Diagram

Description automatically generated

MIDDLESEX UNIVERSITY

THE BURROUGHS, HENDON LONDON

NW4 4BT

**Final Report.**

**Title: Analyzing the Efficiency and Performance of Deep Convolutional Neural Networks, for Facial Recognition using Transfer Learning with Open CV.**

**Student Name: ALMANSUR SA’ID**

**Student ID:M00954151**

**Supervisor Name: Professor CHRIS HUYCK**

Table of Contents

[Chapter 1: Introduction 4](#_Toc163278848)

[1.1 Background 4](#_Toc163278849)

[1.2 Research Objectives 4](#_Toc163278850)

[1.2.1 Project Planning and Design 4](#_Toc163278851)

[1.3 Scope of the Study 5](#_Toc163278852)

[1.4 Structure of the Report 6](#_Toc163278853)

[Chapter 2: Literature Review 7](#_Toc163278854)

[2.1 Convolutional Neural Networks(CNNs) 8](#_Toc163278855)

[2.1.1 Basic concepts and Evolution 8](#_Toc163278856)

[2.1.2 Core Components of CNNs 10](#_Toc163278857)

[2.2 Deep CNN Architectures 12](#_Toc163278858)

[2.2.1 Advancements and Innovations 12](#_Toc163278859)

[2.3 Challenges and Limitations of CNNs 14](#_Toc163278860)

[2.4 Facial Recognition Techniques 15](#_Toc163278861)

[2.4.1 Evolution and Fundamentals 15](#_Toc163278862)

[a. Definition and Fundamentals 15](#_Toc163278863)

[b.  Historical Context and Evolution 15](#_Toc163278864)

[c. Applications Across Domains 16](#_Toc163278865)

[2.4.2 CNNs in Facial Recognition 16](#_Toc163278866)

[2.4.3 Challanges in Facial Recognition 17](#_Toc163278867)

[2.5 Transfer Learning 19](#_Toc163278868)

[2.5.1 Concept and Significance 19](#_Toc163278869)

[2.5.2 Application in Facial Recognition 19](#_Toc163278870)

[2.6 OpenCV For Facial Recognition 20](#_Toc163278871)

[2.6.1 Why use OpenCV 22](#_Toc163278872)

[2.6.2 OpenCV in this Project 22](#_Toc163278873)

[2.6.3 Haar Cascades Explained 22](#_Toc163278874)

[2.6.4 Application in Projects with OpenCV 25](#_Toc163278875)

[Chapter 3: Methodology 26](#_Toc163278876)

[3.1 Project Planning and Design 26](#_Toc163278877)

[3.2 Selection of Tools and Technologies 26](#_Toc163278878)

[3.3 CNN Architecture and Transfer learning models 26](#_Toc163278879)

[3.4 Dataset Collection and Preperation. 29](#_Toc163278880)

[Chapter 4: System Implementation 31](#_Toc163278881)

[4.1 System Architecture and Design For Facial Recognition 31](#_Toc163278882)

[4.2 Overview of the Attendance System 32](#_Toc163278883)

[Chapter 5: Evaluation and Testing 35](#_Toc163278884)

[5.1 Testing Methodology 35](#_Toc163278885)

[5.2 Performance Evaluation, User Feedback and Pilot Testing 35](#_Toc163278886)

[Chapter 6: Ethical Considerations 39](#_Toc163278887)

[6.1 Privacy Concerns 39](#_Toc163278888)

[6.2 Ethical Implications of Facial Recognition 39](#_Toc163278889)

[6.3 Addressing Ethical and Privacy Concerns 39](#_Toc163278890)

[Chapter 7: Conclusion 40](#_Toc163278891)

[7.1 Key Findings 40](#_Toc163278892)

[7.2 Implications for Practice 41](#_Toc163278893)

[7.3 Recommendations for Future Research 42](#_Toc163278894)

[References 43](#_Toc163278895)

[Appendices 46](#_Toc163278896)

[A. Code Explanations and Snippets 46](#_Toc163278897)

# **Chapter 1: Introduction**

## Background

The introduction of Deep Convolutional Neural Networks (CNNs) has significantly changed the field of artificial intelligence (AI), especially in the area of computer vision (Goodfellow et al., 2016). This research offers a thorough examination of the use of CNNs in facial recognition, a field that has seen tremendous growth in popularity as a result of its enormous potential across a range of industries, including social networking, digital identity, and security (Zhao et al., 2003). In order to improve the effectiveness and precision of facial recognition systems, the research focuses in particular on the integration of Transfer Learning techniques with OpenCV, a popular open-source computer vision and machine learning software library (Bradski, 2000).

Since their invention, deep convolutional neural networks have completely changed how images are processed and analysed. CNNs are especially well-suited for intricate visual tasks like facial recognition because of their capacity to learn hierarchical feature representations (LeCun et al., 1998). The architecture of CNNs is examined in detail in this paper, along with the subtleties of each layer's function in feature extraction and pattern identification. To give a thorough grasp of their evolution and their influence on the field of facial recognition, the historical development of CNNs—from simple architectures to sophisticated models like AlexNet (Krizhevsky et al., 2012), VGGNet (Simonyan & Zisserman, 2014), and ResNet (He et al., 2016)—is examined.

Despite its many advantages, facial recognition technology is not without its problems. Variations in lighting, posture, and expressions all have a big impact on how accurately a system recognises a face. This study tackles these issues by outlining the approaches and strategies currently employed in facial recognition. It provides a comprehensive overview of the state-of-the-art in facial recognition technology by delving deeper into the ways that 3D modelling and multi-view training can improve recognition accuracy (Blanz & Vetter, 2003).

The use of transfer learning in CNNs for facial recognition is a major focus of this work. When labelled data is poor or computational resources are restricted, the method of Transfer Learning—where a model generated for one task is utilised as the basis for a model for a second task—is especially helpful (Pan & Yang, 2010). The aim of this work is to explore how facial recognition tasks can be enhanced by fine-tuning pre-trained networks on huge datasets (Yosinski et al., 2014), hence lowering the requirement for enormous amounts of training data and CPU power.

## Research Objectives

### 1.2.1 Project Planning and Design

**Objective Definition**: Although CNNs have demonstrated remarkable results in facial recognition, they still have issues with computational efficiency and the requirement for large amounts of training data (He et al., 2016). In order to tackle these issues, this work focuses on refining CNN models that have already been trained for facial recognition tasks in a variety of dynamic settings.

 This study is unique in that it integrates Transfer Learning with OpenCV for facial recognition. As a powerful and adaptable computer vision tool, OpenCV provides a useful platform for CNN model implementation and testing (Bradski, 2000). The study describes how to apply Transfer Learning to OpenCV in conjunction with pre-trained CNN models to produce accurate and efficient facial recognition.

This work synthesises several approaches and methodology used in the field of facial recognition utilising CNNs and Transfer Learning by examining the material that has already been published. It offers a critical evaluation of the current status of research, stressing its strengths, weaknesses, and possible directions for future investigation.

The study aims to:

1.Examine how well trained CNN models perform on problems involving facial recognition.

2. Examine how Transfer Learning can be used to increase model efficiency.

3. Examine how these models are integrated into the OpenCV environment.

4.Implement Facial recognition by using HaarCascades with OpenCV.

5. Testing a Face Recognition Model in Realtime By creating an Attendance system.

## Scope of the Study

**Scope Determination**: The study will concentrate on well-known pre-trained CNN models, including FaceNet, MTCNN, VGG-Face, and ResNet. It won't take into account complexity seen in the actual world, such as changing illumination or occlusions, and will only work with controlled environments and standard datasets. It also studies how OpenCV works with Haar Cascades in detecting faces from a frontal perspective. Instead of creating brand-new neural network architectures, the study optimises already-existing ones while keeping data and resources limits in mind.

## Structure of the Report

This thesis is organized into eight chapters, each with a distinct function in the exposition of the research. The plan is intended to lead the reader from the basic ideas and literature analysis to the study's actual application, assessment, and wider ramifications.

**Chapter 1: Introduction**  
 This chapter lays forth the framework for the study, including background data on the subject, the goals of the investigation, and the parameters of the study's scope. A summary of the thesis structure is included at the end to help the reader get familiar with the document's structure.

**Chapter 2: Literature Review**  
 A thorough analysis of the literature on transfer learning, facial recognition, convolutional neural networks, and OpenCV's use in image processing. This chapter lays out the study's theoretical framework while noting prior successes and pointing out any gaps that the current investigation seeks to fill.

**Chapter 3: Methodology**

Explains the technologies, instruments, and research design that were chosen for the study. It describes the standards used to select particular CNN architectures, how to gather and prepare datasets, and the project's long-term goals and strategic orientation.

**Chapter 4: Implementation of the System**

Focuses on the research's practical application, which is the creation of a facial recognition attendance system. It highlights the difficulties and solutions associated with implementation and covers system architecture, design considerations, the development process, and the integration of facial recognition technology.

**Chapter 5: Evaluations and Testing**  
 Provides a thorough analysis of the system's performance as well as the testing technique. Accuracy, dependability, efficiency, scalability evaluations, user input, and a comparative study with current systems are all included.

**Chapter 6: Ethical Considerations**

Investigates the ethical and privacy issues surrounding the use of facial recognition technology, particularly as they relate to attendance systems. It talks about how the initiative addresses these issues.

**Chapter 7: Conclusion**  
 Summarises the study and offers final opinions on the project's contributions to the fields of attendance systems and facial recognition. It also reflects on the research process and findings.

**References**

Complies with the selected citation style and includes a list of all the academic works cited in the thesis.

**Appendices**

Includes code listings and explanation of code used in the project. As well as other important images related to the thesis.

# **Chapter 2: Literature Review**

## Convolutional Neural Networks(CNNs)

### Basic concepts and Evolution

***a. Early Concepts and Foundations***

A diagram of a cell body

Description automatically generatedA group of circles on a black background

Description automatically generated

Neural networks are a class of machine learning algorithms that are somewhat inspired by the structure of the human brain. They are designed to recognise patterns and comprehend sensory data through machine perception, tagging, and clustering of raw input. Neurons, which are fundamental computational units with weighted input signals that produce an output signal by applying an activation function, make up the majority of neural networks (Goodfellow et al., 2016).

A subclass of deep neural networks called convolutional neural networks, or CNNs, are made especially to handle input that is presented as several arrays, such as image data (which is composed of height, breadth, and colour channels). CNNs are especially good at many other auto-vision tasks, such as image and video identification and image classification.

