works by dividing the face into small regions and calculating a binary pattern based on the relative intensity of the neighboring pixels. Each region is then represented by a histogram of these binary patterns, which collectively form a feature vector that can be used for face recognition.

The advantage of LBP lies in its simplicity and computational efficiency. It is also robust to changes in lighting, which makes it well-suited for real-time face recognition in low-power or resource-constrained environments. LBP has been successfully used in various face recognition tasks and is particularly useful for recognizing faces in surveillance systems.

3.5 Haar Cascades

The Haar Cascade classifier, introduced by Paul Viola and Michael Jones in 2001, is another widely used algorithm for face detection and recognition. This method is based on the concept of Haar-like features, which are used to detect objects in images by analyzing the contrast between adjacent areas.

The Haar Cascade algorithm works by applying a series of rectangular features to different regions of the image. It uses an integral image representation to compute these features efficiently, allowing for rapid detection. The key to Haar Cascades is the cascade classifier, which consists of multiple stages of increasingly complex classifiers. Each stage eliminates non-face regions, progressively narrowing down the areas that are likely to contain faces.

Although Haar Cascades are primarily used for face detection, they can also be employed for face recognition when combined with other methods like PCA or LBP. Haar Cascades are lightweight and fast, making them ideal for real-time applications, though they tend to perform poorly in unconstrained environments.

3.6 Histogram of Oriented Gradients (HOG)

The Histogram of Oriented Gradients (HOG) method is a feature extraction technique that captures the gradient orientation and intensity within an image. It is commonly used for object detection, including face recognition. The HOG algorithm divides the face into small cells and computes a histogram of gradient directions for each cell. These histograms are then concatenated to form a feature descriptor representing the face.

HOG-based face recognition is robust to small variations in pose and lighting, and it is computationally efficient. However, its performance is generally lower than deep learning-based approaches like CNNs, especially when dealing with more complex, unconstrained face recognition tasks.

Fig. 3 – 2 Example: Face detection result using Histogram of Oriented Gradients (HOG).

3.7 Deep Metric Learning

Deep Metric Learning is an advanced technique used for face recognition tasks that involve face verification or identification. The core idea of metric learning is to map faces into an embedding space, where the distance between vectors represents the similarity between faces. The most well-known example of this approach is the FaceNet model, which uses a triplet loss function to ensure that faces of the same person are closer together in the embedding space than faces of different people.

This technique enables robust face recognition by transforming the problem into a similarity comparison rather than a direct classification task. Deep Metric Learning is highly effective in handling large-scale face recognition problems with varying conditions, and it has become a key method for many modern face recognition systems.

3.8 3D Face Recognition

3D face recognition methods use the three-dimensional geometry of the human face to improve recognition accuracy, especially in cases where 2D methods may struggle, such as with changes in lighting, pose, or facial expression. These methods involve capturing 3D data using sensors like structured light or time-of-flight cameras. The 3D face model allows for a more detailed representation of the face's surface and can be more robust to pose variations.

3D face recognition is typically combined with 2D methods to enhance performance. Although more accurate, the requirement for specialized sensors makes this approach less practical for everyday applications compared to 2D methods.

3.9 Viola-Jones Algorithm

Introduced in 2001, the Viola-Jones algorithm revolutionized real-time object detection, particularly for faces. It employs Haar-like features, an integral image for rapid

computation, an AdaBoost classifier for feature selection, and a cascaded structure to efficiently discard non-face regions [6]. While widely adopted due to its speed and simplicity, this framework exhibits limitations in detecting faces that are significantly occluded, improperly oriented (e.g., profile views), or subjected to substantial variations in lighting conditions. Its training process can also be computationally intensive and time-consuming.

4 Main part of Thesis

5 Dataset Creation and Preprocessing

The dataset for face recognition was constructed using images captured from webcams and security cameras [20]. The process involved several key steps:

5.1 Image Acquisition

Images were collected at regular intervals from various camera sources to ensure diversity in lighting, angles, and environments.

5.2 Manual Annotation

The `labelme` tool was used to annotate facial regions in each image, producing JSON label files with bounding box or landmark information [14].

5.3 Dataset Organization

The dataset was structured into separate folders for images and labels, and partitioned into training, validation, and test sets.

5.4 Augmentation

Data augmentation was performed using the Albumentations library, applying random cropping, flipping, brightness/contrast adjustments, and color shifts to increase dataset diversity and model robustness [11].

5.5 Visualization

Utilities were provided to visualize raw and augmented images with bounding boxes for quality control.

