PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERI
Ministry of Higher Education and Scientific Research

UNIVERSITY OF ECHAHID HAMMA LAKHDAR - EL OUED

FACULTY OF EXACT SCIENCES

Computer Science Department

Presented by:
Beggas Ikram
Guadda Nour
Kahla Chiraz

Under the supervision of :Guia Sana Sahar

ACADEMIC LECANCE

Title:
Automatic Attendance System by Face Recognition

Board of Examiners

Chairman/ President: University of El Oued
Examiner: University of El Oued

Academic year 2023-2024

First Thanks to Our Supervisor
Guia Sana Sahar

We wanted to express our deep gratitude for your invaluable support and guidance in our project. Your dedication and willingness to help made a huge difference, and we couldn't have done it without you. Thank you for your patience and encouragement and for always doing your best to ensure our success. We are truly grateful for all you have done.

Also, the Board of Examiners, thank you for coming. We are honored by your presence.

Abstract

Face recognition is a technology that identifies or verifies a person's identity by analyzing and comparing patterns in facial features. It involves detecting and extracting facial landmarks. The objective of the current report is to build an automatic attendance system using face recognition with deep learning algorithms, which significantly improve the accuracy and efficiency of face recognition systems compared to non-deep learning methods.

Keywords : Automatic attendance system, Face recognition, Deep learning.

ملخص :

تقنية التعرف على الوجه هي تقنية تحدد أو تتحقق من هوية الشخص من خلال تحليل ومقارنة الأنماط في ملامح الوجه. تتضمن هذه التقنية اكتشاف واستخراج معالم الوجه. يهدف التقرير الحالي إلى بناء نظام حضور تلقائي باستخدام تقنية التعرف على الوجه باستعمال خوارزميات التعلم العميق، التي تحسن بشكل كبير دقة وكفاءة أنظمة التعرف على الوجه مقارنة بالطرق غير المعتمدة على التعلم العميق.

الكلمات المفتاحية : نظام الحضور الاوتوماتيكي ، التعرف على الوجه، التعلم العميق.

Contents

List of Figures

Introduction

The Automatic Attendance System by Face Recognition is an innovative application of deep learning technology to streamline and automate the attendance tracking process. Using advanced facial recognition algorithms, the system identifies and verifies individuals in real-time, enabling accurate and efficient attendance management across diverse environments. This project aims to enhance traditional attendance methods, reduce manual efforts, and provide a secure and reliable solution for attendance tracking.

This report consists of three chapters.

Chapter 01: This chapter explores the reasons behind the choice of face recognition for the attendance system and outlines the core concepts and methodologies in face recognition. It provides a comprehensive foundation for understanding the application of face recognition in this context.

Chapter02 :This chapter delves into the fundamentals of deep learning, centered on developing and training artificial neural networks for various tasks.

Chapter 03 :This chapter details the design and implementation of a face recognition system using deep learning. the model in terms of classification and regression tasks, emphasizing the effectiveness of the designed architecture in recognizing and tracking facial features.

Chapter 1

Face Recognition for Attendance Systems

1.1 Introduction

Traditional attendance systems rely on manual methods or smart cards that may be vulnerable to forgery or time loss. Therefore, we present facial recognition technology as an automatic attendance recording system, as this technology provides many advantages such as safety, efficiency, and reducing the risk of transmitting diseases in epidemic situations. Thanks to the uniqueness and difficulty of replicating a face, this technology provides an innovative and integrated solution for accurate and rapid attendance tracking.

1.2 Attendance tracking methods

There is different attendance tracking methods that organizations and educational institutions use. These methods vary in terms of reliability, efficiency, and cost. Here are some common ones:

1.2.1 Manual Attendance Systems:

These traditional methods rely on physical record-keeping. Examples include pen-and-paper registers, time punch cards, or sign-in sheets. Employees or students manually enter their arrival and departure times.

While automated solutions have largely replaced manual systems, some companies still use them for specific purposes [19]. The disadvantages of manual attendance systems are :

- Time-Consuming: Manual attendance tracking can be highly time-consuming, particularly during peak hours when many in-

dividuals need to log their attendance simultaneously. This often results in queuing and delays, making the process inefficient and disruptive.

- Prone to Errors: Human errors are frequent in manual data entry, leading to inaccuracies in attendance records. Issues such as illegible handwriting, missed entries, or accidental modifications can undermine the reliability of the data. These errors can cause incorrect attendance tracking and difficulties in maintaining precise records.

- Absence of Real-Time Updates: Manual systems do not provide real-time updates, meaning supervisors or managers cannot access current attendance data. This lack of immediacy can delay addressing attendance-related issues, such as identifying absenteeism patterns or verifying employee presence. Consequently, decision-making and responsiveness to attendance matters are impaired.

1.2.2 RFID Attendance Systems:

Radio Frequency Identification (RFID) systems use tags or cards with embedded chips. Employees or students swipe their RFID cards near a reader to record attendance. These systems are efficient but may require additional infrastructure.

1.2.3 GPS-Based Attendance Systems:

These systems track location using GPS technology. Employees' or students' mobile devices record their presence when they enter pre-

defined geofenced areas. Useful for fieldwork or outdoor activities.

Collecting and storing location data raises privacy concerns among employees or students. Moreover, GPS tracking-enabled devices can experience significant battery drainage due to the high power consumption of GPS chips.

1.2.4 Biometric Attendance Systems:

Biometrics involve using unique physical characteristics (such as fingerprints, facial features, or iris patterns) to verify identity. Biometric attendance systems are highly accurate and secure. They eliminate the need for physical tokens (like cards) and prevent buddy punching (where one person clocks in for another).

1.3 why face recognition

Implementing facial recognition technology in attendance systems offers many advantages.
here's why we chose facial recognition as our automatic attendance identifier :

- Security: Face recognition is inherently secure because the facial image is used as the ID. Unlike traditional methods such as ID cards or passwords, a person's face is unique and difficult to replicate or forge. This reduces the likelihood of unauthorized access and ensures that only the right individuals are marked present.

- Efficiency:The process of taking attendance through face recognition is quick and efficient. Students can simply walk in front

of a camera, and their attendance is recorded without any physical contact or effort. This significantly reduces the time taken for attendance compared to manual methods.

- Health Considerations:In situations where health considerations are significant, such as during a pandemic, a contactless method like face recognition becomes more appealing. Individuals don't need to touch any surfaces, reducing the risk of disease transmission.

- Anti-Fraud Measures: Facial recognition adds an additional layer of security to attendance tracking. It is more difficult for someone to manipulate or cheat the system by proxy attendance. The system ensures that the person present is the one who is supposed to be there, thus maintaining the integrity of attendance records.

- Scalability: Face recognition systems can easily scale to accommodate a large number of individuals. Whether in schools, universities, or large corporations, the system can handle a high volume of attendance data without compromising on accuracy or speed.

- Integration with Other Systems: Facial recognition technology can be seamlessly integrated with other security and administrative systems. This allows for automated updates to attendance logs, access control, and even payroll systems, ensuring that all records are synchronized and up-to-date.

Facial recognition technology relies on analyzing the unique features of each individual to distinguish them and verify their identity.