Convolutional neural networks (CNNs) process visual information by automatically recognising the spatial hierarchies of features in input images. This is the fundamental idea of CNNs. The biological mechanisms of the human visual cortex, which interprets visual information in a layered, hierarchical fashion, serve as the model for this idea (Hubel and Wiesel, 1968).

The convolutional layer, which is the basic component of a CNN, performs convolution operations on the input data using learnable filters or kernels. By swiping over the input image, these filters extract features from deeper layers, including textures, edges, and more intricate patterns, by multiplying the image's elements and adding up the results (LeCun et al., 1998).

***b. The Advent of the Neocognitron***

A diagram of a diagram of a diagram

Description automatically generated

The creation of the Neocognitron by Kunihiko Fukushima in the 1980s marked the beginning of CNN history. These earlier investigations of the visual cortex served as inspiration for the development of the Neocognitron, a groundbreaking model in neural network research. Although it did not use backpropagation for training, a crucial component in later CNNs, it introduced a hierarchical, multilayered architecture capable of visual pattern recognition and was groundbreaking in the field as displayed in the figure above.(Fukushima, 1980).

***c. LeCun’s Breakthrough with Digit Recognition***

In 1998, Yann LeCun and associates made a major advancement in CNN research. They created the LeNet-5 CNN model, which is intended for digit recognition. Backpropagation, which enables the effective computation of gradients in deep networks, was used to train this model. By correctly identifying handwritten digits from the MNIST database, which became a benchmark dataset in the area, LeNet-5 effectively illustrated the usefulness of CNNs (LeCun et al., 1998).

A diagram of a network

Description automatically generated

Multiple convolutional layers were included in LeNet-5's design, which was then followed by subsampling layers—now often referred to as pooling layers—and finally fully linked layers. The network's number of parameters and computation decreased as a result of the pooling layers shrinking the spatial size of the feature maps, which was done by the convolutional layers. This decrease was essential for controlling the processing load and avoiding overfitting, which is a frequent problem while learning neural networks (LeCun et al., 1998).

***d. Subsequent Developments and Modern CNNs***

The field of CNNs saw an exponential surge in attention and research after LeNet-5. But CNNs didn't really become well-known in the AI world until Krizhevsky et al.'s 2012 release of AlexNet. By far, AlexNet was victorious in the ImageNet Large Scale Visual Recognition Challenge, surpassing LeNet-5 in depth and complexity. Due to this achievement, CNNs are now at the forefront of deep learning research, and their broad use in computer vision has been made possible (Krizhevsky et al., 2012).

A black and white image of a pool

Description automatically generated

A number of new concepts and advancements in CNN architectures were brought forward by the popularity of AlexNet. Significant improvements in deep learning were made during this era with the creation of architectures such as ZFNet, GoogLeNet (Inception), VGGNet, ResNet, and DenseNet (Zeiler and Fergus, 2014; Szegedy et al., 2015; Simonyan and Zisserman, 2014; He et al., 2016; Huang et al., 2017).

### Core Components of CNNs

***a. Definition and Convolutional Layers***

The core component of a CNN, the convolutional layer, is essential to feature extraction and detection. This layer applies a series of learnable filters, often known as kernels, to the input image. Every filter produces a two-dimensional activation map by convolving across the width and height of the input volume, calculating the dot product between the filter and input. As a result of this process, the input is transformed into network learning filters that, according to LeCun et al. (1998), activate when they detect particular kinds of features at particular spatial coordinates.

A diagram of several cubes

Description automatically generated

Simple lines and edges can be found in the uppermost layers, whereas more intricate patterns, such as textures and object components, can be found in the lowermost layers. CNNs are exceptionally good at image processing and computer vision applications because convolutional layers may learn these features in a hierarchical fashion (Goodfellow et al., 2016).

***b. Pooling Layers***

The pooling layers, sometimes referred to as subsampling or downsampling layers, come after the convolutional layers. A pooling layer's main purpose is to gradually shrink the representation's spatial size, which lowers the network's computational load and parameter count as shown in the image below. This reduction aids in managing overfitting in addition to lowering the computational load (Goodfellow et al., 2016).

A diagram of a pooling scheme

Description automatically generated

The most popular type of pooling is called max pooling, in which the maximum value found inside the filter area is selected at each stage as a filter of a given size passes over the input data. In the process, less important data is discarded and the most prominent traits are extracted. Although it is less popular than max pooling, average pooling is a different type of pooling that determines the average value for every patch on the feature map (Scherer et al., 2010).

***c. Fully Connected Layers***

The fully connected (FC) layers are at the very end of a CNN design. As is common in classic neural networks, these layers link every neuron in one layer to every other layer's neuron. Utilising the high-level features that the convolutional and pooling layers have acquired, the fully connected layers divide the input into different classes according to the training dataset (Krizhevsky et al., 2012).

The final fully connected layer in a classification job typically outputs the probabilities of the input falling into each class, and is frequently followed by a softmax activation function. By training, the network learns to categorise the input image into the appropriate category by modifying the weights of the neurons in these layers to minimise the prediction error (Goodfellow et al., 2016).

***d. Additional Components: ReLU and Dropout Layers***

Modern CNN architectures frequently include extra layers like dropout layers and ReLU (Rectified Linear Unit) in addition to these fundamental parts. By using a non-linear activation function, the ReLU layer gives the network non-linearity, which enables it to learn more intricate patterns. However, during training, the dropout layer randomly deactivates a portion of the neurons, which helps to prevent overfitting and strengthens the network (Krizhevsky et al., 2012; Srivastava et al., 2014).

A diagram of a network

Description automatically generated

## Deep CNN Architectures

### Advancements and Innovations

***1. Introduction of Deep Architectures***

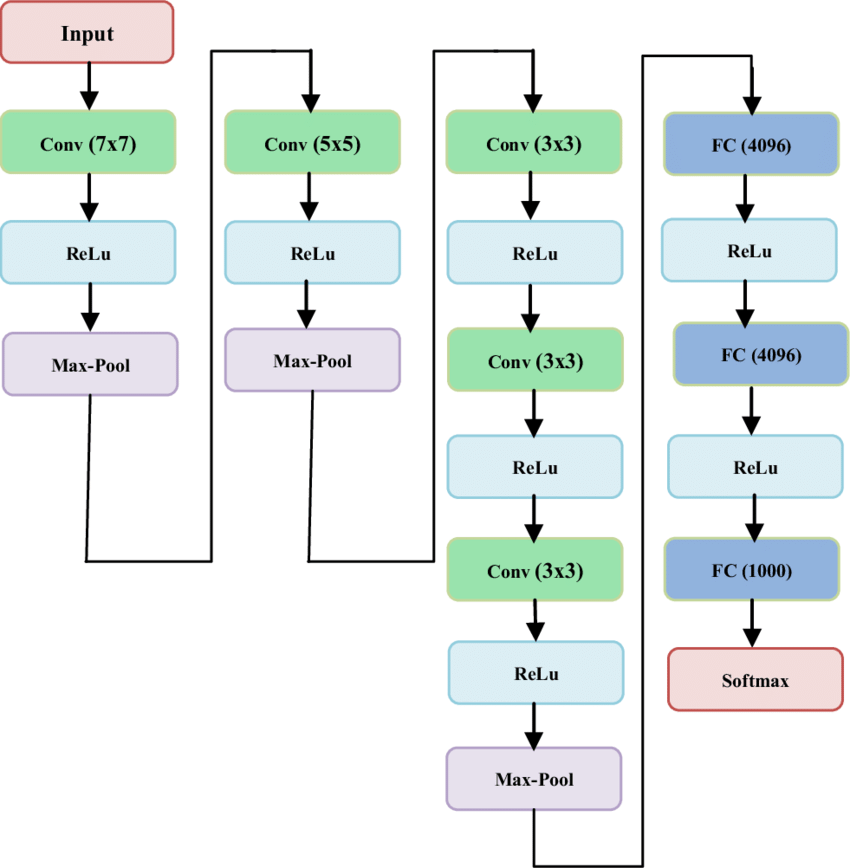
With the advent of deep architectures, particularly with the introduction of AlexNet by Krizhevsky et al. (2012) was a turning point in the development of deep learning. Compared to its predecessors, it was developed with deeper layers, with three completely linked layers placed after five convolutional layers. The network was able to extract a more intricate and abstract hierarchy of information from the photos thanks to this depth. Rectified Linear Units (ReLU) for the activation functions were also introduced for the first time by AlexNet, which greatly accelerated the training process. It also introduced the use of dropout layers, a method of preventing overfitting in which certain neurons are randomly deactivated during training to keep the model from becoming overly dependent on particular properties of the training data (Krizhevsky et al., 2012).

***2. Subsequent Developments in CNN Architectures***

After AlexNet's breakthrough, there was a rush in the creation of new CNN architectures, each of which made a distinct contribution to the field:

* **ZFNet:** Also known as Zeiler & Fergus Net, this technology was developed by Zeiler and Fergus (2014) and played a significant role in expanding our knowledge of CNNs. In essence, it was an improved version of AlexNet. ZFNet made a significant contribution by using deconvolutional layers to illustrate how convolutional layers functioned, giving researchers new insight into how CNNs operate and extract features. Understanding how different CNN layers react to different input data attributes was made easier with the use of this visualisation (Zeiler and Fergus, 2014).

A diagram of a diagram

Description automatically generated

* **GoogLeNet/Inception**: With its inception module, GoogLeNet, commonly referred to as Inception v1, was innovative when it was first presented by Szegedy et al. (2015). This module gave the network the capacity to select filters at different sizes at each layer, enabling it to record data at different scales. Although this architecture employed computing resources more effectively than its predecessors, it was larger and deeper. In order to reduce the overall number of parameters, GoogLeNet also used average pooling rather than fully connected layers at the top of the network (Szegedy et al., 2015).
* **VGGNet**: The simplicity and richness of VGGNet, created by Simonyan and Zisserman (2014), were noteworthy. Its homogenous architecture featured 16–19 layers, each of which had a tiny (3 x 3) convolution filter. This consistency and depth demonstrated how important depth was to the design of effective CNN architectures. Deeper networks can result in greater feature representation and, ultimately, higher performance; this was made possible in large part by VGGNet (Simonyan and Zisserman, 2014).
* **ResNet**: With the introduction of skip connections, He et al. (2016) established Residual Networks (ResNet), which marked a significant innovation. By means of these connections, input from one layer could "skip" one or more levels and be incorporated into the output of a subsequent layer. This architecture made it possible for gradients to go through the network more efficiently, which solved the disappearing gradient issue. ResNet made it possible to train networks with hundreds or even thousands of layers deep while preserving gains in performance (He et al., 2016).
* **DenseNet**: Huang et al. (2017) presented Dense Convolutional Network (DenseNet), which advanced the concept of network connectivity. Every layer in DenseNet is feed-forward coupled to every other layer. Maximum information flow between network levels was guaranteed by this architecture. Because each layer had direct access to the gradients from the loss function and the original input signal, DenseNet was very effective at minimising the issue of vanishing gradients and produced more robust and diverse feature extraction (Huang et al., 2017).