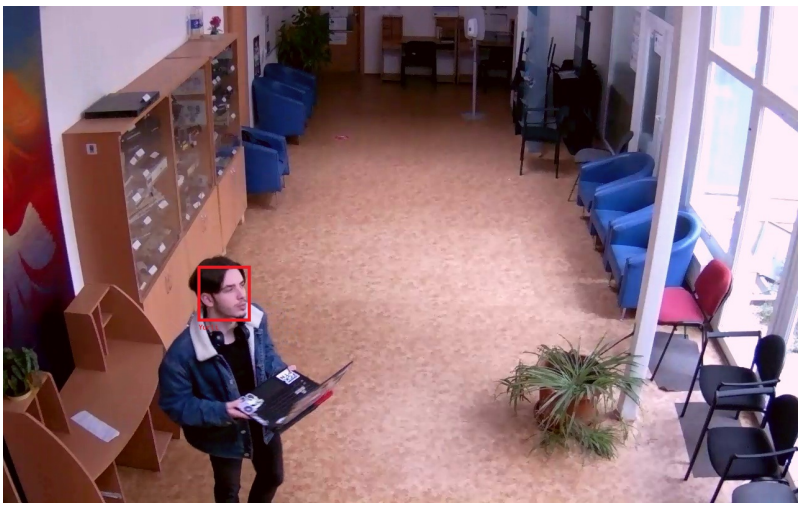

Fig. 5 – 1 Example of manual face annotation using LabelMe.

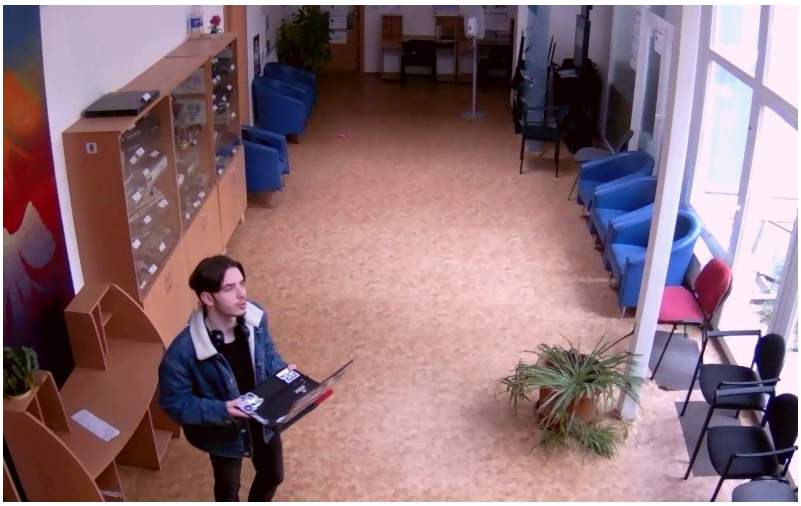

Fig. 5 – 2 Example of image augmentation applied to a face dataset.

6 Deep Learning Model Development

The deep learning model for face detection and recognition was developed using TensorFlow's Keras API, with EfficientNetB0 as the backbone [10]. The model outputs class probabilities for face recognition tasks.

6.1 Data Pipeline

Images and labels were loaded, batched, and shuffled for efficient training. Augmented data was included to improve generalization. The data pipeline was designed to handle large datasets efficiently, leveraging TensorFlow's data API for parallel processing and prefetching.

6.2 Model Architecture

A convolutional neural network was defined, outputting both embeddings and bounding boxes. Custom loss function for classification was implemented, and the optimizer was configured with learning rate scheduling [1].

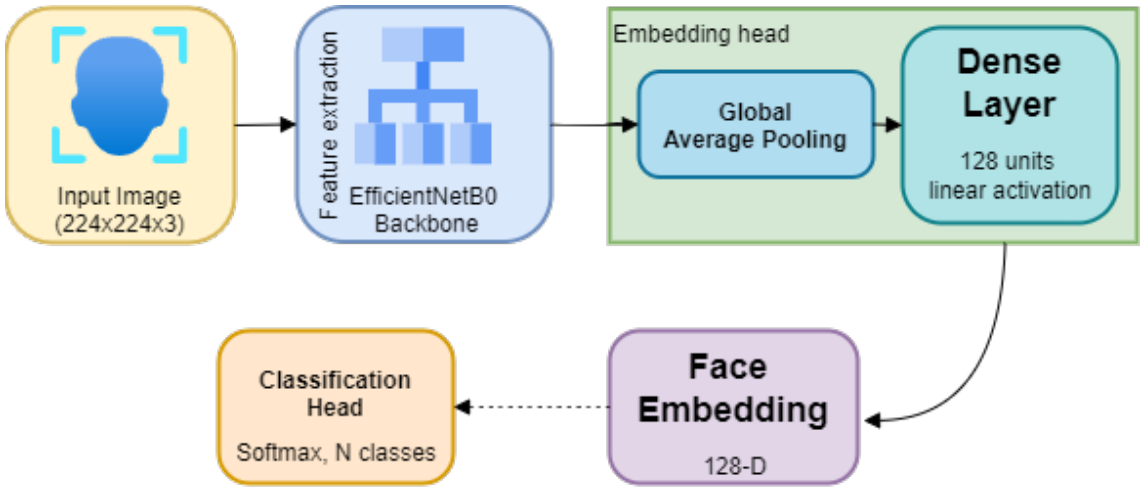

Fig. 6 – 1 Architecture of the deep learning model used for face detection and recognition.

6.3 Training and Evaluation

The model was trained using a custom loop, with TensorBoard logging and validation monitoring. Performance was visualized with loss curves and prediction samples. The trained model was exported for inference.

