PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

UNIVERSITY OF MOHAMED BOUDIAF -M'SILA

order number: . . . . .

Thesis submitted to the

UNIVERSITY OF MOHAMED BOUDIAF – M'SILA

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

In partial fulfillment of the requirements for the degree of

Master in Computer science

By

Doucene, Salah Eddine

Zehani, Mohamed Hached

Title of the thesis

Computer Vision-Based Age, Gender, Ethnicity Recognition

Under the supervision of

Abdessattar Ghemougui

Composition of the jury

Said Gadri	University of Msila	President
Abdessattar Ghemougui	University of Msila	Reporter
Said Hamani	University of Msila	Examiner

June, 2024

Dedications

"First and foremost, thanks to Allah for giving me the health and power to work through this project as a war-tank not stopping for no one and nothing, secondly, i dedicate this thesis to my parents for being there for me throughout my days, good and bad of them, thirdly, i dedicate this to my friends and family for their support and solidarity with me throughout this special journey, and also i dedicate this to Mr. Ghemougui for being one of the greatest teachers I've ever studied under, Cheikh Aissa, my elementary school teacher for being the guide of my success today, and i dedicate this surely to the person that has been with me all from the start of our educational journey back when we started at the age of 4, the real war-tank Doucene Salah eddine, and lastly I dedicate this to all the people that know me."

Zehani Mohamed Hached

"in the name of Allah and the prayers to our Prophet Muhammad peace be upon him, i dedicate this thesis to my dearest mom and dad and my sisters and my brothers that have been there with me throughout my good and bad memories, i also dedicate this to my friends and family that supported me and gave me the push to do what was once far, and i dedicate this special work to Mr. Ghemougui and Cheikh Aissa, my supervisor and my elementary school teacher and guide respectively, and i thank and dedicate this thesis to my guide to becoming the man i am today and the one that taught me all what i know about our religion Cheikh Belkacem Kamel, and i also thank and dedicate this thesis to the real war-tank of a person, my dearest friend and the person that have been with me since we were in elementary and before that Zehani Mohamed Hached, and i lastly dedicate this project to all the people that i know and all the people that know me."

Doucene Salah Eddine

Acknowledgments

This thesis would not have been possible without the guidance of our supervisor and the support our families and friends and the people surrounding us that surely were the fuel for our success.

We are glad to say and express the deepest of gratitude to our supervisor, Mr. Ghemougui Abdessattar. Your guidance and insight throughout our journey to make this thesis and project possible were the most essential part in shaping our work. And truly your patience has been one of a kind so we thank you again.

We can't forget or deny the incredible support and help we got from our families and friends and for that we are truly grateful and from our deepest feelings and hearts we thank every single one of you including the ones that think we forgot them. Thank you for love, encouragement, understanding, especially during this very challenging time.

We would like to also extend our appreciation to the committee (Mr. Kadri, Mr. Hamani), for their time and insights to help us improve with sharing their expertise.

We also thank the University of Mohamed Boudiaf, for being home to us for 5 years of our life, giving us the possibility to make one of the closest friends we have today, and for the valuable informations, and education that we took, and all of the resources it has given us the ability to check.

Contents

List of figures

List of tables

General Introduction

Age, Gender and Ethnicity, are very important aspects in the lives of any person. They help shape people into who they are, and who they would become, influence them and their experiences. They are the first things someone notices about a person. Recognizing them accurately is surely crucial for various real-world applications, such as security and demographic studies.

Age perception shows the changes someone would experience, and the life that has been lived, and the wiseness that have accumulated in someone's mind, throughout their life from their young to old ages.

Gender recognition highlights the experiences of people and how it affects their paths and interactions. And it helps explore the complexities of people and patterns of identity and identification.

Ethnicity recognition on the other hand indicates the culture, traditions, history and many more of a human being. it shows how a person may adapt to a world full of cultures and provides insight and connections to the present and the past.

Even with all the advancements that facial recognition technologies had over the years, we are still incapable of accurately recognizing age, gender and ethnicity; especially under bad lighting conditions, different poses, changes in facial expressions, and recognizing the patterns to determine old people for the lack of their data. A lot of models specified for these researches rely on limited datasets, making the adaptation to diversity harder. While the biases in the data used to train these models, lead to unfair and inaccurate results.

The main goal of this work is to develop an effective and accurate system to recognize age, gender and ethnicity of any given person using Convolutional Neural Networks (CNNs), while keeping in mind to train the model with different facial expressions and lighting conditions and poses. Whilst aiming to improve the system's effectiveness in real-world conditions by classifying age into groups instead of predicting exact numbers for ages, for the sake of enhancing the accuracy.