## Challenges and Limitations of CNNs

Although CNN architectures and their applications have made great strides, there are still a number of obstacles and restrictions facing the field:

* **High Computational Demand:**One of the main obstacles in training and deploying CNNs is its enormous computational requirement. Because of their many layers and parameters, deep neural networks (DNNs) demand a lot of processing power. As a result, they frequently require strong GPUs and large memory capacities in order to train and infer effectively (Canziani et al., 2016). The computational prerequisite of these models may be a challenge for smaller organisations and researchers, as it restricts their use in situations with limited resources.
* **Need for Large Datasets:** Large volumes of labelled training data are usually needed for CNNs to operate well and be able to generalise. This reliance on big data presents problems, especially in fields where it is impossible or impractical to obtain such large and varied datasets. Concerns regarding ethics and privacy are also brought up by the requirement for large amounts of data, particularly when handling sensitive data like private photos (Goodfellow et al., 2016).
* **Issues with Overfitting:** One of the biggest challenges in CNN training is still overfitting. When a model learns the training set too thoroughly—including the noise and outliers—it becomes unable to generalise to new, unobserved data, leading to this problem. Overfitting has been addressed with techniques like dropout, data augmentation, and regularisation, but it is still a serious problem, particularly when working with small or unbalanced datasets (Srivastava et al., 2014).
* **Interpretability:** Concerns about CNN interpretability, sometimes known as the "black box" issue, are becoming more widespread in the industry. It gets harder to grasp how these networks arrive at particular judgements as they get deeper and more complicated. In crucial fields like healthcare, banking, and law—where knowing the reasoning behind a decision is crucial—this lack of openness may provide challenges (Castelvecchi, 2016).

## Facial Recognition Techniques

### Evolution and Fundamentals

Facial recognition is a widely applied use of computer vision technology in many different industries. User authentication processes, security protocols, social media interactions, and other areas are greatly impacted by facial recognition technology. The route of technical growth, especially with the introduction of Deep Convolutional Neural Networks (CNNs), is marked by significant technological advancement combined with challenging issues.

### a. Definition and Fundamentals

Utilising a person's facial characteristics to identify or validate their identity is the fundamental purpose of facial recognition technology. This process analyses features of the face such as the lines of the lips, cheeks, and forehead, as well as the form of the jaw and distance between the eyes. Real-time facial data collection is also an option, as is the use of pre-existing photos or videos. The basis of this work is often complex algorithms that extract, process, and compare various facial features (Jain et al., 2016).

A person with a face id

Description automatically generated

### b.  Historical Context and Evolution

Although facial recognition has been known since the mid-1900s, significant advancements in the discipline were made in the 1990s with the development of automated systems that could recognise faces more correctly (Chellappa et al., 1995). The initial approaches measured many face features using geometric techniques. As machine learning and more potent computer techniques emerged, these methods evolved into increasingly sophisticated algorithms that could handle a variety of facial recognition tasks.

### c. Applications Across Domains

Facial recognition technology finds its use in a wide range of fields.

• Security and Law Enforcement: Used to identify individuals in crowds, airports, and criminal investigations.

• Authentication and Access Control: Used in devices like computers and cellphones to provide secure user authentication.

• Social Media and Entertainment: Used for photo tagging, user experience customisation, and character animation.

• Healthcare: Martinez and Benavente (1998) state that facial analysis is discovering new uses in patient identification and disease diagnosis.

### CNNs in Facial Recognition

The use of neural networks, particularly CNNs, has completely changed the field of facial recognition. By learning and extracting hierarchical feature representations from vast amounts of facial data, CNNs have significantly increased the accuracy and efficiency of facial recognition systems (Lawrence et al., 1997). The development of neural network technology has allowed for previously unattainable levels of accuracy, from straightforward models to complex CNN designs. CNNs are the foundation of contemporary image processing, dating back to the creation of the Neocognitron and LeNet-5.

These networks' hierarchical structure, which resembles that of the human visual brain, made it possible to extract and interpret complicated characteristics from images with efficiency (LeCun et al., 1998).

Their capacity to acquire intricate, hierarchical feature representations is ideal for facial recognition, which requires handling and identifying high-dimensional data with different levels of position, lighting, and expression variability (Zhao et al., 2003). Even under difficult circumstances, CNNs can effectively identify between several individuals by extracting important information from facial photos.

The field of facial recognition technology has advanced significantly over the years, with more complex deep learning-based techniques replacing more conventional principal component analysis-based techniques like Eigenfaces(Turk and Pentland, 1991).

The use of CNNs and deep learning in facial recognition has resulted in significant gains in efficiency and accuracy. The use of CNNs for this task marked the beginning of the transition to deep learning in facial recognition. Two notable examples of this change are Google's FaceNet and Facebook's DeepFace (Taigman et al., 2014). Particularly in unconstrained contexts with fluctuations in lighting, position, and expression, these systems greatly outperformed standard methods by using deep CNNs to learn rich and discriminative representations of facial traits.

The advancement from 2D to 3D face recognition algorithms is noteworthy for resolving problems such as inconsistent lighting and misaligned faces. 3D models provide a more complete and precise representation of the facial structure in addition to being resistant to the previously noted issues (Bowyer et al., 2006).

A collage of a person's face

Description automatically generated

### Challanges in Facial Recognition

Even with its advancements, facial recognition still faces several challenges that could reduce its reliability and accuracy. These challenges become more apparent when Deep Convolutional Neural Networks (CNNs) are used for facial recognition applications.

1. ***Variability in Conditions***

·       **Nature of Variability**: The primary challenge in facial recognition is adjusting to variations in illumination, posture, expression, and occlusions. These factors significantly affect the quality of facial images, which in turn affects the performance of recognition systems.

For instance, a face's features can appear differently in different lighting conditions and have varied orientations depending on their position. Facial expressions have the same temporary distorting effects on facial features as occlusions, such as hats, spectacles, or other objects that block parts of the face (Ricanek & Tesafaye, 2006).

·       **CNNs' Approach to Variability:** CNNs develop intricate feature representations that are largely resistant to these kinds of changes in order to handle these variabilities. The network's capacity to acquire hierarchical features—from basic edges and textures in the top layers to more intricate structures in the lower levels—is partially responsible for this ability. However, stable training procedures and access to large and varied datasets are necessary for CNNs to be trained to this degree of invariance (Zhang et al., 2016).

1. ***Data Augmentation and Multi-Angle Training***

·        **Techniques for Data Augmentation:** In dynamic situations, data augmentation plays a critical role in enhancing CNN's facial recognition performance. Data augmentation involves inflating the training dataset artificially through label-preserving adjustments like image scaling, rotation, and cropping. Perez and Wang (2017) claim that this process helps build a larger dataset with a wider variety of possible permutations, which enables the CNN to learn more broadly applicable features.

·       **Multi-Angle Training Strategies**: Another helpful tactic is multi-angle training, in which the network is taught using facial pictures captured from various angles. This method is crucial for developing a system that recognises faces in a variety of orientations, which is frequently needed in real-world applications. Exposure to a wide range of angles helps the model learn how to deal with positional fluctuations (Masi et al., 2016).

1. ***Overfitting***

·       **The Challenge of Overfitting**: Overfitting is a major issue in machine learning, especially when there is a deficiency in the variety or volume of training data. In the domain of facial recognition, overfitting occurs when a CNN model becomes worthless on new, unseen data because it is too good at identifying the finer points and noise in the training set.

·       **Mitigating Overfitting**: To combat overfitting, several techniques are employed.To counteract overfitting, multiple strategies are utilised:

* **Dropout:** This method involves randomly deactivating a subset of neurons during the training process in order to prevent the network from becoming overly dependent on any one set of features and to promote the learning of more durable and universal features (Srivastava et al., 2014).
* **Regularisation:** Methods like L1 and L2 regularisation prevent the model from learning too complex features that might not transfer well to other domains by including a penalty term in the loss function.
* **Transfer Learning:** This involves using a large dataset to fine-tune a pre-trained model for the specific task at hand. By allowing the model to start with a robust set of learned features, transfer learning reduces the likelihood of overfitting.on smaller datasets (Pan and Yang, 2010).

## Transfer Learning

### Concept and Significance

Transfer learning has emerged as a key technique in the field of deep learning, especially for applications like facial recognition where data availability and diversity may be limited. This section explores the concept of transfer learning, its application to facial recognition using CNNs, and its integration with OpenCV to enhance efficiency and performance.

1. **Definition and Significance:** Transfer learning is the process of using a neural network that has undergone prior training for a similar task on a different machine. It is particularly helpful in deep learning since it reduces the quantity of data collection and processing needed to train new models from scratch because it may leverage pre-existing knowledge (Pan and Yang, 2010).
2. **Effectiveness in Deep Learning**: Applications in deep learning have shown how effective transfer learning can be. It enables the application of neural networks trained on large datasets (such as ImageNet) to specific tasks for which a smaller amount of data may be available. This approach significantly reduces training time and computational resources while maintaining or even improving performance (Yosinski et al., 2014).

