الجمهورية الجزائرية الديمقراطية الشعبية
People's Democratic Republic of Algeria

وزارة التعليم السعالي و البحث العلمي
Ministry of Higher Education and Scientific Research

جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie

Master's thesis

In Telecommunication

Option: Telecommunication Systems

Presented By :

SOUILAMAS Amir

&

YAHIAOUI Raiane

Comparative study of deep neural networks architectures, using CASIA and UbirisV2 datasets for iris recognition

Supervised by: Mme. BOUGHERIRA HAMIDA

Promotion 2023/2024

Acknowledgements

We thank Allah for giving us the patience and the courage to accomplish this project.

We thank Miss Bougherira Hamida for her supervising and her precious advices and guidance throughout this project that made the best come out of us in order to accomplish this thesis.

We thank the members of the jury that accepted to evaluate our work.

We thank our teachers that helped us through our whole academic years for their teachings and efforts in order to get us to this final stage.

We thank all members of the department of electronics that contributed to the accomplishment of our education.

And finally, we would like to thank any person who helped in the making of this project with any small or big information that they provided.

Dedications

To my parents, no words can express how much your support and love means to me, I am forever proud to be your son.

To my brother Nadji and his wife Khadidja, your help and advices in this project and in general were a real boost to me, I am grateful to have both of you by my side.

To my aunt Safia, your emotional support was always present with me before and through this project, may God grant you all what u want.

To my grandmother, your joy and presence were a source of unmeasurable happiness, may you rest in peace.

To my family, who was always a big support in my projects and pushed me to always go further, I am lucky to have you all.

To my friends Imen and Fares, you have been a great support and help in this project and a good ear that listened in both good and bad times, I am happy to have both of you by my side.

Amir

Dedications

To my mom and dad, thank you for your unwavering support, both emotionally and financially, which has been the cornerstone of my journey. Your wisdom has always guided me, and in my moments of exhaustion, you have been my refuge. Your love and care have shaped who I am today. Thank you for believing in me, for your endless encouragement, and for being my rock. I owe you this achievement.

My brothers, a constant source of strength for me. Your presence and encouragement have given me the confidence to achieve my dreams. Thank you for always being there for me, lifting me up, and sharing this journey with me.

Swab, my best friend, thank you for being a guiding light in my life. Your unwavering support and loyalty have been my strength, filling my life with positivity and warmth.

To Samaya, Mrs Samaya thank you for your guidance and encouragement. You saw my potential when I doubted myself and gave me the confidence to believe in my abilities. You were more than just a teacher; You have been a mentor and source of endless motivation.

With all my love and gratitude

Raiane

Abstract

الملخص

الهدف من هذا المشروع هو إجراء دراسة مقارنة بين معماريتين للشبكات العصبية التلافيفية العميقة التي تقوم بالتعرف على قزحية العين (ResNet50 و MobileNetV2).عن طريق نقل التعلم لتقليل وقت التدريب وتكلفة موارد الكمبيوتر، نستخدم Casia-Iris-Tousand. تم تطبيق مجموعة متنوعة من المعالجة المسبقة على قاعدتي بيانات ImageNetإضافة أوزان بنية في تحديد الأنماط المهمة في القزحية: نقاط المفاتيح. ثم قمنا بتحديد أي مجموعة من SIFT. ساعدت خوارزمية UbirisV2و أفضل دقة. أظهر UbirisV2 مع ResNet50البنية ومجموعة البيانات أعطت أفضل النتائج للتعرف على قزحية العين. أعطى ، رسومًا SIFT الذي تمت معالجته مسبقًا باستخدام خوارزمية Casia-Iris-Tousand مع ResNet50النموذج المعتمد على بإمكانية تنفيذه في الأنظمة المدمجة وفي تطبيقات الهاتف MobileNetV2بيانية مع أنماط مكتشفة مركزة على القزحية. يتمتع المحمول

Résumé

Le but de ce projet est de faire une etude comparative entre deux reseaux de neurones convolutifs profonds qui effectuent la reconnaissance par l'iris (ResNet50 et MobileNetV2). Pour diminuer le temp d'entrainement et le cout en ressources informatiques, on utilise le transfer learning en ajoutant les poids de l'architecture ImageNet. On a appliqué une variété de pre-traitement, sur les bases de données Casia-Iris-Thousand et UbirisV2. L'algorithme SIFT a aidé a localiser d'importants motifs dans l'iris : les keypoints. On a ensuite dererminer quelle combinaison d'architecture et de dataset donnait les meilleurs résultats pour la reconnaissance par l'iris. ResNet50 avec UbirisV2 a donné la meilleure precision. Le model basé sur ResNet50 avec Casia-Iris-Thousand prétraité avec l'algorithme SIFT, a montré des histograms avec des motifs detecté concentré sur l'iris. MobileNetV2 a la possibilité d'etre implementé dans des systemes embarqué et dans des applications mobiles.

Abstract

This project's goal is to make a comparative study between two deep convolutional neural networks architectures that perform iris recognition (ResNet50 and MobileNetV2). To lower computational training time and cost, we used transfer learning by adding the ImageNet architecture's weights. We applied various pre-processing, on the databases Casia-Iris-Thousand and UbirisV2. The SIFT pre-processing algorithm helped locate important patterns in the iris: the key points. We then determined which combination of architecture model and dataset was best for iris recognition. Resnet50 with UbirisV2 gave the best accuracy. ResNet50 based model with Casia-Iris-Thousand pre-processed with SIFT, showed histograms with detected patterns concentrated in the iris region. MobileNetV2 has the possibility to be implemented in embedded systems and mobile applicatons.

.

Table of Content

List of Tables

List of Figures

List of Figures

List of Figures

List of Abbreviations

IR	Infra-Red
AI	Artificial Intelligence
CNN	Convolutional Neural Networks
DCNN	Deep Convolutional Neural Networks
SIFT	Scale-Invariant Feature Transform
ResNet	Residual Networks
NIR	Near Infra-Red
Float32	32-bit floating-point numbers
CSV	Comma Separated Values
ReLU	Rectified Linear Unit
ADAM	Adaptive Moment Estimation
XML	Extensible Markup Language
RGB	Red Green Blue
TANH	Hyperbolic Tangeant

General Introduction

Biometric identification is a technology that uses unique traits of a person to confirm their identity [1], there are many ways to perform biometrics, iris recognition, a process to recognize a person with unique patterns in their iris [2], fingerprint and also facial recognition traits are also methods to perform biometric identification [3].

Previously we used to perform iris recognition with methods like Daugman's method that consist of extracting features from segmented irises, but this method has major issues like generalizing complex shapes and poor noise sensitivity which make it hard to perform to use in real life situations [4].

Nowadays with AI and robust deep learning architectures, we overcame these issues, and the databases available today helps the architectures adapt to both clear and noisy images, and with the right pre-processing it becomes even better to locate important regions of the iris, for optimizing the learning process we use transfer learning to combine a pre-trained model and a new model [5], with all this we have the formula to create an optimized model for iris recognition, however these techniques are commonly used today in this field with various datasets and deep learning architectures so how to know which is the best architecture to use and which dataset is most suited for it, this lead to the making of this comparative study to try to find an answer to these questions with the following plan:

In chapter 1 we present a state of the art that contains existing methods for human iris recognition, the classical and AI based method, from there, We the introduce convolutional neural networks (CNN's) and explain the functioning of deep neural network along with some examples of deep convolutional neural networks architectures.

In chapter 2 design our iris recognition models mentioning the pretrained architectures we worked with (ResNet50 and MobileNetV2) implemented using transfer learning, the classification layers implemented, the datasets (Casia-Iris-Thousand, UbirisV2), the pre-processing of the images, the various training parameters we used for optimizing the performances (Batch size, learning rate), and the optimizer we used to compile our model are also shown.

In chapter 3 we will implement the iris recognition models designed previously and provide the results of every step taken in our code, along with the performances obtained after training the models, and we will have three comparisons to make at the end of this chapter and do the necessary discussions about the results of the models and the results of the comparisons.

Chapter 1

State of the art

1. State of the art

1.1. Introduction

The human eye has always been known to have unique features and in today's time thanks to technological advancements we know that the iris contains even more unique features than fingerprints

In this chapter we will speak about the iris and why is it unique to every individual, the techniques of extraction of these features, and the methods used for recognition classical & AI based.

1.2. Potential of iris recognition

Iris recognition has become more accessible due to late technological advances like new cameras, image processing, and the recognition itself, in addition the iris has been preferable than face or fingerprint recognition due to many aspects some of which are mentioned in (Table 1.1) [6]

Benefit	Iris recognition	Facial recognition	Fingerprint
Accuracy	*Excellent*	*Moderate*	*High*
Speed	*Excellent*	*Good*	*Good*
Stability	*Excellent*	*Low*	*Moderate*
Works with gloves	*Yes*	*Yes*	*No*
Works with masks	*Yes*	*No**	*Yes*
Works with glasses/goggles	*Yes*	*No*	*Yes*
Privacy	*Yes*	*No*	*Yes*
Contactless	*Yes*	*Yes*	*No*

Table 1.2:Comparison between different kinds of recognition

Ease of Use: Iris recognition is very convenient as the individual simply needs to look into a camera for a few seconds.

The process captures a video image that is non-invasive and inherently safe. Iris recognition scored far higher than fingerprints in terms of ease of use, speed, and overall user preference in field tests.

Inclusiveness: Iris recognition is more inclusive than fingerprints, with fewer people unable to provide scans at acceptable quality. In one study, over 13% of refugees over age four were unable to provide 4 good fingerprints, but only 2% were unable to provide a high-quality iris. Iris recognition remained almost stable at around 1% failure to capture rate, confirming it is both very secure and inclusive.

Stability and consistency: The iris remain stable and unchanged throughout a person's life. There are no changes to the physical characteristics of the iris even when the person ages. Fingerprint patterns, on the other hand, can change over time due to factors like manual labor, injuries, or aging.

1.3. The human iris

The iris is a part of the human eye that plays a crucial role in vision. And is a flat ring-shaped membrane behind the cornea of the eye it has an adjustable circular opening in the center called the pupil.

The iris is also the pigmented part that expresses eye color. expresses itself during the presence of melanin pigment the more melanin there is the darker the eye color will be [2].

1.3.1. Characteristics of the iris [7]:

The characteristics of the iris that's playing crucial roles in both vision and ocular protection, (Figure 1-1) it detailed anatomical diagram of the human eye.

Color: Iris color is determined by the amount and type of melanin pigment in the iris. Eye color can range from very light blue to very dark brown.

Patterns: The iris has a unique pattern of lines, dots, crypts, furrows, and other structures that are distinct to each individual. These patterns form during fatal development.

Pupillary reflex: The iris controls the size of the pupil, which constricts and dilates to regulate the amount of light entering the eye.

Muscles: The iris contains two sets of smooth muscles that control pupil size. The sphincter pupillae constricts the pupil in response to light, while the dilator pupillae dilates the pupil in low light.

Blood vessels: The iris is highly vascularized and contains many blood vessels that supply oxygen and nutrients to the iris muscles and tissues. These vessels are visible in some people, especially those with lighter colored irises.

Size: Iris size can vary between individuals, but the average diameter is 12 mm. The size of the pupil opening ranges from 2-8 mm depending on lighting conditions.

Thickness: The stroma of the iris, composed of connective tissue and blood vessels, is about 0.5 mm thick. The anterior epithelium is just two cells thick.

Attachment: The iris is attached to the ciliary body and suspensory ligament of the lens, which hold it in place. The outer edge of the iris is continuous with the choroid.

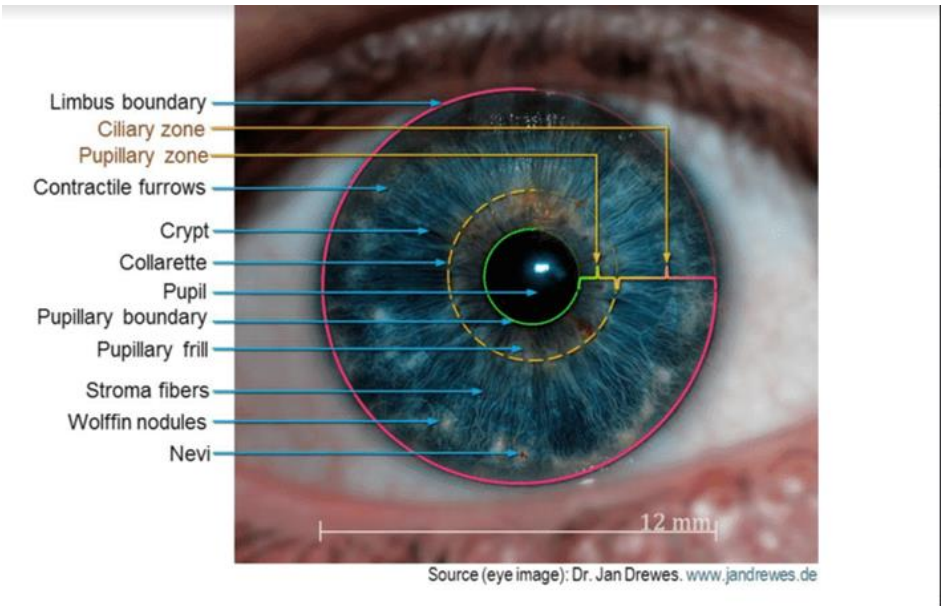


Figure 1-2:Eye characteristics [8]

1.4. Overview of iris recognition technology

Iris recognition just as any other recognition has its own rules, advantages and limitations that are going to be listed here

1.4.1. Iris recognition

Iris recognition is an automated method of biometric identification that uses mathematical pattern-recognition techniques on video images of one or both of the irises of an individual's eyes, whose complex random patterns are unique, stable, and can be seen from some distance.

Iris recognition systems leverage the distinctive features present in the colored tissue of the eye to achieve extremely reliable personal identification. With rapid verification and robust matching capabilities, iris scanning biometrics rival fingerprinting for accuracy while using an internal bodily feature almost impossible to counterfeit [9].

1.4.2. Recognition Rate Criteria

Recognition rate criteria are used in neural networks. These metrics help to determine how well a neural network can perform tasks related to pattern recognition. The evaluation of the ability of a network to recognize different classes or patterns is important through this criterion, which plays an essential role in demonstrating the capabilities of the network [10].

1.4.2.1. Error rate:

The error rate in recognition refers to the quantity of incorrect matches or misclassifications made by a system of iris recognition [10].

It is an imperative metric used to test the performances of iris recognition systems, especially when it comes to security applications [10].

$$ER = \frac{|Approximate - Exact|}{Exact}$$

1.4.2.2. True positives

These are the correctly identified iris patterns. They are the cases where the system correctly matches the iris image with the stored template [10].

$$TPR = \frac{TP}{Actual\ Positive} = \frac{TP}{TP + FN}$$

1.4.2.3. False positives

These are the incorrectly identified iris patterns. They are the cases where the system falsely matches an iris image with the stored template, this results in a false match [10].

$$FPR = \frac{FP}{Actual\ Negative} = \frac{FP}{TN + FP}$$

1.4.2.4. False negatives

These are the incorrectly rejected iris patterns. They are the cases where the system incorrectly identifies a that an image does not match any stored template [10].

$$FNR = \frac{FN}{Actual\ Negative} = \frac{FN}{TP + FN}$$

1.4.2.5. True negatives

These are the correctly denied iris patterns. They are the cases where the system correctly identifies that an iris image does not match any stored data [11].

$$TNR = \frac{TN}{Actual\ Negative} = \frac{TN}{TN + FP}$$

1.4.3. Advantages and limitations of iris recognitions

Iris recognition just like other kinds of recognitions has advantages but also limitations of its own and they will be presented here.

1.4.3.1. Advantages of iris recognition [9]

High accuracy: Iris recognition has extremely low recognition error rate.

Stable over time: The detailed iris structure remains unchanged through adult life, enabling strong matches against enrolment scans decades later.

Non-invasive process: Iris scans can be acquired easily at a short distance without any contact with the body.

Hard to spoof: It requires high resolution cameras focused properly on the eye to fake an iris biometric trait.

1.4.3.2. Limitations of iris recognitions [12]

Unique Biometric Characteristic: Iris recognition is limited by the requirement that the iris pattern must be unique for each individual. If this uniqueness is compromised due to factors like ocular diseases or injuries, the accuracy of iris recognition can be affected.

Environmental Factors: The effectiveness of iris recognition can be hindered by environmental factors such as lighting conditions, occlusions, or reflections that may distort the iris image. These factors can impact the quality of the captured iris image, leading to potential recognition errors.

Invasive Nature: While iris recognition is non-intrusive compared to other biometric methods like fingerprints, it still requires proximity for image capture. This can be considered invasive in certain scenarios, especially in terms of privacy concerns or user acceptance.

Cost and Complexity: Implementing iris recognition systems can be costly due to the need for specialized hardware and software. The complexity of iris recognition technology may also pose challenges in terms of system integration and maintenance.

Security Concerns: Despite being a highly accurate biometric modality, iris recognition systems are not immune to security threats such as spoofing attacks where fake iris images are used to deceive the system. Continuous advancements in anti-spoofing techniques are essential to mitigate such risks.

1.5. Fundamentals of iris recognition

1.5.1. Iris Recognition pipeline

Iris recognition involves a series of steps (Figure1-3) from image capture to feature encoding and matching to accurately identify people based on the unique pattern of the iris.

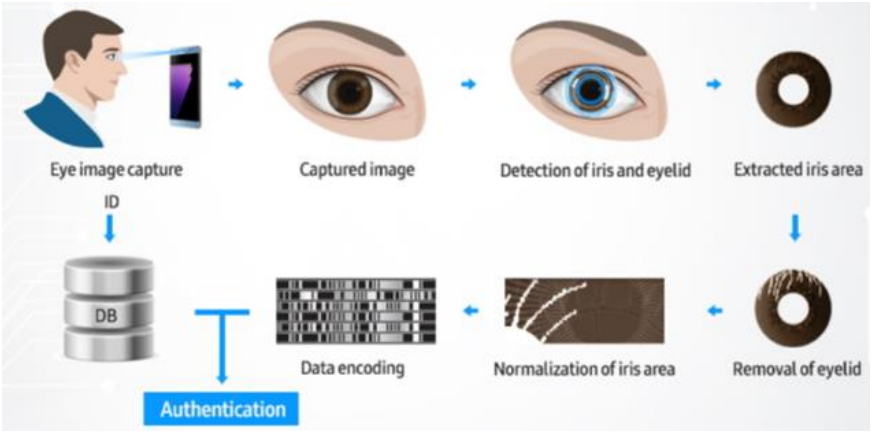


Figure1-4:Iris recognition pipeline [13]

1.5.1.1. Image acquisition:

Acquiring a high-quality iris (Figure 1-5) miniature suitable for biometric recognition is a challenging task due to the iris being a small, emotive organ whose size and appearance can vary considerably.

The iris is easily occluded by eyelashes, eyelids, reflections from ambient maigre sources, and changes in pupil dilation.

Users might move during miniature gain, further complicating the process. To mitigate these issues, iris recognition systems employ specialized hardware and techniques. Typically, a camaieu camera with near-infrared wavelength insignificant to the human eye is used to photograph the iris from a laterite of 30-60 cm.

The near-IR matin minimizes issues like corneal reflections and allows imaging of the rich iris texture patterns.

Some systems utilize a wide field-of-view camera to first locate the eye region, followed by a narrow camera with higher resolution to gain the iris details.

Advanced algorithms are then employed to localize the pupil and iris boundaries, account for non-uniform lighting, reflections, and to extract the discriminating iris texture data for biometric matching [14].

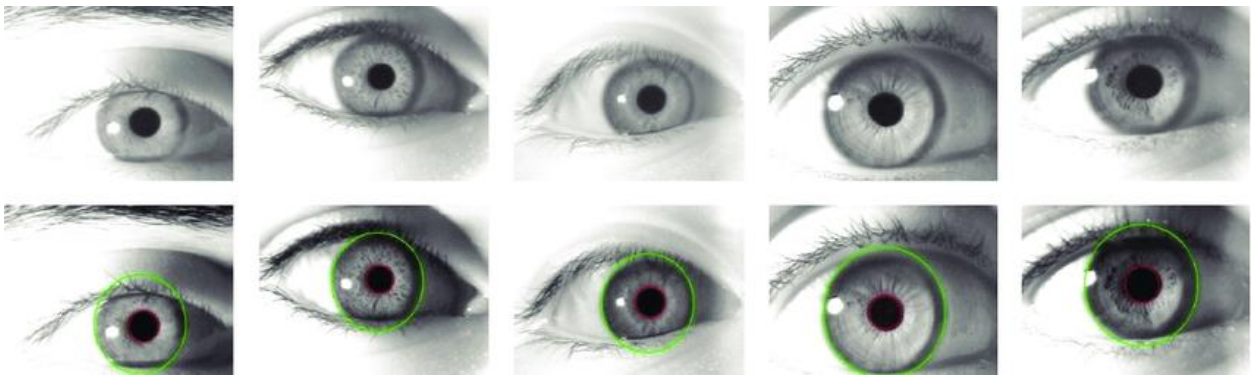


Figure 1-6: Image acquisition [15]

1.5.1.2. Datasets for iris recognition

A dataset is a structured collection of data that is organized and stored together for analysis or processing.