Nevertheless, Reducing biases in the models is important for us too and that is by using a diverse dataset that goes by the name of (UTKFace) to train the models, trying to ensure fair results for all population groups.

Moreover, Applying the system in various fields is an aim for later because this technology helps with security and surveillance, marketing, advertising, customer analysis, and providing personalized user experiences, especially when we add to the system other factors like emotion recognition and other factors.

Afterwards, Recognizing age, gender, ethnicity, requires a method called CNN. The models are trained on a large dataset of over 20000 images (UTKFace), and a smaller one that we made contatining less than 500 images to identify patterns and traits linked to age, gender and ethnicity. With the trained models we can then estimate the age, gender and ethnicity of any person from their facial images. While categorizing age into groups to improve accuracy.

After all we've said, the important things to know are the challenges and limitations we may face working on making a good system, possibility of usage of the application under any circumstance regardless of whatever may arise, such as lighting and facial expressions, and most importantly, the correlation between age, gender and ethnicity in human beings.

Regarding the chapters that we chose to work on, they are 4. The first one, gives a general view over AI and its applications including diving deeper into computer vision, and CNNs, whilst the second chapter is about the definitions and the state of the arts, the third chapter is about the training and datasets used, and also the implementation of our models, then their evaluations.

Finally, the general conclusion will summarize the findings of various approaches and give some perspectives on future works.

Also, after the general conclusion, we have an appendix to state all the needed things to state but aren’t essential for the thesis.

Chapter 1: Introduction to Artificial Intelligence

1. Introduction

Age, gender and ethnicity recognition is a well known area in computer vision and artificial intelligence research especially age and gender recognition systems. These systems aim to automatically recognize and classify people based on their age, gender and ethnicity/race using facial images.

2. Artificial Intelligence

The concept of artificial intelligence refers to the way in which the capabilities of human intelligence are simulated. It is a part of computer science that deals with the process of designing intelligent systems that exhibit a set of characteristics that are linked to intelligence related to many human behaviors.[1]

it is an aspect of computer science that relies on providing a variety of methods, techniques, and tools to create models and solutions to issues by simulating the behavior of individuals.[2]

Artificial intelligence is also categorized into two types, the first type is weak artificial intelligence, which focuses on a set of specific and narrow tasks, such as self-driving cars, and the other type is strong artificial intelligence, which is known as artificial general intelligence (AGI), and this type is able to perform most of the cognitive functions that humans may have, in addition to applying intelligence to more than one problem.[3]

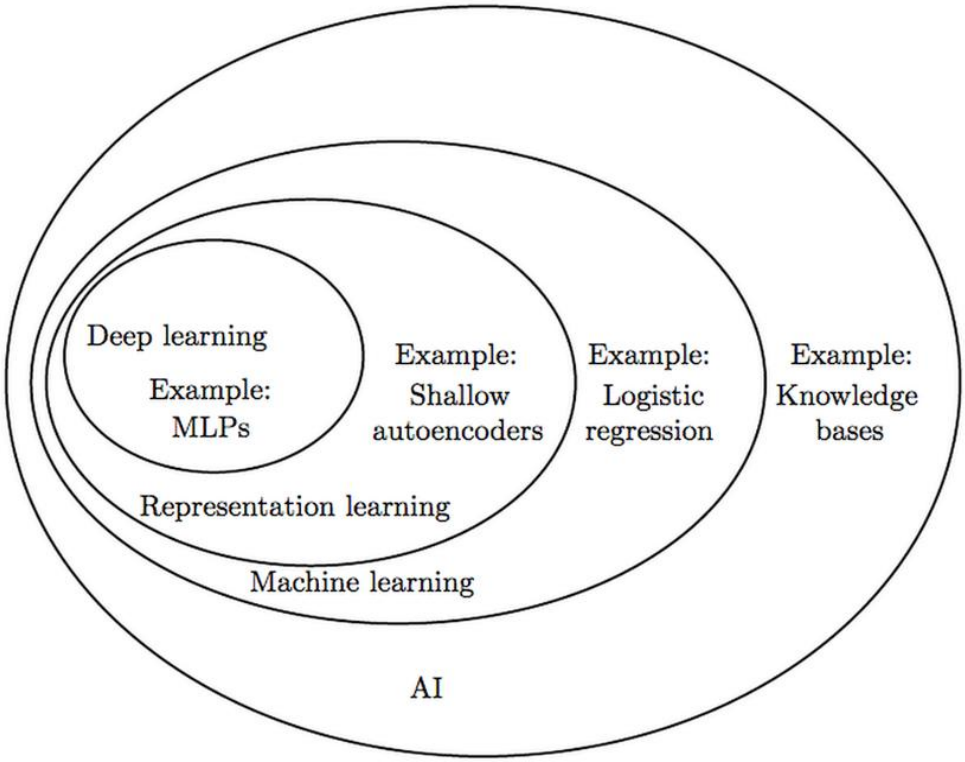

Fig. 1: Artificial Intelligence Domains [4]

2.1. AI technologies

Artificial intelligence technologies are the set of software, tools, and algorithms used to develop and implement artificial intelligence, and among these technologies we mention the main ones: Machine learning, deep learning, computer vision, expert systems and robotics, these technologies are characterized by rapid and continuous development thanks to research efforts.