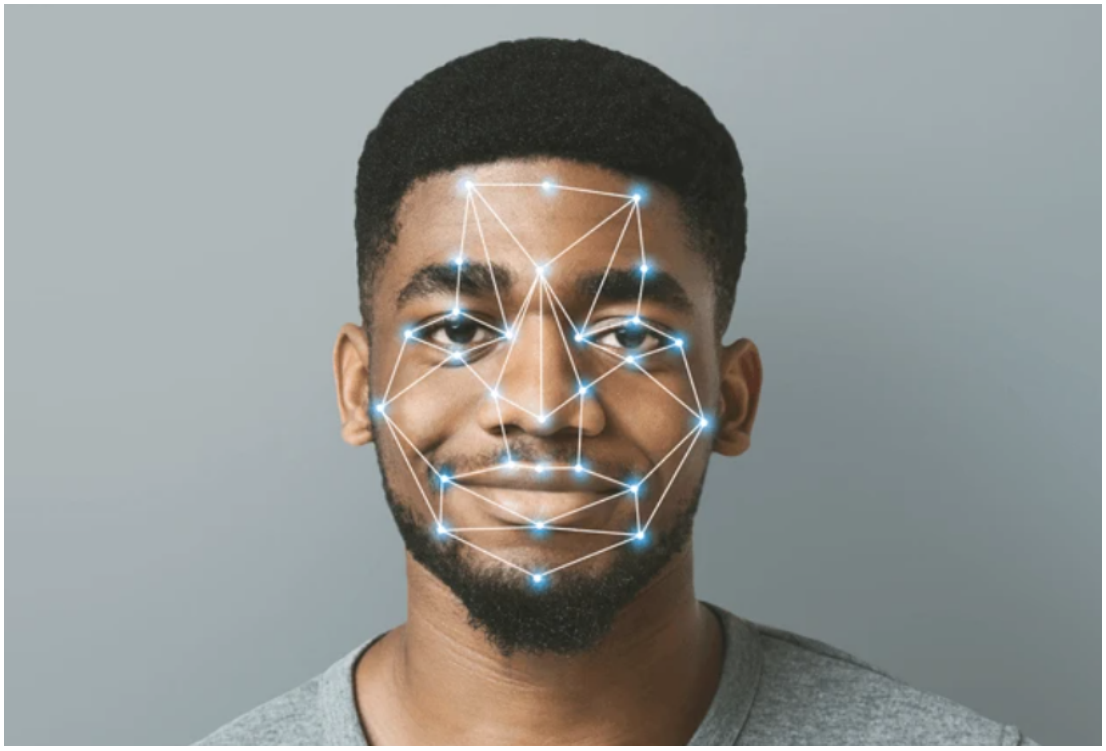

Figure 1.1: The face of a person with the main facial features identified (eyes, nose, mouth ....etc)

1.4 Face Recognition methods

1.4.1 Traditional Methods

Eigenfaces Method: This method utilizes Principal Component Analysis (PCA) to represent faces as eigenfaces, allowing for dimensionality reduction and face recognition. The steps involved are:

1. Data Collection: Gather a large set of face images.

2. Compute the Average Face: Calculate the average of all face images.
3. Subtract the Mean: Subtract the average face from each image to get a set of mean-centered images.
4. Compute the Covariance Matrix: Use these mean-centered images to compute the covariance matrix.
5. Principal Component Analysis: Extract the eigenfaces (eigenvectors) from the covariance matrix.
6. Project Images: Project new face images onto the eigenspace (spanned by the eigenfaces) to obtain a reduced dimensional representation.
7. Comparison and Recognition: Compare the new face representation with known faces using distance metrics such as Euclidean distance [5].

Fisherfaces: Similar to Eigenfaces but uses Fisher's Linear Discriminant Analysis (LDA) instead of PCA. LDA focuses on maximizing the ratio of between-class variance to within-class variance. The steps are:

1. Data Collection: Gather a set of face images.
2. Compute Within-Class and Between-Class Scatter Matrices: Calculate matrices that capture the variance within each class and the variance between different classes.
3. Linear Discriminant Analysis: Extract the Fisherfaces (discriminant vectors) that maximize the separation between different classes.

4. Project Images: Project new face images onto the discriminant subspace defined by the Fisherfaces.

Local Binary Patterns (LBP): This method analyzes local texture patterns of facial images, particularly effective for facial texture analysis. The steps are:

1. Divide Image into Cells: Split the face image into small regions (cells).
2. Compare Pixels: For each pixel in a cell, compare it to its neighboring pixels.
3. Generate Binary Patterns: Create a binary code for each pixel based on whether the neighboring pixels are greater or smaller.
4. Histogram of Patterns: Compute a histogram of these binary patterns for each cell.
5. Concatenate Histograms: Concatenate the histograms from all cells to form a single feature vector.
6. Comparison and Recognition: Compare the feature vector of the new face image with those of known faces using distance metrics.[32]

Haar Cascades: This method employs Haar-like features and cascading classifiers for efficient object detection, including faces. The steps are:

1. Extract Haar-like Features: Compute Haar-like features (e.g., edge, line, and four-rectangle features) from the images.

2. Train Classifiers: Use machine learning techniques to train a series of simple classifiers on these features using positive (face) and negative (non-face) images.
3. Create a Cascade of Classifiers: Form a cascade where each stage consists of a classifier. The cascade is designed so that most non-face regions are quickly discarded while most face regions pass through.
4. Face Detection: Pass the image through the cascade to detect potential face regions.
5. Face Recognition: Recognize faces based on the detected features.[27]

These traditional methods laid the groundwork for face recognition technology, though modern approaches like Convolutional Neural Networks (CNNs) now provide higher accuracy and robustness.

1.4.2 Modern Approaches

Convolutional Neural Networks (CNN):Deep neural networks, particularly CNNs, have shown remarkable success in face recognition tasks by learning hierarchical features directly from images,eliminating the need for handcrafted features.[21]

Siamese Networks:Trains a neural network to differentiate between pairs of images, which is useful for one-shot learning tasks.[18]

Triplet Loss: Utilizes triplets of images (anchor, positive, and negative) to enforce similarity for positive pairs and dissimilarity for negative pairs during training.[33]

Generative Adversarial Networks (GAN):GANs can generate realistic facial images and can be used for data augmentation or improving the quality of training datasets.[2]

3D Face Recognition:Utilizes depth information or 3D models of faces to enhance recognition accuracy and robustness to pose variations.[31]

Template Matching:Compares facial features with pre-defined templates, measuring the similarity between the input face and stored templates.[15]

Facial Landmarks Detection:Identifies key points on the face, such as eyes, nose, and mouth, to characterize facial features and improve recognition accuracy.[14]

An automatic attendance system using facial recognition is a qualitative leap in attendance management. It combines high security and efficient performance with ease of expansion and integration with other systems. By exploiting deep learning techniques, this system can significantly improve the accuracy and speed of attendance recording, making it an ideal choice for schools, universities, and large enterprises.

1.5 Conclusion

In this chapter, we introduce an attendance system using face recognition. Deep learning methods, known for their higher accuracy, are employed in our project. In the next chapter, we will explore deep

learning in greater detail, focusing on the methods used in our system.

Chapter 2

The deep learning technique

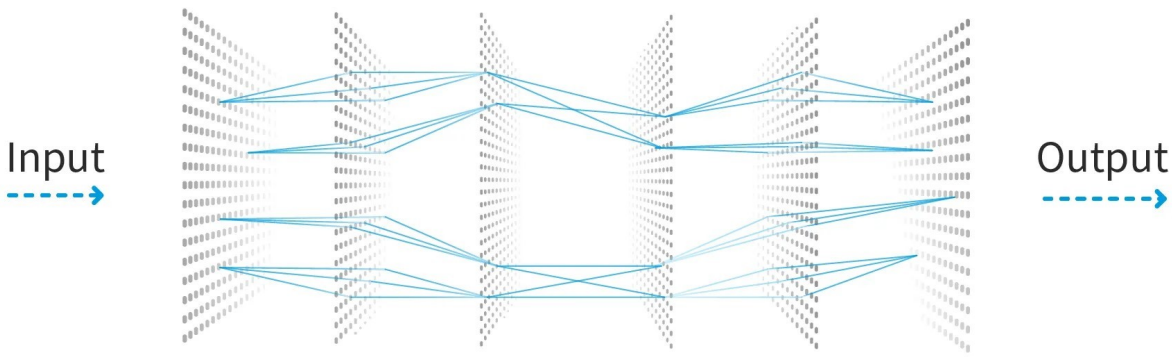

Figure 2.1

2.1 Introduction

Deep learning is a branch of machine learning that focuses on developing and training artificial neural networks to perform tasks without the need for explicit programming. This involves using deep neural networks, which consist of multiple layers, to transform input data through a series of mathematical operations. In this chapter, we will cover how deep learning is used in face recognition, along with other advanced techniques such as transfer learning, VGG models, MobileNet, and ResNet, and how to apply them to object detection.