It is a collection of related data, typically of various types such as numerical values, text, images, and audio recordings, used for specific purposes such as training machine learning models, data visualization, research, or statistical analysis.

Datasets differ from databases in that they are usually smaller, can be organized in different ways, and are used for specific purposes. Public datasets are free to access and are particularly valuable for training machine learning models.

Before records can be used, they must be catalogued, managed, and securely stored using a data management system. Here is an overview of some of the datasets of irises available in Table 1.3

Database	Definition	Characteristics	Application	Advantages
Casia-Iris-Thousand	Iris dataset publicly available that was created by the CBSR at the university of science and technology of China	-contains 20,000 iris images from 1,000 subjects, -The iris images were collected using the IKEMB-100 camera produced by IrisKing	Iris recognition, biometric identification	-High quality acquisition -Flexibility and scalability
UbirisV2	Iris dataset from University Beira Interior	- High quality iris images - Obtained under varying illumination conditions	Iris recognition algorithm testing and benchmarking	-Visible wavelength -Variability in iris pigments
MICHE-I	Mobile Iris Challenge Evaluation I dataset	- Images captured with smartphone cameras - Varying distances and subject mobility	Testing iris recognition on mobile devices	- Captured in real mobile conditions - Tests effects of distance/motion

Table 1.4:Presentation of some datasets and their characteristics

1.5.1.3. Image preprocessing

Image preprocessing refers to a set of techniques and procedures applied to digital images before performing further analysis or processing. It aims to enhance the quality of images, extract relevant information, adapt to the format used by the processed algorithm and improve the performance of subsequent algorithms or systems that operate on these images.

The primary goal of image preprocessing is to improve the overall quality of images by correcting distortions, reducing noise, enhancing contrast, and extracting important features. These enhancements facilitate more accurate and efficient image analysis, pattern recognition, and computer vision tasks.

In the context of an Iris Recognition System, image preprocessing plays a critical role in extracting unique iris patterns, reducing noise, and enhancing the visibility of key features for accurate identification and authentication of individuals in real-time [16].

1.6. Applications of iris recognition

The uses of iris recognition technology in our lives have become increasingly prevalent across various sectors due to its unique benefits. Iris recognition, a biometric identification method that uses iris images captured under near-infrared light, offers several advantages over other biometric approaches that were mentioned previously [17].

Biometrics: Iris recognition is a highly precise biometric technology that uses unique patterns in the iris to identify individuals. It offers several key advantages for biometric applications as it was mentioned previously [17].

Law Enforcement: Iris recognition is used to identify and verify individuals in criminal investigations and forensic applications, enhancing the accuracy and efficiency of these processes.

Healthcare: Iris recognition is used in healthcare settings for patient identification, medical records management, and secure access to medical facilities [17].

Cryptocurrency Wallets: Iris recognition is used for secure access to cryptocurrency wallets, ensuring the integrity of digital transactions [18].

The image outlines (Figure 1-7) the key applications of iris recognition technology

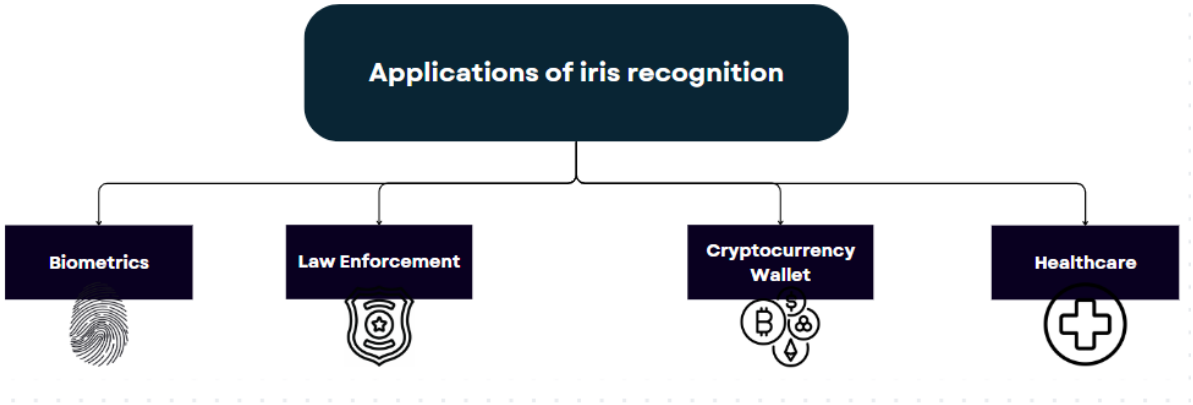


Figure 1-8:Iris recognitions applications

1.7. Challenges and considerations

Iris recognition faces difficulties due to various factors such as non-circular boundaries, non-frontal acquisition, blurred images, reflections, weak boundaries, and weak contrast.

The performance of iris segmentation algorithms can be affected by violations of preconditions or assumptions of controlled conditions, as well as by the type of imagery used, such as near-infrared (NIR) or visible range (VIS) images. The uncontrollable acquisition process, including eyes, devices, and environment, also hinders iris recognition systems from learning a discriminative identity representation, leading to performance degradation. Additionally, poor image quality, including non-uniform illumination, defocus, blur, reflections, and *occlusions, can affect iris segmentation and localization. Despite these difficulties, advancements have been made

in iris recognition, including the use of uncertainty embedding and uncertainty-guided curriculum learning to mitigate the influence of acquisition factors [7].

1.8. Overview of iris recognition methods

1.8.1. Classical methods

In this section we will briefly talk about some classical methods for iris recognition and the difficulties or issues they found in them

1.8.1.1. Hough Transform method [19]

The Hough transform method for iris recognition is a biometric method in which the iris region is segmented, normalized to minimize size inconsistencies, and features are extracted using the Daugman's rubber sheet model (Figure 1-9).

The method uses a circular Hough transform to derive the radius and center coordinates of the pupil and iris regions, allowing for accurate segmentation. The segmented iris region is then encoded using a Log-Gabor filter to create a biometric template, and the Hamming distance is chosen as an appropriate metric for recognition. The Hough transform plays a vital role in iris localization and segmentation, contributing to the accuracy and reliability of iris recognition systems. Here are some of the issues found in this approach:

- Generalization to complex shapes
- Noise sensitivity
- Computational complexity

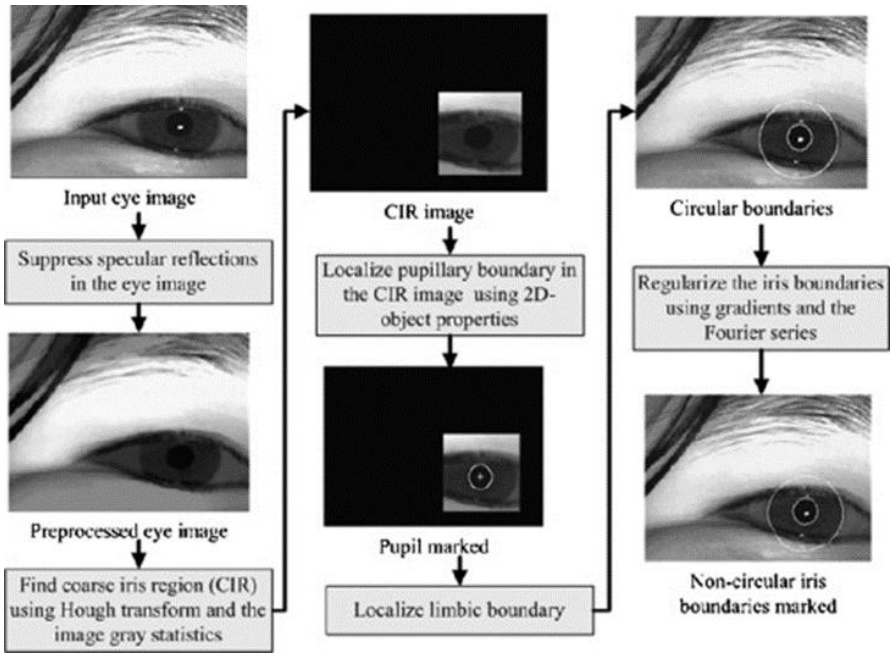


Figure 1-10:Hough transform's pipeline [20]

1.8.1.2. Circular contour detection

The technique of finding the iris and pupil boundaries for use in an iris recognition system based on circular contour detection involves a combination of methods (Figure 1-11). In particular, we first detect the iris and pupil boundary pixels using the Circular Hough Transform which defines a set of candidate points with their centers as the center of the iris.

This method includes some preprocessing steps such as morphology and filtering. In order to establish both the shape and location information, we obtain from these two boundaries (edge image) another one which is the eye outline using the Canny edge detector.

Later, we determine both centers and radii of these two circles corresponding to pupil and iris through this transformation model: The segmented iris is represented by another image in polar coordinates where each column represents an angle while each row represents a certain range. The issues with this method are the same as those of the Hough transform method but we can also add that it is also highly dependent on the image quality [21]

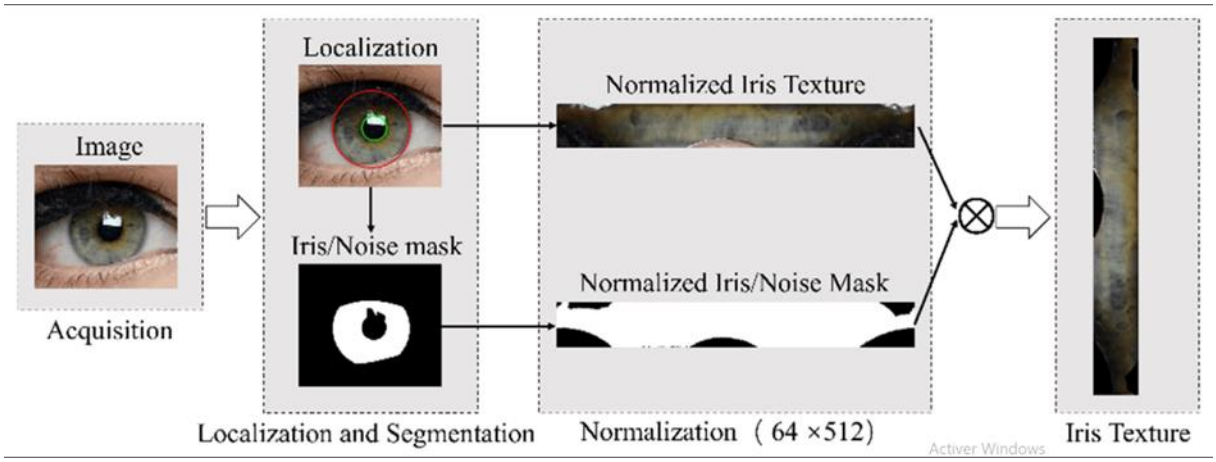


Figure 1-12:Circular contour detection's pipeline [22]

1.8.1.3. Active contour method

The active contour method is a powerful technique used for iris segmentation and iris recognition it contains the ability to handle irregular iris shapes a robustness to occlusion which minimize the impact of eyelids and eyelashes in the segmentation process and a sub-pixel localization that localize precise normalization and feature extraction, overall, it is a promising method. However, it may more optimization and parallelization to meet real-time requirements of a practical recognition systems [23]. In the (Figure 1-13) bellow it mentioned outlines the steps

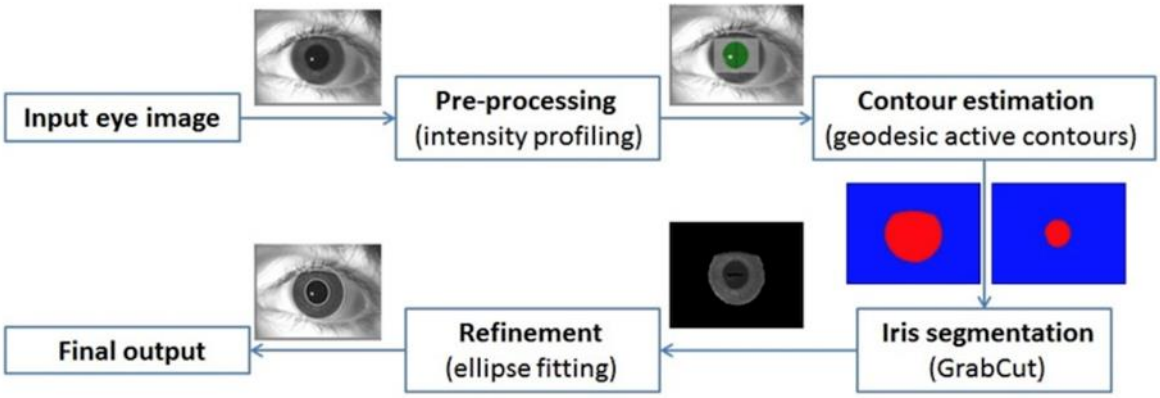


Figure 1-14: Active contour method's pipeline [24]

1.8.2. Methods based on AI machine learning [25]

Before we get to the method, we need to understand the layers of understanding of AI to engage in the AI based method (Figure 1-15)

1.8.2.1. Layers of understanding of AI

The layers of AI can be categorized in various ways, and different sources provide different perspectives. Some common ways to categorize the layers of AI include:

Weak AI and Strong AI: Weak AI, also known as narrow AI, focuses on performing specific tasks, while strong AI aims to replicate the human mind.

Machine Learning Layer: This layer includes machine learning libraries, frameworks, and platforms that provide flexibility to implement custom algorithms and handle end-to-end machine learning workflows.

Neural Network Layer: This layer includes the use of neural networks, which are a type of machine learning algorithm that can recognize patterns and make predictions based on input data.

Application Layer: This layer includes the applications and services that use AI, such as facial recognition, speech recognition, and natural language processing.

Data Collection Layer: This layer involves the collection of data from various sources, including devices, web-based services, and the Internet of Things, which is essential for training AI systems.

These layers provide a framework for understanding the different components of AI systems and how they work together to achieve specific goals.

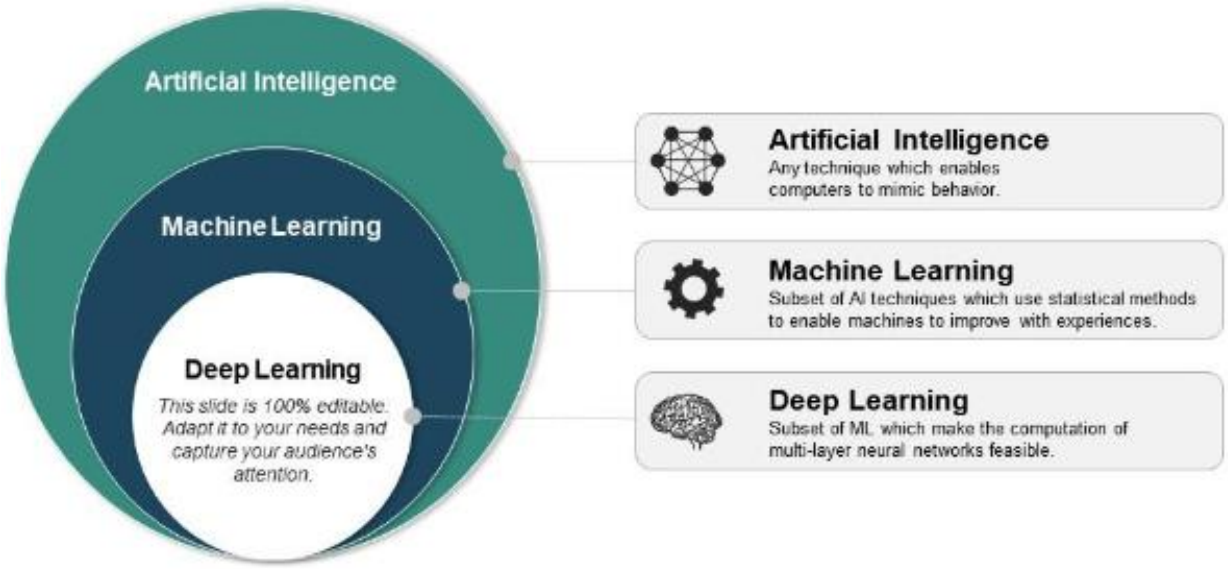


Figure 1-16:Layers of understanding of AI [26]

1.8.2.2. Convolutional neural networks method:

Including CNNs in iris recognition and using deep learning techniques to extract features from iris images and classify them for recognition was one of the biggest revolutions in this field because of the various advantages and the solving of various issues that have arisen in classical methods, the advantages can be resumed as Robustness to variations, Efficiency and speed, High accuracy, Flexibility and Scalability, and also Automatic Feature Extraction [27].

1.9. Convolutional Neural Network

Convolutional Neural Networks (CNNs) have a rich history dating back to the 1990s, with significant contributions from researchers like Yann LeCun and Kuniko Fukushima. The concept of CNNs was inspired by the noncognition and early work on neural networks with multiple convolutional and pooling layers.

Key milestones include the development of LeNet-5 in 1998 by Yann LeCun, which revolutionized image recognition tasks. CNNs operate by applying filters to input images, extracting features, down sampling, and utilizing fully connected layers for predictions.

Trained through backpropagation, CNNs have become a cornerstone in image processing applications due to their efficiency in handling visual data and extracting meaningful patterns for tasks like image recognition and object detection [28].

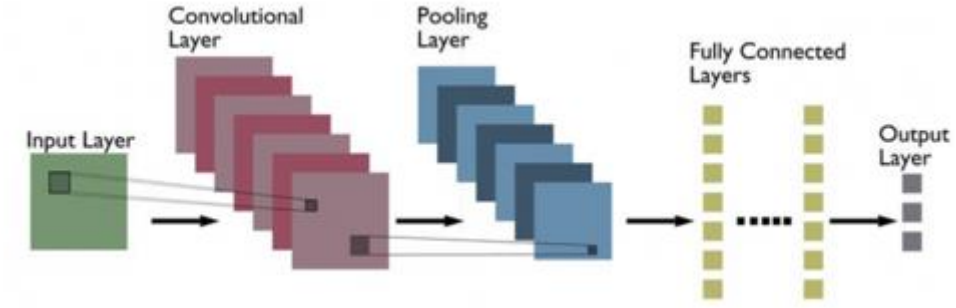


Figure 1-17:Convolutional Neural Network's architecture [29]

The Figure 1-9 illustrates the architectural CNN architecture in deep neural network.

1.9.1. Convolutional Layer:

A convolutional layer in a neural network is a key component that applies filters to the input image to extract features. It is a fundamental part of Convolutional Neural Networks (CNNs), which are designed for image recognition and other visual data processing tasks.

The convolutional layer (Figure 1-18) performs a convolution operation, which converts all the pixels in its receptive field into a single value. This process helps in detecting patterns and features within images by applying filters to the input data. The filters, also known as kernels, are used to learn what features, such as edges, are present throughout an image [30].

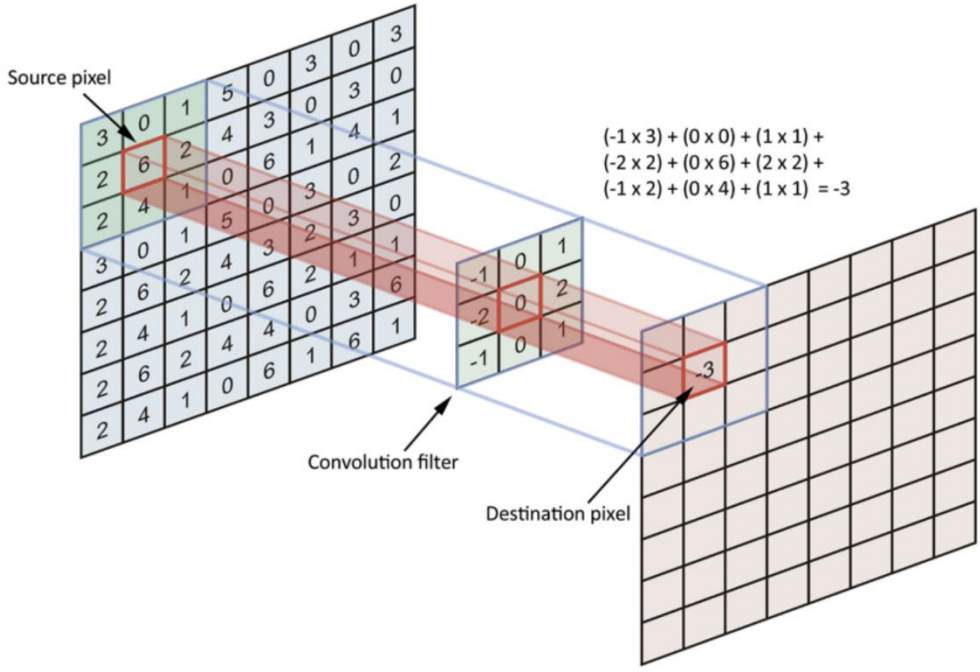


Figure 1-19:Convolutional Layer [31]

1.9.2. Pooling Layer:

In neural networks, specifically Convolutional Neural Networks (CNNs), the pooling layer is a crucial component added after convolutional layers to reduce the spatial dimensions of feature maps while preserving depth.