- Machine Learning: It is a field in computer science that is concerned with developing systems to make them more human-like. It relies on data and algorithms to train systems so they improve their performance over time. While its usage and applications vary, such as voice recognition, machine translation and data analysis.
- Deep Learning: It is a branch of machine learning that focuses on using multi-layer NNs (Neural Networks) to understand and analyze data. It aims to represent data at multiple representational levels and automatically learn distinctive features. Its usage also vary, such as in image recognition, machine translation and text analysis.

- Computer Vision: It is a field that uses machines to recognize people, places and objects in images with accuracy that reaches human levels, even exceeds it sometimes by how quick and efficient it acts using complex models of AI. It works by extracting useful features from images to then analyze and categorize it and understand it.
- Expert Systems: It is a computer Application for decision-making in real-life situations,[5]; it aims to simulate the human logic of experts in a particular field of knowledge.
- Natural Language Processing (NLP): Natural Language Processing (NLP) is a sub-science of artificial intelligence, which in turn is a branch of informatics and interferes heavily with language sciences that provide linguistic characterization to the computer. This science enables us to create software that can analyze, simulate and understand natural language, as this latter is used in many fields, including machine reading of texts, automatic text or speech generation, information extraction, and machine translation.[5]
- Neural Network (NN): A neural network is a system designed to simulate the way the human mind performs a particular task. It is a huge, evenly distributed processor made up of simple processing units, these units are nothing but computational elements called neurons or have a neuronal property where they store practical knowledge and empirical information and make it available to the user by adjusting the weights.[5]

2.2. Historical Development of AI

In the mid-20th century, a few scientists began to explore a new approach to building intelligent machine based on new discoveries in neuroscience, a new mathematical theory of information, the development of cybernetics and, above all, the invention of the digital computer, and then inventing a machine that can simulate the human computational thinking process.[5]

The modern field of AI research was founded at a conference on the campus of Datmouth College in the summer of 1956, these attendees became the leaders of AI research for several decades, especially Marvin Herbert Simon, Allen Newell and Lee Minsky, whom founded the AI labs at MIT.[5]

MIT, Carnegie Mellon University and Stanford University, they and their students wrote programs that amazed most people. The computers they made were solving algebra problems, proving logical theorems and speaking English.

By the mid-1960s this research was generously founded by the U.S. Department of Defense and these researchers made the following predictions:

- In 1965: Herbert Simon, "In twenty years, machines will be able to do any job a human being can do".
- In 1967: Marvin Minsky, "Within a generation, the problem of creating artificial intelligence will be largely solved".

But, they failed to recognize the difficulty of some of the issues they faced in 1974.

Criticisms of AI, and constant pressure from Congress to fund more productive projects, the U.S. and British governments cut funding for all AI-Oriented exploratory, which was the first setback for AI research.

In the early 1980s, AI research experienced a renaissance with the ongoing success of expert systems, which is an AI program that mimics the knowledge and analytical skills of one or more human experts.

By 1985, AI research profits in the market reached more than a billion dollars. Governments started funding again and a few years later, starting with the collapse of the lisp machine market in 1987, AI research once again experienced a setback but this time longer.

In the 1990s and early 21st century, artificial intelligence made even greater strides being used in logistics, data mining, medical diagnostics and many other areas, throughout the technology industry.

The success is due to several factors, the most important of which are:

- The great computational power of today's computers.
- The increased focus on solving specific sub-problems.
- The creation of new relationships in AI and other areas working on similar issues.
- Researchers began to adhere to strong mathematical methods and rigorous scientific standards.

2.3. AI Applications

There are many applications of AI worth noting in many different and diverse fields and here we state some of them.