2.2 what is the deep learning

Deep learning is a sub-field of machine learning that focuses on the development and training of artificial neural networks to perform tasks without explicit programming. It involves the use of deep neural networks, which are composed of multiple layers (hence ”deep”) that transform input data through a sequence of mathematical operations.[10]

2.3 Deep Learning for Face Recognition

Input Data:Images of faces are used as input data. Each pixel in the image represents a value for the intensity of red, green, and blue (RGB) colors.[12]

Convolutional Layers: The first layers in a CNN are typically convolutional layers. These layers use filters (small windows)

to scan the input image. Each filter captures different features, such as edges, textures, or patterns.[23]

Activation Function: After convolution, the results pass through an activation function (commonly ReLU - Rectified Linear Unit). The activation function introduces non-linearity, allowing the network to learn complex patterns.[3]

Pooling Layers: Pooling layers follow convolutional layers to reduce spatial dimensions and computational complexity. Max pooling, for example, retains the most significant information from a group of pixels, discarding less important details.[8]

Flattening: The output from the convolutional and pooling layers is then flattened into a one-dimensional vector. This vector represents the extracted features from the input image.[28]

Fully Connected Layers: These layers connect every neuron from the previous layer to every neuron in the current layer. These layers learn more complex patterns and relationships in the features.[16]

Output Layer: The final layer produces the output of the network. For facial recognition, it could be a probability distribution over predefined classes (e.g., identities). Softmax activation is often used to convert raw scores into probabilities.[30]

Loss Function: A loss function measures the difference between the predicted output and the ground truth (actual labels). The goal during training is to minimize this loss.[24]

Backpropagation: The optimization algorithm (e.g., Stochastic Gradient Descent) adjusts the weights of the neural network in the backward direction, guided by the calculated gradients from the loss function.[29]

Training: The entire process of forward pass (making predictions), calculating loss, backpropagation, and weight adjustment is repeated iteratively on a large dataset. The model learns to recognize facial features and their relationships. Once trained, the deep learning model can take a new image as input and output the predicted class or label, representing the identity or characteristics of the face in the image. The sequence of mathematical operations encoded in the weights of the neural network captures the learned representation of facial features.[11]

2.4 Transfer Learning

Transfer learning, used in machine learning, is the reuse of a pre-trained model on a new problem. In transfer learning, a machine exploits the knowledge gained from a previous task to improve generalization about another. For example, in training a classifier to predict whether an image contains food, you could use the knowledge it gained during training to recognize drinks.[11]

To build Transfer Learning model we should :

- Select a Pre-trained Model: Choose a model pre-trained on a large dataset (e.g., ImageNet).
- Replace the Final Layer: Adapt the model to the specific task by replacing the final classification layer.

- Fine-Tune the Model: Train the modified model on the new dataset.

2.5 Common pre-trained models

2.5.1 VGG (Visual Geometry Group)

It is a deep machine learning model developed by the Visual Geometry Group team at the University of Oxford. It is deep and simple, consisting of cascading transformation layers and a simple structure. It has been successfully used in image classification, medical image analysis, and many other applications.[11]

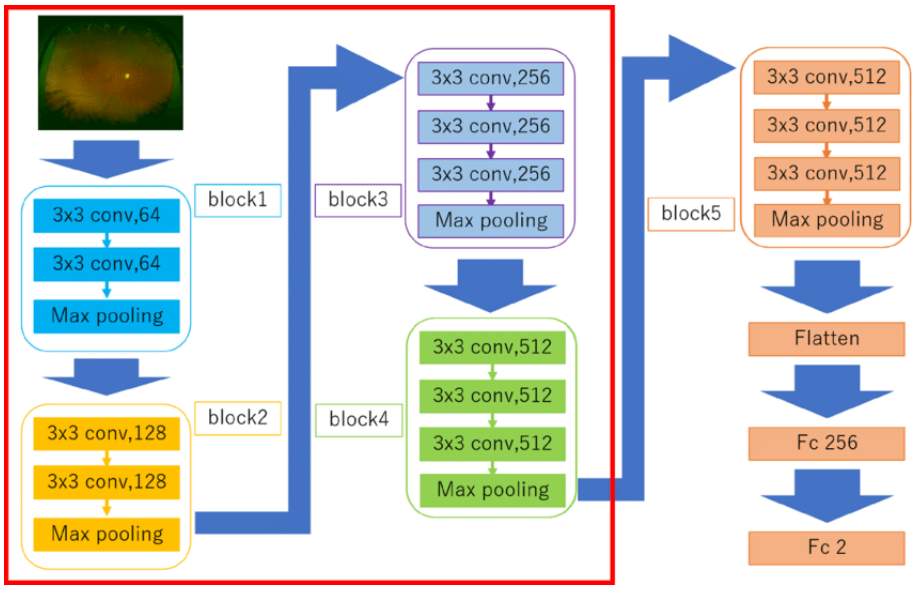

Figure 2.2: Overall architecture of the Visual Geometry Group-16 (VGG-16) model. VGG-16 comprises five blocks and three fully connected layers. Each block comprises some convolutional layers followed by a max-pooling layer.

2.5.2 MobileNet

It is a template specifically for mobile devices developed by Google. It is resource efficient and fast, making it suitable for AI applications on resource-constrained devices such as smartphones.[4] It includes techniques that allow optimizations in memory usage and improvements in performance.

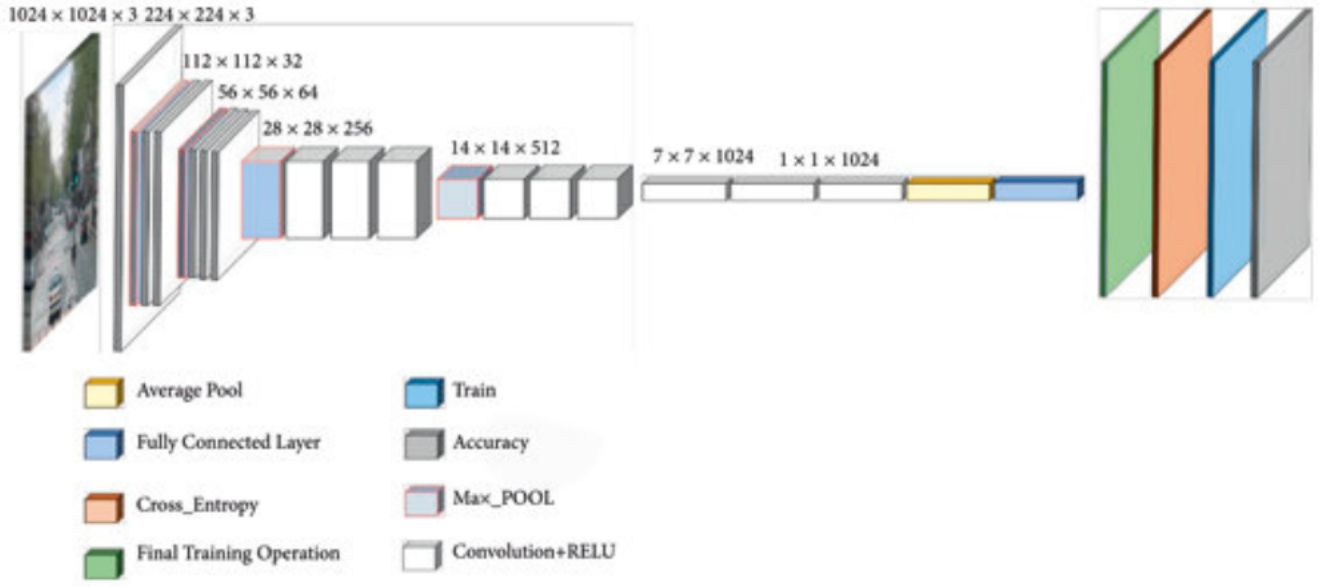

Figure 2.3: The architecture of MobileNet

2.5.3 ResNet (Residual Network)

It is a deep machine learning model developed by researchers at Microsoft Research. It is characterized by the ability to deal with common problems in deep learning such as missing and minimizing drift. It is based on the idea of ”skip connections”, which allows information to be passed directly from layer to layer, which helps reduce information loss.[9]