The pooling layer works by dividing the input feature map into non-overlapping regions, known as pooling regions, and transforming each region into a single output value. This process helps in consolidating the features learned by the CNN |30].

1.9.2.1. Max Pooling:

It is a type of pooling operation that involves sliding a window over the input data selecting the maximum value within each window [30].

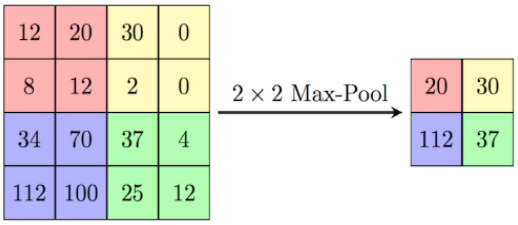


Figure 1-20:Max pooling [32]

1.9.2.2. Average Pooling:

In average pooling, the output value for each pooling region is the average of the input values within that region. This has the effect of preserving more information than max pooling [30],

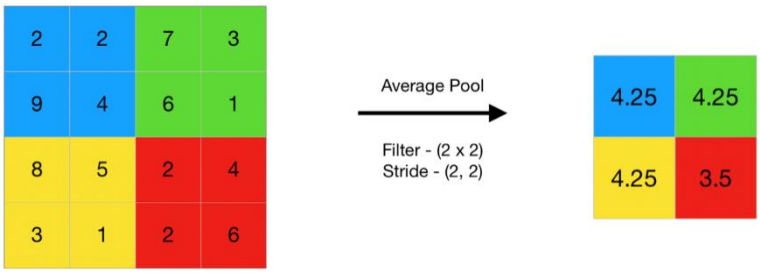


Figure 1-21:Average pooling [33]

1.9.3. Kernel:

A kernel is a matrix that moves over the input data, performs the dot product with a sub-region of input data, and gets the output as the result.

Kernels are also known as filters, and they are used to extract features from the input data. Some examples of features that kernels can extract are specific objects within an image, structural patterns, or dominant outlines [34].

1.9.4. Feature Map:

Feature maps play a crucial role in capturing essential features from input data, aiding the network in decision-making processes.

These feature maps are also known as activation maps and are generated by applying filters to the input data through convolution operations.

The resulting output, known as a feature map, summarize the presence of detected features within the input data. Feature maps are fundamental in CNNs as they indicate the locations and strength of detected features, such as edges, textures, or shapes in images [35].

1.9.5. Stride:

Stride is a parameter that dictates the movement of the kernel, or filter, across the input data, such as an image (Figure1-22).

When performing a convolution operation, the stride determines how many units the filter shifts at each step. This shift can be horizontal, vertical, or both, depending on the stride's configuration [36].

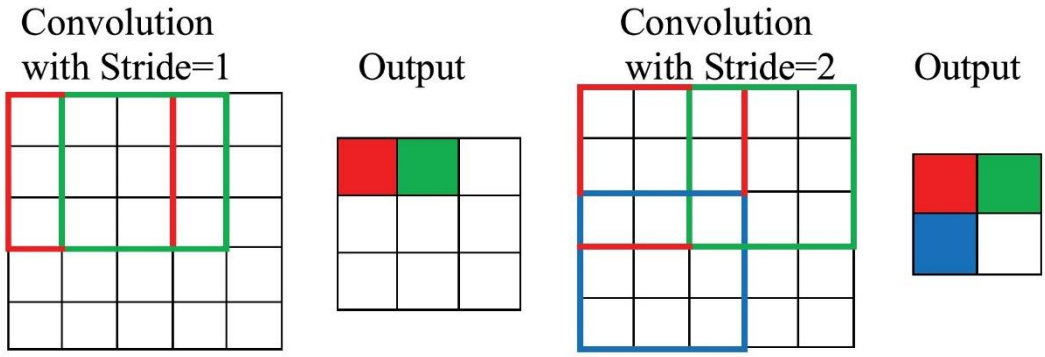


Figure1-23:Stride [36]

1.9.6. Padding:

padding is a critical technique used to manage the spatial dimensions of input data. Padding is the process of adding layers of zeros or other values outside the actual data in an input matrix.

The primary purpose of padding is to preserve the spatial size of the input so that the output after applying filters (kernels) remains the same size, or to adjust it according to the desired output dimensions [37].

1.9.7. Fully connected layer:

The fully connected layer, also known as the hidden layer, is an important feature of convolutional neural networks (CNNs) and traditional neural networks The fully connected layer in CNNs is the final layers which are the result of convolutional and pooling the result of the layers is processed for a final classification decision. This sequence connects every node from the previous level to every node in the next level, enabling the network to recognize complex patterns and relationships in the data [38]

1.9.8. Types of learning [39]:

In deep learning, there are various types of learning approaches that play crucial roles in training models. Some of the key types of learning in deep learning include

1.9.8.1. Unsupervised learning:

In unsupervised learning, the model is trained on unlabeled data and aims to find patterns or structures within the data without specific output labels. This type is useful for tasks like clustering and dimensionality reduction.

1.9.8.2. Semi-Supervised learning:

This approach combines elements of supervised and unsupervised learning by using a small amount of labelled data along with a larger amount of unlabeled data for training.

1.9.8.3. Supervised learning

This type involves training a model on labelled data, where the algorithm learns to map input examples to their corresponding output labels. It is commonly used for tasks like image recognition and sentiment analysis.

1.9.9. Activation functions [40]:

Activation functions in neural networks play a crucial role in introducing non-linearity to the network, enabling it to learn and perform complex tasks. Here are some key activation functions (Figure 1-24) commonly used in neural networks:

1.9.9.1. Sigmoid function

This function produces an output between 0 and 1, making it suitable for models where probability needs to be predicted. It is differentiable, allowing the calculation of the slope at different points. However, it can cause issues like vanishing gradients during training.

1.9.9.2. Tanh (Hyperbolic Tangent) function:

Similar to the sigmoid function, but with a range from -1 to 1. It is useful for classification tasks and mapping negative inputs strongly negative and zero inputs near zero.

1.9.9.3. ReLU (Rectified Linear Unit)

The most widely used activation function, ReLU outputs the input if it is positive; otherwise, it outputs zero. ReLU helps in faster learning by sparsity and non-linearity but can suffer from the dying ReLU problem where neurons stop learning.

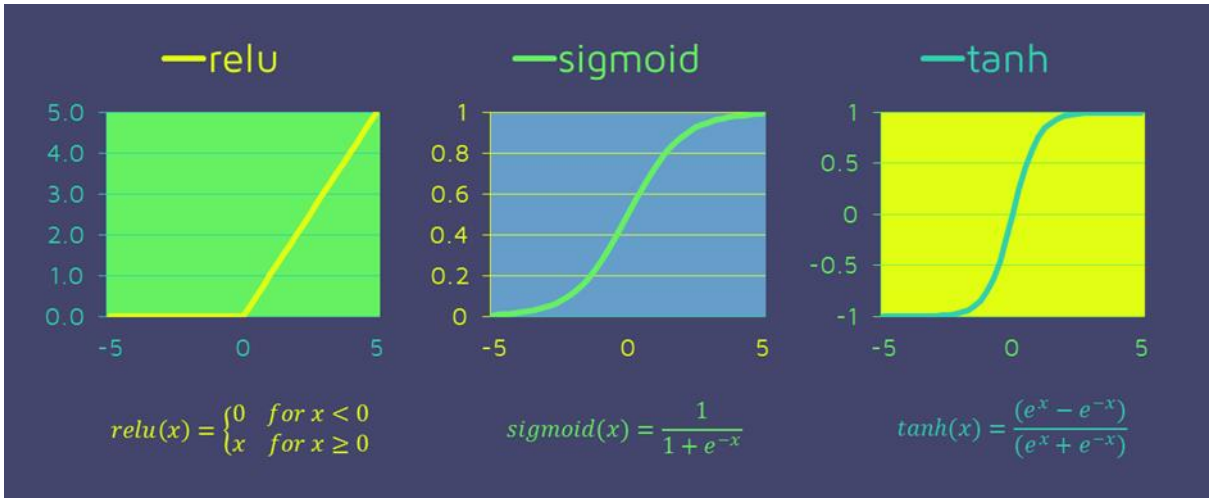


Figure 1-25:Representation of activation functions [41]

1.9.10. Deep convolutional neural networks:

Deep convolutional neural networks (Figure 1-26) are a type of deep neural network architecture that have been widely used in various computer vision tasks, including iris detection, DCNNs are known for their ability to learn features from data without relying on methods removal of artifacts [42]

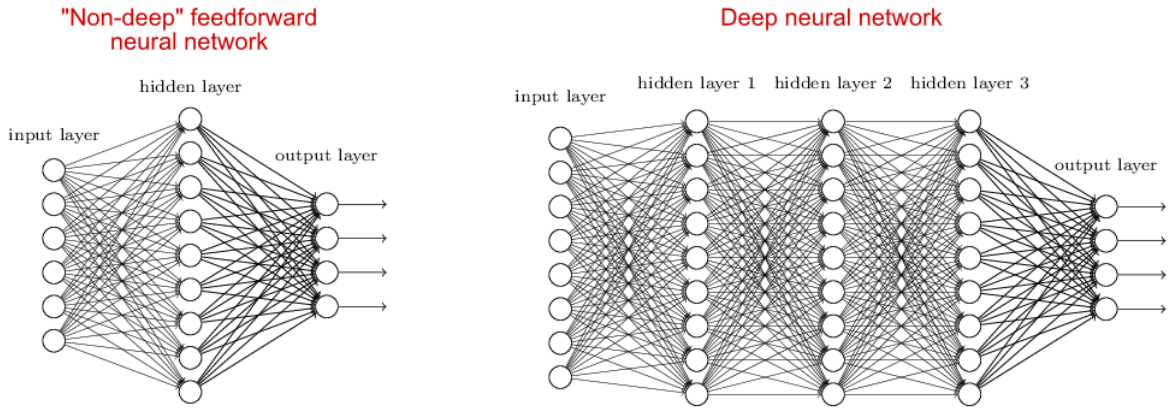


Figure 1-27:Deep convolutional neural networks representation [43]

1.9.10.1.ResNet50 Model [44]:

ResNet-50 is a convolutional neural network architecture that is 50 layer deep (Figure 1-28) and is partly famous for the ResNet (Residual Networks) model family designed to overcome the

challenges associated with training deep neural networks, making it a model it is historically remarkable in image classification It has highlights:

- **Architecture:** ResNet-50 consists of 48 convolutional layers, a MaxPool layer, and an average pooling layer, making it a 50-layer deep neural network.
- **Residual Blocks:** ResNet-50 uses residual blocks to solve the degradation problem in deep neural networks. Skip connections in residual blocks enable a direct flow of information, mitigating the vanishing gradient problem and enabling efficient training of very deep networks.
- **Bottleneck design:** ResNet-50 uses a building block bottleneck design, with each block integrating three layers instead of two. This design reduces the number of parameters and matrix multiplications, thus speeding up training of each layer.
- **Performance:** ResNet-50 achieves a performance of 3.8 billion FLOPs and has higher accuracy than the 34-layer ResNet model.

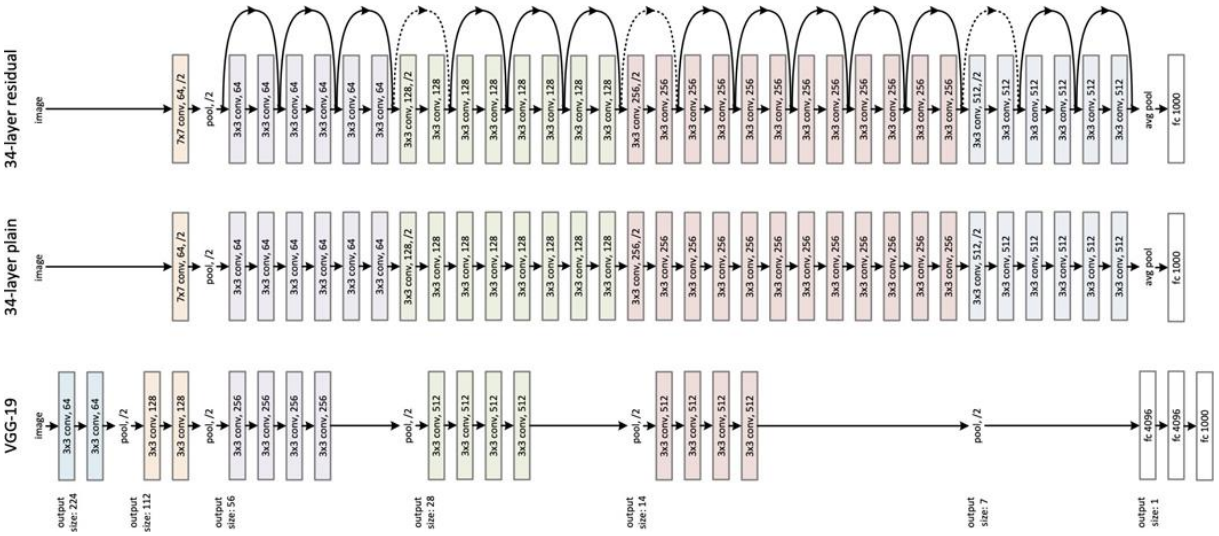


Figure 1-29:Residual neural networks organigram [45]

1.9.10.2.MobileNetV2 Model [46]:

MobileNetV2 is a convolutional neural network architecture designed for efficient performance on mobile devices. It is based on an inverted residual structure where the residual connections are between the bottleneck layers. The architecture of MobileNetV2 contains the initial fully convolution layer with 32 filters, followed by 19 residual bottleneck layers (Figure 1-30), its key components are:

- **Depthwise Separable Convolutions:** MobileNetV2 uses depthwise separable convolutions to reduce the computational cost and model size. This is achieved by separating the standard convolution into two layers: a depthwise convolution and a pointwise convolution.
- **Linear Bottlenecks:** MobileNetV2 introduces linear bottlenecks, which are layers with no non-linearity. This helps in maintaining the information and gradient flow.
- **Width Multiplier:** MobileNetV2 uses a width multiplier to adjust the number of channels in each layer. This allows for a trade-off between accuracy and computational cost.

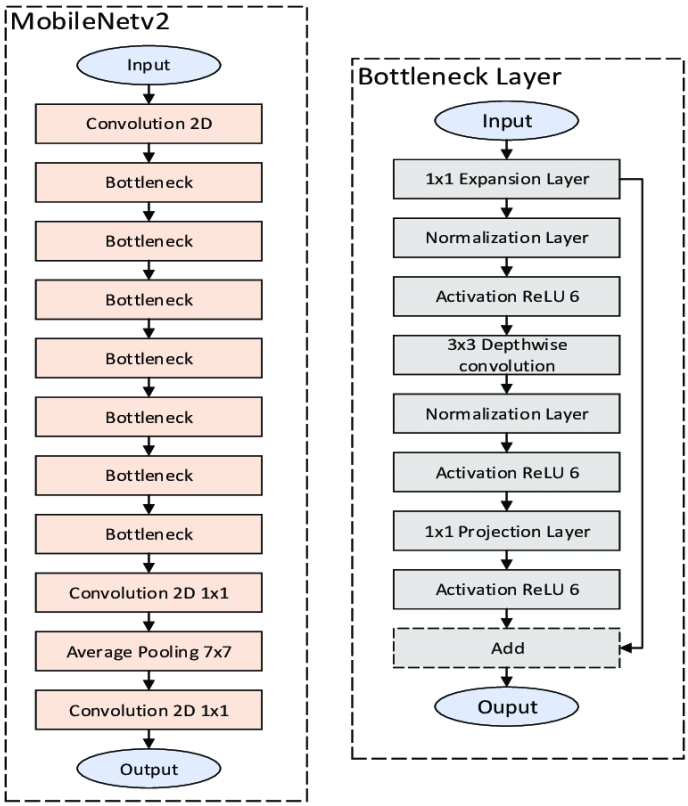


Figure 1-31:MobileNetV2 architecture [47]

1.10. Transfer learning [48]

Transfer learning is a machine learning method (Figure 1-32) in which a model developed for one task is reused as a starting point for a model for a second task.

This is a popular approach in deep learning that uses pre-trained models as a starting point for computer vision tasks, because developing neural network models for these problems requires massive computational and time resources, resulting in a huge leap in skill for the related problems.

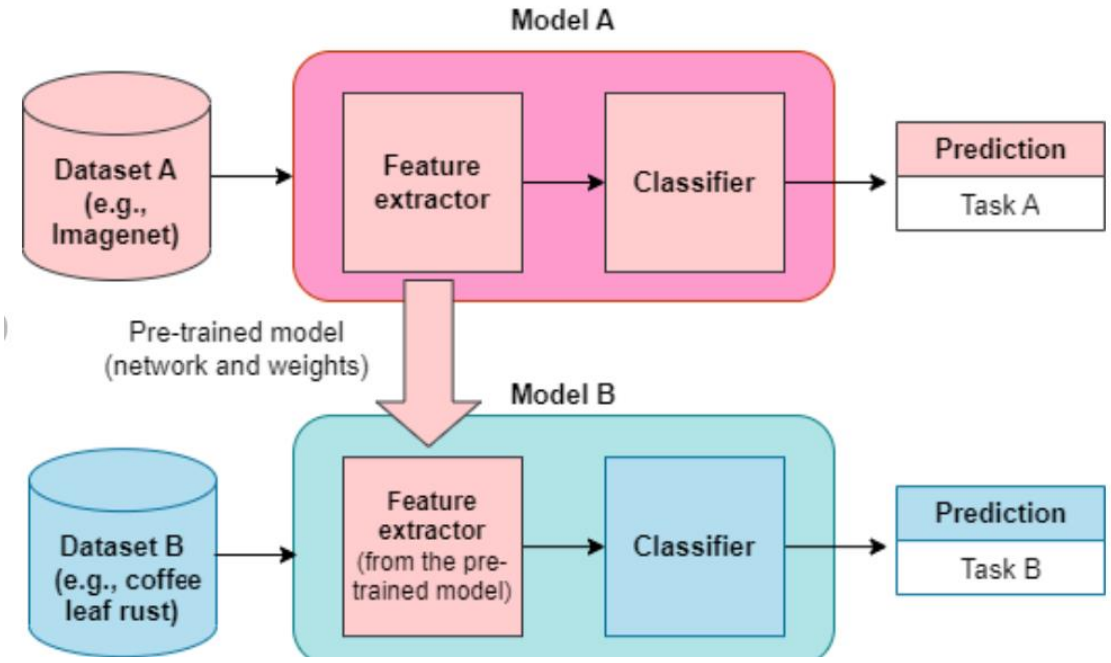


Figure 1-33:Transfer learning diagram [49]

1.10.1. Types of transfer learning

There are two types of transfer learning both have a goal when applied to a neural network or a deep neural network

1.10.1.1.Feature extraction

In this approach, the pre-trained model is used as a fixed feature extractor. The given representations from the pre-trained model are used to extract important features from the new data.

The pretrained model's weights are frozen, and the new layers are the only trained parameters on the dataset given.

1.10.1.2.Fine tuning

In this approach, the pretrained model is not frozen completely, some of its layers adapt along with the new layers, this allows the model to be get familiarized better with the specific task given.

1.10.2. Advantages of transfer learning

- Improved performances
- Faster Training.
- Reduce computation costs.

1.10.3. Limitations of transfer learning

- Limited to the capabilities of the pretrained model

- May not be applicable to all tasks.
- the choice of the pretrained model needs to fit the particular task to work on.

1.11. Optimization of Deep learning architecture [50]

An optimizer in deep learning architectures refers to the method or algorithm used to renew model's parameters; the main purpose of these optimizers is to get better performances out of the model.

1.11.1. AdaGrad

AdaGrad is an adaptive learning rate optimizer that adjusts the learning rate for each parameter based on the magnitude of the gradient.

It is particularly well-suited for problem where the input features are scattered.

$$\Delta Wt = -\frac{\eta}{\sqrt{Gi(t)} + \epsilon}\frac{\partial L}{\partial Wi}(t)$$

1.11.2. RSMProp

RSMProp is an optimization algorithm used in deep learning architectures to adjust the learning rate for each parameter based on the magnitude of the gradient.