### Application in Facial Recognition

a.     **Application in Facial Recognition**: Transfer learning is the technique of using CNNs that have previously been trained on a large and diverse dataset to perform facial recognition tasks when it comes to face identification. The primary layers of these CNNs are altered for use in more specific facial recognition applications after being trained to extract generic features.

b.     **Case Studies and Examples**

* **DeepFace**: One of the best examples of transfer learning in action for face recognition is Facebook's DeepFace model. By using a deep CNN that has been pre-trained on a sizable facial dataset, it achieves facial verification performance that is almost human-level (Taigman et al., 2014).
* **OpenFace**: Another notable example of how transfer learning can be beneficial for broader applications is OpenFace. This open-source application recognises faces using a neural network that has already been trained.
* **Networks like VGGNet and ResNet**: Face recognition tasks have made extensive use of these networks as their foundation. Their deep architectures facilitate the extraction of intricate facial features—features that are essential for precise identification. These networks are useful for transfer learning because they can extract a variety of facial features from large datasets (Parkhi et al., 2015; He et al., 2016).

c.     **Advantages**

* **Reduced Training Time and Data Requirement**: Using pre-trained networks reduces the training time and data requirements significantly.
* **Enhanced Performance**: Transfer learning can improve accuracy, especially for problems with little to no labelled data..
* **Flexibility and Adaptability:**It allows for adaptability to new activities without necessitating a substantial degree of retraining.

d. **Recent Progress**: The previous few years have seen a substantial advancement in facial recognition technology thanks to transfer learning. This progress is mostly due to transfer learning's ability to leverage attributes and data from previously trained models, which improves the effectiveness and precision of facial recognition tasks. Studies like Schroff et al. (2015) developed groundbreaking models like FaceNet, which combined deep CNNs with transfer learning to dramatically enhance facial recognition accuracy. For instance, FaceNet achieved notable improvements by employing a deep network trained on an enormous dataset to provide a compact Euclidean embedding for each face (Schroff et al., 2015)

## OpenCV For Facial Recognition

a.**Overview of OpenCV:** OpenCV (Open Source Computer Vision Library) is a well-known open-source software library for machine learning and computer vision. It provides a broad range of methods and instruments for image processing and computer vision applications, including facial recognition.

A logo with a colorful circle

Description automatically generated with medium confidence

There are various ways in which OpenCV and transfer learning can be combined:

* **Facilitating Pre-Processing:** OpenCV can be used to preprocess images in order to get them ready for CNN inference and training. This includes resizing, normalisation, and augmentation activities.
* **Model Integration:** Pre-trained models can be imported into OpenCV environments for further deployment and optimisation. OpenCV supports many deep learning frameworks, making it versatile for applications using transfer learning.

**b.Performance Enhancement**: Facial recognition systems can be made more accurate and efficient by fusing OpenCV's powerful image processing capabilities with CNNs' expertise in transfer learning.

**c.Practical Implementations**:Integrating OpenCV with transfer learning enables real-time facial recognition, which is crucial for applications such as interactive systems and monitoring.

**d.Resource Optimization:** Transfer learning in conjunction with OpenCV can improve

efficiency on low-processing-power devices without compromising accuracy.

**e.Addressing Challenges**: OpenCV and transfer learning work well together to solve some of the problems of facial recognition, namely the need for large datasets and the variety of surrounding variables. Strong image processing capabilities in OpenCV can enhance the quality of CNN input data, while transfer learning ensures effective learning even in the presence of sparse data.

1.     **Fine-Tuning in Transfer Learning**:Fine-tuning entails modifying a pre-trained network's weights marginally to make it more suitable for the facial recognition task. In order to customise the generalised features discovered from massive datasets for more specialised facial recognition applications, this procedure is essential.

2.     **Strategies for Handling Overfitting**:To avoid overfitting and guarantee that the models perform well when applied to fresh, untested facial data, transfer learning models frequently make use of strategies like data augmentation, dropout layers, and regularisation.

3.     **Cross-Domain Transfer Learning**:Cross-domain transfer learning, in which a model trained on a particular kind of dataset (such as general objects) is modified for facial recognition, has been the subject of recent research. The robustness and usefulness of facial recognition systems are increased by this method.

4.     **Incremental Learning in Facial Recognition**:Incremental learning techniques emphasise updating the facial recognition model on a continual basis as new data becomes accessible, enabling the model to change and adapt with time.

5.     **Combining Traditional Techniques with Deep Learning**:To improve facial recognition performance, some studies have tried with merging deep learning with conventional machine learning algorithms. The advantages of both approaches are used in this hybrid strategy.

### Why use OpenCV

* **Versatility and Ease of Use:** OpenCV is a favourite option for developers and researchers due to its wide variety of features and interoperability with multiple programming languages.
* **Efficient Image Processing Capabilities**: Because of its superior image processing and transformation capabilities, OpenCV is a great choice for the preparation stages needed for transfer learning (Bradski, 2000).
* **Seamless Integration with Deep Learning Frameworks**: Pre-trained deep learning models can be easily integrated with OpenCV, enabling effective fine-tuning and use in facial recognition tasks.

### OpenCV in this Project

* The choice of OpenCV in this project is driven by its robustness, efficiency, and compatibility with deep learning models. OpenCV’s tools enable the effective implementation of transfer learning techniques, enhancing the performance of CNNs in facial recognition.
* By leveraging OpenCV’s functionalities, this project aims to overcome typical challenges in CNNs, such as computational intensity and data limitations, while ensuring high accuracy and performance in facial recognition.

### 2.6.3 Haar Cascades Explained

Viola and Jones made a significant contribution to the field of computer vision with the introduction of Haar cascades in their landmark 2001 publication, especially in the area of object detection. The efficacy of this technology is especially well-known in face detection tasks. By combining OpenCV, a feature-rich open-source computer vision toolkit, with Haar cascades, developers may quickly and accurately perform face identification in a variety of applications. Here, we explore how Haar cascades function inside the OpenCV framework and explain how this method is used in face detection projects.

A diagram of a structure

Description automatically generated

**How Haar Cascades Work**

Gaining an understanding of Haar features, integral pictures, the AdaBoost algorithm, and cascading classifiers is necessary in order to comprehend how Haar cascades function in OpenCV to recognise frontal faces. Let's dissect these elements to provide a thorough explanation.

Using a machine learning-based methodology, Haar cascades train a cascade function using a large number of both positive and negative images. It then recognises items in subsequent photos by using the features it has learned. First, the basic contrast features known as Haar features—which are akin to Haar wavelets—are extracted. In the context of face detection, these features—which are chosen from a subregion of an image—are utilised to efficiently capture the existence of object-like patterns, such as the nasal bridge or the line separating the forehead from the backdrop (Viola & Jones, 2001).

A diagram of a process

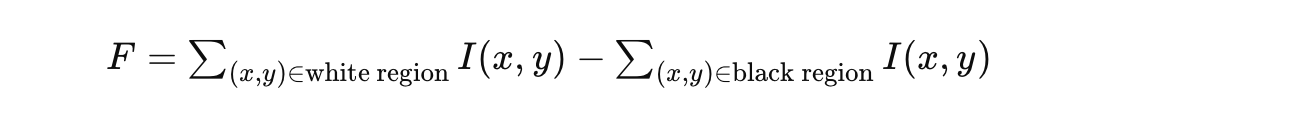
Description automatically generated

The detection method quickly determines the sum of pixel values in image subregions by using an integral image idea. This method makes real-time processing possible by greatly speeding up the feature calculating stage. Then, from a bigger set of features, a machine learning algorithm, usually AdaBoost, picks a few key features and builds classifiers using these features. Because these classifiers are set up in a cascade, simpler classifiers are executed first, swiftly excluding negative sub-windows from additional processing and cutting down on the amount of time needed to compute object detection in an image (Viola & Jones, 2001).

1. **Haar Feautures**  
 Originally introduced by Papageorgiou et al. (1998), Haar features are straightforward digital picture features used for object recognition that were later employed by Viola and Jones (2001) for face detection. In essence, they are particular arrangements of contrasting rectangular areas in a picture. These properties, when used for face detection, are able to recognise traits like the nasal bridge in a frontal face being brighter than the eyes.

A collection of two or three rectangles could be the appearance of a common Haar feature. The difference in the total pixel brightness between these regions determines the value of a Haar feature.

Mathematically, if I (x,y)*I*(*x*,*y*) represents the pixel intensity at position (x,y)(*x*,*y*), then for a two-rectangle feature, the feature value F*F* can be calculated as shown below



**2. Integral Images**

Integral Images, also called summed area tables, were presented by Viola and Jones (2001) as an efficient way to compute the sum of pixel intensities over rectangular areas. Regardless of the size of the rectangle, an integral image enables the quick computation of the sum of the pixel values in any rectangular area using just four array references.

Given an image I*I*, its integral image II*II* at a location (x,y)(*x*,*y*) is the sum of all pixels above and to the left of (x,y)(*x*,*y*), inclusive:

A black and white symbol

Description automatically generated

**3.AdaBoost Algorithm**

A smaller subset of useful features is chosen from a larger set using the AdaBoost (Adaptive Boosting) algorithm, which is then used to train classifiers using these features. According to Freund and Schapire (1997), this approach entails selecting the best feature iteratively at each stage, taking into account the features' capacity to classify instances while assigning greater weight to examples that were previously misclassified.

For face detection, each Haar feature is considered a weak classifier. AdaBoost combines these weak classifiers into a strong classifier, defined as a weighted sum:  
 

where C(x)*C*(*x*) is the strong classifier, hi(x)*hi*​(*x*) are the weak classifiers (Haar features), αi*αi*​ are the weights determined by AdaBoost, and T*T* is the total number of selected features.

**Cascading Classifiers**

A cascaded technique was suggested by Viola and Jones (2001) to shorten computing times and increase detection accuracy. This method divides the classifiers into phases, with a powerful classifier at each stage. If an image region fails any step, it's immediately eliminated as not containing a face; yet, for an image region to be classified as a face, it needs to pass every stage.

By adopting a cascade approach, computational resources are directed towards face-like regions that show promise, thereby significantly reducing the amount of calculations required for regions lacking faces.

**Implementation in OpenCV**

An intuitive interface is offered by OpenCV (Open Source Computer Vision Library) for the application of Haar cascades in face detection. Pre-trained Haar cascade models for identifying faces and other objects are included in the library. Loading the Haar cascade file, turning the picture to grayscale (because Haar features don't need colour information), and then applying the detectMultiScale function to detect faces are the steps in the process.

### 2.6.4 Application in Projects with OpenCV

In the context of a project that utilizes OpenCV for face detection, Haar cascades offer several advantages:

**Rapid Detection**: Haar cascade-based detection is exceptionally quick thanks to the use of integral pictures and cascading classifiers, making it perfect for real-time face detection applications. For the purpose of identifying faces in photos or video streams, OpenCV offers pre-trained Haar cascade models.