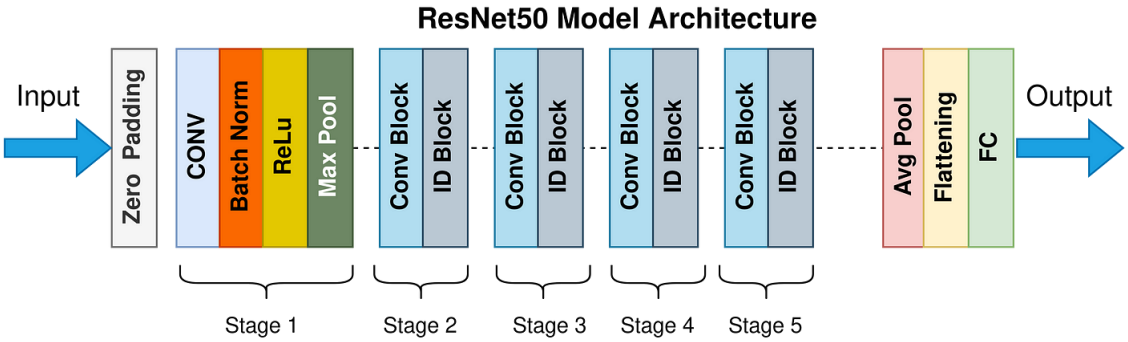

Figure 2.4: ResNet Model architecture

These models are used in many applications such as classification, object detection, image analysis, machine translation, etc., and represent a good starting point for building powerful artificial intelligence systems.

2.6 Object Detection

Object detection allows us to classify the types of things found while also locating instances of them within the image.

Typically, the object detection process includes several basic steps:

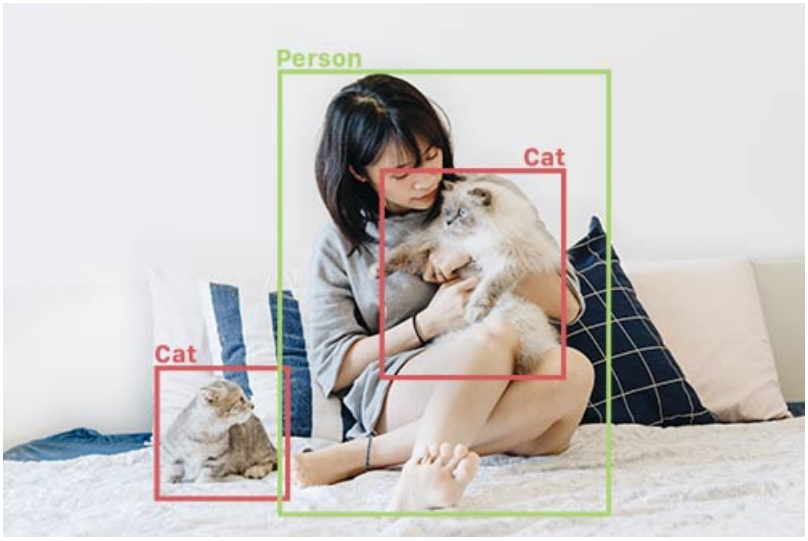

Figure 2.5: an image that contains two cats and a person.

- Image Grid Partitioning: The image is divided into a grid of smaller boxes, and each box is processed independently.

- Detection Model Utilization: A detection model is employed, typically a neural network such as Faster R-CNN, YOLO (You Only Look Once), SSD (Single Shot MultiBox Detector), or others. These models are trained on a dataset containing labeled classifications and bounding box annotations of the objects to be detected.

- Object Classification and Localization: The model classifies the objects within each box and accurately determines their locations.

- Drawing Detection Boxes: Boxes are drawn around the detected objects to highlight them in the image.

- Post-Processing: Additional processing is performed on the detection results, such as filtering out unnecessary results or merging overlapping detection boxes.[11]

2.7 Popular Object Detection Models:

2.7.1 YOLO

(You Only Look Once), developed by Joseph Redmon , frames object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. It looks at the whole image at test time so its predictions are informed by global context in the image. YOLO is known for its speed, making it suitable for real-time

applications.[25]

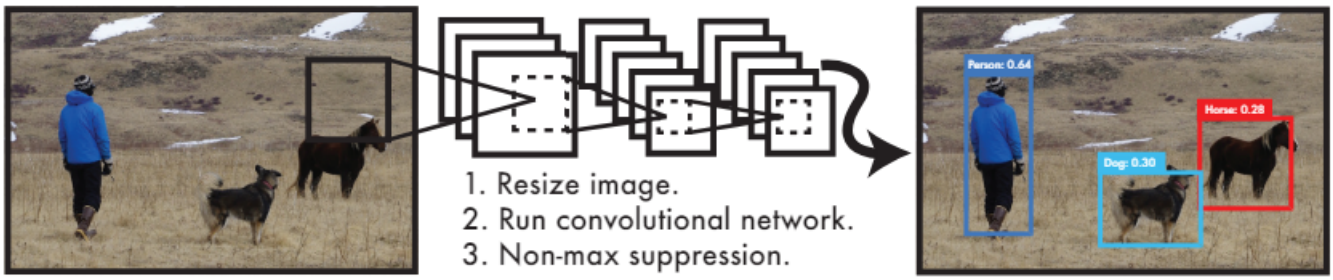

Figure 2.6: The YOLO Detection System.

2.7.2 RCNN

Region-based Convolutional Neural Network (R-CNN) one of the pioneering models that helped advance the object detection field by combining the power of convolutional neural networks and region-based approaches.[22]

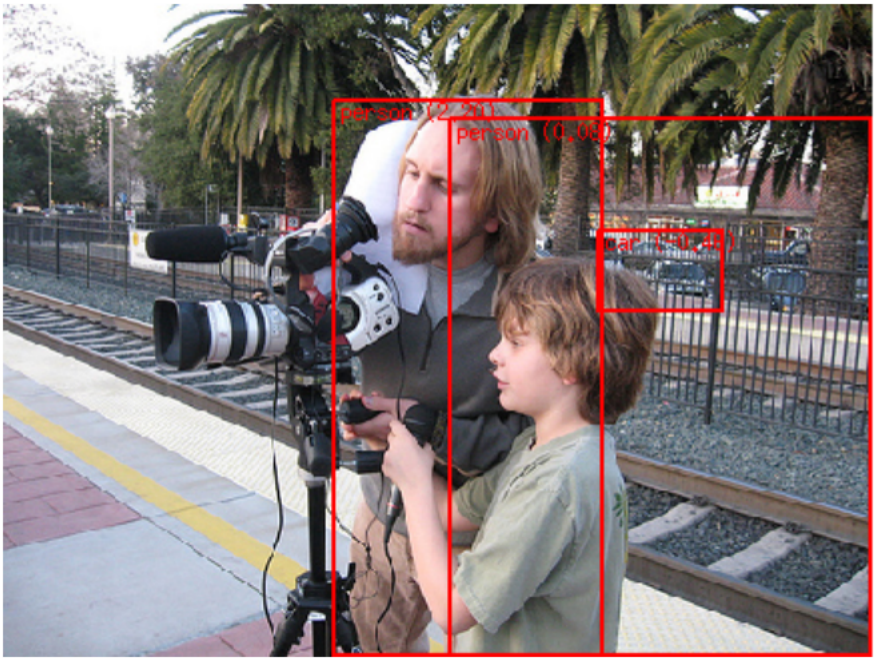

Figure 2.7: RCNN Exemple Result.

2.7.3 SSD :

(Single Shot MultiBox Detector): SSD is another real-time object detection model that predicts bounding boxes at multiple scales. It balances speed and accuracy.

2.8 Conclusion

By exploiting models such as VGG, MobileNet, and ResNet, performance can be significantly improved on different devices, including mobile devices. These technologies represent a strong foundation for building powerful and effective artificial intelligence systems in many practical applications.

Chapter 3

Design and implementation

In this chapter, we discuss the design and implementation of a face recognition system using deep learning techniques.