- Personalized shopping: Artificial Intelligence technology is used to create recommendation engines through which you can engage better with your customers. These recommendations are made in accordance with their browsing history, preference, and interests. [6]
- Autonomous Vehicles: Automobile manufacturing companies like Toyota, Audi, Volvo, and Tesla use machine learning to train computers to think and evolve like humans when it comes to driving in any environment and object detection to avoid accidents.[6]
- Spam Filters: The email that we use in our day-to-day lives has AI that filters out spam emails sending them to spam or trash folders, letting us see the filtered content only. The popular email provider, Gmail, has managed to reach a filtration capacity of approximately 99.9%.[6]
- Facial Recognition: Our favorite devices like our phones, laptops, and PCs use facial recognition techniques by using face filters to detect and identify in order to provide secure access. Apart from personal usage, facial recognition is a widely used Artificial Intelligence application even in high security-related areas in several industries.[6]
- Healthcare: Artificial Intelligence is widely used in the field of healthcare and medicine. The various algorithms of Artificial Intelligence are used to build precise machines that are able to detect minor diseases inside the human body. Also, Artificial Intelligence uses the medical history and current situation of a particular human being to predict future diseases. Artificial Intelligence is also used to find the current vacant beds in the hospitals of a city that saves the time of patients who are in emergency conditions.[7]
- Insights and Analysis: With the help of AI, a collection of large datasets, that includes clinical data, research studies, and public health data, to identify trends and patterns. This inversely provides aid in surveillance and public health planning.[7]
- Telehealth: This feature enables doctors and healthcare experts to take close monitoring while analyzing data to prevent any uncertain health issues. Patients who are at high risk and require intensive care are likely to get benefited from this AI-powered feature.[7]

- Gaming: Artificial Intelligence is really dominating the field of the gaming industry. Artificial Intelligence is used to make a human-like simulation in gaming. This enhances the gaming experience. Apart from that, AI is also used to design games, predict human behavior, to make the game more realistic. Various modern games use real-world simulation for gaming using AI.[7]
- Quality Assurance: Testing games & ensuring their performance gets easier allows testers to perform rigorous testing in comparatively less time. It empowers and fixes all the game mechanics and any other potential bugs that can hinder performance.[7]
- Game Assistance: AI algorithms offers virtual assistance during gaming sessions that include tips, tutorials, and other useful resources. This feature help players to be in the game & understand the metrics during the whole time session.[7]
- Animation: To make games more realistic, machine learning and artificial intelligence algorithms are being used in today's gaming industry. Techniques such as Neural network empowers stimulation and facial expressions for an immersive experience.[7]

3. Computer Vision

Computer Vision is a field of AI that focuses on enabling computers to process, analyze, and understand digital images and videos.

It involves the extraction of high-dimensional data from the real world to produce numerical or symbolic information that can be used for decision-making.

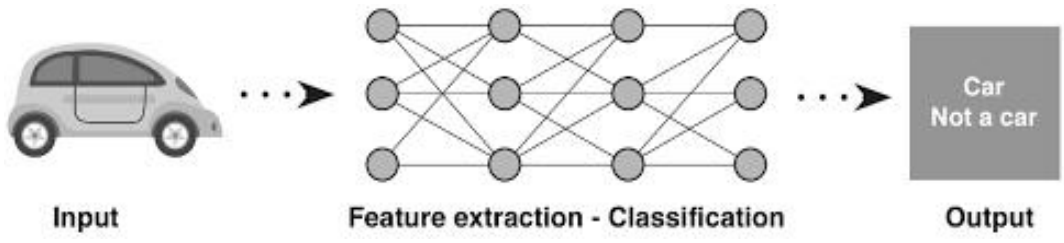

Fig. 2: A demonstration of features extraction leading to output. [8]

It also aims to mimic both the way that we humans see and the way we sense the world that we see around us, that is with the involvement of training computers and systems to identify and understand objects and beings that it sees, by extracting patterns in images and videos.

Using machine learning and neural networks, computer vision systems can derive meaningful information from visual inputs and take actions or make recommendations based on the information.

3.1. Computer Vision Techniques

There are many techniques in computer vision and these are some of the most common ones:

- Image Processing: It includes several techniques such as filtering and morphological operations to optimize the images for further analysis. [9]
- Feature Detection and Description: This technique is done by identifying key points or features in an image and describing them in a way that is unaffected by changes in size, orientation, and lighting.
- Object Detection and Recognition: this is done by localizing objects within images or videos, often using deep learning models such as convolutional neural networks (CNNs). [10]
- Semantic Segmentation: This is based on dividing the image into regions with a certain semantic meaning, such as distinguishing between different objects or backgrounds.
- Tracking and Motion Analysis: Monitoring the movement of objects over time in videos.
- 3D Reconstruction: Creating 3-Dimensional models of objects or scenes from multiple images or videos frames.