**Versatility**: Although Haar cascades are most famous for face identification, they may be trained to recognise a wide range of objects. Because of its adaptability, the project's capabilities can be expanded beyond face detection to include the identification of characteristics or items relevant to the project's particular requirements.

**Accessibility:** The integration of Haar cascades into projects is made easier by OpenCV's implementation of them. Because using OpenCV just requires a small number of lines of code to invoke a Haar cascade detector, even individuals with no background in machine learning or computer vision may do complex object recognition (Bradski, 2000).

**Customization**: OpenCV offers tools to train bespoke Haar cascade classifiers with user-provided images for projects with specific needs. This implies that developers can design detectors specifically for particular project objects of interest.

# **Chapter 3: Methodology**

## Project Planning and Design

**Project Overview:** With the use of Haar cascades and OpenCV, I hope to create a face recognition system that can identify faces in pictures or video streams. After detection, I want to use machine learning models to identify specific faces by analysing the generated signatures.

**Face Detection:** Using Haar cascade classifiers to implement face detection,  
The goal of my project is to create a technique that will enable me to create distinct signatures for faces that have been identified.

**Face Recognition:** My objective is to identify faces by matching the generated signatures to an already-existing database of well-known people.

## Selection of Tools and Technologies

**Hardware Requirements:** Macbook air 2019 8gb ram, 256 gb rom.

**Programming Language:** I will use Python 3.9

**Libraries/Frameworks:**

* + I plan to use OpenCV for face detection with Haar cascades.
  + Facenet would be used for Facial Recognition
  + TensorFlow/Keras will be used for any involved neural network models.
  + NumPy for numerical computations and handling arrays.
  + PIL (Python Imaging Library) for image processing tasks.

**Development Environment:** I will utilize Jupyter Notebook for prototyping and testing.

## CNN Architecture and Transfer learning models

I chose to use the FaceNet model, which is renowned for its sophisticated convolutional neural network architecture, in my research. Following testing of several CNN models, FaceNet performed better than the others, leading to this decision. FaceNet is a sophisticated architecture with a wide network of convolutional layers. To aid in dimensionality reduction, these are cleverly organised into modules or blocks that combine pooling layers, activation functions like ReLU, and convolutional procedures. My FaceNet implementation is based on the Inception-ResNet architecture, which combines the robust representation offered by residual connections with the computational efficiency of the Inception model.

FaceNet can efficiently interpret input images by extracting increasingly abstract and higher-level features from the raw pixels thanks to this smart architecture. The network eventually casts these features into an embedding space where the Euclidean distances reflect similarity between the faces. For the face recognition tasks in my study, this embedding space is essential since it calculates the distances between faces' embeddings, enabling precise face comparison.

Through the use of FaceNet in my project, I significantly improved our capacity for precise facial recognition. This further demonstrated how important transfer learning is when using cutting-edge machine learning methods to solve real-world problems. FaceNet's design, which is based on neural networks and deep learning, gives the recognition system a strong base and allows me to achieve impressive accuracy levels with a small amount of processing power.

For my project, I tested a number of CNN architectures, and FaceNet performed far better than the others, obtaining training accuracy of 88.85% and testing accuracy of 99.04%. The performance of the other models I examined, including MTCNN, VGGNet, ResNet, and MobileNet, was noticeably worse, particularly when it came to processing a single image. ResNet and MobileNet followed suit, exhibiting high training accuracies but testing accuracies of 35.5% and 29.29%, respectively. For instance, MTCNN had a training accuracy of 87.81% but a testing accuracy of just 48.60%; VGGNet, having a flawless training score, plummeted to 36.44% in testing. These models were prone to problems such as overfitting and insufficient training and testing data.

Based on my findings, I have concluded that model accuracy is greatly increased when the dataset has a more balanced distribution. Moreover, the effectiveness of training and testing is increased when FaceNet is used as an embedding tool. But I had trouble with other CNNs, particularly with Dlib. The installation of Make, Xcode, and Homebrew was necessary for the setup process, and I ran across multiple file verification problems, which increased the difficulty of completing the project.

A screenshot of a computer program

Description automatically generated

*Fig. .Training and Testing accuracy of MobileNet*

A screenshot of a computer

Description automatically generated

*Fig. Training and Testing accuracy of FaceNet*

A screenshot of a computer code

Description automatically generated

*Fig. Precision, Recall and F1 score of FaceNet*

A screenshot of a number

Description automatically generated

*Fig. Training and Testing accuracy of MTCNN*

A screenshot of a computer

Description automatically generated

*Fig. Training and Testing accuracy of ResNet*

A screenshot of a computer

Description automatically generated

*Fig. Training and Testing accuracy of VGGNet*

## Dataset Collection and Preperation.

In my facial recognition project, data preprocessing is essential to make sure the input photos are properly formatted so the FaceNet model can analyse them and provide accurate embeddings. The input photos were carefully cropped and taken from a range of real-world, everyday situations that friends and colleagues submitted. This method was specifically chosen to guarantee a high level of realism and dependability in the real time environments, hence improving the realism and applicability of real-time simulations.In order to properly prepare the photos for the model, my preprocessing pipeline consists of multiple crucial phases. The data preprocessing stages and their importance in my project are examined in detail below:

1. **Face Detection**: My preprocessing workflow starts with identifying faces in the pictures. This is crucial because FaceNet needs the facial region of an image in order to produce precise embeddings. For this challenge, I use a Haar Cascade Classifier, which is a well-known and effective face detection technique. This classifier locates the face regions in each image by going through them in a predetermined directory.
2. **Managing Non-Detections:** Occasionally, a picture might not include any faces at all, or the classifier might fail to find them. My pipeline is set up to skip these photos and move on to the next in such cases. This method preserves the quality and relevancy of the FaceNet input data by guaranteeing that only photos with distinct faces are analysed further.
3. **FaceNet Image Preprocessing:** Following a face's successful detection, the area around the face is removed and fine-tuned to conform to FaceNet's input requirements. This improvement consists of:

i)Converting the image from BGR (Blue, Green, Red) to RGB in order to comply with FaceNet's requirement for RGB and OpenCV's default reading format.

ii)Adjusting the extracted face region to a standard size of 160 x 160 pixels, which satisfies FaceNet's requirement for input dimensions that are constant.  
Increasing the size of the face picture array to meet FaceNet's batch input requirement even for processing single images.

1. **Embedding Generation:** I use FaceNet to create embeddings after preprocessing the facial photos. In essence, these embeddings are high-dimensional vectors that hold the extracted facial features from the pictures. Accurate face recognition is made possible by FaceNet's meticulous training, which places comparable face embeddings closer together in the vector space.
2. **Creating a Facial Embeddings Database**: I set the filename (sans extension) as the key and gather the created embeddings for each image as I process it. This database serves as the foundation for my project's recognition phase, allowing for an effective comparison of an input image's embeddings with those in the database to determine which faces match.
3. **Persistence:** Lastly, I use pickle to serialise and store this embeddings database on disc. This storage technique improves system performance by eliminating the need to reprocess photos for each project run.

# **Chapter 4: System Implementation**

## System Architecture and Design For Facial Recognition

1. **Step 1: Configuring the Environment**

Installing Python and the required libraries (OpenCV, TensorFlow/Keras, NumPy, PIL) will be my first step.  
I'll set up my Jupyter Notebook for testing and developing programmes.

1. **Step 2: Gathering and Preparing Data**

In order to test the face detection and recognition system, I will compile a dataset of face photos.

In the event that I use supervised learning for face recognition, I intend to label photographs.

1. **Step3: Recognition of Faces**  
     
   I'll use OpenCV's Haar cascade classifiers to implement facial detection.

To make sure the face detection module is reliable, I'll test it on a variety of photos.

1. **Step 4: Creation of Face Signatures**

If required, I will investigate and put into practice a technique for creating facial signatures that might entail feature extraction from CNN models that have already been trained, such as FaceNet.

The produced signatures will be kept in an organised manner for future comparison.

1. **Step 5: Identification of Faces**  
     
    My goal is to create a face recognition module that verifies the signature of a detected face by comparing it to a database of known signatures.  
   I'll use similarity metrics to get precise facial recognition.
2. **Step 6: Testing and Integration**  
     
   I'll combine the modules for face detection, signature creation, and recognition to create a seamless system.I intend to properly test the system using a variety of images and scenarios.

My goal is to optimise the system by making adjustments to settings and improving algorithms.

## Overview of the Attendance System

I created an automatic facial recognition system for attendance tracking. This is how I configured it:

1. **Loading the database**: I started by loading an already-existing database of people I knew from a file called data.pkl. In essence, this database was a Python dictionary that mapped individual identifiers (such as names or IDs) to their distinct facial signatures, which are numerical vectors that are derived from the FaceNet model and reflect each person's face.
2. **Initializing My Already Trained Model:** I created facial embeddings from photos using the FaceNet model, a complex deep learning model. The distinct characteristics of a person's face are captured by these high-dimensional vector embeddings. This model was initialised by me to process pictures that were taken from the video feed.
3. **Video Capture:** I began using cv2 to record live footage from the computer's built-in camera.Video Capture (0). I would identify and detect faces in this live broadcast.
4. **Face Detection:** I used a Haar cascade classifier intended for frontal face detection to find faces inside the video frames. For real-time detection applications, Haar cascades are a highly effective method.
5. **Face Recognition:** This is what I performed for each face that I found:

* The face was extracted and scaled to 160x160 pixels to match the input specifications of the FaceNet model.
* FaceNet was used to create a facial embedding for the resized face.
* Euclidean distance was used to compare this embedding with those in the database to determine which one was the closest match. I regarded the face as identified with the appropriate person if the distance was less than a predetermined threshold.

1. **Debounce Logic:** I put in place a debounce system to prevent logging the same person more than once in a short period of time. I looked up the time since a face's last log in to see if it was recognised. The logging attempt was rejected if this length was shorter than the designated debounce period (300 seconds or 5 minutes). The attendance was logged otherwise.
2. **Recording Attendance:** A person's attendance was recorded as soon as they were identified and cleared the debounce check. This required creating a CSV file called attendance\_log.csv and storing their ID and the current timestamp in it. The attendance record was contained in this file.
3. **Display and Termination:** The user was able to view the video feed in real-time, along with any faces that were identified. The script processed every video frame forever in order to identify and detect faces. The script ended by shutting off the video feed and releasing the camera if the user hit the ESC key.