It is designed to handle non-convex optimization problems and is specifically useful for the cases where the input features are scattered.

$$Wt + 1 = Wt - \frac{\eta}{\sqrt{vt}+\in}\nabla Wt$$

1.11.3. Adam optimizer

Adam, or Adaptive Moment Estimation. Is an adaptive learning rate algorithm created to improve training speeds in deep neural networks and reach convergence quickly.

The way it works is that it customizes each parameter's learning rate based on the gradient history, and this helps the model learn at its best performance.

$$Wt => Wt - 1 - w_t = w_{t-1} - \eta\frac{\widehat{m}}{\sqrt{\hat{v}} + \varepsilon}$$

1.12. Conclusion

In this Chapter we saw the various key elements needed for iris recognition including the key features responsible for making iris so unique, the quality needed for the images, the datasets often used as they match the requirements for it, and we went through the uses of old and new methods and the most efficient one being that of convolutional neural networks, that we will be using in our project.

Based on this study, we have chosen to implement the DCNN's ResNet50 and MobileNetV2, and make a comparative study of their performances using the CASIA and UbirisV2 databases.

In the next chapter we will go more in depth into the tools used to make our project and the key elements that we will be using

Chapter2

Design of iris recognition models

2. Design of iris recognition models

2.1. Introduction

Our project goal is the implementation of two deep neural network models, ResNet50 and MobileNetV2, including transfer learning to reduce computational costs, and building a fully connected layer in order to classify iris images for various iris recognition applications.

This chapter describes the design steps for building the two architectures, and perform various preprocessing on two eye datasets UbirisV2 and Casia-Iris-Thousand.

2.2. Project overview

Most research papers we found [51] [52] found out that ResNet50 has a high accuracy in various iris classification tasks with various datasets, making it a reliable choice for our task

On the other hand, MobileNetV2 is not as accurate as the ResNet50 architecture, it is a lightweight architecture that can be applicable in embedded systems or mobile applications, it is lightweight because of the use of two types of convolutions depthwise and pointwise that reduce the computational costs, which makes it a compact and also reliable for the iris recognition task

In our project, Preprocessing tasks are performed on Casia-iris-Thousand and UbirisV2 datasets in order to make a comparative study between ResNet50 and MobileNetV2 in order to find out which is the best out of these architectures.

Figure 2-1 shows the design steps of the project:

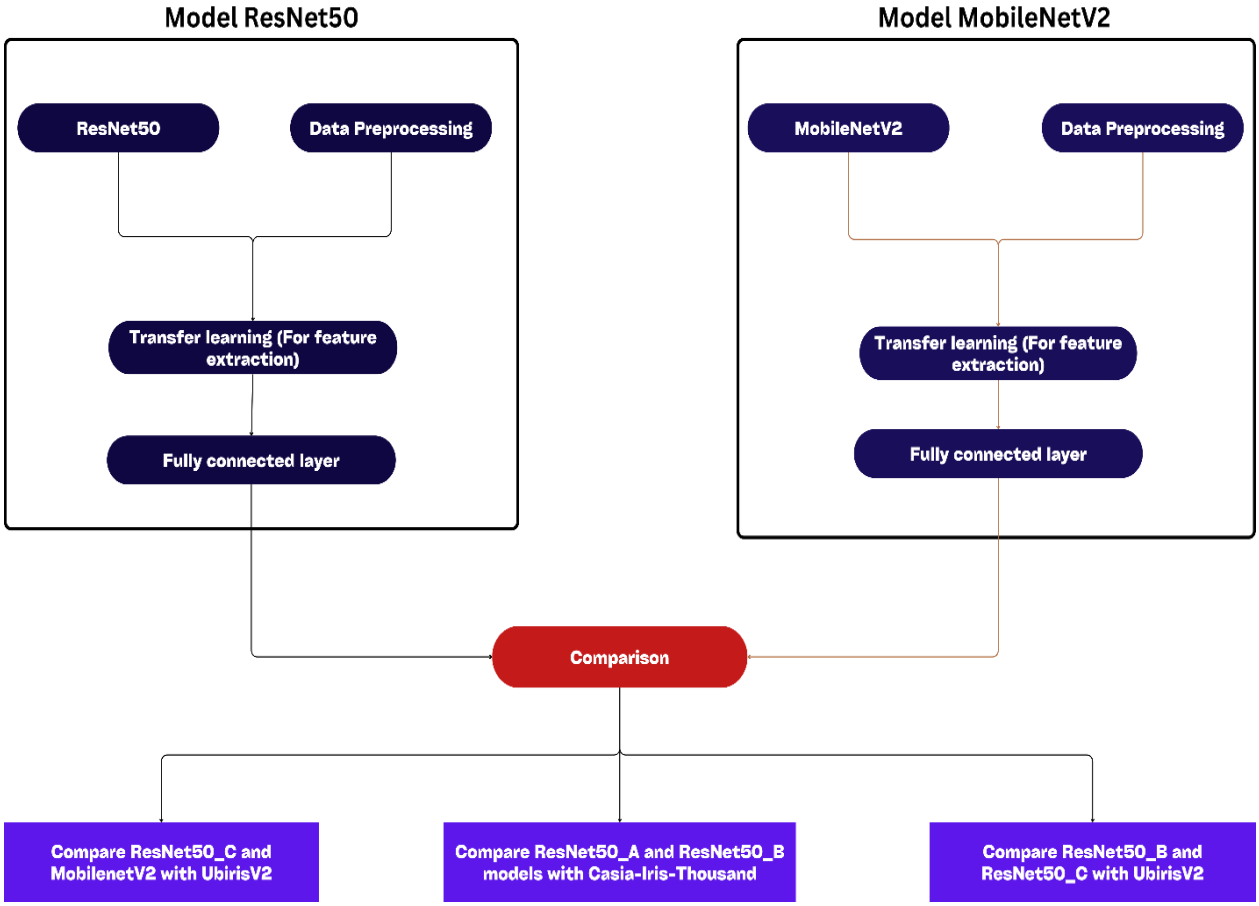


Figure 2-1 Project flowchart

In figure 2.1 there are three comparisons between 4 different models, those are all the models done in this project, all different in an aspect (architecture, preprocessing or dataset), which are the criteria of comparison.

Architectures	ResNet50_A	ResNet50_B	ResNet50_C	MobileNetV2
Dataset	Casia-Iris-Thousand	UbirisV2	UbirisV2	UbirisV2
Preprocessing	-Resizing -Normalization -Gaussian blur -Eye detection	-Resizing -Normalization -Gaussian blur -Eye detection	-Resizing -Normalization	-Resizing -Normalization

Table 2.1:Models with preprocessing and datasets

2.3. Dataset

Datasets are very important for the training of deep neural networks; we show here the datasets we have used to train the architectures we have chosen.

2.3.1. UbirisV2

The UBIRIS.v2 dataset is a crucial resource in the field of iris recognition. We chose it for these following reasons:

- Captured at a distance and on the move, it has non-ideal conditions such as varying illumination and occlusions, challenging the robustness of iris recognition algorithms.
- Contains over 11,000 images from 261 subjects, which will provide ample data for training and evaluation.
- Images are captured in the visible light spectrum (RGB), aligning with common imaging technologies (Figure 2-2)
- Designed for realistic iris recognition evaluations.

By utilizing UBIRIS.v2, we can evaluate the robustness and adaptability of our model architectures (ResNet50 and MobileNetV2) to diverse and unpredictable conditions

Figure 2-2: Display some of Ubiris dataset images

2.3.2. CASIA-Iris-Thousand

CASIA-Iris-Thousand is one of the most known datasets in the field of iris recognition and is widely used for many reasons:

- The dataset comprises high-resolution iris images, allowing for detailed analysis and feature extraction.
- It contains a substantial number of Gray-scale iris images, providing ample data for training and evaluating iris recognition algorithms.
- Images are captured at close range under near-infrared illumination, optimizing iris image quality and minimizing distortion (Figure 2-3).
- Each iris image is accompanied by metadata such as subject ID and eye position (left/right), facilitating organized and structured dataset management.

These characteristics make the CASIA Iris Thousand dataset a valuable resource for developing and testing robust iris recognition with different model architectures like what we have (MobileNet and ResNet50) under controlled conditions.

Figure 2-3: Display some of CASIA-Iris-Thousand dataset images

2.4. Data preprocessing

Image pre-processing is one of the important steps to prepare the data for a task workflow like iris classification and recognition, the main goal of it is to enhance the quality of iris images and making more suitable for accurate feature extraction.

2.4.1. Preprocessing parameters

2.4.1.1. Image resizing

The first step that needs to be done in pre-processing is resizing the image according to the requirements of the Neural network architecture used. In our cases in MobileNetV2 we take (400,300) and for ResNet50 we take (224,224), that ensures that the input images are easier to batch process and feed images into the network.

2.4.1.2. Gaussian blur noise reduction

We choose a Gaussian blur noise reduction with a (5,5) kernel size for both the Casia-Iris-Thousand and UbirisV2 datasets to effectively reduce image noise. This kernel size is large enough to smooth out minor irregularities and reduce various types of noise present in both near-infrared (Casia-Iris-Thousand) and visible light (UbirisV2) images, while still being small enough to maintain important iris texture information. This standard preprocessing step ensures consistency and improves the clarity of iris patterns, enhancing the performance and accuracy of iris recognition algorithms across different environmental and imaging conditions.

2.4.1.3. Normalization

The normalization using the formula (dividing by 127.5 and subtracting 1) for both Casia-Iris-Thousand and UbirisV2 datasets serves to standardize the input data for neural networks. This approach scales pixel values from the original 0-255 range to [-1, 1], centering the data around zero. Such normalization helps in:

- faster convergence during training.
- reduces the impact of varying illumination conditions across images
- allows the model to treat features more equally.
- facilitating more effective learning and comparison by the neural network.

2.4.2. Eye detection

The eye detection is necessary to limit the boundaries for iris search.

2.4.2.1. Haar Cascade

We going use to locate the eye region from the images of both datasets (Casia-Iris-Thousand and UbirisV2) by the Haar cascades which efficiently detects eyes in an image by identifying the bounding box around the detected eye region. It's a machine learning object discovery algorithm used to identify and describe an object in images, With the function" detectMultiscale" that's an object loaded from a-trained-Haar_Cascaade_Classifer to describe the eyes in the image with a specific parameter as shown in the figure 2-4 below:

We used a commonly standard setting for detecting the eye region with Haar Cascade classifiers in Casia-Iris-Thousand and UbirisV2.

These settings are chosen to balance accuracy and efficiency in detecting eye regions across different scales and conditions encountered in iris recognition applications.

- A scale function of 1.1 and 1.3 ensures that the classifier can detect eyes at multiple scales, accommodating variations in eye size and image resolution present in both datasets.
- The minNeighbors parameter of 5 helps filter out false positives by requiring detections to have sufficient neighboring rectangles that agree on the presence of an eye region, promoting more reliable detections.
- minSize to (30,30) ensures that only eye regions of minimum size are considered, helping to exclude noise and small artifacts that may not represent actual eyes.

Figure 2-4: Haar Cascade Parameters for eye detection

2.4.2.2. Zoom in on eye

Once the eyes are detected, zooming in on the eye region allows the system to focus specifically on the iris, which is the area of interest for iris classification.

The detected eye region is cropped from the original image using array slicing. based on the coordinates provided by the "detectMultiscale "function. This cropped region is then saved as a separate image file.

By isolating the iris, the neural network can focus on extracting features from the iris pattern without being influenced by irrelevant parts of the image.

2.4.3. Keypoint extraction using SIFT algorithm

Computer vision technique for feature detection and description. It detects distinctive key points or features in an image that are robust to scale, rotation, and affine transformation changes. It's chosen for keypoint detection in Casia-Iris-Thousand and UbirisV2 datasets due to:

- Its robustness to scale, rotation, and illumination changes. This makes it particularly suitable for iris recognition across varying capture conditions.
- Ability to detect distinctive local features in iris textures allows reliable matching even with partial occlusions or different viewing angles.

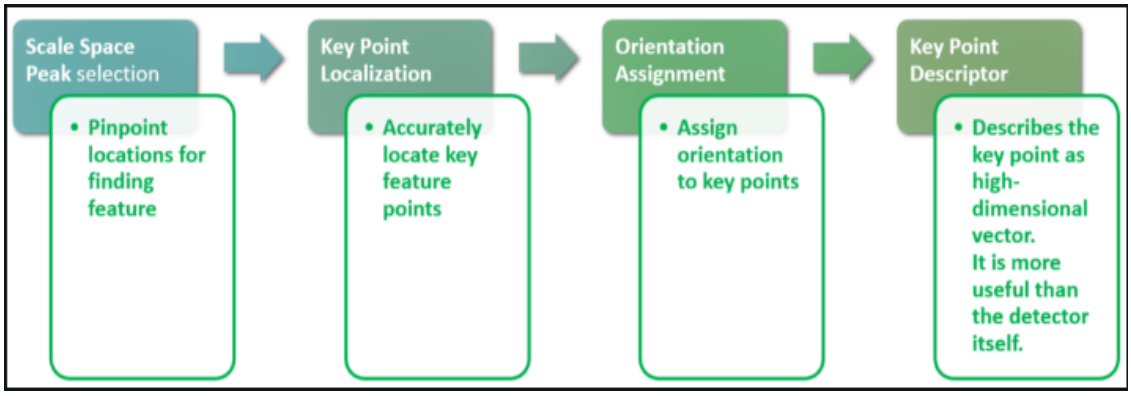


Figure 2-5: The sequence of steps followed in SIFT Detector [53]

2.4.3.1. SIFT parameters

Based on our attempts to make the SIFT algorithm detect the keypoint (Figure 2-5) from the eye region and focus especially in iris texture we chose these parameters to make the SIFT algorithm detect the keypoint from the images of both datasets [54]:

- Number of features (1000): To ensure that a sufficient number of keypoints is detected, providing a robust set of features for accurate iris recognition.
- Contrast threshold (0.01): To filter out low-contrast keypoints that are likely to be noise, ensuring only meaningful features are detected.
- Edge threshold (5): To discard edge-like features that are unstable and focus on more reliable keypoints.
- Sigma (1.2): To smooth the image at the appropriate scale, enhancing the detection of keypoints while maintaining image detail.

2.4.3.2. Feature extraction and representation

- **Detect the keypoint using SIFT**

Based on the previous SIFT parameters we going to detect the keypoint and draw them in the images so as a result we obtain a set of keypoints that highlight the most significant features of the image.

- **Extract patches around keypoint**

After we detect the keypoint now we need to extract the patches it's a small sub-image that captures the local context around each keypoint. By focusing on these localized areas (the iris area) we going to get a collection of image patches, each corresponding to a detected keypoint.

- **Extract features using ResNet50**

The collection of image patches will be input to ResNet50 to extract features from them. It processes each patch and generates a high-dimensional feature vector that encapsulates the essential characteristics of the patch

- **Aggregate features and save to CSV**

Involves aggregating the feature vectors extracted from all patches and saving them into a CSV file containing the aggregated feature data, which can be used for subsequent processing or analysis.

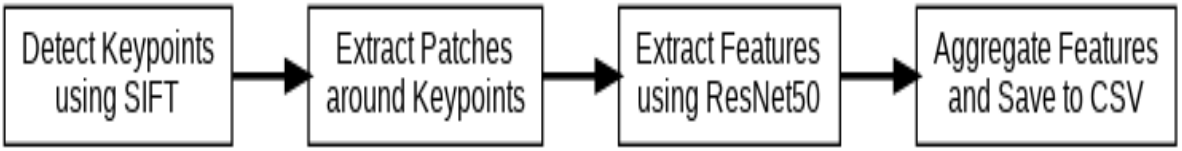


Figure 2-6: Feature extraction and representation

The primary goal of the process (Figure 2-6) is to extract meaningful features from images using a combination of traditional computer vision techniques (SIFT for keypoint detection and patch extraction) and deep learning (ResNet50 for feature extraction). These extracted features are then aggregated and stored in a structured format (CSV file). This feature extraction process aims to transform raw image data into a format that is suitable for downstream tasks such as image classification, object recognition, or further analysis in various applications of computer vision.

2.5. Architecture Design

Here we will go more in depth into the architectures we chose for the project, from their implementation to the fully connected layer adapted to the iris recognition and classification task.

2.5.1. ResNet50 design

In the Figure 2-7 below is the full architecture of ResNet50 with its 48 convolutional layers, a max pooling layer at first and an average pooling layer at last.

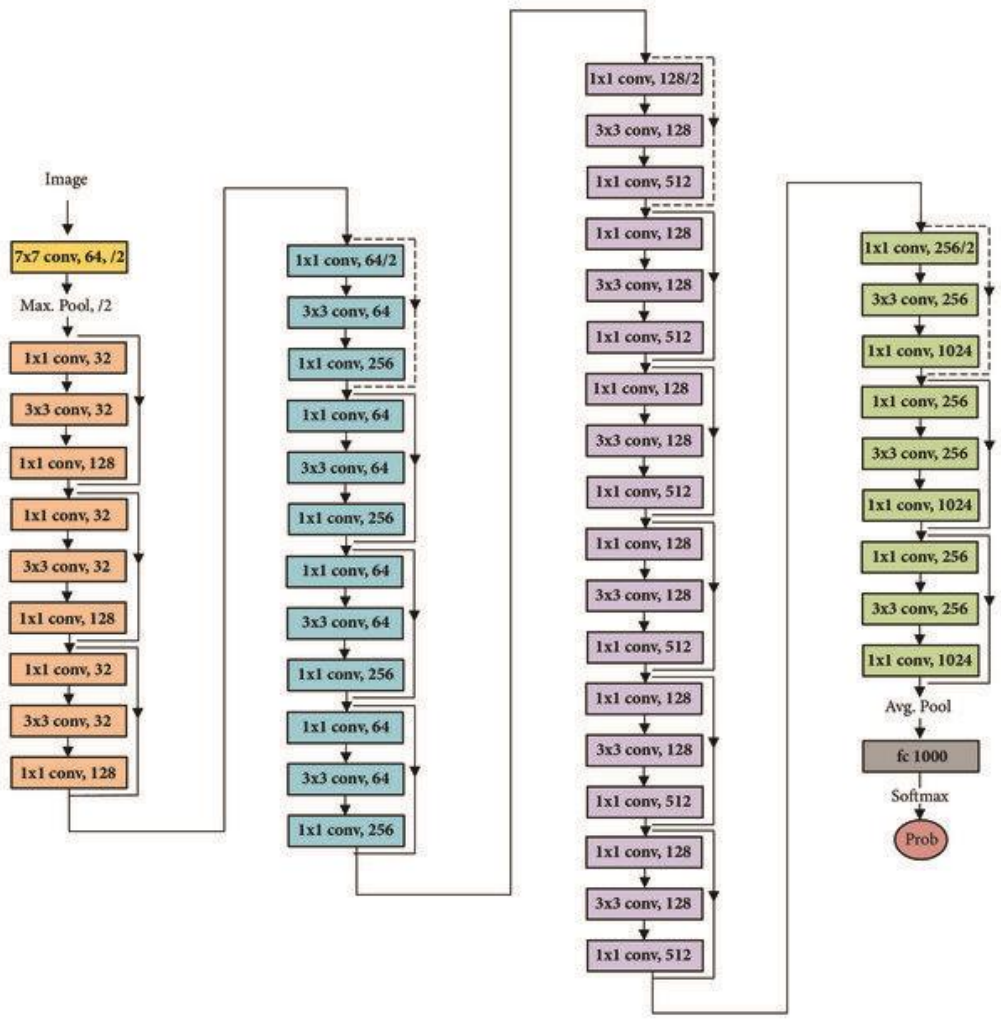


Figure 2-7: Full ResNet50 architecture [55]

2.5.1.1. ResNet50 from TensorFlow

The TensorFlow library is necessary in order to implement ResNet50, in the keras subsection there are various architectures and one of them is ResNet50,

While implementing the architecture we remove the top layer (fully connected) in order to replace it with our classification layer

2.5.1.2. ResNet50 with transfer learning

As mentioned previously we need the right pretrained model that fits our task in order to perform any kind of transfer learning.

We chose the ImageNet architecture due to its training on more than 14 million images, we freeze its weights in order to work with it as a feature extractor.

2.5.1.3. Implementing fully connected layer

The fully connected layer in our case will be a dense layer. But before we put two layers, global average pooling layer, and the dropout layer.

- **Global average pooling**

  The global average pooling is a pooling operation that condenses all the feature maps into one map, we use it in the fully connected to generate one feature map for each category of the classification task in the last layer [56].

- **Dropout layer**

  Dropout is a layer utilized in Deep neural network architectures for iris classification to prevent overfitting (Figure 2-8). By randomly deactivating a fraction of neurons during training, Dropout encourages the network to learn more robust features that generalize better to unseen iris images [57].

  In this case due to the robustness of ResNet50, it has more chances to overfit, the dropout layer therefore helps to prevent this overfitting.