3.2. Computer Vision Applications

There are many applications that we could state so we took the most common and researched ones in our world today:

- Autonomous Vehicles: Computer Vision enables vehicles to perceive their environment and make decisions autonomously, crucial and important for tasks like navigation and obstacle avoidance.
- Surveillance and Security: Monitoring and analyzing video feeds for identifying security threats or unusual activities.

- Medical Imaging: Assisting healthcare professionals in tasks like tumor detection, organ segmentation, and disease diagnosis from medical images.
- Augmented Reality: Overlaying digital information onto the world, enhancing user experiences in fields like gaming, education, retail and much more.
- Industrial Automation: Utilizing computer vision for quality control, object tracking and robotic manufacturing processes.
- Retail and E-commerce: enhancing customer experiences through features like visual search, recommendation systems, and inventory management.
- Agriculture: Monitoring crop health, yield estimation, and automated harvesting through aerial and ground-based imagery analysis.

4. Convolutional Neural Networks (CNNs)

A convolutional neural network (CNN) is a type of deep learning algorithm, and it is well-suited for analyzing visual data, such as images. They use principles from linear algebra, specifically convolution operations, to extract features and identify patterns within images. [11]

CNNs are designed to automatically and adaptively learn spatial hierarchies of features through back-propagation by using multiple building blocks, such as convolution layers, pooling layers, and fully connected layers [Fig.3]. CNNs have become the state of the art for many visual applications, including image classification and object recognition.[12]

The architecture of a CNN typically consists of an input layer, an output layer, and multiple hidden layers in between. The core building block of a CNN is the convolutional layer, which applies convolution operations to the input data to extract features. The earlier layers of a CNN focus on simple features, such as colors and edges, while the deeper layers recognize larger elements or shapes of the object. This hierarchical approach allows CNNs to capture hierarchical patterns and spatial dependencies within images. [13]

CNNs are trained using a large dataset of labeled images, where the network learns to recognize patterns and features associated with specific objects or classes. They can also be adapted to work with audio and other signal data, making them versatile for various tasks.

CNNs have applications beyond image recognition and analysis, including natural language processing, drug discovery, health risk assessments, and depth estimation for self-driving cars.

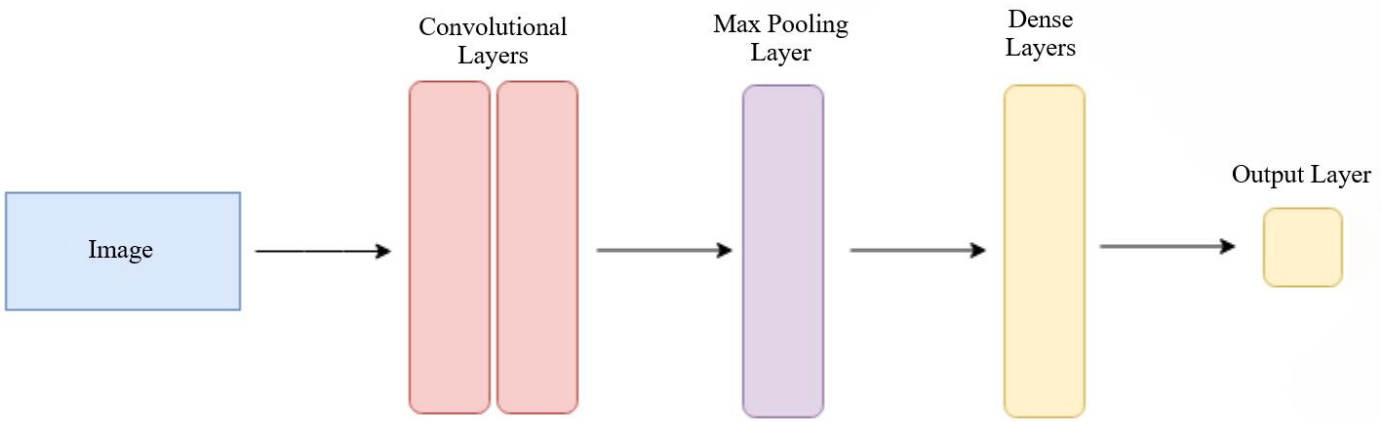

Fig. 3: A simple representation of a CNN architecture. [14]

5. Conclusion

Through what we covered in this chapter, we were able to know the conceptual framework of AI and Computer Vision, as well as some different applications of AI and Computer Vision and an attempt to define each of them. We will move on to the next chapter, in which we will discuss the general concepts of age, gender and ethnicity recognition as well as mentioning some previous studies.

Chapter 2: Definitions & State of the Arts

1. Introduction

After presenting general concepts about AI technologies and their important applications in the previous chapter, we now turn to age, gender and ethnicity recognition and face recognition technologies with an attempt to give an overview of them in addition to presenting some previous studies.