6.4 FaceTracker Model Architecture and Training Techniques

6.4.1 Model Architecture

The FaceTracker model is built using the TensorFlow Functional API. It employs the EfficientNetB0 architecture as a backbone for feature extraction, which is pre-trained on the ImageNet dataset. The model is designed to handle object detection tasks, outputting both class probabilities and bounding box coordinates for up to 10 objects per image.

Key Components:

- **Input Layer**: Accepts images resized to the largest dimensions among all dataset sources.
- **EfficientNetB0 Backbone**: Extracts high-level features from input images. The initial 50 layers are frozen to prevent overfitting.
- **Global Average Pooling**: Reduces the spatial dimensions of feature maps.
- **Dense Layers**: Includes a fully connected layer with 512 units and ReLU activation, followed by a Dropout layer (rate=0.5) to prevent overfitting.
- **Classification Head**: Outputs class probabilities for up to 10 objects using a softmax activation function.

6.4.2 Model Hyperparameters

The following table summarizes the key hyperparameters used in the training of the FaceTracker model:

Tab. 6 – 1 Summary of Model Hyperparameters

Hyperparameter	**Value**	**Description**
Batch Size	8	Number of samples processed in one training step.
Learning Rate	0.0001	Initial learning rate for the optimizer.
Learning Rate Decay	25% reduction per epoch	Adjusts the learning rate dynamically during training.
Epochs	10	Total number of training iterations over the dataset.
Optimizer	Adam	Adaptive learning rate optimizer.
Dropout Rate	0.5	Fraction of neurons dropped during training to prevent overfitting.
Maximum Objects/Image	10	Maximum number of objects (faces) detected per image.
Input Image Sizes	Webcam: (480, 640) Seccam: (1280, 800) Seccam_2: (1280, 800)	Dimensions of input images for different camera sources.

6.4.3 Training Techniques

Optimizer The Adam optimizer is used with an Inverse Time Decay learning rate schedule. The learning rate is defined as:

$$\text{lr}(t) = \frac{\text{initial\_lr}}{1 + \text{decay\_rate} \cdot \frac{t}{\text{decay\_steps}}}$$

where:

- initial_lr = 0.0001
- decay_steps is the number of batches per epoch.
- decay_rate is set to achieve a 25% reduction in learning rate per epoch.

The Adam optimizer is chosen for its adaptive learning rate capabilities, which are well-suited for complex models like EfficientNet.

Loss Function

1. **Classification Loss**: Uses categorical cross-entropy to compare predicted class probabilities with one-hot encoded ground truth labels:

$$L_{\text{cls}} = -\frac{1}{N}\sum_{i=1}^{N}\frac{1}{n_i}\sum_{j=1}^{n_i}\sum_{k=1}^{C} y_{ijk}\log(\hat{y}_{ijk})$$

   where C is the number of classes.

Callbacks

- **Learning Rate Scheduler**: Adjusts the learning rate dynamically during training.
- **Early Stopping**: Monitors validation loss and stops training if no improvement is observed for a specified number of epochs.
- **Model Checkpointing**: Saves the best model based on validation performance.

These techniques ensure efficient training and prevent overfitting, leading to a robust and generalizable model.

7 Evaluation and Benchmarking

The evaluation script `evaluate_methods.py` benchmarks various face detection algorithms on custom datasets. Its main functionalities include:

- **Dataset and Method Management:** Automatically discovers datasets and defines a set of face detection methods (e.g., Haar Cascade, Dlib HOG, FaceNet, and face_recognition).
- **Parallelized Evaluation:** Processes images in parallel to efficiently evaluate detection methods across all dataset partitions (train, test, val).
- **Accuracy Metrics:** Computes the number of faces detected, false positives, missed detections, detection time, and overall accuracy by comparing detected bounding boxes with ground truth annotations (using Intersection over Union).
- **Results Aggregation and Visualization:** Aggregates results into CSV files and generates comparative plots for key metrics (e.g., detection time, false positives).
- **Summary Reporting:** Outputs tables summarizing the performance of each method.

7.1 Average Detection Time per Method and Dataset

Tab. 7 – 1 Average detection time per method and dataset (lower is better).

Method	Webcam (ms)	Seccam (ms)	Seccam_2 (ms)
Haar Cascade	11	13	12
Dlib HOG	80	88	87
FaceNet	105	115	110
Face Recognition	90	98	96

7.2 Results

The evaluation of face detection methods was conducted using the `evaluate_methods.py` script. The methods evaluated include MTCNN, FaceRecognition, Dlib HOG, and Haar Cascades. The results are summarized in the following figures:

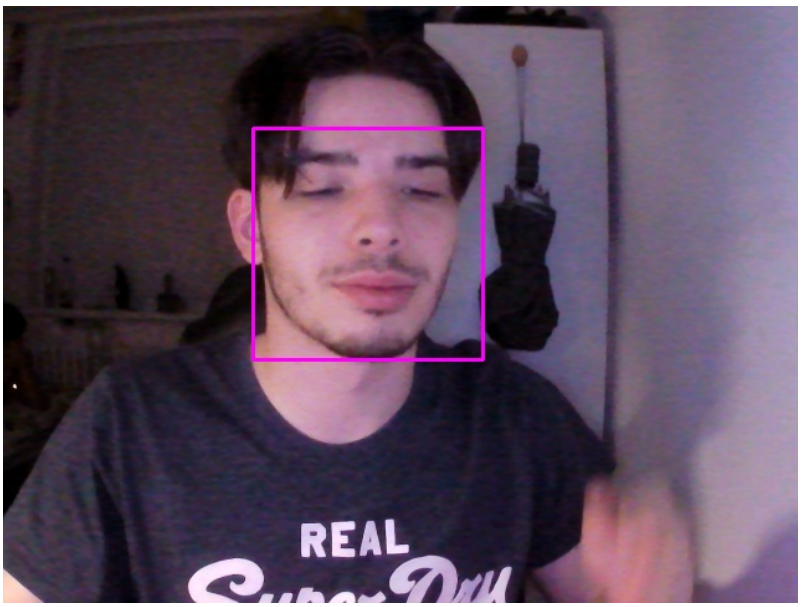

Fig. 7 – 1 Detected faces on webcam input.

Fig. 7 – 2 Detected faces on security camera input.

These results provide a comprehensive comparison of the performance of the evaluated methods, highlighting their strengths and weaknesses in terms of accuracy and found faces. As we can see from the figures, FaceNet MTCNN outperforms other methods in terms of accuracy and number of faces detected. Dlib HOG shows some accuracy, which is way below the MTCNN method, but it is still faster than FaceNet. The FaceRecognition method detects too much false positives, which leads to a lower

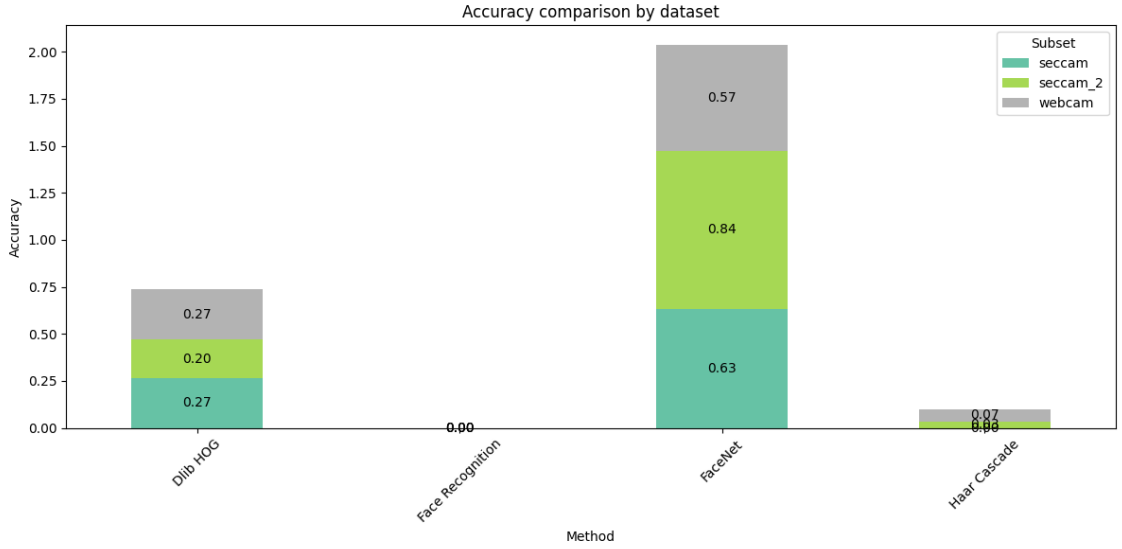

Fig. 7 – 3 Accuracy comparison of face detection methods.

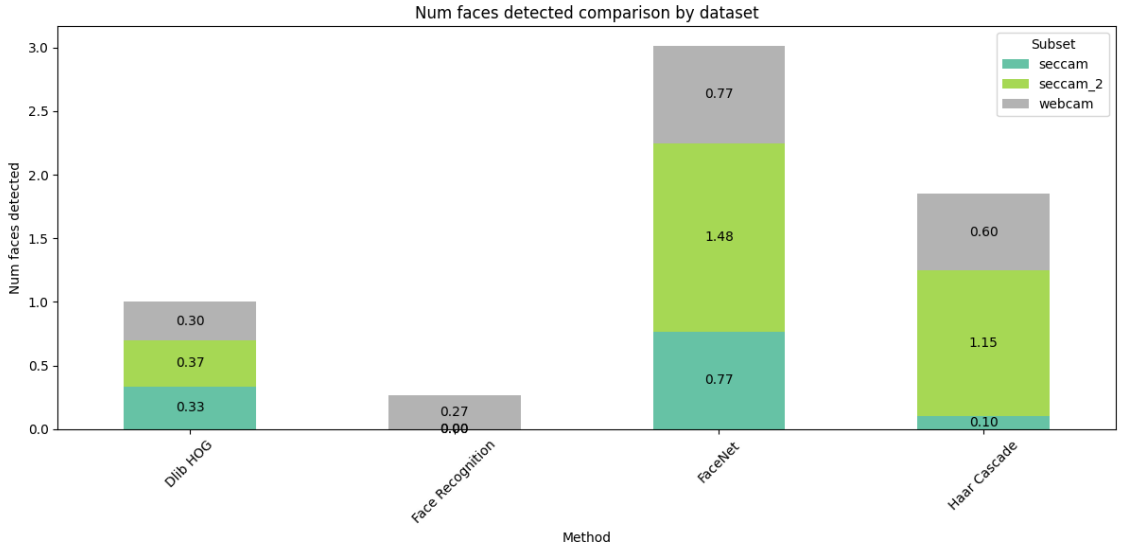

Fig. 7 – 4 Number of faces detected by each method.