3.1 Software environment

3.1.1 Python

Python programming language is a high-level programming language developed by Guido van Rossum in the late 1990s. Python is known for its ease of understanding and use, making it suitable for a wide range of software applications. It is used in various fields including software development, data science, scientific computing, web development, and artificial intelligence. Python features a clean and organized syntax, with comprehensive support for libraries and tools, making it a preferred choice for developers in diverse fields.[26]

Figure 3.1: Python Logo

Python's support for modules and packages promotes program modularity and code reuse. The Python interpreter and its substantial standard library are freely distributable and accessible in source or binary form for all major platforms.

3.1.2 Computing Resources

Google Colaboratory (Colab) is a powerful and free web-based platform for running and developing machine learning models. It has aided many researchers and developers in accomplishing their projects due to a range of features and resources it offers.[1]

here is why we used COLAB

1- Access to Computing Resources:The ability to access CPUs, GPUs, and TPUs via the cloud provided by Google allows deep learning models to be trained quickly and efficiently

2- Sharing and Collaboration: the ability of sharing notebooks with others and collaborate on projects in real-time, enabling effective teamwork

3- Programming Libraries: : Google Colab comes with popular Python libraries pre-installed, including TensorFlow, PyTorch, Keras, and others, making it easy to use them in deep learning model development.

3.1.3 Keras

Keras works by providing a user-friendly interface for building and training neural networks. It allows users to define models using high-

level abstractions called layers, which are stacked together to form a network. Users can then compile the model with optimization algorithms and loss functions, and train it on labeled data using a variety of training techniques. Keras also supports multiple backend engines , which handle the low-level operations necessary for training and inference. This modular and flexible design makes it easy to experiment with different architectures and adapt models to various tasks.[20]

Figure 3.2: keras logo

3.1.4 Numpy

At its core, NumPy introduces a new data structure called ndarray (n-dimensional array), which allows for efficient storage and manipulation of numerical data. These arrays can be created using various functions provided by NumPy or by converting existing Python lists.[13] NumPy also offers a wide range of mathematical functions for array manipulation, including element-wise operations, linear algebra routines, Fourier transforms, and random number generation. These functions are implemented in highly optimized C and Fortran code, making them fast and efficient even for large datasets.[13]

Figure 3.3: Numpy's logo

3.1.5 Tensorflow

It offers a vast, adaptable ecosystem of tools, libraries, and community resources that enables academics to advance the field's state-of-the-art and developers to quickly create and deploy ML-powered applications.[6]

Figure 3.4: Tensorflow Logo

3.1.6 Opencv

Being a BSD-licensed product, OpenCV makes it simple for businesses to use and modify the code. OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. It was created to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.[17]

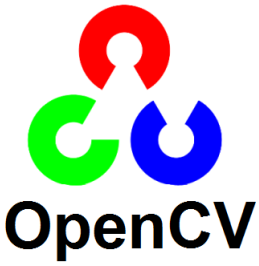

Figure 3.5: OpenCV Logo

3.2 Design

The face recognition system involves identifying a person from a digital image or a video frame. Convolutional Neural Networks (CNNs) form the backbone of the face recognition system. It learn to extract relevant features from facial images. These features are then used to

classify faces by comparing them to known faces.

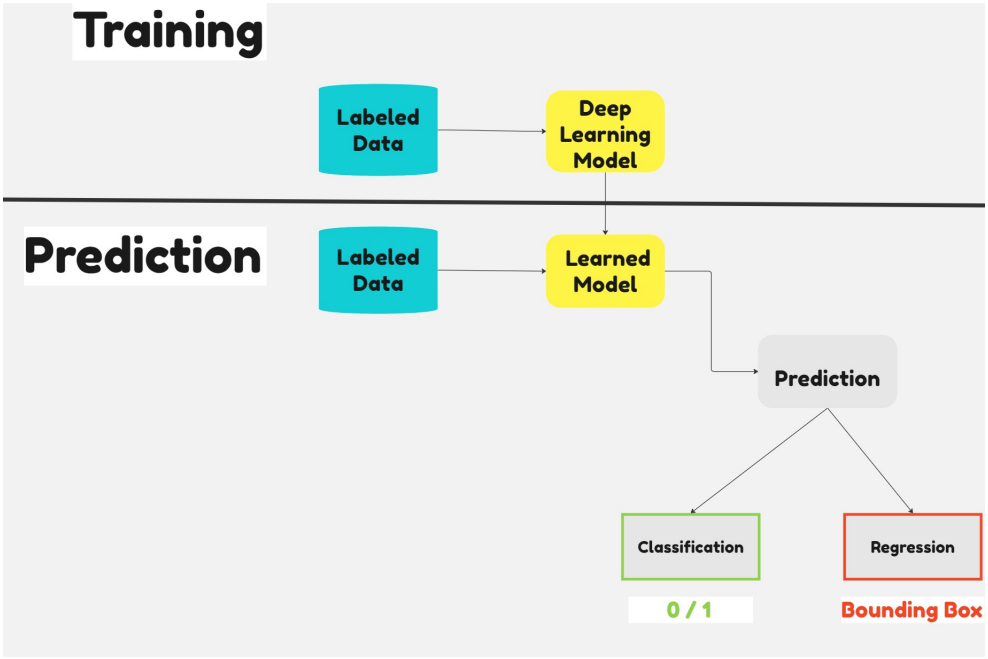

Figure 3.6: Training and Prediction

Using this process, we can develop deep models that leverage Labeled data effectively, allowing us to make accurate and efficient predictions for a variety of problems and challenges.

As illustrated in figure 3.7 the face recognition system locate faces in the acquired image and extract features using a pretrained CNN then compare the extracted features with known faces.

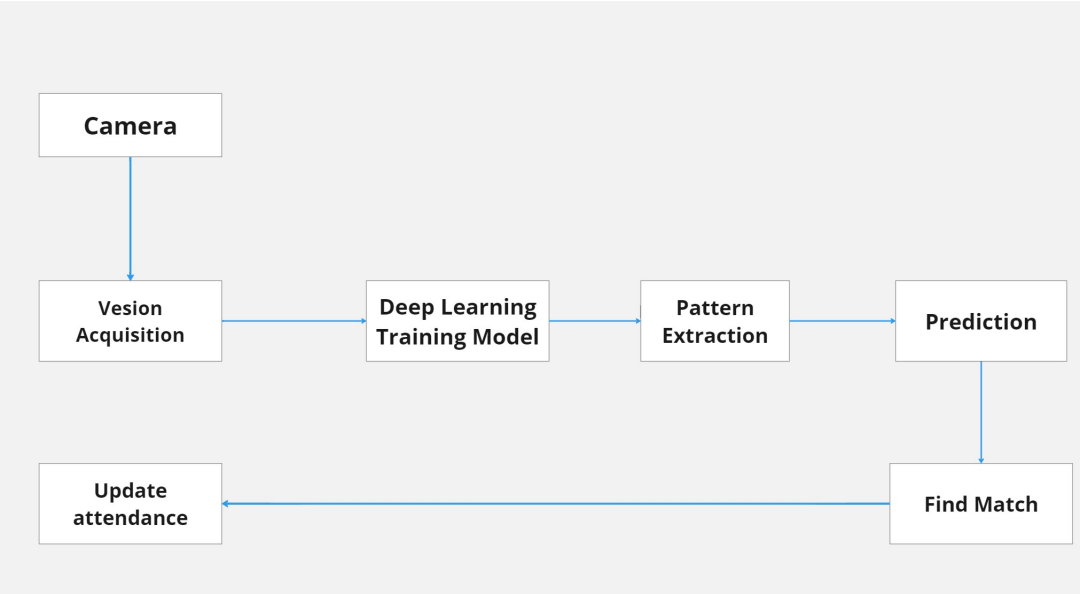

Figure 3.7: Face Recognition Pipeline

3.3 Implementation

3.3.1 Data collection

The attendance face recognition system aims to accurately identify students. To achieve this, we first collect data consisting of images of students that will be recognized, along with images of common faces. All data is labeled before the training process begins.