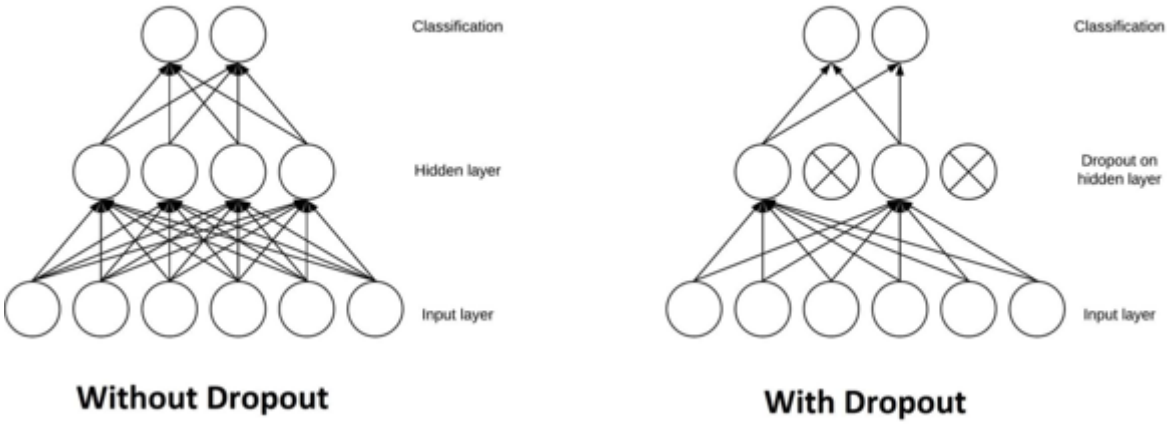


Figure 2-8: Dropout illustration [58]

- **Dense layer**

The dense layer, also known as the fully connected layer it is used to connect between all the previous layers to every neuron in the current layer [59].

This layer is necessary for our work in order to perform iris classification, which is what we want to do.

2.5.1.4. Training the fully connected layer with pre-processed data

Once we create our model for iris recognition and classification, we train the architecture using the pre-processed datasets as it mentioned in the figure bellow (Figure 2-9)

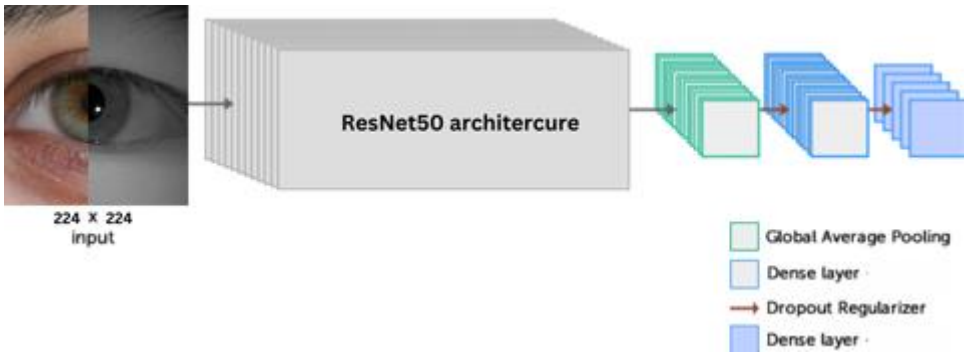


Figure 2-9: Graphic representation of ResNet50 based model

2.5.2. MobileNetV2 design

In Figure 2-10 we have the MobileNetV2 Architecture with along with the bottleneck layers.

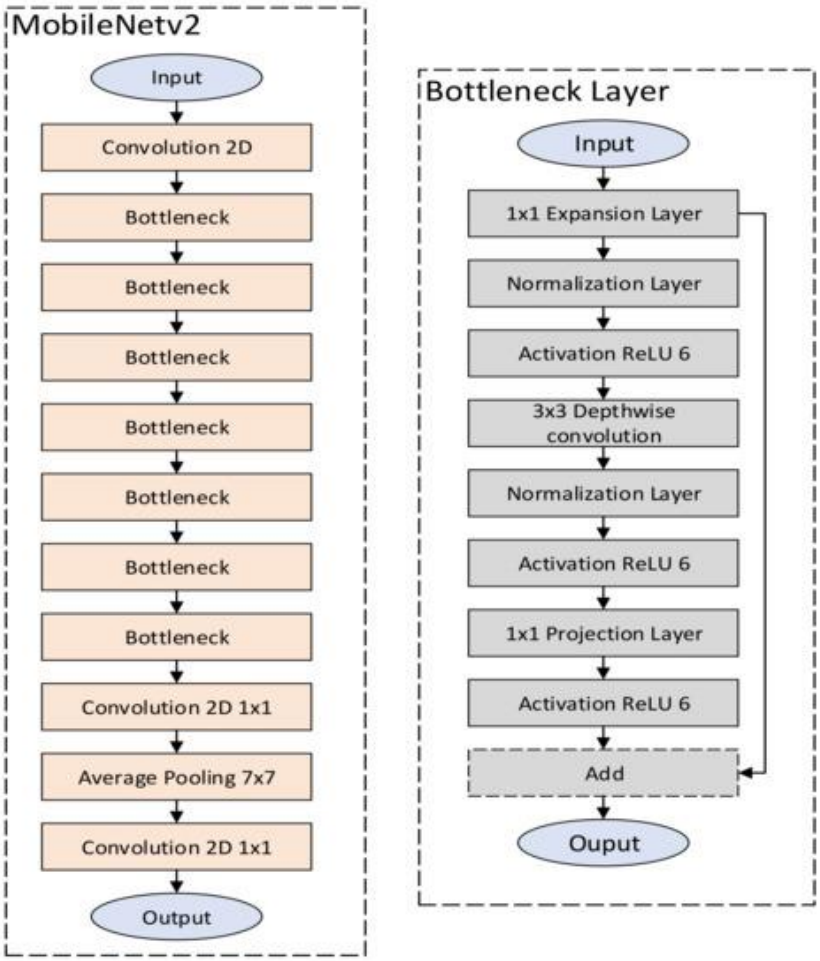


Figure 2-10: Full MobileNetV2 architecture with Bottleneck layer [47]

The MobileNetV2 contains a 2D convolutional layer at first 7 bottleneck layers and a convolutional layer after the bottleneck layers, an average pooling layer and another convolutional layer at the end.

The MobilNetV2 architecture contains two types of convolutional layers in order to reduce computational costs by reducing the number of parameters processed by the model [60].

Depthwise convolution: the depthwise convolutional layer processes each channel separately [40].

Pointwise Convolution: the pointwise convolutional layer combines the output of the depthwise convolutional layer across the channel [60].

2.5.2.1. MobileNetV2 from TensorFlow

Same as in ResNet50 the MobileNetV2 architecture is included in the Keras subsection of TensorFlow,

While implementing the MobileNetV2 architecture, we remove the fully connected layer in order to replace it with our classification layer

2.5.2.2. MobileNetV2 with transfer learning

Here as well we choose the ImageNet model in order to perform transfer learning for feature extraction, meaning we freeze the weights of the pretrained model.

2.5.2.3. Implementing fully connected layer

The fully connected layer for MobileNetV2 will be a classification layer composed of a global average pooling layer, a dropout layer even if MobileNetV2 is not as robust as ResNet50 but preventing overfitting is necessary, and a dense layer, as previously explained in section 2.5.1.3.

2.5.2.4. Implementing fully connected

Once we create our model for iris recognition and classification, we train the architecture (Figure 2-11) using the pre-processed datasets.

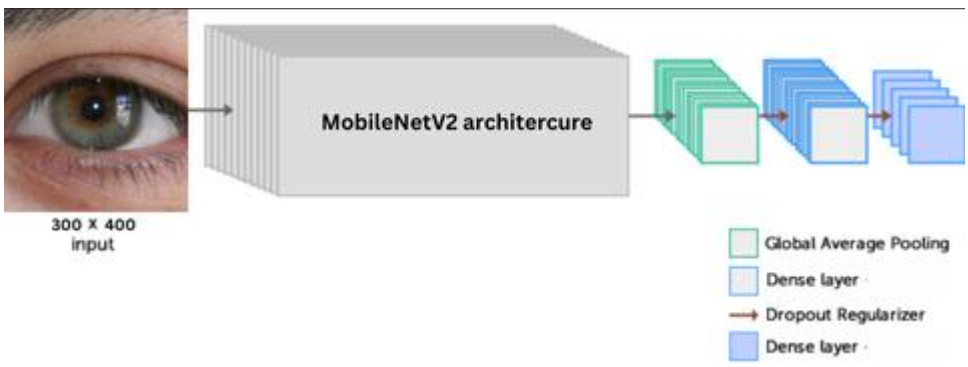


Figure 2-11: Graphic representation of MobileNetV2 model

2.6. Training Parameters

Training parameters important for the optimization of our results, These specific values either make our models work at optimal performances or make them fall off.

2.6.1. Batch size

A batch size is a training parameter that defines the number of samples passed to the neural network at once during the training [61],

In the datasets we have there are a lot of images so in order to optimize the training time and to have better classification, using a batch size is important.

2.6.2. Loss function

A loss function in neural networks is a function that measures how good the model's predicted outputs match with the true output labels [62].

In our case we will be using the Cross Entropy loss function which is a widely used loss function that calculates the difference between the true labels and the predicted probabilities

2.6.3. Epoch

An epoch in machine learning refers to one complete pass of the whole dataset through the algorithm or neural network, it is a hyperparameter that controls the training process of the model. Each epoch includes both forward and backward pass through the dataset [63].

2.6.4. Learning rate

The Learning rate is a training parameter that controls how much to change the model in response to the calculated error each time the model weights are updated [64].

Choosing the learning rate can be quite difficult as a too small value could result in a longer training process that could get stuck, however a value too big as well has a downside which is the learning of a sub-optimal set of weights results in a too fast or an unstable model behaviour.

2.7. Adam optimizer

Previously we said that the optimization methods are to improve performances, and this meant reducing the loss function by changing the model's parameters such as the weights or learning rate.

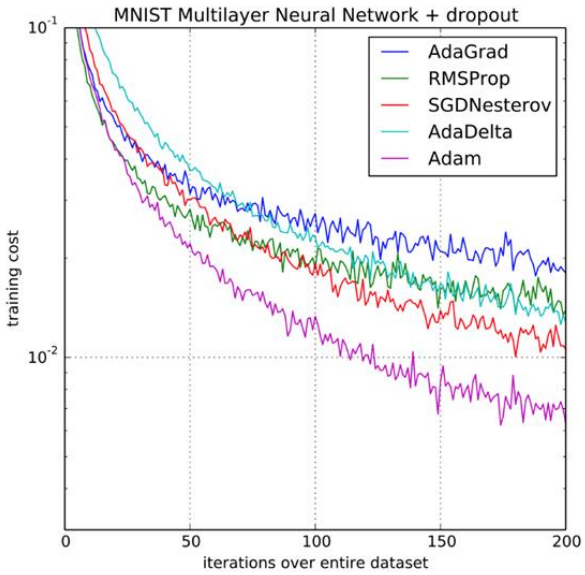


Figure 2-12: Comparison between Adam and other optimizers [65]

In Figure 2-12 we see that Adam is the most performant out of the optimizers presented so we use Adam as our optimizer for our deep learning models.

Conclusion

In this chapter we presented the DCNN models ResNet50 and MobileNetV2 and the datasets Casia-Iris-Thousand and UbirisV2, we will use for iris recognition.

In the next chapter their implementation and results will be shown, and their results will be discussed.

Chapter 3

Realization of iris recognition models

3. Realization of iris recognition models

3.1. Introduction

In this chapter after talking previously about the tools and architectures we will be using we are going to put all of this into work and present our work with precision and see the results of this one.

This chapter will contain 3 comparative studies the first one will be a comparison between 2 architectures (ResNet50 and MobileNetV2) to see which one is better when it comes to performances (accuracy and loss) to see which one is more performant.

And the third between of two identical models with the same dataset, but different pre-processing steps.

For the second comparison it will be between two identical models, with the same pre-processing steps, but with different datasets (Casia-Iris-Thousand and UbirisV2) and see which one will have the best performances according to the pre-processing steps.

And the third between of two identical models with the same dataset, but different pre-processing steps.

3.2. Overview of path to follow

This overview will give a better idea of the path to follow to get to the results we wish to obtain and do the comparative study between.

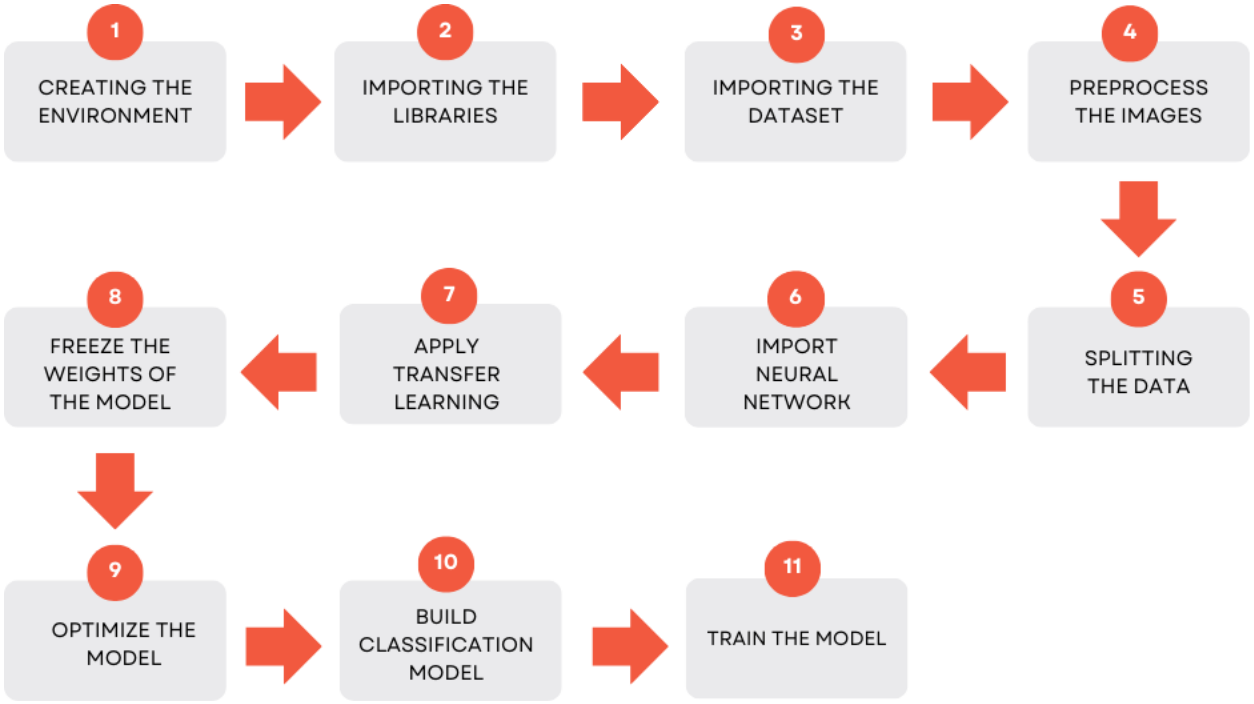


Figure 3-1: Path to follow

In figure 3-1 shows all the steps that we will go through in the making of our deep learning models to make it clear and organized

3.3. Creating the environments

In this section we will be showing the code for the creation of the environment and implementation of the libraries along with the setting of the paths to the databases:

```
from __future__ import print_function

import tensorflow as tf
import os
```

Figure 3-2: Importing libraires

```
import os

TRAIN_DIR = "/content/drive/MyDrive/PFE/UbirisV2_Dataset/Train_Dataset"
TEST_DIR = "/content/drive/MyDrive/PFE/UbirisV2_Dataset/Val_Dataset"
os.chdir(TRAIN_DIR)
```

Figure 3-3: Setting path to the databases

3.4. Splitting the data

This diagram explains how we divided the whole database to 2 sections

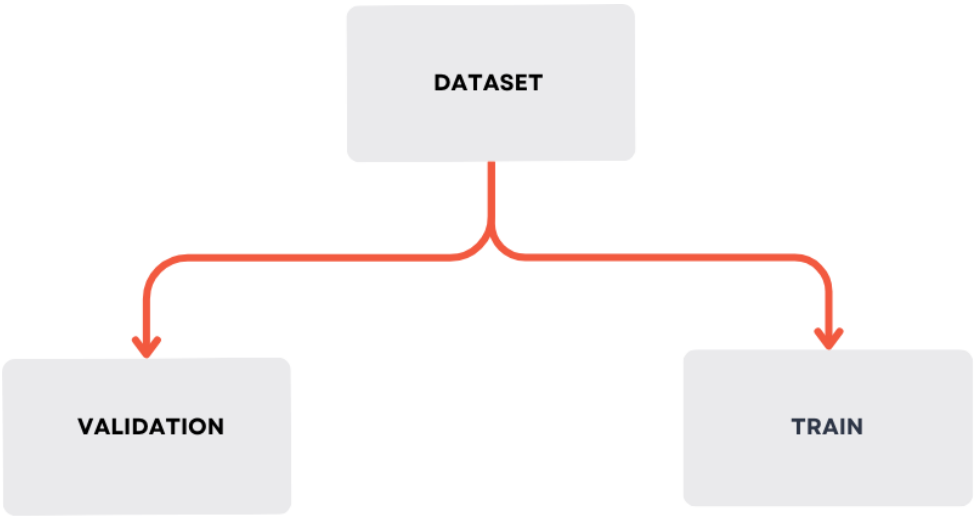


Figure 3-4: Splitting the data diagram

Either before or after the preprocessing we made sure to split the data in 2 parts one for the training (80%), and one for the validation (20%) to prevent any overfitting and also to make our models performances better overall.

3.5. Data preprocessing

As mentioned previously image preprocessing is an important step when talking about iris recognition, here we will be going in depth into the code used to preprocess the images

3.5.1. Preprocessing parameters

3.5.1.1. Image resizing

In this step we resize the images according to the MobileNetV2 as said previously the image height needs to be 400px and its width needs to be 300px. and for ResNet50 the height needs to be 224px and its width either needs to be 244px.

3.5.1.2. Normalize the data

```
[ ]  preprocess_input = tf.keras.applications.resnet.preprocess_input
     # equivalent to transforming the input with: x / 127.5 - 1 (also does the work of the 2 cells above)
```

Figure 3-5: Code of data normalization

3.5.1.3. Apply Gaussian blur

We use noise reduction especially for the UbirisV2 dataset since the images were taken outdoors unlike for Casia-Iris-Thousand, we need to reduce the noise occlusion caused by the sun.

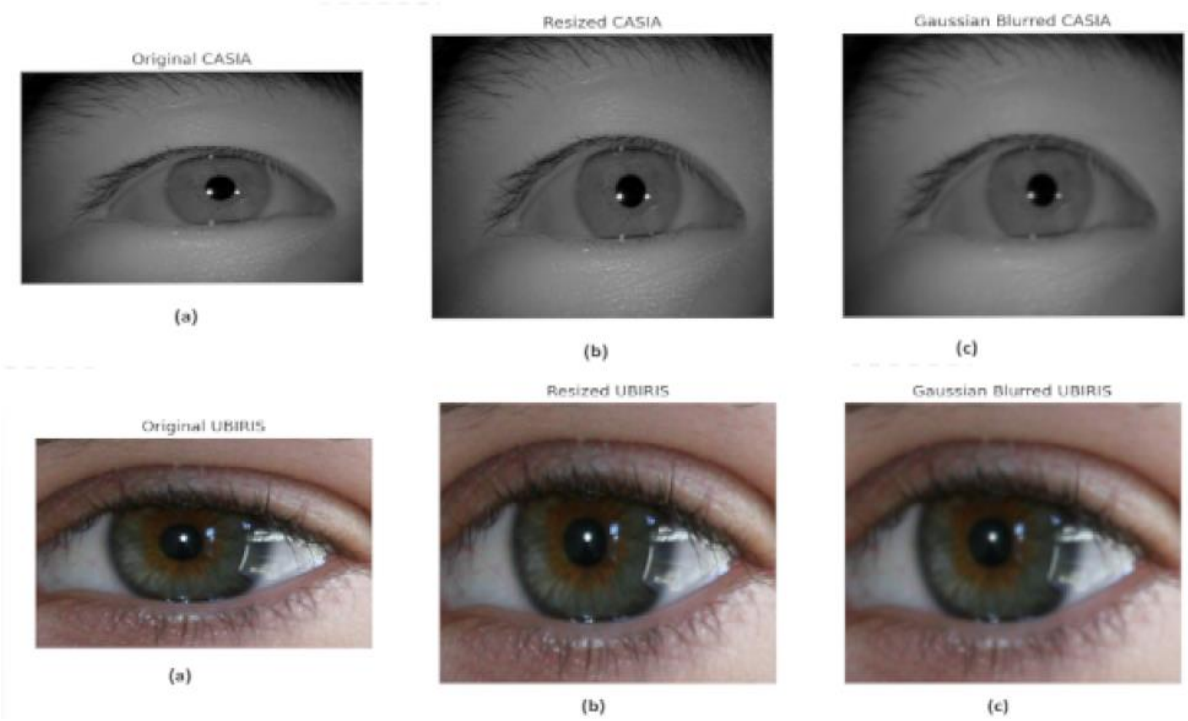


Figure 3-6: Result of applying preprocessing parameters

In Figure 3-6 we show the result of applying the preprocessing parameters in both of the datasets.