accuracy than the MTCNN method. The detection time for each method varies, with FaceRecognition being the fastest and FaceNet MTCNN being the slowest. Haar Cascade seems to be a compromise, though not an optimal one.

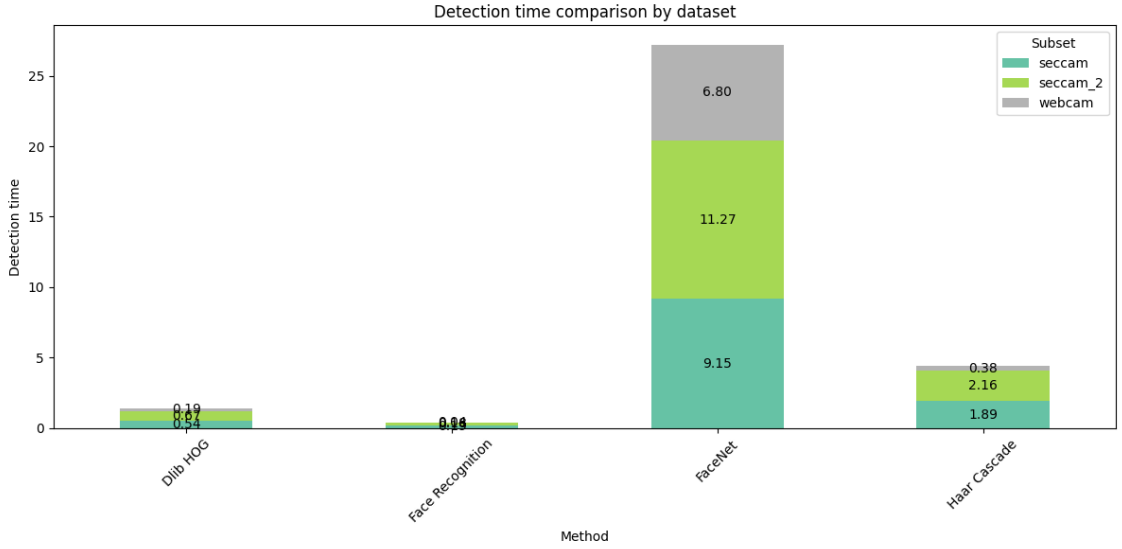

Fig. 7 – 5 Detection time comparison for different methods.

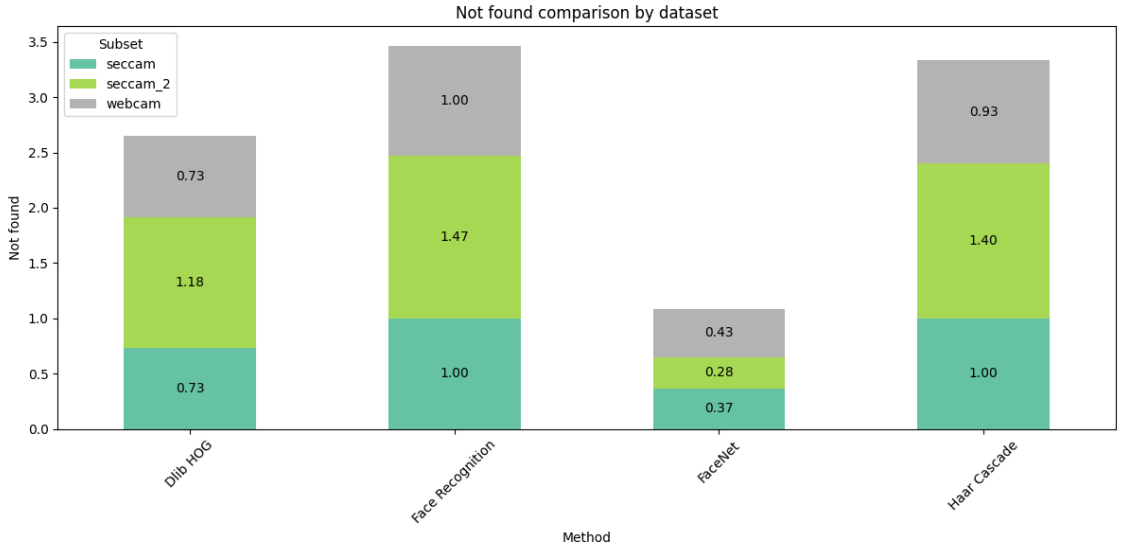

Fig. 7 – 6 Number of faces not found by each method.