2. Face Recognition

Facial recognition is a method of identifying or verifying a person's identity by analyzing and comparing patterns of facial features captured from an image or video. This technology has attracted a lot of attention due to its potential applications in security, law enforcement, and personal identification. It involves the use of deep learning algorithms and computer vision techniques to extract and analyze facial features such as the distance between the eyes, the shape of the nose, and the angle of the jaw line. Face recognition has been applied in various fields such as cell phones, social media, and surveillance systems. [26]

Facial recognition technology (FRT), once a futuristic fantasy, has become an undeniable presence in our world. It has exploded onto the world stage, transforming from a figment of science fiction into a force shaping our daily lives.

This powerful tool can uniquely identify individuals using their facial characteristics, promising a future filled with convenience and security. From unlocking smartphones with a glance to streamlining airport security procedures, FRT's potential applications seem endless.

However, this exciting technological advancement is not without its complexities. Ethical considerations regarding privacy, bias, and the potential for misuse cast a long shadow over its widespread adoption. As we navigate the landscape of FRT's impact, a deeper understanding of its evolution, its multifaceted applications, and the ethical considerations surrounding it becomes crucial.

2.1. History of Facial Recognition

The implementation of facial recognition is not new, in fact, its roots date back to the 1960s when facial recognition was manually implemented by Woodrow Wilson Bledsoe. Bledsoe is largely considered the father of facial recognition for developing a system that classified photos of faces through a RAND tablet, which was a graphical computer input device.

With this device, Bledsoe manually recorded the coordinate locations of facial features such as a person's mouth, nose, eyes, and even their hairline. Back then, computers weren't powerful enough to handle complex calculations.

So, people had to manually record facial features in a log and then compare new photos to those entries by hand. It wasn't very efficient, but it was a starting point to start searching about and studying facial recognition.

In the late 1980s, a new mathematical technique called linear algebra (developed in the mid 1800s), gave facial recognition a big boost. Researchers figured out a way to represent faces using just a small set of key features, instead of needing a ton of data. This was like finding a shortcut to identify faces. This method, called Eigenfaces, laid the groundwork for the powerful facial recognition systems we have today. [15]

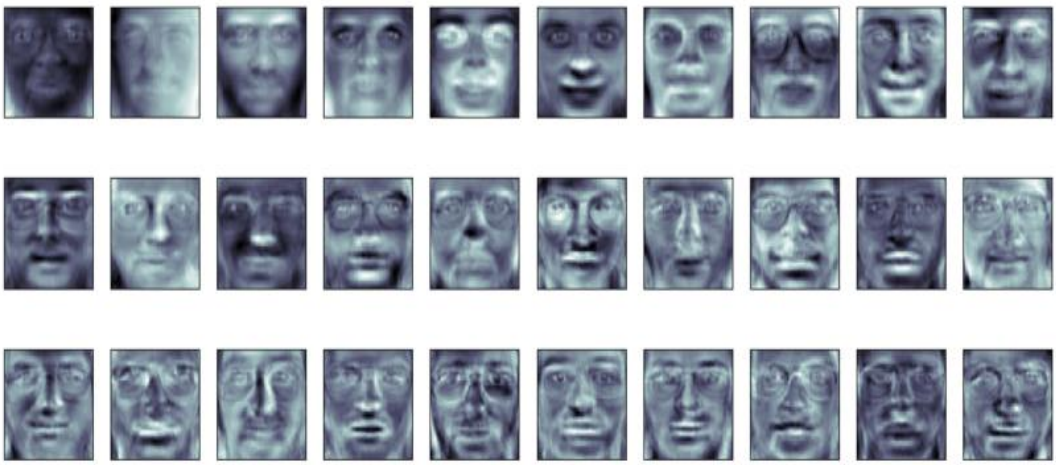

Fig. 4: Examples of Eigenfaces. [15]

Eigenfaces is a machine learning method that help store more information in smaller sizes with only linear algebra fundamentals. It is a method that is useful for face recognition and detection by determining the variance of faces in a collection of face images and use those variances to encode and decode a face in a machine learning way without the full information reducing computation and space complexity. [27]

It wasn't the only one that laid the groundwork of modern facial recognition systems though but also programs like FERET.

These initiatives involved building vast databases of facial images. This seemingly simple step – collecting a large number of faces for testing – proved crucial. It allowed researchers to experiment with different methods of identifying people from their faces.

The government then took the reins in the 2000s, sponsoring competitions like FRVT and FRGC. These programs pushed the boundaries of facial recognition even further. By pitting different systems against each other, they accelerated advancements in accuracy.