3.3.2 Labelme

LabelMe typically operates through a web-based user interface.

```
E:\>labelme
2024-05-06 19:59:50,068[ INFO] __init__:get_config:67-Loading config
                                    file from : C\Users\LAPTA\.
                                   labelmerc
```

The total number of labeled images is 90. By effectively using LabelMe, we were able to build an improved dataset to diversify our

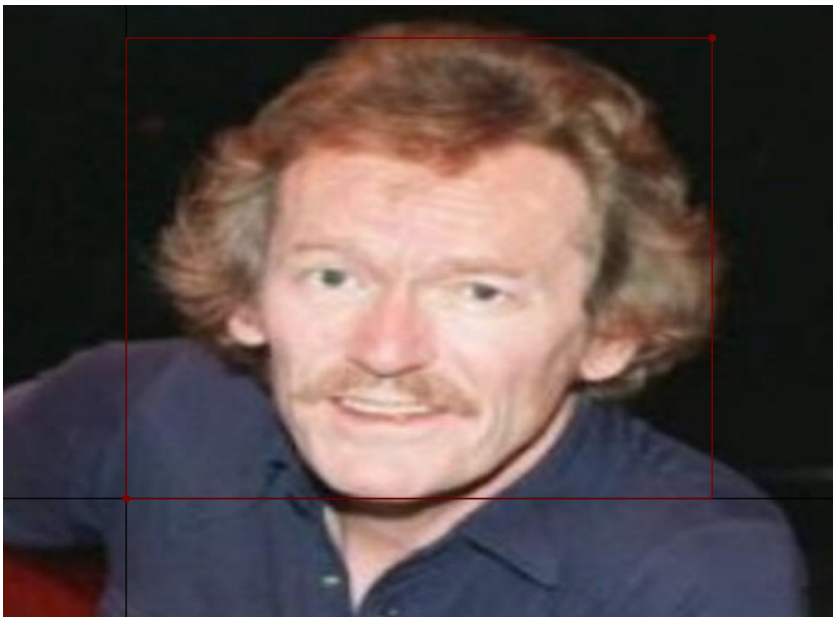

Figure 3.8

deep learning model, improving its performance and accuracy.

3.3.3 Data Augmentation

It is a process in which we modify the original data in certain ways to create new copies of it. Data augmentation techniques include a variety of transformations that can be applied to the original data.[7] In the following, we detail the transformations utilized in data augmentation to enhance the diversity and quality of the collected data.

Random Crop: A random crop is performed with a specified width and height (in this case, 320x320 pixels),This operation helps introduce spatial invariance by randomly selecting a region of interest from the original image.

Horizontal Flip: The image is horizontally flipped with a probability of 50This transformation helps diversify the dataset by

creating mirror images.

Random Brightness and Contrast: Random brightness and contrast adjustments are applied with a probability of 20Brightness and contrast variations enhance the model's robustness to different lighting conditions.

Random Gamma Correction:Gamma correction is applied with a probability of 20Gamma correction adjusts the pixel intensity values, which can be useful for handling non-linear lighting variations.

RGB Shift: A random shift in the red, green, and blue channels is performed with a probability of 20This helps simulate color variations and improves the model's ability to generalize.

Vertical Flip: The image is vertically flipped with a probability of 50Similar to horizontal flip, this transformation introduces additional variations.

Bounding Box Parameters: The bbox-params specify how bounding boxes (if present) should be handled during augmentation. The format is set to 'albumentations', and the label fields are specified as 'class-labels'.

Data augmentation techniques are used to increase the diversity of data available for training, which helps reduce the difference between the training set and the actual set to be processed. This reduces the model's skewness and enhances its ability to generalize to new data that it has not previously processed.

```
import albumentations as alb
```

```
augmentor = alb.Compose([alb.RandomCrop(with=320,height=320),
   alb.HorizontaFlip(p=0.5),
   alb.RandomBrightnessContrast(p=0.2),
   alb.RandomGamma(p=0.2),
   alb.VerticalFlip(p=0.5)],
bbox_params=alb.BoxParams
(format='albumentation',label_fields=['class_labe]
```

Code Listing 3.1: Application of Augmentation process

```
for partition in ['train',test',val]:
 for image in os.listdir(os.path.join('/content/drive/MyDrive/data',
                                      partition, 'images' )):
    img= cv2.imread(os.path.join ('/content/drive/MyDrive/data',
                                        partition, 'images' , image))

    coords =[ 0,0,0.00001,0.00001]
    label_path = os.path.join('/content/drive/MyDrive/data', partition,
                                        'labels' f'{ image.split(".")[0]
                                        }.json' )):
   if os.path.exists(label_path):
        with open(label_path, 'r') as f:
              label = json.load(f)
        coords[0] =label[ 'shapes'][0]['points'][0][0]
         coords[1] =label[ 'shapes'][0]['points'][0][1]
          coords[2] =label[ 'shapes'][0]['points'][1][0]
         coords[3] =label[ 'shapes'][0]['points'][1][1]
         coords = list(np.divide(coords,[640,400,640,480]))
    try:
     for x in range(60) :
        augmented=augmentor(image=img,bboxes=[ coords], class_labels=['
                                            face'])
        cv2.imwrite(os.path.join('/content/drive/MyDrive/data',
                                            partition, 'images' , image)
                                            )
         annotation={}
        annotation['image'] = image
         if os.path.exists(label_path):
           if len (augmented['bboxes'])==0;
             annotation['bbox'] = [0,0,0,0]
              annotation['class'] = 0

              else:
                annotation['bbox'] = augmented['bboxes'][0]
                    annotation['class'] = 1
               else:
                 annotation['bbox'] = [0,0,0,0]
```

```
                annotation['class'] = 0
            with open(os,pathh.join('/content/drive/MyDrive/aug_data'
             parttion ,'labels',f'{ image.split(".")[0]}.{x}.json'),json
                                                    .dump(annotation,f)

    except Exception as e:
        print(e)
```

For each image, it is read using OpenCV (cv2.imread) and then the corresponding label data is loaded from the JSON file. Augmentation is applied to the image using a function that makes changes such as rotation, reflection, zoom in and out.After that, the optimized image is saved and the corresponding label data is updated for the changes made.This process is repeated 60 times,Finally, the updated label data is saved in new JSON files, which have different names for each image.

3.3.4 Dataset splitting

After augmentation, we obtained a total of 5220 images, which were divided into three subsets: the training dataset, the validation dataset, and the test dataset. Specifically, 3600 images (70% of the data) were allocated to the training set, while 780 images (14%) were allocated to the validation set. Additionally, 840 images (16%) were designated for testing. The following table illustrates the distribution of data between these sets.

	Total	Percentage
Test	840	16%
Train	3600	70%
Val	780	14%
Total	5220	100%

3.3.5 Build the model

Code Listing 3.2: Import necessary packages

```
import os
import time
import uuid
import cv2
import tensorflow as tf
import json
import numpy as np
from matplotlip import pyplot as pl
import albumentations as alb
from tensorflow.keras.models import Model
from tensorflow.keras.models import Input,Conv2D,Denes,
                                    GlobalMaxPooling2D
from tensorflow.keras.applications import VGG16
from tensorflow.keras.models import laod_model
```

First import the necessary packages required for the code Listing 3.4.

Code Listing 3.3: Installing the drive for Google Colab

```
from google.colab import drive
drive.mount('/content/drive')
```

The dataset is saved in a Google Drive account, we link Google Drive storage to Google Colab environment.3.3.