This system's development was a thorough experiment in using facial recognition technology to address a real-world issue. I developed a system that greatly expedites the attendance recording process and provides an insight into the potential of facial recognition technology in common applications by fusing complex models like FaceNet with real-time video processing and implementing logic to handle the nuances of attendance logging.

* 1. Implementation challenges and Solutions

As I worked on this face recognition system, I ran into a number of obstacles that affected many facets of the project, such as problems with environmental setup and model training. Below is a detailed explanation of some of the hurdles I faced and possible solutions.

* + **Model Training And Overfitting:** A major obstacle was faced when training the facial recognition models. The main problem was overfitting, which was partly brought on by the small amount of training images. I had to carefully choose and test on a minimum dataset in order to lessen this, attempting to strike a compromise between the limitations of the available data and the requirement for model correctness.
  + **Hardware Limitations** : My laptop constantly overheated and shut down due to the strain of training and testing models. This presented a risk to the hardware itself in addition to interfering with the process.
  + **Difficulties with Software and Environment:** Jupyter Notebooks were originally recommended due to their interactive development environment. But I regularly ran into kernel problems, which meant I had to redo work and lose my progress. I was forced by this instability to abandon my initial Conda system, which came with its own set of difficulties, and establish a virtual environment on my local machine.
  + **Suggestions for Training Environments:** Considering the problems with local settings, I strongly advise using cloud-based notebooks for testing and training models, such Google Colab. These platforms reduce some of the hardware limitations encountered by providing greater stability and improved access to computational resources.
  + **Data Set homogeneity:** The significance of dataset homogeneity was another realisation. A dataset that has an equal distribution of samples from various classes performs the model more accurately and broadly. This is especially important for facial recognition jobs because there is a lot of variation in faces.
  + **Problems with Dlib and HaarCascades:**

**Dlib:** There were unique difficulties in utilising Dlib for facial recognition. It was difficult to install; other tools like Homebrew, Xcode, and CMake were needed. This made the setup procedure more difficult.

**Haar Cascades:** Despite being lightweight, Haar Cascades had trouble with accuracy, particularly in dimly lit areas or in the presence of impediments like headgear and facial coverings. These drawbacks highlighted the requirement for a face detection system that is more reliable in a variety of settings.

* + **Training on the LFW Database:** The Large Face in the Wild (LFW) database was used to emphasise the limitations of available resources. With my limited computational capabilities and numerous code terminations, training a model from scratch was not feasible due to the dataset's sheer size of over 1,500 photos.
  + **Resolving the Resource Shortage:** In the end, employing a pre-trained model was a practical solution. It enabled me to get a functional degree of accuracy for the attendance system without having to use the substantial resources needed for training from beginning.

Upon evaluation, these difficulties highlighted the difficulties involved in creating a practical application such as an attendance system based on facial recognition. All of the challenges, from the complexities of training models to the practical choices about hardware and software settings, provided insightful insights into striking a balance between aspiration and reality. These experiences also emphasise the value of flexibility and resourcefulness in the face of technical difficulties, emphasising the ongoing interaction between creative problem-solving and technology limitations.

# Chapter 5: Evaluation and Testing

## 5.1 Testing Methodology

The facial recognition-based attendance system's testing phase was an insightful process that revealed the model's capabilities as well as the difficulties and constraints posed by the technologies used, notably OpenCV's use of Haar cascades for face detection. This step was tackled systematically by my colleagues and me, who divided it into discrete sections to assess the system's performance in different scenarios and conditions. Below is a thorough rundown of every phase and the knowledge we acquired:

## 5.2 Performance Evaluation, User Feedback and Pilot Testing

**Initial testing with simple Ui and Camera.**

In order to conduct real-time face identification and recognition tests, the model was integrated with a basic user interface (UI) and live video feeds were captured using my laptop's native camera. Although the goal of this configuration was to replicate a common user scenario, it had several drawbacks:

**Accuracy and Mismatches:** Between the identified faces and the signatures kept in our database, we found a sizable number of mismatches. One of my friends was classified as a "unknown" entity by the algorithm, while I was mistakenly recognised as "Mukhtar" who is another of my friend in the database. This demonstrated the difficulties the model faced in correctly matching the facial signatures from the real-time video stream with those that had been previously saved in the database, highlighting the difficulties brought on by varying lighting, a range of facial emotions, and the general calibre of the video feed.A person with glasses and a green rectangle

Description automatically generated

*Fig. Me being mislabelled as “Mukhtar”*

A white wall with a black towel

Description automatically generated

*Fig. Yasir being mislabelled as Unknown*

**Limitations of Haar Cascades:** The disparities and errors observed highlighted the drawbacks of utilising Haar cascades in OpenCV for face detection. Haar cascades demonstrated worse reliability when confronted with the intricacies of real-world facial recognition, such as fluctuating ambient conditions and obstacles, despite their efficiency for simple detection tasks.

**Enhanced testing with improved Conditions**

Motivated by the first results, I moved on to the second testing phase with the goal of improving the settings and fixing the earlier-found issues. At this point, there were:

**Improved Image Quality**: I considerably reduced the environmental factors that had previously impacted the system's performance by making sure that there was better illumination and using higher quality images for the database.

**Accurate Recognition:** A noticeable increase in accuracy was the result of the modifications made. My Friend and I were appropriately recognised by the algorithm as "Almansur" and "Yasir," respectively. This achievement validated the modifications made after the initial testing and showed the model's potential for accurate facial recognition under ideal circumstances.

A person taking a selfie

Description automatically generated

*Fig. My face correctly identified and recognised by the algorithm.*

A person sitting on the floor

Description automatically generated

*Fig. Yasir’s face correctly recognised by the algorithm.*

**Real World Application In the Attendance System.**

Encouraged by the successful results of the second phase, I proceeded to test the system's ultimate objective: taking attendance. At this point, there were:

**Integration with Attendance System:** In order to evaluate the facial recognition model's applicability in a real-world setting, I integrated it with the attendance tracking system.

**Accurately Logging and Successful Recognition:** There were no issues with the system's ability to accurately log our attendance or recognise us. This testing phase verified that, under certain conditions, the previously highlighted issues were addressed and the model could operate as intended within the attendance system.

A close-up of a document

Description automatically generated

*Fig. Image of attendance getting logged for me.*

A screenshot of a computer

Description automatically generated

*Fig. Image of attendance getting Logged for Yasir and Mukhtar.*

A number with numbers and numbers

Description automatically generated with medium confidence

*Fig. Attendance successfully logged in the csv file.*

**Insights and future Directions**

From the first difficulties to the final success, the extensive testing procedure yielded several important insights:

**Importance of Optimising Conditions:** It was clear that the system's performance was greatly impacted by the surrounding environment and image quality. Achieving maximum accuracy requires optimising these elements.

**Limitations of detecting technologies:** The shortcomings of Haar cascades brought to light the necessity of more advanced or supplementary detection techniques, particularly for a variety of dynamic and varied real-world applications.

**Potential for Real-World Application**: In spite of early difficulties, the attendance system's effective integration and performance highlighted the model's potential for useful applications, if the right modifications and optimisations were made.

# Chapter 6: Ethical Considerations

A number of ethical issues arise when implementing real-time face recognition systems, like the one this project discusses. Although there are many advantages to these technologies, there are also many ethical, privacy, and potential misuse concerns. Based on the current state of the world, this conversation examines these issues and provides solutions.

## Privacy Concerns

The emergence of facial recognition technology presents noteworthy obstacles to individual privacy. Public anonymity is compromised by the ability to recognise or authenticate people from digital photos or video frames (Bowyer, 2004). This is made worse by real-time applications, which make it possible to track people without their knowledge and gather private biometric information. More intrusive data collecting techniques may result from the need for operational precision under changing circumstances, such as lighting or camera angles. For example, making up for inadequate lighting might mean stepping up surveillance, which would increase privacy violations. Haar cascades, which are most useful for frontal face detection, might need camera placement that directly captures faces, sometimes in ways that subjects are unaware of or do not consent to (Viola & Jones, 2004).

## Ethical Implications of Facial Recognition

The potential of facial recognition technology to conduct extensive profiling and surveillance poses serious ethical concerns. According to Garvie et al. (2016), the use of technology in public areas to track and monitor behaviour might be interpreted as a kind of social control that goes against ethical standards pertaining to consent and autonomy. Due to dataset biases, there is a danger of misidentification, which raises issues of justice and discrimination and exacerbates already-existing societal disparities, especially among specific demographic groups (Buolamwini & Gebru, 2018).

Discussions on the moral use of facial recognition technology have been sparked by its widespread use worldwide. Diverse cultural perspectives regarding surveillance and privacy may lead to different legislative frameworks being adopted in different regions. This discrepancy can result in a patchwork of safeguards and vulnerabilities globally, making the ethical assessment of the technology more difficult (Pagallo, 2017).

## 6.3 Addressing Ethical and Privacy Concerns

A diversified approach is required to mitigate the ethical and privacy concerns related to facial recognition technologies. Strong legal frameworks that are developed and upheld can provide people with protection against unauthorised surveillance while also ensuring that the use of these technologies is consistent with society norms and values (Kaminski & Malgieri, 2019). Furthermore, potential abuses can be avoided by addressing ethical issues early in the technology's design and development process. To lower the danger of misidentification, this entails putting in place procedures for informed consent, openness, and the ability to opt out as well as creating more inclusive and objective datasets (Mittelstadt, 2019).

A further important factor in determining the moral application of facial recognition technology is stakeholder participation and public dialogue. Policymakers can better grasp the public's concerns and ideals by including a diverse range of society members in conversations about the deployment of these technologies. This can result in more democratic and socially conscious policies (Wachter, 2018).

The trade-off between technical efficacy and privacy is highlighted by the effect of lighting on model performance. Systems may need to use lighting conditions or viewing angles that make the people being watched feel uncomfortable in order to guarantee accurate facial recognition (Zhang & Gao, 2019). This technical necessity may make it acceptable to employ illumination conditions that people might find offensive in any other situation, thus undermining their right to privacy.

# Chapter 7: Conclusion

## 7.1 Key Findings

Although my built facial recognition system worked well in a few scenarios, it had trouble correctly identifying people in other situations. These difficulties are mostly caused by the built-in limitations of the technologies and techniques used, particularly the face identification technique that makes use of Haar cascades. I go into further depth about these restrictions and how they affect the functionality of the system below.