As a result, facial recognition technology became significantly more reliable leading to improved accuracy and reliability in facial recognition systems.

The rise of social media platforms like Facebook in the 2010s marked another turning point. Facebook's use of facial recognition for photo tagging brought the technology directly to everyday users. While convenient, it sparked concerns about privacy.

The idea of having your face constantly analyzed rasied questions about data security and potential misuse. Despite these concerns, facial recognition continued its march forward.

Smartphones like the recent iPhone X, as an example, incorporated facial recognition for unlocking devices, showcasing its potential for everyday use beyond social media.

Today, the applications of facial recognition technology extend far beyond unlocking phones. Companies like NEC are using it to enhance security at airports and borders, aiming to streamline processes and improve safety.

Additionally, the concept of "paying by face" is emerging, hinting at a future where our faces could replace traditional payment methods.

However, the rise of facial recognition is not without challenges. Privacy remains a major concern. Questions linger about how facial data is stored and used. Additionally, there are concerns about bias and accuracy, particularly regarding the technology's ability to correctly identify people of color.

The future of facial recognition is uncertain. It holds the potential to revolutionize various industries, offering enhanced security and convenience.

However, ethical considerations and potential misuse cannot be ignored. As we move forward, it is crucial to strike a balance between technological advancement and the protection of individual privacy. Only through responsible development and implementation can facial recognition technology truly fulfill its potential for a safer and more efficient future.[16]

2.2. Present-day Facial Recognition

Today, FRT is rapidly integrating itself into various aspects of our lives. It is transforming the way we approach security, law enforcement, access control, financial transactions, and even our smartphones.

Airports and border control facilities now rely on FRT to facilitate security checks and identify potential threats.

Law enforcement agencies utilize FRT to aid in identifying suspects and missing persons by comparing captured faces to databases.

Businesses and organizations are implementing FRT for access control purposes, whether it's recognizing authorized personnel or securing high-risk areas.

Additionally, the concept of "paying by face" is gaining traction, potentially revolutionizing financial transactions. Imagine a future where a simple glance can replace the need to swipe your credit card.

Furthermore, FRT has become a popular feature on smartphones, offering a convenient and secure alternative to traditional passwords. As FRT continues to advance, it holds the potential to reshape our daily lives in ways we could never have imagined.

3. Age, Gender and Ethnicity Recognition

Age, gender, and ethnicity recognition systems are advanced tools that analyze facial features to identify and classify individuals based on their age, gender, and ethnicity. These systems have various applications, such as enhancing security measures and improving user experience in different industries.

3.1. Age Estimation

Age and age recognition are important concepts that have implications in various domains, including cognitive processes, biometric recognition, computer vision, and social institutions. Age plays a significant role in understanding human development, cognitive abilities, and the impact of aging on different aspects of life.

Age recognition, on the other hand, is crucial for tasks such as identification, verification, and face recognition in various applications. Age recognition is a crucial component of biometric systems, which are used for personal authentication and identification purposes.

Automated fingerprint recognition systems, for example, rely on age recognition to process a diverse range of fingerprints and support consistent detail extraction for identification and verification, it is also important for enforcement. [17] It helps identify criminals and suspects, which is crucial in forensic investigations. This feature ensures that the identification process is accurate. [18]

3.2. Gender Recognition

Gender and gender recognition are important topics that have significant implications for individuals and society as a whole. Understanding gender and recognizing the diverse experiences of individuals can promote respect and equity. Gender recognition in itself refers to the ability of AI systems to identify and predict the gender of individuals based on their facial features. This technology utilizes

machine learning algorithms that analyze and compare facial characteristics that distinguish male faces from female faces, such as nose length, cheekbone shape, lip thickness, and eyebrow position.[19]

Fig. 5: An image showing a system recognizing faces and classifying them. [19]

Gender recognition has various potential applications. For example, it can be used in healthcare services to tailor the diagnosis and treatment of certain conditions that differ between males and females, such as cardiovascular diseases, autoimmune disorders, or mental health. Whilst, companies and shops could use the technology to gather information about the gender distribution of their costumers.

3.3. Ethnicity Prediction

Ethnicity is a complex social construct that refers to a person's cultural background, including shared customs, traditions, language, and heritage. It is often associated with a person's racial or national identity. It can be self-identified or ascribed by others based on physical characteristics, cultural practices, or ancestry.

facial recognition algorithms, has been used to predict or detect ethnicity based on facial features. These algorithms analyze facial characteristics such as skin color, eye shape, and facial structure to deduce demographic data and make predictions about a person's ethnicity.