3.3.6 CNN Layers

Convolutional Neural Network (CNN) layers are the building blocks of convolutional neural networks, a type of deep learning model used for tasks involving image recognition, object detection, and classification.

```
def build_model():
input_layer = Input(shape=(120,120,3))
mobNet = mobileNettv2(include_toop=False, wights= 'imagenet',input_shape
                                    =(120,120,3))(input_layer)
pool = MaxPooling2D(pool_size=(2,2), strides=1,padding='same')(vgg)
avrg = AveragePooling2D(pool_size=(2,2),strides=1,padding='same'(vgg)
conc=conactenat([pool,avrg,] axis = -1)
drop = Dropout(0.5)(conc)
#classification Model
f1 = GlobalMaxPooling2D()(drop)
class1 = Dense(2048,activation='relu')(f1)
class2 = Denese(1, activation='sigmoid')(class1)
#Bounding box model
f2=GlobalMaxPooling2D()(mobNet)
regress1 = Denes(2048,activation='relu')(f2)
regress2 = Denes(4,activation='sigmoid')(regress1)

facetracker = Model(inputs=input_layer, outputs=[class2, regress2])
return facetracker
```

We start by defining the build_model() function. We extracted features from the input image using MobileNetV2 and employed Max Pooling and Average Pooling to reduce the spatial dimensions of the extracted features. Additionally, we used Dropout to reduce the redundant correlation between layers and to prevent overfitting.