3.5.2. Eye Detection

3.5.2.1. Load Haar Cascade

The code bellow's (Figure 3-7) function is to download and load the Haar cascade, It is an XML file used for detecting eyes or parts of them.

```
# URL of the Haar cascade for eye detection
eye_cascade_url = "https://raw.githubusercontent.com/opencv/opencv/master/data/haarcascades/haarcascade_eye.xml"

# Download the Haar cascade XML file
urllib.request.urlretrieve(eye_cascade_url, "haarcascade_eye.xml")

# Load the Haar cascade for eye detection
eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_eye.xml')
```

Figure 3-7: Code of loading the Haar cascade

The first step when using Haar cascade is to convert the images to grayscale. Following this, we will set specific parameters as indicated in the upcoming table.

3.5.2.2. Haar Cascade parameters

This table lists the Haar cascade parameters used for eye detection in images from Casia-Iris-Thousand and UbirisV2 datasets.

Dataset	Casia-Iris-Thousand	UbirisV2
Scale Function	1.1	1.3
minNeighbors	5	5
minSize	(30,30)	/

Table 3.1: Haar cascade parameters for eye detection

We used on the Casia-Iris-Thousand parameters that result in moderate sensitivity, on the other hand we used higher sensitivity parameters on the UbirisV2 datase (Table 3.1)

3.5.2.3. Zooming on the eye

This code in Figure 3-8 iterates through each detected eye and crops the corresponding region from the image to zoom on the eye.

```
            # Zoom in on each detected eye
for i, (eye_x, eye_y, eye_width, eye_height) in enumerate(eyes):
    # Crop the eye region
    eye_region = image[eye_y:eye_y + eye_height, eye_x:eye_x + eye_width]
```

Figure 3-8: Zooming on the eye code

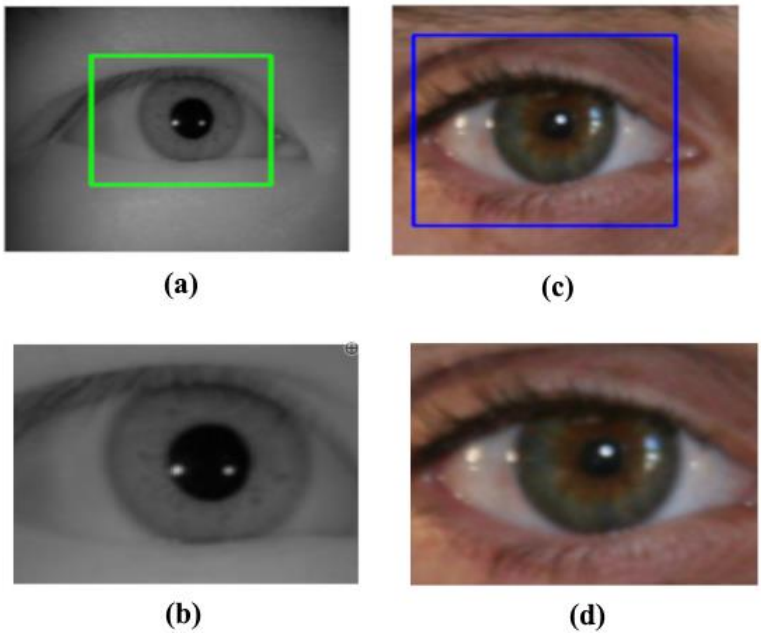


Figure 3-9: visualization of the detection and zoom in on the eye for CASIA (a, b) and Ubiris V2 (c, d)

In this Figure 3-9we can see that Haar cascade draw a boundary box in the images then we zoom in on eye to be the region of iris clearer.

3.5.3. Keypoint extraction by using the SIFT algorithm

After eye detection and the division of the results into train and Val set directories, now we'll work with the train path for detecting the keypoints from the eye images.

3.5.3.1. Creating the SIFT detection with its specific parameters

Based on the SIFT parameters in the Table 3.2 we create a SIFT detector

Parameters	Number of features	Contrast threshold	Edge threshold	Sigma
Values	1000	0.01	5	1.2

Table 3.2: SIFT parameters

3.5.3.2. Draw the keypoint on the images

Based on the previous parameters given to the SIFT algorithm is going to give us the keypoints.

```
image_with_keypoints = cv2.drawKeypoints(image, keypoints, None, flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
```

Figure 3-10: Code to draw keypoints

```
keypoint_counts.append(len(keypoints))
image_names.append(file)
```

Figure 3-11: Code to count keypoints for the histogram

In figure 3-11 The code written is made to have the keypoints detected in both datasets in a histogram for better visualization.

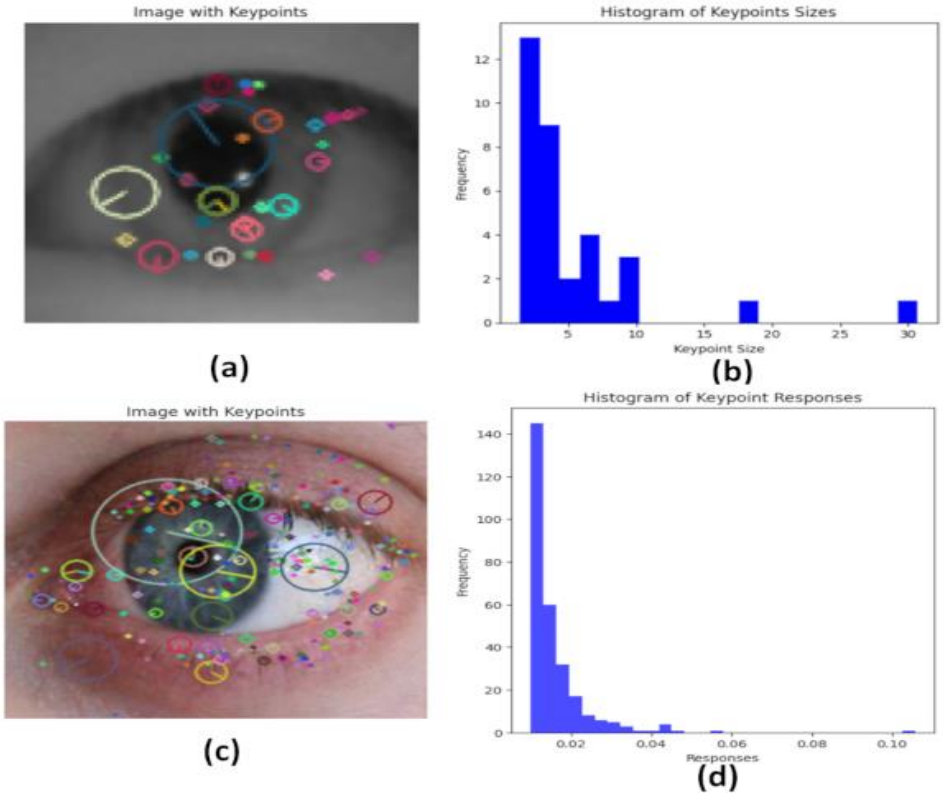


Figure3-12: Result of drawing keypoints and histogram for Casia-Iris-Thousand (a, b) and UbirisV2 (c, d)

The keypoints detected by SIFT vary across images in both datasets due to differences such as the presence of eyelashes. SIFT identifies the edges and corners of eyelashes, the pupil center, and the iris boundary as keypoints.

Despite these variations, the majority of keypoints are located in the iris region due to its distinct texture patterns and significant intensity changes in the iris

3.5.3.3. Extract patches

After detecting the keypoint by using the SIFT algorithm, now we need to extract patches around these keypoint, that helps in focusing on the most informative parts of the image; in this case, this part is the iris region.by following this step:

- Extract Square patches around keypoint from the image
- Resize patches to a specific size (224,244) pixel
- Convert to NumPy array
- Pre-process patches
    - Convert patches to ‘float32’
    - Normalize pixel values

3.5.3.4. Store features and manage them in a CSV file

After extracting the patches that contain features and labels, we need to store and manage them in a CSV file for further machine-learning tasks such as classification and recognition. and Due to the large size of our dataset, which requires considerable processing time, we use a batch size of 32. We aggregate the features and append them with labels to each batch before saving them to the CSV file.

```
# List to store features and labels temporarily
batch_features = []
batch_labels = []
```

Figure3-13: List to store features and labels temporarily

For each image that is extract features, we can have a CSV file containing the features extraction data as Figure 3.14:

	A	B	C	D	E	F	G	H	I
1	File,Label,Features								
2	S5000R09_eye_1.jpg,000,"[0.7080345749855042, 0.027905747294425964, 0.11183861643075943, 0.07592562586069107, 0								
3	.7994979619979858, 0.030491143465042114, 0.018268566578626633, 0.7737178206443787, 0.01807558909058571, 0.0204								
4	S5000R09_eye_1.jpg,000,"[0.6198136806488037, 0.2836475670337677, 0.07314325124025345, 0.0373939648270607, 0.11								
5	12200891599059 1, 0.1040290370583534 2, 0.21426844596862793, 0.013423117808997631, 0.008149778470396996, 0.01966								
6	S5000R09_eye_1.jpg,000,"[0.5050244927406311, 2.665626049041748, 0.6404591202735901, 0.0016653562197461724, 0.13								
7	, 0.7026134729385376, 0.20837043225765228, 0.0, 0.0010810821549966931, 0.29096174240112305, 0.0, 0.00183116213884								
8	S5000R09_eye_1.jpg,000,"[0.7002314925193787, 0.8233599066734314, 0.22641241550445557, 0.03984684497117996, 0.08								
9	6133346558, 7.910723686218262, 1.2086917161941528, 0.0017044453416019678, 0.0, 2.76456880569458, 0.1130919679995								
10	S5000R09_eye_1.jpg,000,"[0.44673025608062744, 0.14741337299346924, 0.05025206506252289, 0.2648704946041107, 0.5								
11	425789549946785, 2.8644392490386963, 0.04608374834060669, 0.023397738113999367, 0.0, 0.1481533646583557, 3.7235								
12	S5000R09_eye_1.jpg,000,"[0.23965628445148468, 0.09997010231018066, 0.14810696244239807, 0.1809227466583252, 0.1								
13	873, 0.23548221588134766, 0.15475066006183624, 0.11402146518230438, 0.8719303011894226, 0.22603105008602142, 0.(								
14	S5000R09_eye_1.jpg,000,"[0.29452335834503174, 0.08862189948558807, 0.122538261115551, 0.1764184534549713, 0.161								

Figure3-14:CSV file of features extraction

3.6. Implementing the architectures

In our work we have 2 architectures but we used the ResNet50 architecture 3 times for the same purpose but not with the same preprocessing or dataset.

3.6.1. Implementing ResNet50 architecture

In order to implement the ResNet50 architecture we use the TensorFlow library, we will be using It for feature extraction.

3.6.2. Implementing MobileNetV2 architecture

In order to implement the MobilenetV2 architecture we use TensorFlow library and write the code line: **tf. keras. applications. MobileNetV2**.

3.7. Transfer learning for feature extraction

As said previously to apply transfer learning we need to find the right architecture to build our pre-trained model.

In all our models we chose ImageNet (Figure3-15) that was the best choice since it is trained on more than 14 million images, therefore it was the best choice.

```
# Apply Transfer learning
IMG_SIZE = (IMG_HEIGHT, IMG_WIDTH)
IMG_SHAPE = IMG_SIZE + (3,)
base_model = tf.keras.applications.MobileNetV2(input_shape=IMG_SHAPE,
                                               include_top=False,
                                               weights='imagenet')
```

Figure3-15: Code of applying the transfer learning

As said previously freezing the weights will make the new layers the only trainable parameters on the new dataset given.

```
[ ]  # Extract features

     base_model.trainable = False
     base_model.summary()
```

Figure3-16: Code for freezing the weights

```
Total params: 2257984 (8.61 MB)
Trainable params: 2223872 (8.48 MB)
Non-trainable params: 34112 (133.25 KB)
```

```
Total params: 2257984 (8.61 MB)
Trainable params: 0 (0.00 Byte)
Non-trainable params: 2257984 (8.61 MB)
```

Figure3-17: Results of freezing the weights

As we can see freezing the weights made the trainable parameters (Figure3-17) go from 2223872 to 0 which means that we have successfully froze the weights.

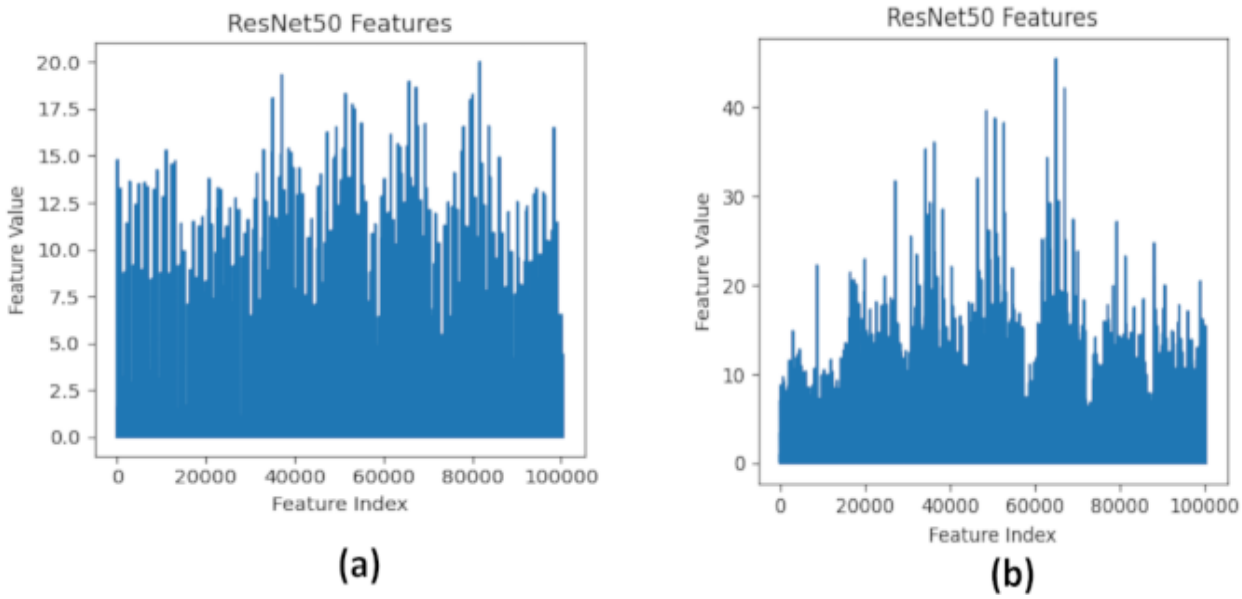


Figure3-18: Visualization of features extraction in Casia-iris-Thousand (a) and UbirisV2 (b)

The Figure3-18shows two plots related to the ResNet50 CNN model, visualizing the feature values learned by the model.

3.8. Building the classification layer

To create a classification layer, we need to first create a dense layer that will connect all the neurons to the previous layer's neurons.

We need to give also to this dense layer the number of classes we have in our case it is 50 Classes.as we can see in Figure 3-19 below :

```
# Classification layer
prediction_layer = tf.keras.layers.Dense(N_CLASSES)
prediction_batch = prediction_layer(feature_batch_average)
print(prediction_batch.shape)
```

Figure 3-19: Code for classification layer

3.9. Creating the model

Here we will present the model that will be training on the classification of the images and therefore recognizing the individuals and classifying them in the right classes.

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 400, 300, 3)]     0

 tf.math.truediv (TFOpLambd  (None, 400, 300, 3)       0
 a)

 tf.math.subtract (TFOpLamb  (None, 400, 300, 3)       0
 da)

 mobilenetv2_1.00_224 (Func  (None, 13, 10, 1280)      2257984
 tional)

 global_average_pooling2d (  (None, 1280)              0
 GlobalAveragePooling2D)

 dropout (Dropout)           (None, 1280)              0

 dense (Dense)               (None, 50)                64050

=================================================================
Total params: 2322034 (8.86 MB)
Trainable params: 64050 (250.20 KB)
Non-trainable params: 2257984 (8.61 MB)
```

Figure3-20: Model layers

In this code(Figure3-20) we set the inputs to their right size we defined above two variables IMG_HEIGHT and IMG_WIDTH as we showed above, they're the standard proportions for ResNet50 and MobileNetV2.

3.10. Setting training parameters

In order to optimize our model, we need to choose the right training parameters(Table 3.3) that will suit the model and make it work at its best performances.

Model	ResNet50_A	ResNet50_B	ResNet50_C	MobileNetV2
Batch size	32	32	32	64
Dropout	30	30	30	30
Learning rate	0.001	0.001	0.001	0.001
Epochs	20	20	20	20

Table 3.3: Training parameters

3.11. Compile the model

In order to compile our model, we chose to add an optimizer and a function to calculate the loss here is the code used for that in figure 3-21:

```
base_learning_rate = 0.001
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=base_learning_rate),
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

Figure 3-21**:** Code of compiling model

3.12. Results

After setting the training parameters and compiling the model we start the training by executing the program and wait for the training to end and plot the results using the seaborn library:

3.12.1. Result of ResNet50_A

ResNet50_A is the model that is using the SIFT algorithm and ResNet50 architecture for feature extraction and Casia-Iris thousand as its dataset

Figure 3-22**:** Accuracy and loss graphs for ResNet50_A

- **Accuracy and Loss**

We can in figure 3-22 see that the accuracy curve is increasing greatly from the 1st to the 15th epoch, then it increases normally until settling at its final value of 83.38%.

For the lost curve we observe that it is decreasing greatly from the 1st to the 15th epoch, then it is decreasing until its final value of 0.8323.

- **Validation accuracy and loss**

We can see that the Val_Accuracy curve increases greatly from the beginning to the 6th epoch then decreases a little bit, in the next epochs it is having the same pattern until it starts to settle at its final value of 80.30%.

We see that the Val_Loss has nearly the opposite behavior of the Val accuracy except for sudden increases, and at the end it settles at its final value of 0.8030.

The Table 3.4 show the parameters with it result (Value) for ResNet50_A

Parameters	Value
Accuracy	83.38%
Loss	0.5500
Val_Accuracy	80.30%
Val_Loss	0.8030

Table 3.4:Results of ResNet50_A

3.12.2. Result of ResNet50_B

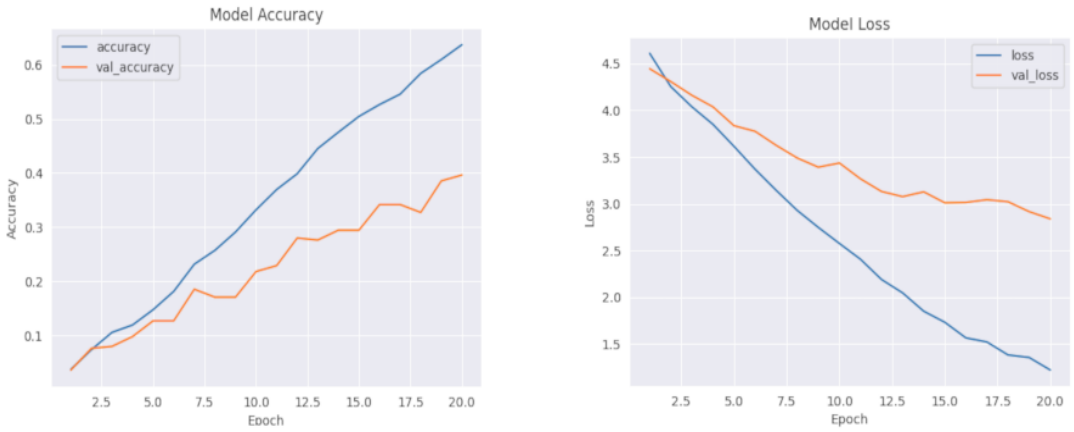


Figure 3-23: Accuracy and loss graphs for ResNet50_B

- **Accuracy and Loss**

We can see in figure 3-23 that the accuracy curve increases greatly from beginning to end without until settling at its peak value of 63.69%.

The loss curve has the opposite behavior to the accuracy's, it decreases greatly throughout all epochs to settle at its lowest value of 1.2223.