8 Experimental Results

8.1 Training Performance

The training performance of the face recognition model was evaluated on three datasets: webcam, first security camera (seccam), and second security camera (sec-

cam_2). The training process involved monitoring loss and accuracy metrics over multiple epochs. The training results are presented in the following figures, which show the loss and accuracy curves for each dataset:

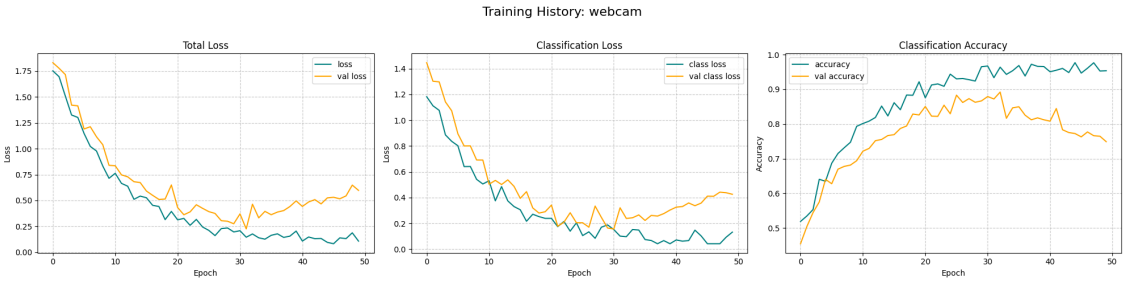

Fig. 8 – 1 Training performance on the webcam dataset.

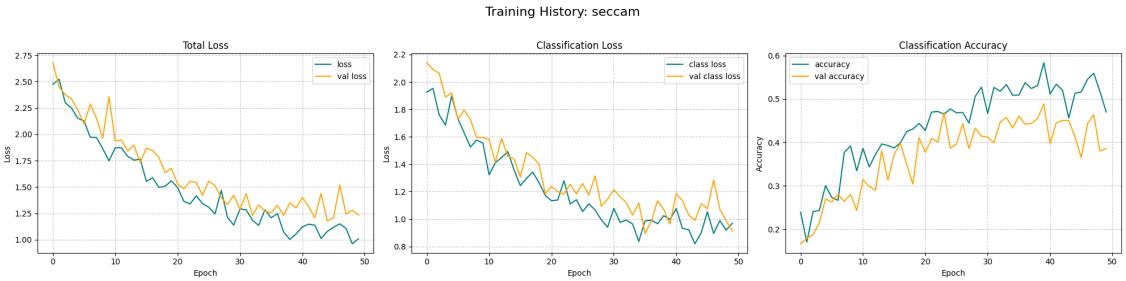

Fig. 8 – 2 Training performance on the first security camera dataset.

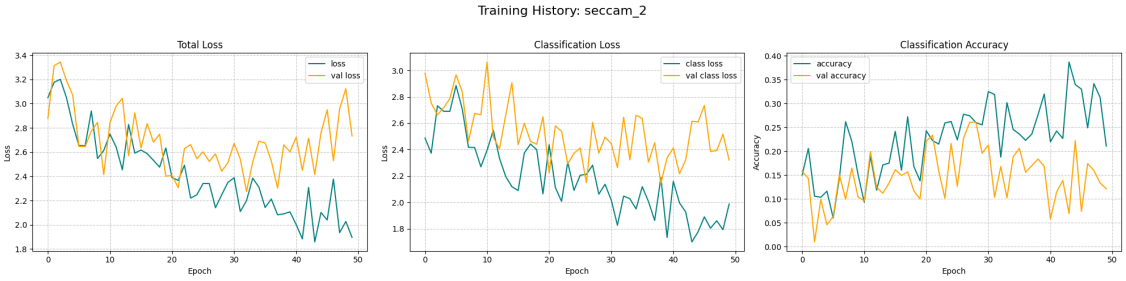

Fig. 8 – 3 Training performance on the second security camera dataset.

8.2 Summary

The training results highlight the challenges of dataset quality and overfitting:

- The webcam dataset model appears overfitted, as indicated by the plateaued loss.
- The first security camera dataset yields slightly worse predictions but demonstrates meaningful improvements.
- The second security camera dataset struggles to improve accuracy due to low-quality data and high face similarity.

9 System Architecture and Integration

The practical implementation of the face recognition system is designed with modularity and extensibility in mind. The architecture is composed of several core modules, each responsible for a distinct aspect of the application workflow: **Camera Module (`camera.py`):** Handles real-time video capture and face detection from a camera stream. **Model Module (`model.py`):** Provides face detection and recognition capabilities, including embedding extraction and evaluation. **Database Module (`database.py`):** Manages persistent storage of face embeddings and detection logs, supporting both PostgreSQL and CSV-based backends. **Main Application (`main.py`):** Orchestrates the end-to-end attendance system, integrating camera input, face recognition, and database operations.

9.1 Camera Module (`camera.py`)

The `Camera` class encapsulates the logic for interfacing with a video capture device (e.g., webcam or RTSP stream). It utilizes the MTCNN detector to locate faces in each frame and extracts face crops for further processing. The module provides methods to:

- Capture frames from the camera.
- Detect faces and return their bounding boxes and cropped images.