3.4 The results

Classification

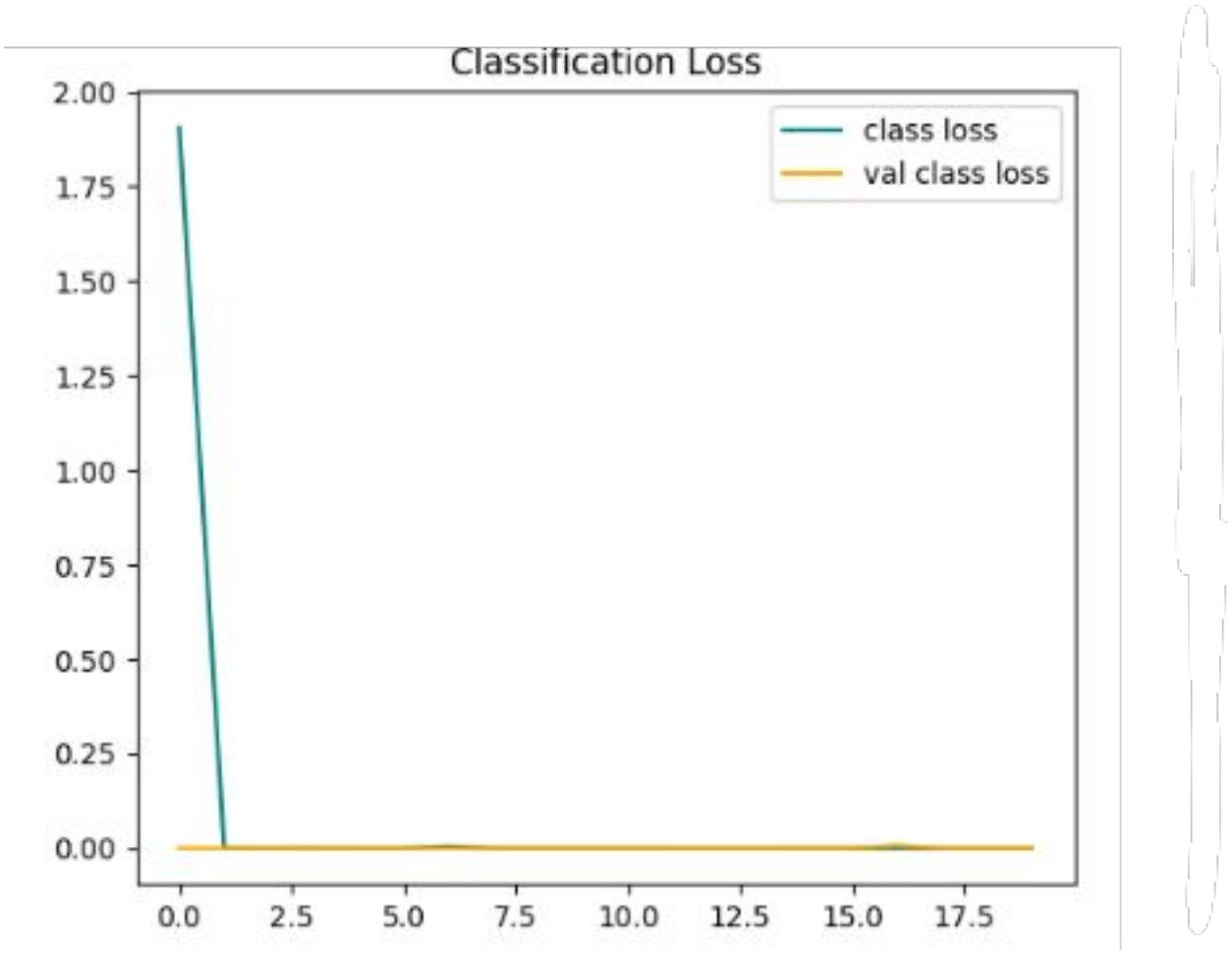

Figure 3.9: Statement of the classification loss function

Regression

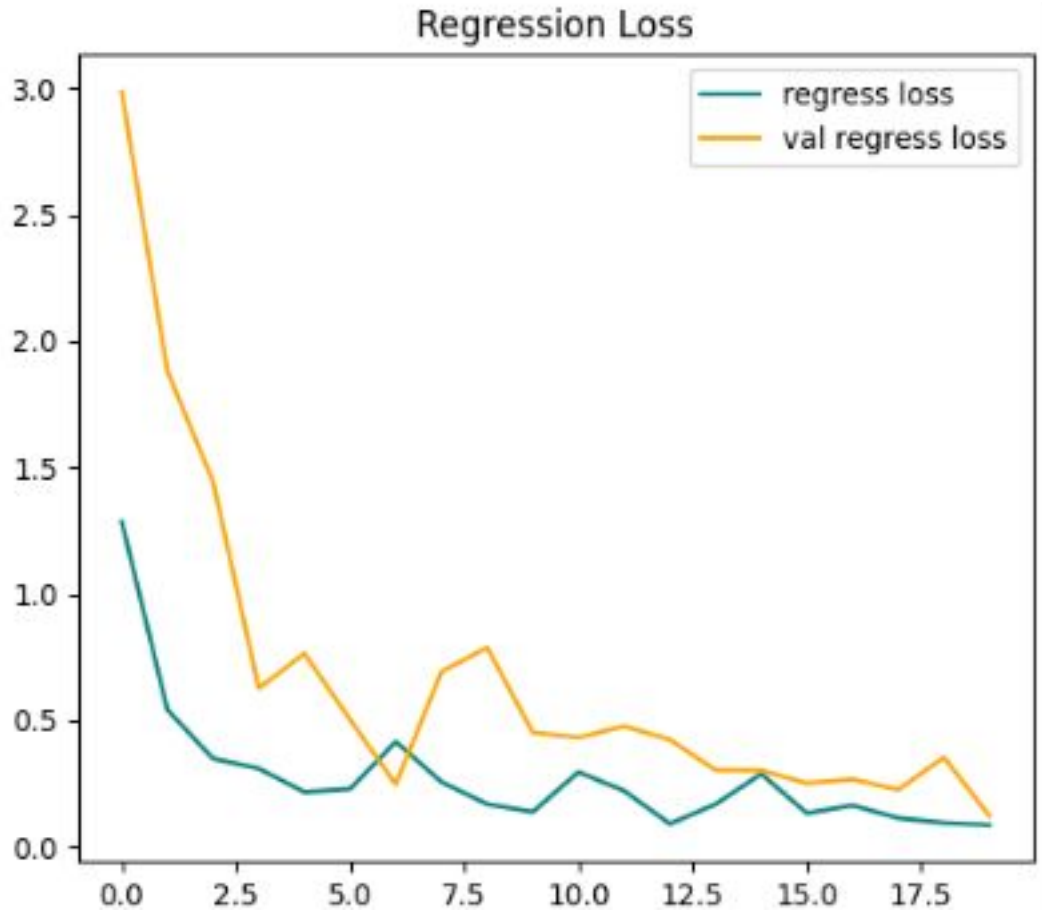

Figure 3.10: Statement of the regression loss function

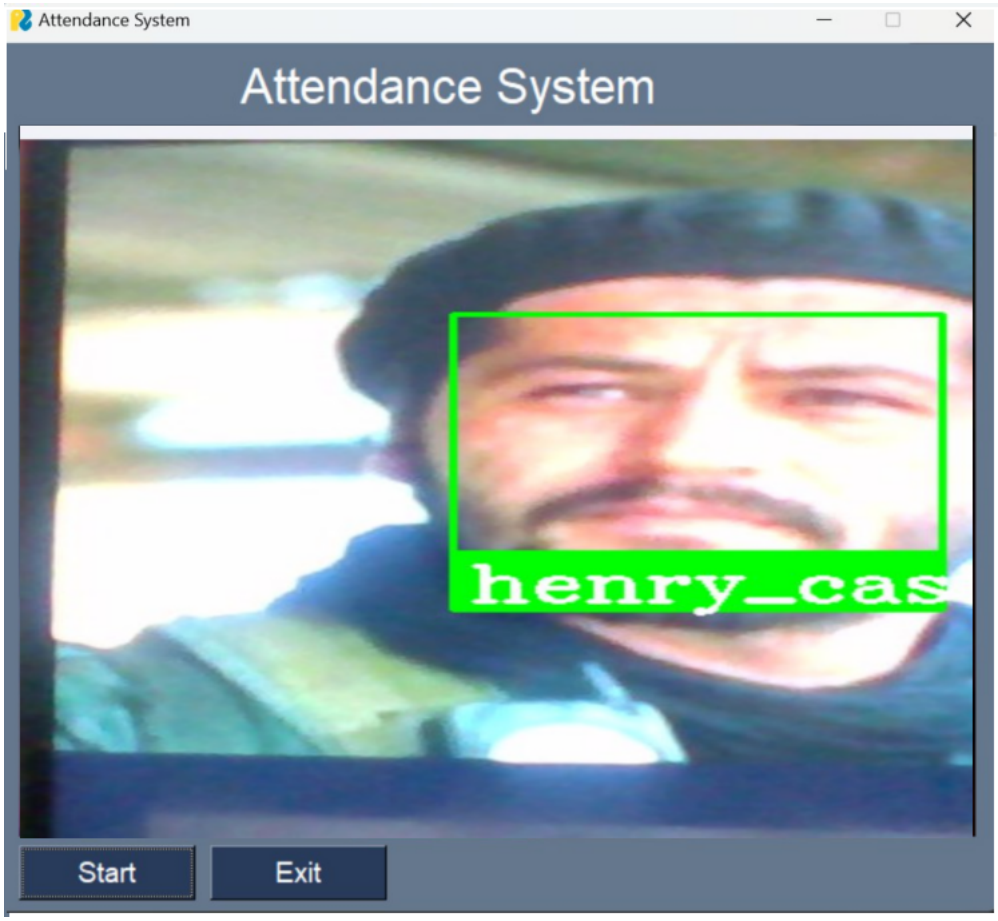

Figure 3.11: The system recognized the face and identified it with a green frame with the person's name displayed

When a person's face is registered, it is compared to faces stored in the database. If the person is identified, their presence is recorded.

3.5 Conclusion

In this chapter, we discussed the application side and described the tools that were used to execute the suggested attendance system.

Conclusion

The implemented system exhibits strong face recognition capabilities, which are essential for an efficient and accurate attendance management solution. Performance metrics verify the model's accuracy and generalization capabilities, confirming its suitability for real-world applications. There are some additions that could further improve the system and make it more streamlined and accurate. Implementing customizable alert systems for anomalies such as detection of unknown faces, low attendance rates, or other pre-determined conditions. Also, there is potential in the future to enhance our model by Upgrading to more advanced and efficient modeling architectures such as ResNet, FaceNet or MobileFaceNets. Integrating attention mechanisms to better deal with different lighting conditions and occlusions.

Bibliography

[1] "A classification and review of tools for developing and interacting with machine learning systems". In: 2022, pp. 1092–1101.

[2] Antreas Antoniou, Amos Storkey, and Harrison Edwards. "Augmenting image classifiers using data augmentation generative adversarial networks". In: 2017.

[3] Chaity Banerjee, Tathagata Mukherjee, and Eduardo Pasiliao. "Feature representations using the reflected rectified linear unit (RReLU) activation". In: (2020).

[4] Han Cai et al. "Enable deep learning on mobile devices: Methods, systems, and applications". In: ().

[5] Dulal Chakraborty, Sanjit Kumar Saha, and Md Al-Amin Bhuiyan. "Face recognition using eigenvector and principle component analysis". In: *International Journal of Computer Applications* ().

[6] Rory Conlin et al. "Keras2c: A library for converting Keras neural networks to real-time compatible C". In: (2021).

[7] Ekin D Cubuk et al. "Autoaugment: Learning augmentation strategies from data". In: 2019.

[8] Hossein Gholamalinezhad and Hossein Khosravi. “Pooling methods in deep neural networks, a review”. In: (2020).

[9] Daniel Gibert, Carles Mateu, and Jordi Planes. “The rise of machine learning for detection and classification of malware: Research developments, trends and challenges”. In: (2020), p. 102526.

[10] Yoshua Bengio Goodfellow and Aaron Courville. *Deep Learning*. 2016.

[11] Yoshua Bengio Goodfellow and Aaron Courville. *Deep Learning*. MIT press, 2016.

[12] Gaurav Goswami, Mayank Vatsa, and Richa Singh. “RGB-D face recognition with texture and attribute features”. In: (2014).

[13] Pramod Gupta and Anupam Bagchi. “Introduction to NumPy”. In: 2024.

[14] Mahmoud Hassaballah et al. “Facial features detection and localization”. In: (2019), pp. 33–59.

[15] Tal Hassner et al. “Pooling faces: Template based face recognition with pooled face images”. In.

[16] Geoffrey E Hinton et al. *How neural networks learn from experience*. 1992.

[17] Joseph Howse. “The opencv library.” In: 27 (2013).

[18] Shou-Ching Hsiao et al. “Malware image classification using one-shot learning with siamese networks”. In: *Procedia Computer Science* (2019).

[19] Jibble. *What are the Different Types of Attendance Systems?* 2024. URL: %5Curl%7Bhttps://www.jibble.io/article/types-of-attendance-systems%7D (visited on 04/03/2024).

[20] "Keras2c: A library for converting Keras neural networks to real-time compatible C". In: ().

[21] Asifullah Khan et al. "A survey of the recent architectures of deep convolutional neural networks". In: *Artificial intelligence review* (2020).

[22] Kaidong Li et al. "Object detection with convolutional neural networks". In: 2020, pp. 41–62.

[23] Yijun Li et al. "Deep joint image filtering". In: 2014, pp. 1629–1640.

[24] Giorgio Patrini et al. "Making deep neural networks robust to label noise: A loss correction approach". In.

[25] Joseph Redmon et al. "You only look once: Unified, real-time object detection". In: 2016, pp. 779–788.

[26] Guido van Rossum and Fred L Drake. *An introduction to Python*. Network Theory Limited, 2006.

[27] Ali Sharifara, Mohd Shafry Mohd Rahim, and Yasaman Anisi. "A general review of human face detection including a study of neural networks and Haar feature-based cascade classifier in face detection". In: 2014.

[28] J Sujanaa, S Palanivel, and M Balasubramanian. "Emotion recognition using support vector machine and one-dimensional convolutional neural network". In: (2021).

[29] Linnan Wang et al. “Accelerating deep neural network training with inconsistent stochastic gradient descent”. In: (2017), pp. 219–229.

[30] Guihua Wen et al. “Ensemble of deep neural networks with probability-based fusion for facial expression recognition”. In: (2017).

[31] Chenghua Xu et al. “Automatic 3D face recognition from depth and intensity Gabor features”. In: (2009), pp. 1895–1905.

[32] Bo Yang and Songcan Chen. “A comparative study on local binary pattern (LBP) based face recognition: LBP histogram versus LBP image”. In: *Neurocomputing* ().

[33] Ye Yuan et al. “In defense of the triplet loss again: Learning robust person re-identification with fast approximated triplet loss and label distillation”. In: 2020, pp. 354–355.