**Sensitivity to Facial Expressions:** When a person's facial expression changed, the system's capacity to identify them changed considerably. This sensitivity results from the fact that the model's facial embeddings collect minuscule facial characteristics that vary depending on an individual's expression. Because of this, the embeddings for the same person's frown and smile may appear sufficiently different to alter the accuracy of recognition.

**Effect of Low Light Conditions:** The system's operation in low light presented another important difficulty. Even though Haar cascades are effective for frontal face detection in well-lit environments, they have trouble correctly identifying faces in dimly lit environments. This restriction results in lower facial recognition accuracy because the cascade relies on visual characteristics that become less noticeable in dark settings.

**Problems with Non-Uniform Lighting:** In the same way that low light levels affected system performance, so did non-uniform or contrasting lighting circumstances. If the lighting is too strong and parts of the face are in darkness, the resulting image may not match the facial signatures in the database accurately enough for identification. This problem emphasises how essential uniform, consistent lighting is to the system's optimal functioning.

**Obstacles and Facial Coverings:** The system was biassed to identify fully visible, unobstructed frontal faces because face detection was done using Haar cascades. Therefore, any kind of covering or obstruction—like hats, masks, or even glasses, to some extent—could have a major effect on the system's capacity to identify and detect faces. These objects have the potential to mask important facial features or change the way the detection algorithm interprets the form and curves of the face, which might cause recognition errors.

The complexity and variability of human faces, combined with the external factors that affect recognition, are the fundamental causes of these difficulties. The use of Haar cascades for detection has two drawbacks: although it is effective and straightforward, it also limits the system's ability to adjust to the variety of real-world scenarios that face recognition is intended to function in. It would probably take a diverse strategy to address these restrictions, including but not limited to:

Improving the face embeddings robustness which can be achieved by applying more advanced models or training methods that produce embeddings that are less susceptible to changes in expression or partial obstructions.

Enhancing low-light performance which could involve applying pre-processing techniques to standardise illumination in photos or investigating more sophisticated detection algorithms that are less sensitive to changes in lighting.

Adapting to obstructions: Researching techniques for detection and recognition that are especially made to deal with partially-veiled faces or identify people using traits that aren't covered up.

In summary, although the existing system shows the promise of face recognition technology for uses such as automatic attendance tracking, overcoming its limits will need a deeper comprehension of the technology as well as the variety of environments in which it must function. This knowledge, along with continuous developments in machine learning and computer vision, portends more adaptable and trustworthy facial recognition systems down the road.

## 7.2 Implications for Practice

Under controlled conditions, the facial recognition model that I constructed has an impressive accuracy of 99.0%. But because it relies on Haar cascades for face detection, there are notable differences from this high accuracy rate in its performance during testing and real-world application. This disparity and the difficulties it presents for real-time application call for a more thorough analysis and clarification of the underlying problems, as well as ideas for future advancements.

**Difficulties in Implementing Haar Cascades in Real-Time:**

**Performance in Various Environments:** Although Haar cascades offer effective face detection, their performance is noticeably less reliable in a variety of environmental settings. Variations in illumination, obstacles, and even different facial expressions can have a big impact on how accurate face detection is, which in turn affects how the recognition process works as a whole. The high accuracy rate of the model is usually attained in ideal circumstances, which are not always representative of the intricate reality of real-world settings.

**Real-Time Processing Restrictions:** The system's implementation in real time brought Haar cascades' limits to light. Under perfect circumstances, the identification procedure is efficient on a per-frame basis; however, in dynamic contexts or with video streams of different quality, it can become noticeably slower and less dependable. As a result, the system may appear erratic and unpredictable, and performance variations may erode user trust and the system's usefulness.

**Accuracy vs. Usability Trade-off:** A crucial trade-off between theoretical correctness and practical usability is highlighted by the great accuracy attained in controlled experiments contrasted with practical implementation issues. Essentially, even though the model is capable of reliably identifying faces in a lab context, its performance is reduced because of the intricacies of real-world applications, which involve factors that are difficult to regulate or account for.

## 7.3 Recommendations for Future Research

Overcoming these obstacles necessitates a comprehensive strategy that goes beyond simple adjustments to the current paradigm. Some things to think about for upcoming upgrades are:

**Advanced Detection Algorithms**: Investigating increasingly complex face detection algorithms that can more elegantly navigate obstacles and withstand changes in environmental conditions. For real-time applications, machine learning models trained on a variety of datasets with a wide range of situations may provide better detection reliability.

**Optimising Real-Time Performance:** Improving the model's execution and performance in real time. This could entail reducing the computational load without noticeably sacrificing accuracy, using hardware acceleration when feasible, and optimising the code for speed.

**Hybrid Detection Approaches:** To increase stability and dependability, combining Haar cascades with other detection techniques. For instance, Haar cascades could be used in conjunction with deep learning-based detectors to enable quick processing under less demanding circumstances.

**Environmental Adaptation:** Applying pre-processing techniques to normalise photos taken in different scenarios prior to the face identification algorithm processing them. Detection reliability could be improved by using methods like dynamic histogram equalisation for lighting correction or filters to lessen the influence of obstructions.

In conclusion, even though the facial recognition model shows remarkable theoretical accuracy, there are issues with its practical implementation that draw attention to the drawbacks of using Haar cascades alone for face detection with OpenCV. Enhancements to the algorithm are necessary, but so is a more comprehensive analysis of the operational environment and the possible integration of complementary technologies in order to address these problems and increase real-time performance and dependability.

# **References**

1. Blanz, V., & Vetter, T. (2003). Face recognition based on fitting a 3D morphable model. IEEE Transactions on pattern analysis and machine intelligence, 25(9), 1063-1074.
2. Bowyer, K. W., Chang, K., & Flynn, P. (2006). A Survey of Approaches and Challenges in 3D and Multi-modal 3D+2D Face Recognition. Computer Vision and Image Understanding, 101(1), 1-15.
3. Bradski, G. (2000). The OpenCV Library. Dr. Dobb's Journal of Software Tools.
4. Canziani, A., Paszke, A., & Culurciello, E. (2016). An Analysis of Deep Neural Network Models for Practical Applications. arXiv preprint arXiv:1605.07678.
5. Castelvecchi, D. (2016). Can we open the black box of AI?. Nature News, 538(7623), 20-23.
6. Chellappa, R., Wilson, C. L., & Sirohey, S. (1995). Human and Machine Recognition of Faces: A Survey. Proceedings of the IEEE, 83(5), 705-741.
7. Fukushima, K. (1980). Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. Biological Cybernetics, 36(4), 193–202.
8. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
9. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. Proceedings of the IEEE conference on computer vision and pattern recognition, 770-778.
10. Hubel, D.H., & Wiesel, T.N. (1968). Receptive fields and functional architecture of monkey striate cortex. The Journal of Physiology, 195(1), 215-243.
11. Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K.Q. (2017). Densely Connected Convolutional Networks. Proceedings of the IEEE conference on computer vision and pattern recognition, 4700-4708.
12. Jain, A. K., Ross, A., & Nandakumar, K. (2016). Introduction to Biometrics. Springer.
13. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Advances in neural information processing systems, 25, 1097-1105.
14. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.
15. Masi, I., Rawls, S., Medioni, G., & Natarajan, P. (2016). Pose-Aware Face Recognition in the Wild. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 4838-4846.
16. Martinez, A. M., & Benavente, R. (1998). The AR Face Database. CVC Technical Report.
17. Pan, S. J., & Yang, Q. (2010). A Survey on Transfer Learning. IEEE Transactions on Knowledge and Data Engineering, 22(10), 1345-1359.
18. Parkhi, O. M., Vedaldi, A., & Zisserman, A. (2015). Deep Face Recognition. British Machine Vision Conference.
19. Perez, L., & Wang, J. (2017). The Effectiveness of Data Augmentation in Image Classification Using Deep Learning. arXiv preprint arXiv:1712.04621.
20. Ricanek, K., & Tesafaye, T. (2006). Morph: A Longitudinal Image Database of Normal Adult Age-Progression. Proceedings of the 7th International Conference on Automatic Face and Gesture Recognition (FGR06), 341-345.
21. Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A Unified Embedding for Face Recognition and Clustering. Proceedings of the IEEE conference on computer vision and pattern recognition, 815-823.
22. Scherer, D., Müller, A., & Behnke, S. (2010). Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. In: Artificial Neural Networks – ICANN 2010. ICANN 2010. Lecture Notes in Computer Science, vol 6354. Springer, Berlin, Heidelberg.
23. Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv:1409.1556.
24. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research, 15, 1929-1958.
25. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going Deeper with Convolutions. Proceedings of the IEEE conference on computer vision and pattern recognition, 1-9.
26. Taigman, Y., Yang, M., Ranzato, M., & Wolf, L. (2014). DeepFace: Closing the Gap to Human-Level Performance in Face Verification. Proceedings of the IEEE conference on computer vision and pattern recognition, 1701-1708.
27. Turk, M., & Pentland, A. (1991). Eigenfaces for Recognition. Journal of Cognitive Neuroscience, 3(1), 71-86.
28. Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How Transferable are Features in Deep Neural Networks? Advances in Neural Information Processing Systems, 27.
29. Zeiler, M.D., & Fergus, R. (2014). Visualizing and Understanding Convolutional Networks. European Conference on Computer Vision, 818-833.
30. Zhao, W., Chellappa, R., Phillips, P. J., & Rosenfeld, A. (2003). Face recognition: A literature survey. ACM Computing Surveys (CSUR), 35(4), 399-458.