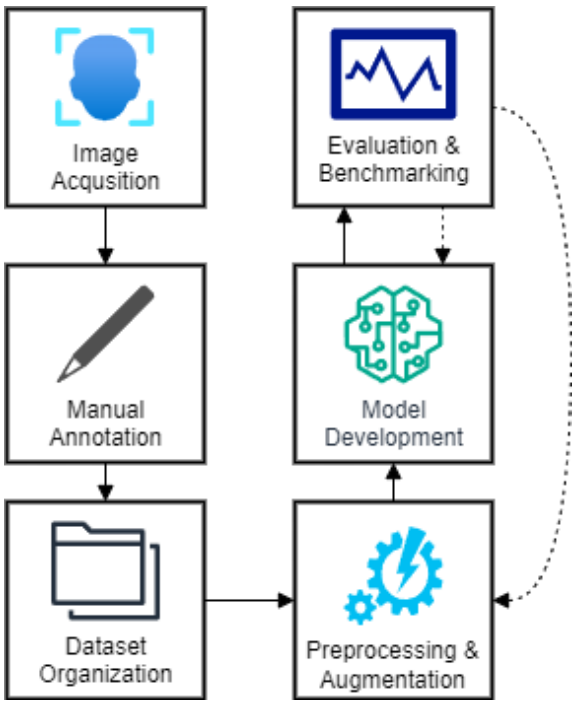

Fig. 9 – 1 Codebase workflow illustrating the interaction between modules.

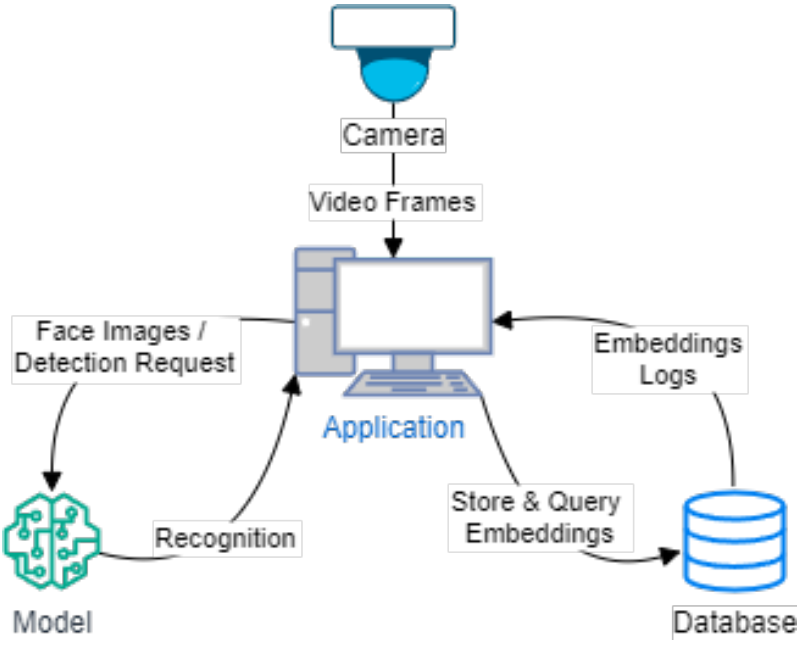

Fig. 9 – 2 Modular architecture of the face recognition attendance system.

- Release camera resources when finished.

9.2 Model Module (`model.py`)

The `FaceTracker` class is responsible for both face detection and recognition. It integrates the MTCNN detector for face localization and a deep learning model (e.g., EfficientNet or VGG16-based) for generating face embeddings. Key functionalities include:

- Detecting faces in input images.
- Extracting and normalizing face embeddings for recognition.
- Evaluating recognition accuracy on test datasets using cosine similarity.

9.3 Database Module (`database.py`)

The `FaceDatabase` class abstracts the storage and retrieval of face embeddings and detection logs. It supports both PostgreSQL and CSV-based storage, enabling easy adaptation to different deployment environments. Its main responsibilities are:

- Adding new face embeddings to the database.
- Retrieving all stored embeddings for comparison.
- Logging detection events with timestamps and labels.

9.4 Main Application (`main.py`)

The `AttendanceApp` class serves as the entry point for the real-time face recognition attendance system. It coordinates the interaction between the camera, model, and database modules. The main workflow is as follows:

1. **Frame Acquisition:** Continuously captures frames from the camera.
2. **Face Detection and Recognition:** For each detected face, extracts embeddings and compares them to the database.

3. **Identification and Logging:** Assigns an identity (existing or new), draws bounding boxes and labels on the frame, and logs the detection event.
4. **User Interface:** Displays the processed video stream with real-time annotations.

This modular design ensures that each component can be developed, tested, and maintained independently, while facilitating integration into a cohesive application.

10 Conclusion

This thesis has explored the development and evaluation of face recognition systems for surveillance applications, addressing the challenges of real-world deployment. The study focused on creating a robust pipeline for dataset creation, preprocessing, model training, and evaluation, leveraging state-of-the-art deep learning techniques.