- **Validation accuracy and loss**

We can see that the Val_Accuracy curve is increasing, settling, and sometimes decreasing throughout all epochs and at the end it ends up settling at 39.64%

The Val_Loss curve in figure 3-23 has an opposite behavior to the Val_Accuracy, and in some epochs (10, 14) we see that it increases which means that there is a slight overfitting in the model. The Table 3.5 show the parameters with it result (Value) for ResNet50_B

Parameters	Values
Accuracy	63.69%
Loss	1.2223
Val_Accuracy	39.64%
Val_Loss	2.8391

Table 3.5:Results of ResNet50_B

3.12.3. Result of ResNet50_C

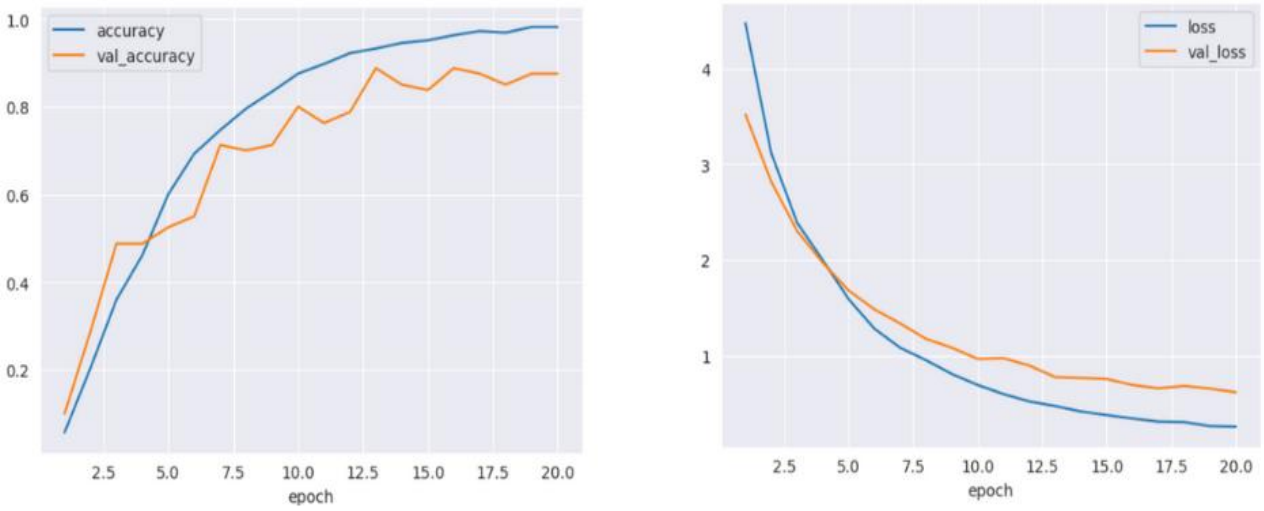


Figure 3-24**:** Accuracy and loss graphs for ResNet50_C

- **Accuracy and Loss**

We can in figure 3-24 that the accuracy grows greatly from the 1st to the 10th epoch, then it starts slowly settling when it gets to 18th epoch to get finally to its last value of 96,96%.

We can see that the loss curve is decreasing greatly from the 1st to the 12th epoch then it slowly gets lower from there to the 18th epoch then it settles to its final value of 0,2558.

- **Validation accuracy and loss**

We can see that the Val_Accuracy curve starts to increase greatly from the 1st to the 3rd epoch then settles for an epoch and then continues in the same process of growing and settling and sometimes decreasing a little at the end settling to its final value of 86,25%

When it comes to the Val_Loss we see that in most epochs the loss is decreasing just like the loss but in the 15th epochs we see the loss slightly increasing which means that there is a slight overfitting in the model.

The Table 3.6show the parameters with it result (Value) for ResNet50_C

Parameters	Values
Accuracy	96.96%
Loss	0.2558
Val_Accuracy	86.25%
Val_Loss	0.6268

Table 3.6: Results of ResNet50_C

3.12.4. Result of MobileNetV2

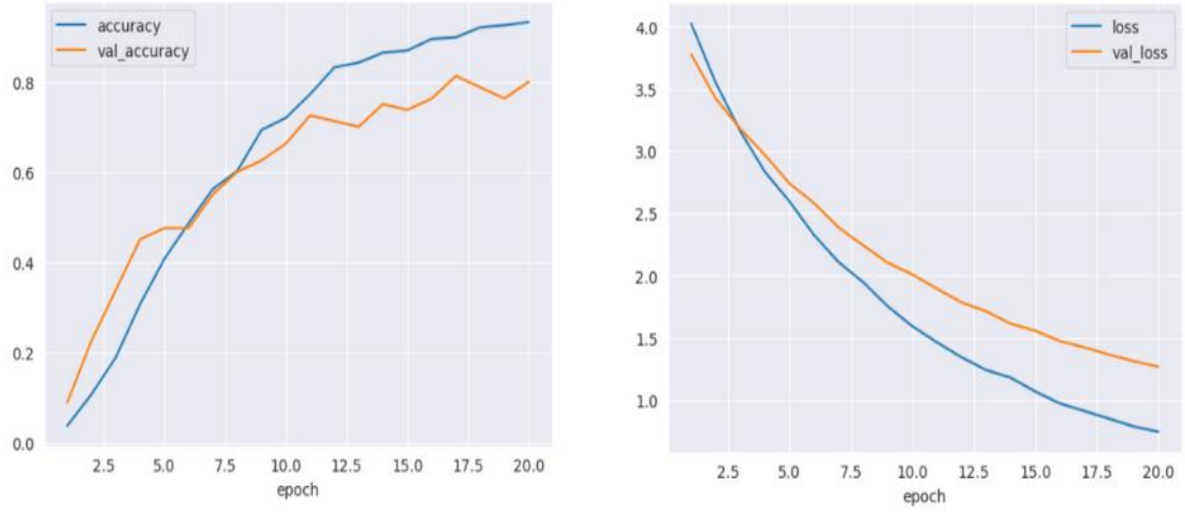


Figure 3- 25: Accuracy and loss graphs for MobileNetV2

- **Accuracy and Loss**

We can see that the accuracy curve increases greatly from the 1st to the 6th epoch then increases at a normal pace until reaching the 18th epoch then starts settling to its final value of 93.72%

For the loss curve we notice that the curve greatly decreases from beginning to end to settle at its final value of 0.7474.

- **Validation accuracy and loss**

We can see that the Val_Accuracy is growing greatly from the 1st to the 3rd epoch then starts settling for 3 epochs and then grows significantly until reaching the 11th epoch, for the next epochs it keeps increasing then decreasing until settling at the value of 80%.

We can see that the Val_Loss decreases greatly throughout all epochs to finally settle at its final value of 1.2664.

The Table 3.7 show the parameters with it result (Value) for MobileNetV2

Parameters	Values
Accuracy	93.72%
Loss	0.7474
Val_Accuracy	80%
Val_Loss	1.2664

Table 3.7:Results of MobileNetV2

3.13. Discussion

In this section we will be having comparisons, one between 2 neural network architectures that are ResNet50 and MobileNetV2, the second between the same dataset but with different preprocessing steps and the third will be between 2 databases Casia-Iris-Thousand and UbirisV2. in all of three comparison we will focus on the performance result in each one of them.

Model	Accuracy	Loss	Validation_accuracy	Validation_Loss
ResNet50_A	83.38%	0.5500	80.30	0.8030
ResNet50_B	63.69%	1.2223	39.64%	2.8391
ResNet50_C	96.96%	0.2558	86.25%	0.6268
MobileNetV2	93.72%	0.7474	80%	1.2664

Table 3.8: Results of all models

3.13.1. Comparison between ResNet50_C and MobileNetV2

Since this is a comparison between 2 architectures, we will mostly focus on the performances of both architectures, which means the accuracy, Val_Accuracy, loss, and Val_Loss and see which one is better overall.

- **Accuracy**

We will start first by comparing the accuracies to see which one is more precise.

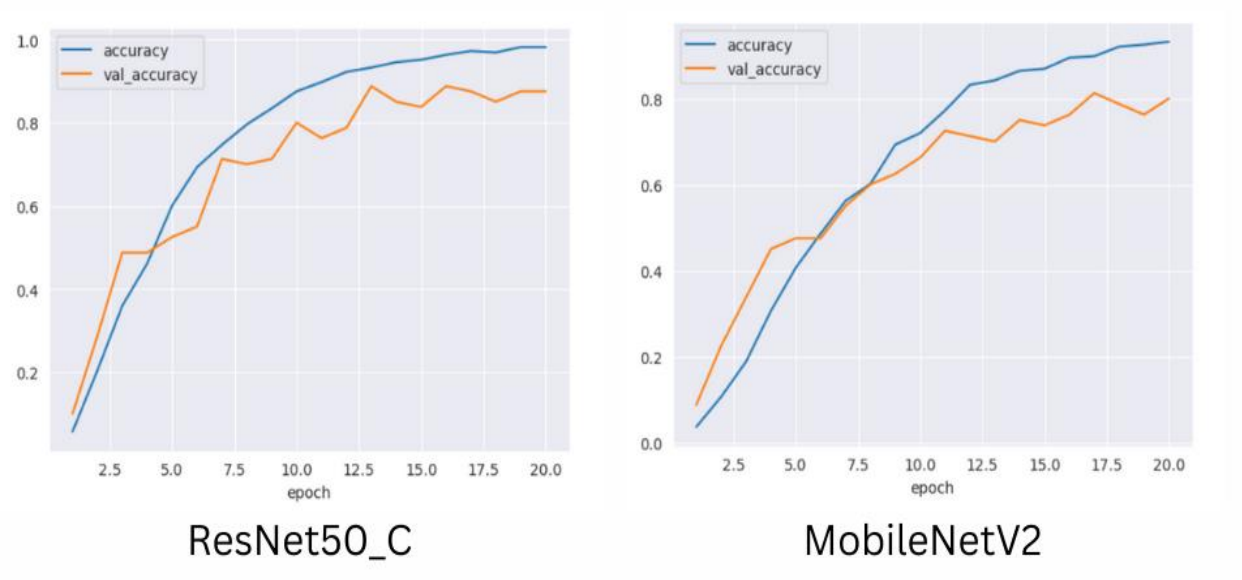


Figure 3-26: Comparison of the accuracies of ResNet50_C and MobileNetV2

Based on what we saw before and comparing the results side by side we can see clearly that the ResNet50_C architecture is better when it comes to both its accuracies (accuracy, Val_Accuracy), which means for precision the ResNet50_C architecture comes out on top.

- **Loss**

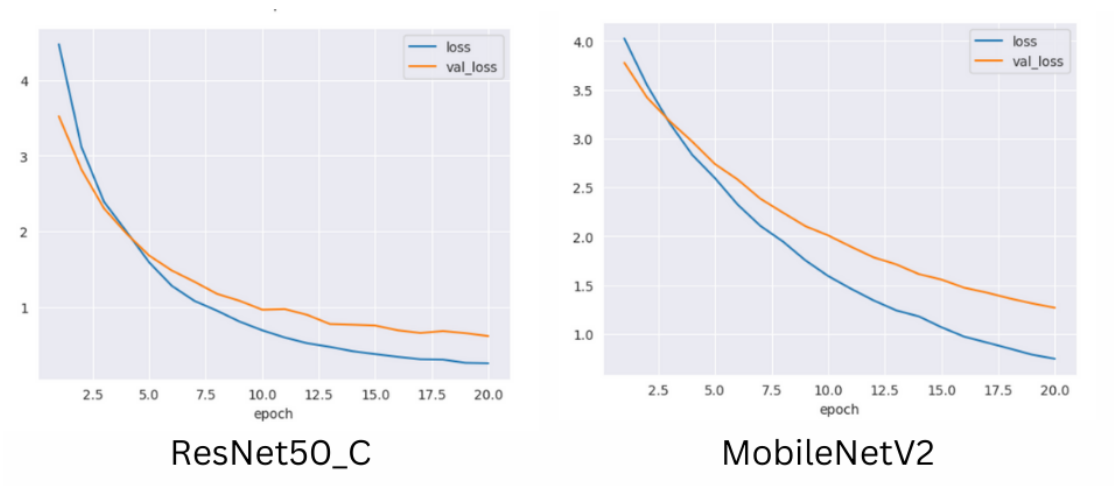


Figure 3-27**:** Comparison of the losses of ResNet50_C and MobileNetV2

Also, here we can see that when it comes to the loss, ResNet50 is indeed better, we see that in the result the MobileNetV2's loss is nearly double that of the other.

However, we can observe a slight overfitting of the ResNet50 architecture which is not the case for the MobileNeV2's architecture.

- **Conclusion of comparison**

Overall, the ResNet50 architecture comes out on top in all the parameters analysed which is not really a surprised it is a more rigid architecture with more layers and more complexity to implement, when in the other hand MobileNetV2 is more adapted to embedded systems or phones applications where it can be implemented and perform pretty well, and for the ResNet50 architecture it is better suited for bigger systems that need a lot of performances to be implemented in.

3.13.2. Comparison with ResNet50_B and ResNet50_C

We will compare between two performances of ResNet**50**_B that's is training based on a lot of preprocessing steps such as resizing, normalization, Gaussian blur, eye detection, and keypoint detection, and the ResNet**50**_C that we just apply two preprocessing parameters is image resizing and data normalization.

- **Accuracy**

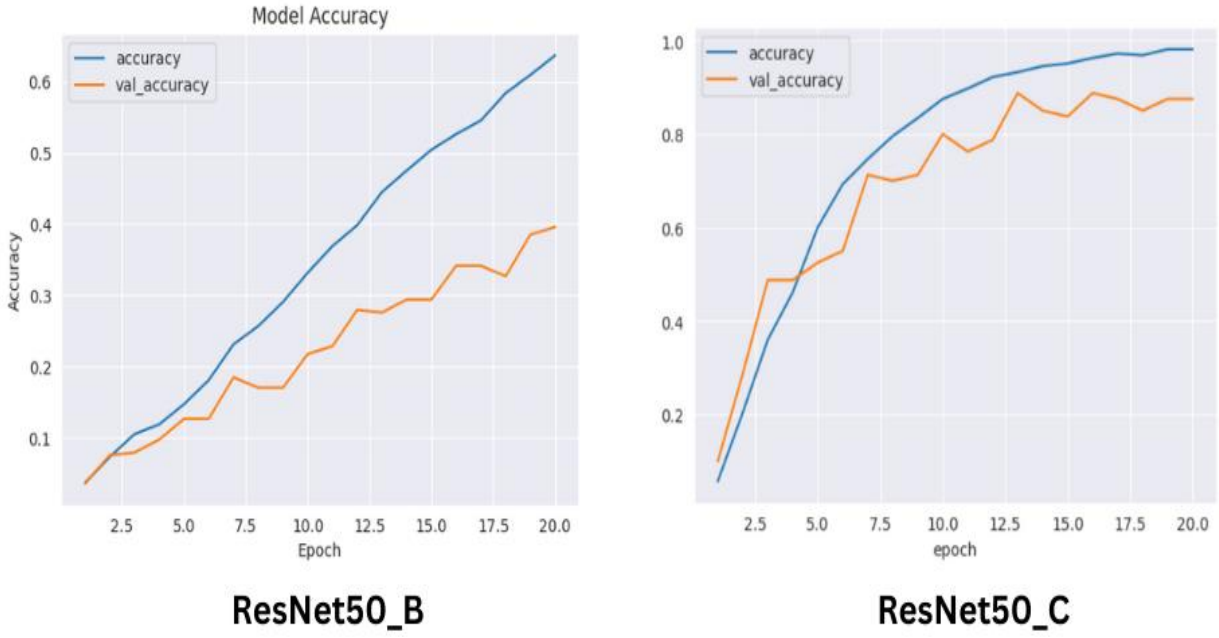


Figure3-28: Comparison of the accuracies of ResNet50_B and ResNet50_C

ResNet50_B shows rapid initial improvement in training accuracy, levelling off at around 0.8, with validation accuracy plateauing lower, indicating some overfitting. In contrast, ResNet**50**_C also demonstrates rapid initial training accuracy improvement but stabilizes at a higher level close to 1.0, with validation accuracy converging more closely to training accuracy, suggesting improved generalization and less overfitting.

- **Loss**

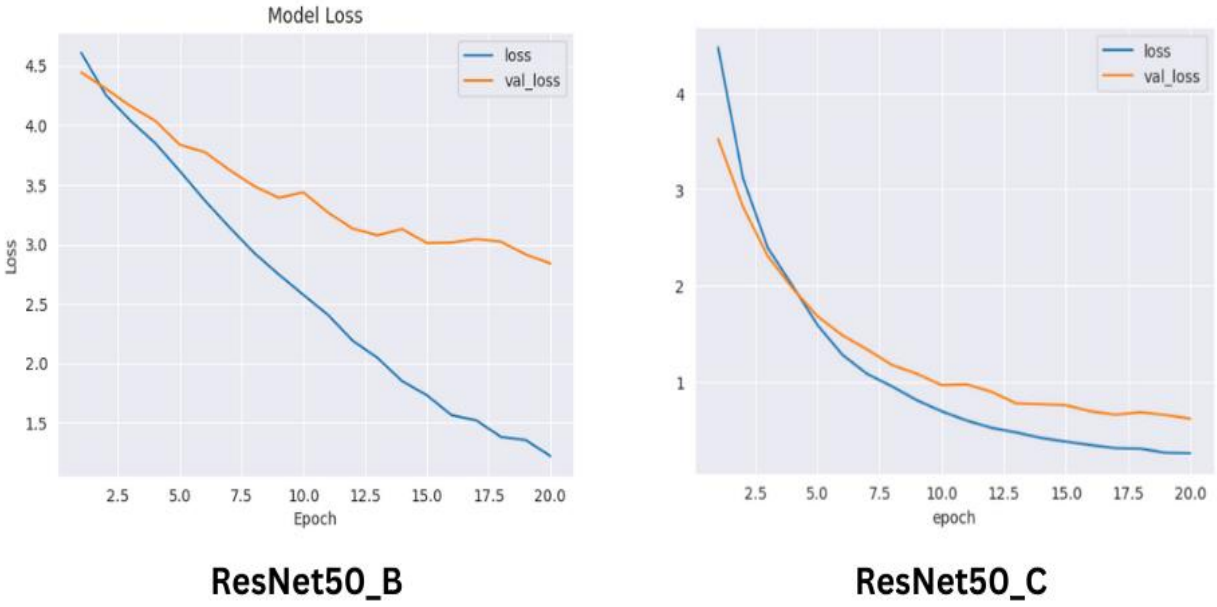


Figure3-29: Comparison of the loss of ResNet50_B and ResNet50_C

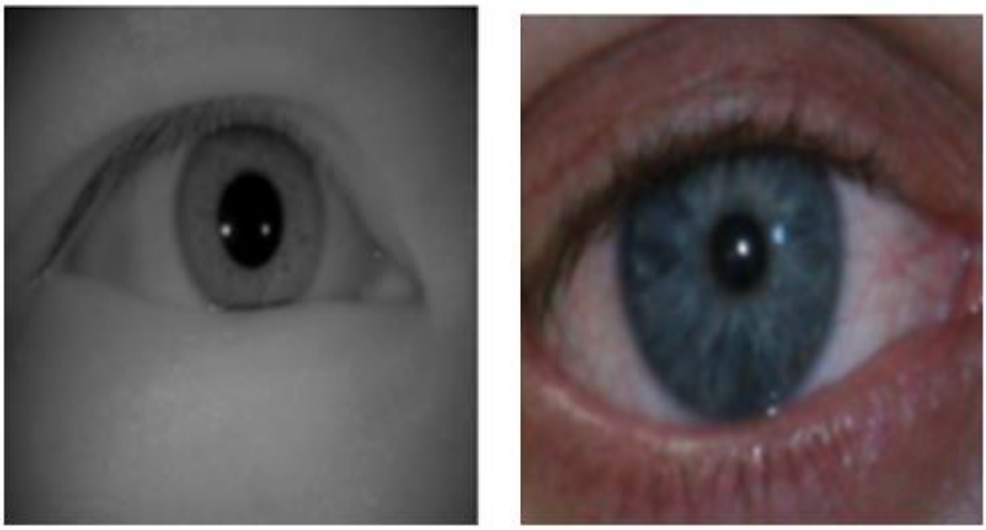
In the ResNet50_B, the model shows rapid initial improvement in training loss but struggles with overfitting, as indicated by the slower decrease and higher validation loss. In contrast, the ResNet50_C, depicted in the second image, exhibits smoother and more closely aligned training and validation loss curves, suggesting better generalization and less susceptibility to overfitting.

Based on the result that we have and the comparison that we make, we conclude that the more preprocessing steps doesn't mean a better performance, the opposite as in this case the accuracy and the loss result is clearly different when we apply more of the preprocessing functions eve if there's a result in keypoints and features extraction.

3.13.3. Comparison between ResNet50_A and ResNet50_B

For this comparison it will be between two datasets (Casia-Iris-Thousand) using the same architecture and preprocessing steps for both of them, to see which one will give us better features to extract and also more importantly better performances overall.