# ***Appendices***

### A. Code Explanations and Snippets

**Code for Signature processing and facial embedding storage.**

*# Import necessary libraries*

*import os*

*from os import listdir*

*from PIL import Image*

*from numpy import asarray, expand\_dims*

*from matplotlib import pyplot*

*from tensorflow.keras.models import load\_model*

*from keras\_facenet import FaceNet*

*import numpy as np*

*import pickle*

*import cv2*

*# Initialize the Haar Cascade for face detection and the FaceNet model for face embeddings*

*HaarCascade = cv2.CascadeClassifier(cv2.samples.findFile(cv2.data.haarcascades + 'haarcascade\_frontalface\_default.xml'))*

*MyFaceNet = FaceNet()*

*# Define the directory containing images and prepare an empty dictionary to store embeddings*

*folder = 'Images/'*

*database = {}*

*# Iterate over each image file in the folder*

*for filename in listdir(folder):*

*# Construct the full file path*

*path = os.path.join(folder, filename)*

*# Read the image using OpenCV*

*gbr1 = cv2.imread(path)*

*# Skip processing if the image couldn't be loaded*

*if gbr1 is None:*

*print(f"Failed to load the image: {filename}")*

*continue*

*# Detect faces in the image*

*Img\_face = HaarCascade.detectMultiScale(gbr1, 1.1, 4)*

*# Skip the image if no faces are detected*

*if len(Img\_face) == 0:*

*print(f"No faces detected in the image: {filename}")*

*continue*

*# Extract the first detected face (x1, y1, width, height)*

*x1, y1, width, height = Img\_face[0]*

*x2, y2 = x1 + width, y1 + height # Calculate bottom-right corner*

*# Convert the image from BGR to RGB color space*

*gbr = cv2.cvtColor(gbr1, cv2.COLOR\_BGR2RGB)*

*# Crop the face region from the image*

*face = gbr[y1:y2, x1:x2]*

*# Resize the face to 160x160 pixels (expected by FaceNet)*

*face = cv2.resize(face, (160, 160))*

*# Add a batch dimension to the face image array*

*face\_expanded = expand\_dims(face, axis=0)*

*# Generate an embedding for the face using FaceNet*

*signature = MyFaceNet.embeddings(face\_expanded)*

*# Store the embedding in the dictionary with the filename (without extension) as the key*

*database[os.path.splitext(filename)[0]] = signature*

*# Serialize the dictionary of embeddings and save it to a file*

*myfile = open("data.pkl", "wb")*

*pickle.dump(database, myfile)*

*myfile.close()*

***Import Libraries:*** *The code starts by importing the required Python modules and libraries, which offer the following functionalities: machine learning model management (tensorflow.keras, keras\_facenet), image processing (PIL, cv2), numerical operations (numpy), and data serialisation (pickle).*

***Initialization:*** *A pre-trained model is used to initialise the Haar Cascade classifier, which is utilised to identify faces in images. Additionally, the FaceNet model—which creates facial embeddings—is initialised. For the next tasks of face recognition and picture processing, these initializations are essential.*

***Setup Directory and Database:*** *An empty dictionary is ready to hold the face embeddings, and the directory containing the photos to be processed is supplied. Filenames (without their extensions) will be mapped by the dictionary to the appropriate face embeddings.*

***Image Processing Loop****: Every image file in the designated directory is iterated through by the code. We carry out the following actions for every image:*

***a.Load Image****: OpenCV is used to read the image into memory.*

***b.Face Detection****: Faces in the image are identified via the Haar Cascade classifier. The image is skipped if there are no faces found.*

***c.Face Extraction and Conversion****: To prepare it for model input, the detected face region is extracted, shrunk to the dimensions requested by FaceNet (160x160 pixels), and converted to RGB colour space (from BGR, which is OpenCV's default). A batch dimension is also added.*

***Embedding Generation****: The FaceNet model creates an embedding for every processed face. This embedding, which can be readily compared to other embeddings, is a high-dimensional numerical representation of the face that captures its distinctive traits.*

***Embedding Storage:*** *Using the filename of the image (without the extension) as the key, the created embedding for every face is kept in the dictionary that was initially initialised. By doing this, a dataset of face embeddings that may be used to identify specific people in the photos is created.*

***Serialisation and Saving:*** *Lastly, pickle is used to serialise (transform into a byte stream) the dictionary containing all of the face embeddings, and the dictionary is then saved to a file. In subsequent sessions, this file can be imported to access the face embeddings without requiring that the photos be reprocessed.*

**Real time Testing of Facial Recognition**

*import cv2*

*import numpy as np*

*from PIL import Image*

*from numpy import asarray, expand\_dims*

*# Initialize the video capture*

*cap = cv2.VideoCapture(0)*

*# Load the Haar cascade for face detection*

*HaarCascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade\_frontalface\_default.xml')*

*# Assuming MyFaceNet is correctly defined and loaded*

*MyFaceNet = FaceNet() # Placeholder - ensure this is correctly initialized*

*while True:*

*\_, gbr1 = cap.read()*

*wajah = HaarCascade.detectMultiScale(gbr1, 1.1, 4)*

*if len(wajah) > 0:*

*x1, y1, width, height = wajah[0]*

*else:*

*x1, y1, width, height = 1, 1, 10, 10*

*x1, y1 = abs(x1), abs(y1)*

*x2, y2 = x1 + width, y1 + height*

*# Directly use the BGR image for face extraction and resizing*

*face = gbr1[y1:y2, x1:x2]*

*face = cv2.resize(face, (160, 160))*

*# Prepare the face for the FaceNet model*

*face\_expanded = expand\_dims(face, axis=0)*

*signature = MyFaceNet.embeddings(face\_expanded)*

*min\_dist = 100*

*identity = ' '*

*for key, value in database.items():*

*dist = np.linalg.norm(value - signature)*

*if dist < min\_dist:*

*min\_dist = dist*

*identity = key*

*cv2.putText(gbr1, identity, (x1, y1-10), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255, 255, 0), 2, cv2.LINE\_AA)*

*cv2.rectangle(gbr1, (x1, y1), (x2, y2), (0, 255, 0), 2)*

*cv2.imshow('res', gbr1)*

*k = cv2.waitKey(5) & 0xFF*

*if k == 27: # Esc key to stop*

*break*

*cv2.destroyAllWindows()*

*cap.release()*

***First Setup Video Capture Initialization:*** *Using OpenCV's VideoCapture function, the webcam is initialised for video capture.*

***Haar Cascade Classifier:*** *The classifier for facial recognition is loaded. Because of its speed, this classifier works well for face detection in real-time applications.*

***FaceNet Model Initialization:*** *Assuming it has been properly loaded and defined elsewhere, the FaceNet model is initialised. This approach creates high-dimensional vectors, or embeddings, for faces that can be utilised to differentiate between various people.*

***Processing in Real Time Loop***

***a.Frame Capture:*** *It takes one frame at a time from the camera in an endless loop.*

***b.Face detection****: The Haar Cascade classifier is used to process each frame in order to identify faces. The coordinates of faces detected are returned by the classifier.*

***c.Face Processing:*** *Using the coordinates that the Haar Cascade classifier supplied, if at least one face is found, the face region is extracted. Next, the size of this region is adjusted to match the FaceNet model's predicted dimensions.*

***d.Embedding Generation****: It uses the FaceNet model to create an embedding for every face that is detected. This is feeding the face region through the model after, if needed, transforming it into a format that fits the model.*

***Real Time Testing of Attendance system***

*import cv2*

*import numpy as np*

*from numpy import expand\_dims*

*from datetime import datetime*

*import pandas as pd*

*from keras\_facenet import FaceNet*

*import pickle*

*# Load database*

*with open("data.pkl", "rb") as myfile:*

*database = pickle.load(myfile)*

*# Initialize FaceNet*

*MyFaceNet = FaceNet()*

*# Initialize the video capture*

*cap = cv2.VideoCapture(0)*

*# Load Haar cascade for face detection*

*HaarCascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade\_frontalface\_default.xml')*

*# Debounce logic variables*

*debounce\_time = 300 # Seconds*

*last\_recognized = {}*

*def log\_attendance(identity):*

*# Check debounce time*

*if identity in last\_recognized and (datetime.now() - last\_recognized[identity]).total\_seconds() < debounce\_time:*

*return # Skip logging if within debounce time*

*last\_recognized[identity] = datetime.now()*

*# Log attendance*

*now = datetime.now().strftime("%Y-%m-%d %H:%M:%S")*

*new\_entry = pd.DataFrame([[identity, now]], columns=["ID", "Timestamp"])*

*new\_entry.to\_csv("attendance\_log.csv", mode='a', header=False, index=False)*

*print(f"Logged attendance for {identity} at {now}")*

*while True:*

*\_, frame = cap.read()*

*faces = HaarCascade.detectMultiScale(frame, 1.1, 4)*

*for (x, y, w, h) in faces:*

*face = frame[y:y+h, x:x+w]*

*face = cv2.resize(face, (160, 160))*

*face\_expanded = expand\_dims(face, axis=0)*

*signature = MyFaceNet.embeddings(face\_expanded)*

*min\_dist = 100*

*identity = None*

*for key, value in database.items():*

*dist = np.linalg.norm(value - signature)*

*if dist < min\_dist:*

*min\_dist = dist*

*identity = key*

*if identity:*

*log\_attendance(identity)*

*cv2.imshow('Attendance System', frame)*

*if cv2.waitKey(5) & 0xFF == 27: # ESC key*

*break*

*cv2.destroyAllWindows()*

*cap.release()*

***Initial Setup***

***Database Loading:*** *Pickle is used to load an existing database (data.pkl) containing face embeddings and matching identities. Individuals can be identified using this database.*

***FaceNet Initialization****: Creates embeddings for faces found in the video feed by initialising the FaceNet model.*

***Video Capture Initialization:*** *Configures the default webcam (device 0) for video capture.  
The Haar Cascade classifier is loaded in order to identify faces in the video feed. In terms of real-time face detection, this approach is effective.*

***Debounce Logic Setup****: Sets up variables for debounce logic to stop the same person from being logged in repeatedly. It specifies a dictionary to record the most recent time each identity was recognised, as well as a debounce duration of 300 seconds.*

***Ongoing Video Processing***

***Frame Capture:*** *The script records frames from the webcam in an endless loop.  
Face detection: The Haar Cascade classifier, which yields the coordinates of detected faces, is used to process each frame in order to identify faces.*

***Face Processing and Recognition****: For every face that is detected, the face region is extracted, resized to the 160x160 pixel dimensions that FaceNet expects, and a face embedding is created.Next, using the Euclidean distance to determine which embedding is the closest match, it compares this one with those in the database. The match is the identity with the shortest distance.*

***Attendance Logging****: The system logs the attendance to attendance\_log.csv by attaching the identity and the current timestamp to the identity if it recognises it but does not do so within the debounce period (to prevent spamming).Using a debounce method, it determines whether a person's attendance has been recorded in the recent debounce\_time seconds. If not, it skips to prevent multiple entries and logs the attendance.*

***Termination and User Interface***

***Presentation and Exit:*** *The "Attendance System" pane contains the live video stream. The ESC key releases the webcam and closes any OpenCV windows in addition to ending the loop and, so, the programme.*

*A screenshot of a computer

Description automatically generated*

*Fig. Snapshot overview of the /Images Folder.*