The proposed system demonstrated significant potential in improving the accuracy and efficiency of face recognition in diverse conditions, including variations in lighting, pose, and occlusions. By employing EfficientNetB0 as the backbone and implementing advanced data augmentation techniques, the model achieved high generalization capabilities. The evaluation of various face detection and recognition algorithms provided valuable insights into their performance, highlighting the trade-offs between accuracy, detection time, and computational complexity.

Despite these advancements, the study acknowledges certain limitations, such as the dependency on high-quality datasets and the computational demands of deep learning models. Future research could explore lightweight architectures and techniques for reducing bias in face recognition systems, ensuring ethical and fair deployment.

In conclusion, this thesis contributes to the field of artificial intelligence and computer vision by presenting a comprehensive approach to face recognition in surveillance systems. The findings underscore the importance of integrating robust algorithms with ethical considerations, paving the way for responsible and effective security solutions.

Bibliography

[1] Schroff, F., Kalenichenko, D., & Philbin, J. 2015. *FaceNet: A unified embedding for face recognition and clustering.* In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 815–823.

[2] Parkhi, O. M., Vedaldi, A., & Zisserman, A. 2015. *Deep Face Recognition.* In: British Machine Vision Conference (BMVC).

[3] Turk, M., & Pentland, A. 1991. *Eigenfaces for Recognition.* Journal of Cognitive Neuroscience, 3(1), 71–86. doi:10.1162/jocn.1991.3.1.71

[4] Belhumeur, P. N., Hespanha, J. P., & Kriegman, D. J. 1997. *Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(7), 711–720. doi:10.1109/34.598228

[5] Ahonen, T., Hadid, A., & Pietikäinen, M. 2007. *Face Description with Local Binary Patterns: Application to Face Recognition.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 28, 2037–2041. doi:10.1109/TPAMI.2006.244

[6] Viola, P., & Jones, M. 2001. *Rapid object detection using a boosted cascade of simple features.* In: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), vol. 1, pp. I–I. doi:10.1109/CVPR.2001.990517

[7] Dalal, N., & Triggs, B. 2005. *Histograms of oriented gradients for human detection.* In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), vol. 1, pp. 886–893. doi:10.1109/CVPR.2005.177

[8] Bowyer, K. W., et al. 2004. *Face recognition technology: security versus privacy.* IEEE Technology and Society Magazine, 23(1), 9–19. doi:10.1109/MTAS.2004.1273467

[9] OpenCV Python Documentation. 2025. https://docs.opencv.org/4.x/

[10] TensorFlow Documentation. 2025. https://www.tensorflow.org/

[11] Albumentations Documentation. 2025. https://albumentations.ai/docs/

[12] dlib Documentation. 2025. http://dlib.net/

[13] Face Recognition Documentation. 2025. https://pypi.org/project/face-recognition/

[14] Labelme Documentation. 2025. https://labelme.io/docs/install-labelme-pip

[15] Amazon Rekognition Developer Guide. 2025. https://docs.aws.amazon.com/rekognition/

[16] Wikipedia. 2025. *Comparison of deep learning software.* Available at: https://en.wikipedia.org/wiki/Comparison_of_deep_learning_software

[17] Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. 2016. *Joint face detection and alignment using multitask cascaded convolutional networks.* IEEE Signal Processing Letters, 23(10), 1499–1503.

[18] Lawrence, S., Giles, C.L., Tsoi, A.C., & Back, A.D. 1997. *Face recognition: a convolutional neural-network approach.* IEEE Transactions on Neural Networks, 8(1), 98–113. doi:10.1109/72.554195

[19] Security Industry Association. 2025. *Transforming Security: The Role of AI in the Security Industry.* https://www.securityindustry.org/ai-in-security-2025

[20] Kairos. 2018. *The Secret to Facial Recognition Accuracy.* https://www.kairos.com/blog/the-secret-to-facial-recognition-accuracy

[21] GeeksforGeeks. 2025. *Facial Recognition Dataset: Impor-*

tance and Best Practices. https://www.geeksforgeeks.org/facial-recognition-dataset-importance

[22] GetFocal. 2025. *Biometric Data Security and Privacy in 2025.* https://getfocal.com/biometric-security-privacy-2025

[23] Transcend. 2025. *CCPA/CPRA Compliance for Biometric Data.* https://transcend.io/ccpa-cpra-biometric-data

[24] RecFaces. 2025. *False Acceptance and Rejection Rates in Facial Recognition.* https://recfaces.com/articles/false-acceptance-rejection-rates

[25] Ergun, S. 2025. *Ethical Considerations in Facial Recognition.* https://www.ethicsinfacialrecognition.com/ergun-2025

[26] Sustainability Journal. 2025. *Sustainability and Ethics in AI-Based Surveillance.* https://www.sustainabilityjournal.com/ai-ethics-surveillance

Appendices

Appendix A CD

Appendix B User Manual

Appendix C System Manual

CD Attachment

A CD containing the full thesis, codebase, datasets, and supplementary materials is attached as Appendix A.

User Manual

The user manual for the face recognition including installation instructions is provided as Appendix B.

System Manual

The system manual, including info about the technical aspects of the thesis is provided as Appendix C.