- **Preprocessing of the dataset**



(a) (b)

Figure 3-30: image preprocessing for Casia-Iris-Thousand (a) and UbirisV2 (b)

Our datasets share some preprocessing steps and parameters, such as resizing, noise reduction, normalization, and eye detection.

As seen in the (Figure3-30) Casia-Iris-Thousand pre-processed images(a) maintain a consistent appearance with minimal noise, while UbirisV2 pre-processed images (b), although improved, show residual contrast and noise.

These differences in results are due to the varying data collection environments. where the Casia-Iris-Thousand images were captured in controlled indoor environments with consistent illumination and imaging conditions. In contrast, the UbirisV2 dataset images were captured in uncontrolled outdoor environments with varying natural lighting.

- **Feature extraction**

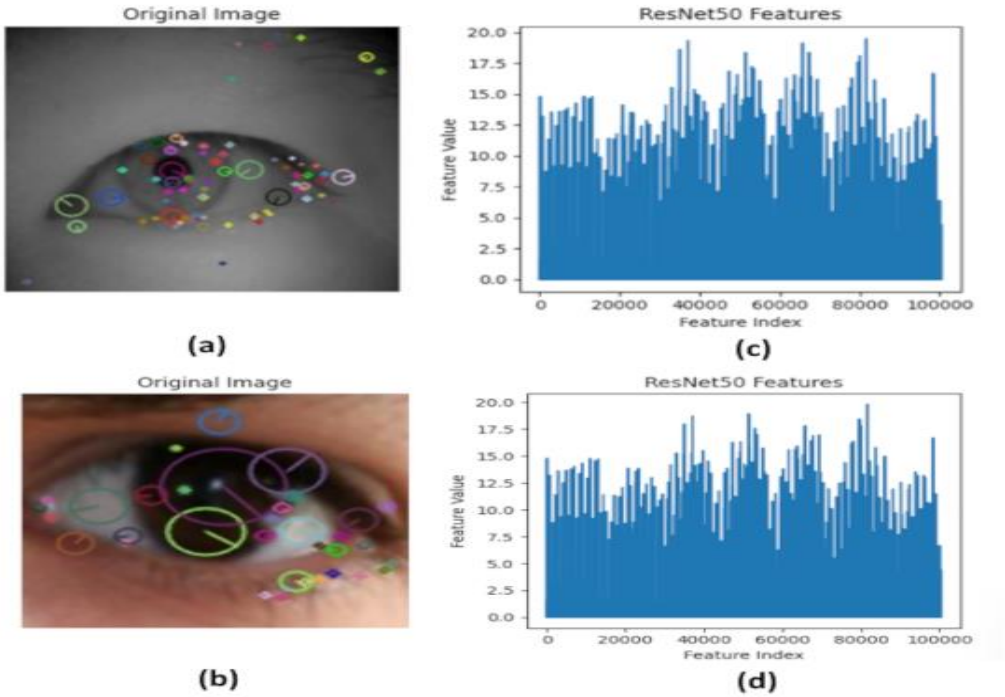


Figure 3-31: Visualization of keypoint images and Histogram of ResNet50 features for Casia-Iris-Thousand (a, c) an UbirisV2 (a, d)

Keypoint detection using the SIFT algorithm and feature extraction yielded varying results across the two datasets as we seen in the (Figure 3.30) below:

- In the Casia-Iris-Thousand dataset, keypoints were more consistent and concentrated within the iris region (image (a)).
- while the UbirisV2 dataset presented challenges with a broader distribution of keypoints (image (b)).

The feature extraction process using the pre-trained ResNet50 model was consistent across both datasets. The same ResNet50 architecture, pre-trained on the ImageNet dataset, was used to extract features from patches around the detected keypoints in the iris images. However, the visual representations of the extracted features, as shown in (Figure 3.30) indicate differences in quality and characteristics between the datasets:

- For the Casia-Iris-Thousand dataset (image (c)), the feature visualizations are more consistent and exhibit uniform patterns, indicating distinctive and representative iris features.
- In contrast, the UbirisV2 dataset (image (d)) displays higher diversity and variation in patterns, reflecting the complexities of uncontrolled imaging conditions.

Consequently, the features obtained from the Casia-Iris-Thousand dataset are more distinctive and representative of the iris patterns, whereas the features extracted from the UbirisV2 dataset are more complex and varied. due to these reasons:

- The Casia-Iris-Thousand dataset images are in gray-scale, while the images of the UbirisV2 dataset are RGB which are mismatching whit the SIFT parameters.
- Differences in preprocessing outcomes.
- **Classification performances**

All of the previous comparisons can make a difference in the model training where the training loss and accuracy curves for both datasets, provide insights into the classification performance achieved using the extracted features and the ResNet50 model with a new classification layer.

- For the Casia-Iris-Thousand dataset (Figure 3.22), the training loss steadily decreased, and the accuracy curve reached approximately 0.99 (99%) on the training set. This high training accuracy indicates that the model effectively learned the patterns and features present in the Casia-iris-thousand dataset.
- In contrast, for the UbirisV2 dataset (Figure3.23), while the training loss also decreased, the accuracy curve plateaued at around 0.95 (95%) on the training set. This slightly lower training accuracy, suggests that the model encountered more challenges in learning the patterns and features from the UbirisV2 dataset.

However, the lower training accuracy achieved on the UbirisV2 dataset aligns with the more varied and complex features extracted, as discussed in the feature extraction comparison. The challenging imaging conditions and the diverse features present in the UbirisV2 dataset may have made it more difficult for the model to learn robust and discriminative representations, leading to a slightly lower classification performance compared to the Casia-Iris-Thousand dataset.

- **Conclusion of Comparison**

In conclusion, the Casia-Iris-Thousand dataset outperforms the UbirisV2 dataset due to its controlled imaging conditions, which result in more consistent pre-processing, keypoint detection, and feature extraction. Casia-Iris-Thousand's grayscale images and stable environment lead to more reliable keypoints and distinctive iris features. This consistency enables higher classification accuracy, reaching approximately 83% compared to UbirisV2's 64%. The UbirisV2 dataset, with its varied outdoor conditions and complex RGB images, presents greater challenges, resulting in less effective keypoint detection and feature extraction, ultimately impacting classification performance.

3.14. Conclusion

As a conclusion of our work in this chapter we can say first that the preprocessing of the data is an important matter but it doesn't necessarily mean better performances.

For our first comparison between our two architectures, we saw that more rigidity truly means better performance but also means more complexity which has its downsides as we saw a slight overfitting in the ResNet50 architecture, on the other hand a more lightweight architecture means less complexity so no overfitting which is also something the architecture's layers work on and also more deployable in many hardware devices like mobiles or embedded systems.

For our second comparison we used the architecture with better performances on two datasets one with black and white pictures and one with coloured pictures which we applied to a specific preprocessing to see which one reacts better to it, to see also which one will give more features, and more importantly which will have the best performances.

In the end we ended up having the results and did the both comparisons and did the analysis which gave us an answer to which of the architectures is better, and which of the two datasets is better to work with.

General conclusion

In the beginning after doing some research we decided to make a comparative study, due to the variety of work done in the field of iris recognition using deep learning and also the variety of work we have done on our own, our problematic initially was iris recognition using ResNet50 then became which of these various architectures is better and which dataset is more suited for it , effectively large eye datasets Casia-iris thousand and UbirisV2 we used did not have the same resolution, neither the same eye background nor the position in the images. This greatly impacted the results therefore preprocessing the datasets was crucial.

This led us to study the iris specific features that make differences between individuals. We decided to create new databases from Casia-Iris-Thousand, and UbirisV2, by applying the SIFT algorithm, that extracted keypoints from eye images.

We named the pretrained model ResNet50 using different databases and preprocessing: ResNet50_A, ResNet50_B, ResNet50_C

We trained and dismissed other preprocessing tools either because they were less efficient (SURF) or too time consuming (labelme)

When comparing the performances of ResNet50_A, ResNet50_B, ResNet50_C, MobileNetV2, ResNet50_C came out on top with 96.96% accuracy.

ResNet50 based architecture proved to be robust, and MobileNetV2 embedded.

Furthermore work should consider improving the accuracy of the MobileNetV2 models as embedded applications on IOT's for real time applications that are greatly.

Datasets, preprocessing is another important parameter, since it has a high influence on the training process, prediction and classification. In particular preprocessing tools for effective iris segmentation should be developed.

Bibliographical References

[1] Heckathorn, D. D. Respondent-Driven Sampling: a new approach to the study of hidden populations. Social Problems, 44(2), 174–199. https://doi.org/10.2307/3096941. (1997)

[2] Dantcheva, A., Elia, P., & Ross, A. What else does your biometric data reveal? A survey on Soft Biometrics. IEEE Transactions on Information Forensics and Security, 11(3), 441–467. https://doi.org/10.1109/tifs.2015.2480381. (2016)

[3] Cintas, C., and others. Automatic ear detection and feature extraction using Geometric Morphometrics and convolutional neural networks. IET Biometrics, 6(3), 211–223. https://doi.org/10.1049/iet-bmt.2016.0002. (2017)

[4] Mostofa, M. Generative Adversarial Network and its application in aerial vehicle detection and biometric identification system. https://doi.org/10.33915/etd.11758. (2023)

[5] Quinn, J., McEachen, J., Fullan, M., Gardner, M., & Drummy, M.. Dive into deep learning: Tools for Engagement. Corwin Press. (2023).

[6] Alan Gelb , Anit Mukherjee and Anna Diofasi. Iris Recognition: better than fingerprints and falling in price. https://www.cgdev.org/blog/iris-recognition-better-fingerprints-and-falling-price. (2016).

[7] Moazed, K. T. The Iris: Understanding the Essentials. Springer Nature. (2020)

[8]https://www.researchgate.net/figure/Complexity-and-uniqueness-of-human-iris-Fine-textures-on-the-iris-form-unique-biometric_fig3_315477410

[9]Wayman, J. L., Jain, A. K., Maltoni, D., & Maio, . Biometric systems: Technology, Design and Performance Evaluation. Springer Science & Business Media. (2005b).

[10] De Telecomunicações, I.-. I. (n.d.). Iris recognition: Analysis of the error rates regarding the accuracy of the segmentation stage. IT - Instituto De Telecomunicações. https://www.it.pt/Publications/PaperJournal/3831

[11] Ratha, N. K., Connell, J. H., & Bolle, R. M. Enhancing security and privacy in biometrics-based authentication systems. IBM Systems Journal, 40(3), 614–634. https://doi.org/10.1147/sj.403.0614. (2001b)

[12] S., Sujanthi. (2023). Iris Liveness Detection using Deep Learning Networks. 1188-1196. doi: 10.1109/ICOEI56765.2023.10125665

[13] https://www.frandroid.com/marques/samsung/381623_zoom-capteur-diris-samsung-galaxy-note-7

[14] Short papers of the 10th Conference on Cloud Computing, Big Data & Emerging Topics. In Facultad de Informática (UNLP) eBooks. https://doi.org/10.35537/10915/139373. (2022)

[15] https://asp-eurasipjournals.springeropen.com/counter/pdf/10.1155/2010/938737.pdf

[16] Iakovidis, D. K., Ooi, M., Kuang, Y. C., and others. Roadmap on signal processing for next generation measurement systems. Measurement Science & Technology, 33(1), 012002. https://doi.org/10.1088/1361-6501/ac2dbd. (2021).

[17] Rathgeb, C., Uhl, A., & Wild, P. Iris Biometrics: From Segmentation to Template Security. Springer Science & Business Media. (2012b)

[18] Zwitter, A., Gstrein, O. J., & Yap, E. Digital identity and the blockchain: universal identity management and the concept of the "Self-Sovereign" individual. Frontiers in Blockchain, 3. https://doi.org/10.3389/fbloc.2020.00026. (2020)

[19] Bodade, R. M., & Talbar, S. N. Iris analysis for biometric recognition systems. (2014)

[20] https://www.sciencedirect.com/science/article/abs/pii/S0165168412002630

[21] Bowyer, K. W., & Burge, M. J. Handbook of iris recognition. Springer. (2016)

[22] https://www.mdpi.com/1424-8220/23/4/2238

[23] A novel iris recognition system based on active contour. IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/5704946. (2010, November 1)

[24]https://cvrl.nd.edu/projects/?project_name=Iris%20Segmentation%20using%20Geodesic%20Active%20Contours%20and%20GrabCut

[25] Bostrom, N. *Superintelligence: Paths, Dangers, Strategies*. Oxford University Press, USA. (2014)

[26]https://www.slideteam.net/artificial-intelligence-ai-machine-and-deep-learning-layers-powerpoint-topics.html

[27] Quinn, J., McEachen, J., Fullan, M., Gardner, M., & Drummy, M. Dive into deep learning: Tools for Engagement. Corwin Press. (2019b)

[28] Brownlee, J. Deep learning for computer vision: Image Classification, Object Detection, and Face Recognition in Python. Machine Learning Mastery. (2019)

[29] https://ai.plainenglish.io/a-hands-on-guide-to-document-image-classification-daed233ebeed

[30]https://www.researchgate.net/figure/Application-dune-convolution-sur-une-image-96_fig4_369170743

[31] Li, Z., Wang, S., Fan, R., Cao, G., Zhang, Y., & Guo, T. Teeth category classification via seven-layer deep convolutional neural network with max pooling and global average pooling. International Journal of Imaging Systems and Technology, 29(4), 577–583. https://doi.org/10.1002/ima.22337. (2019)

[32]https://www.researchgate.net/figure/Example-dune-operation-de-max-pooling_fig4_337635267

[33]https://www.researchgate.net/figure/Convolution-kernel-mapping-a-a-standard-two-dimensional-convolution-kernel-and-b-a_fig2_364962873

[34] Krig, S. Computer Vision Metrics. In Apress eBooks. https://doi.org/10.1007/978-1-4302-5930-5. (2014)

[35] Krig, S. Computer Vision Metrics. In Apress eBooks. https://doi.org/10.1007/978-1-4302-5930-5. (2014)

[36] https://halil7hatun.medium.com/convolutional-neural-networks-cnns-95321b1f63ff

[37] Ramos, A. A., Cheung, M. C. M., Chifu, I., & Gafeira, R. Machine learning in solar physics. Living Reviews in Solar Physics, 20(1). https://doi.org/10.1007/s41116-023-00038-x. (2023)

[38] Alom, Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., Hasan, M., Van Essen, B. C., Awwal, A. a. S., & Asari, V. K. A State-of-the-Art Survey on Deep learning theory and architectures. Electronics, 8(3), 292. https://doi.org/10.3390/electronics8030292. (2019)

[39] GeeksforGeeks. Types of machine learning. GeeksforGeeks. https://www.geeksforgeeks.org/types-of-machine-learning/ . (2023, November 29)

[40] Sharma, S. Activation functions in neural networks - towards data science. Medium. https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6. (2022, November 20)

[41] https://quantdare.com/from-the-neuron-to-the-net/activation_functions-2/

[42] Kim, P. (2017). MATLAB Deep learning: With Machine Learning, Neural Networks and Artificial Intelligence. Apress.

[43] https://www.researchgate.net/figure/The-key-distinction-between-classic-neural-networks-and-modern-deep-neural-networks-Deep_fig3_366973865

[44] Khan, S., Rahmani, H., Shah, S. a. A., & Bennamoun, M. A guide to Convolutional Neural Networks for Computer Vision. Springer Nature. (2022)

[45]https://www.researchgate.net/figure/ResNet-architecture-image-taken-from-11_fig1_331671014

[46] Zhang, K., Ying, H., Dai, H., Li, L., Peng, Y., Guo, K., & Yu, H.. Compacting deep neural networks for Internet of Things: Methods and applications. IEEE Internet of Things Journal, 8(15), 11935–11959. https://doi.org/10.1109/jiot.2021.3063497. (2021b)

[47]https://www.researchgate.net/figure/The-architecture-of-MobileNetV2-DNN_fig1_361260658

[48] What is transfer learning? | IBM. (n.d.). https://www.ibm.com/topics/transfer-learning

[49]https://www.researchgate.net/figure/Schematic-representation-of-transfer-learning-Typically-dataset-B-is-much-smaller-than_fig1_369802799

[50] Bhattacharjee, J. Practical Machine Learning with Rust: Creating Intelligent Applications in Rust. Apress. (2019).

[51] Boyd, A., Czajka, A., & Bowyer, K. Deep Learning-Based feature extraction in iris recognition: use existing models, fine-tune or train from scratch? arXiv (Cornell University). https://doi.org/10.48550/arxiv.2002.08916. (2020)

[52] Khade, S., Gite, S., & Pradhan, B. Iris liveness detection using multiple deep convolution networks. Big Data and Cognitive Computing, 6(2), 67. https://doi.org/10.3390/bdcc6020067. (2022).

[53] https://www.geeksforgeeks.org/sift-interest-point-detector-using-python-opencv/

[54] GeeksforGeeks. What is the difference between SIFT and SURF? GeeksforGeeks. https://www.geeksforgeeks.org/what-is-the-difference-between-sift-and-surf/. (2024, June 13)

[55] https://onlinelibrary.wiley.com/doi/10.1155/2018/1463546

[56] Gou, J., Yu, B., Maybank, S. J., & Tao, D. Knowledge distillation: a survey. International Journal of Computer Vision, 129(6), 1789–1819. https://doi.org/10.1007/s11263-021-01453-z. (2021).

[57] Tang, W., Sun, J., Wang, S., & Zhang, Y. Review of AleXNeT for medical image Classification. ICST Transactions on E-education and E-learning, 9. https://doi.org/10.4108/eetel.4389. (2023)

[58] https://wenkangwei.github.io/2020/11/13/DL-DropOut/

[59] Sze, V., Chen, Y., Yang, T., & Emer, J. S. Efficient Processing of deep Neural Networks: A tutorial and survey. Proceedings of the IEEE, 105(12), 2295–2329. https://doi.org/10.1109/jproc.2017.2761740. (2017)

[60] Redaction. MobileNet, optimisation de la convolution pour les réseaux de neurones embarqués. - Quantmetry. Quantmetry. https://www.quantmetry.com/blog/mobilenet-optimisation-de-la-convolution-pour-les-reseaux-de-neurones-embarques/. (2020, September 10)

[61] Taşova, U. The Dictionary of Artificial Intelligence. Entropol.

[62] Erickson, B. J., & Kitamura, F. (2021). Magician's corner: 9. Performance Metrics for Machine learning models. Radiology. Artificial Intelligence, 3(3), e200126. https://doi.org/10.1148/ryai.2021200126. (2023)

[63] Delange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., & Tuytelaars, T. A continual learning survey: Defying forgetting in classification tasks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1. https://doi.org/10.1109/tpami.2021.3057446. (2021)

[64] Friedman, J. H. Greedy function approximation: A gradient boosting machine. Annals of Statistics, 29(5). https://doi.org/10.1214/aos/1013203451. (2001)

[65]https://www.researchgate.net/figure/Diagram-of-the-optimizer-comparison-20_fig4_351448263

ANNEXE

Python and Libraries

Python programming language:

Python is an interpreted, high-level, object-oriented programming language known for its simplicity, readability, and ease to use.

it is statistically one of the most used programming languages in today's time due to its various applications like web development, data science, but here the most important is machine learning.

Libraries in Python:

Libraries in python are a collection of related modules that provide pre-written code, tools, and functions for specific tasks.

It simplifies the coding process by proposing reusable code that can be used in different programs.

- **NumPy:**

NumPy (Numerical Python) is a Python library used for working with arrays, it provides a strong N-dimensional array object and advanced function for integrating C/C++ code. NumPy is a fundamental package for scientific computing with Python.

- **TensorFlow:**

TensorFlow is an open-source machine learning framework developed by Google, designed to be versatile and used for various ML tasks, as training and deployment,

It provides a comprehensive ecosystem of tools, libraries of its own and community resources that give room to research and development.

- **Keras:**

Keras is an open-source, high-level neural network API written in Python. It is designed to be user friendly to make it accessible to any developer whether it's a beginner or advanced in machine learning, it is built on top of other frameworks such as TensorFlow.

- **OpenCV:**

OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library. It is designed to come up with a comprehensive set of tools and algorithms for real-time computer vision applications.

- **Scikit-learn:**

Scikit-learn is a free software machine learning libraries in Python. It is designed to provide a comprehensive set of tools and algorithms for various machine learning tasks, including mostly classification, regression and clustering.

- **Matplotlib:**

Matplotlib is a comprehensive library for creating static, animated and interactive visualizations in Python, it allows users to create publication-quality plots and customize visual styles and layouts, it is open-source and can be used in plenty of environments.

- **Seaborn:**

Based on Matplotlib, Seaborn is a Python data visualisation library, it provides a high-level interface for drawing and informative statistical graphics, Seaborn is designed to make it easy to make informative and attractive statistical graphics.