Fig. 6: An image showing an AI recognizing age, gender and race. [20]

Research studies have also shown that AI models can predict race or ethnicity with high accuracy using medical images, such as chest x-rays. These models have demonstrated the ability to detect patterns in facial landmarks that differentiate between different ethnic groups. [21]

However, it is worth noting that human experts are often unable to make such predictions by looking at medical images, making the accuracy of AI models in this context remarkable.[21]

4. Age, Gender and Ethnicity Literature Review

Facial recognition technology has made significant advancements in age and gender recognition, and researchers have been exploring its applications in race and ethnicity recognition too, whilst age and gender are frequently studied together, race and ethnicity are often investigated independently.

4.1. Age and Gender Literature Review

Age and gender, 2 independent problems that have always been studied together by many people including us, the estimation of them has seen such an advancement in recent years.

In the work of MiVOLO-V2, the capabilities of the recent models were compared in a specialized task for age and gender estimation using the MiVOLO state-of-the-art model, yielding interesting results and analyses about the strengths and weaknesses of the combined models, as well as an attempt to modify the ShareGPT4V model for this specific task in order to achieve state-of-the-art results. [22]

One of the results obtained, is that this model will not be practical in production as it is too expensive compared to a specialized model like MiVOLO, but it can be very useful in some tasks such as data annotation, specifiying that MiVOLO-V2 managed to get an accuracy of 97.39% on the Adience gender dataset. [22]

ViT-hSeq is another model to improve accuracy of age and gender classification using facial feature proccessing, the model outperforms ther previous state-of-the-art implementations, increasing the accuracy by 10% for age estimation, while achieving 84.91% for age estimation on the Adience age dataset. [31]

One of the other models that ranked 3rd in both age estimation and gender estimation, named MiVOLO-D1, using both face details and body images of people, to make the model adaptable and able to give good results even in the absence of faces. This model even outperformed humans in most identifying ages of people. [32]

RetinaFace + ArcFace + MLP + Skip connections is a study that explored MLP designs for improved performance. The researchers tested various MLP architectures using age and gender datasets, giving them an accuracy of 90.66%. [33]

CGP presents a method for continuous deep learning that combines deep model compression, critical weight selection, and incremental mediation of networks. The goal of the study was to learn new tasks while retaining knowledge from previous tasks, using an easy to implement method, preventing forgetting previous tasks, allowing for model expansion, and keeping the model compact by compressing and selecting important information. The method shows that knowledge from previous tasks can improve the model performance on new tasks, giving us and accuracy 89.66% on the Adience gender dataset. [34]

AL-ResNets-34-IMDB-wiki introduced a new method for accurately estimating age in real-world- settings. The method combined Residual Networks with long short-term memory networks to improve the accuracy of age estimation. The model is fine-tuned and further fine-tuned on IMDB-WIKI-101 dataset and then target age datasets to extract overall facial features, while an LSTM module was introduced to pinpoint age-sensetive regions and extract local features, achieving 67.47%. [35]

In R-SAAFc2 paper proposed the interactive activation functions and how to apply them to CNNs in regression tasks. When CNNs were evaluated experimentally using SAAFs, they showed improved results on a set of positional age estimation datasets, including Adience age dataset, with an accuracy of 67.3%. [36]

Rank	Model	Accuracy	Year
1	**MiVOLO-V2**	**97.39%**	**2024**
2	**ViT-hSeq**	**96.56%**	**2024**
3	**MiVOLO-D1**	**96.51%**	**2023**
4	**RetinaFace + ArcFace + MLP + Skip connections**	**90.66%**	**2021**
5	**CPG (Single Crop Pytorch)**	**89.66%**	**2019**

Tab. 1: Age And Gender Classification on Adience Gender. [29]

Rank	Model	Accuracy	Year
1	**ViT-hSeq**	**84.91%**	**2024**
2	**MiVOLO-V2**	**69.43%**	**2024**
3	**MiVOLO-D1**	**68.69%**	**2023**
4	**AL-ResNets-34 + IMDB-WIKI**	**67.47%**	**2018**
5	**R-SAAFc2 +IMDB-WIKI**	**67.3%**	**2017**

Tab. 2: Age And Gender Classification on Adience Age. [30]

MiVOLO-V2 [22] is a state-of-the-art (as of 4 march 2024) open-source neural network for gender estimation. It is a multi-input transformer model that leverages the latest vision transformer technology to perform age and gender recognition in challenging real-world conditions. The model integrates both age and gender estimation tasks into a unified dual input/output model. It not only utilizes facial information but also incorporates body images to improve accuracy.[22]

It was trained using mutliple datasets combined to create a very large dataset of images (IMDB-clean / UTKFace / FairFace / Adience / AgeDB / LAGENDA), and additionaly extending the train dataset to contain over 800k samples.

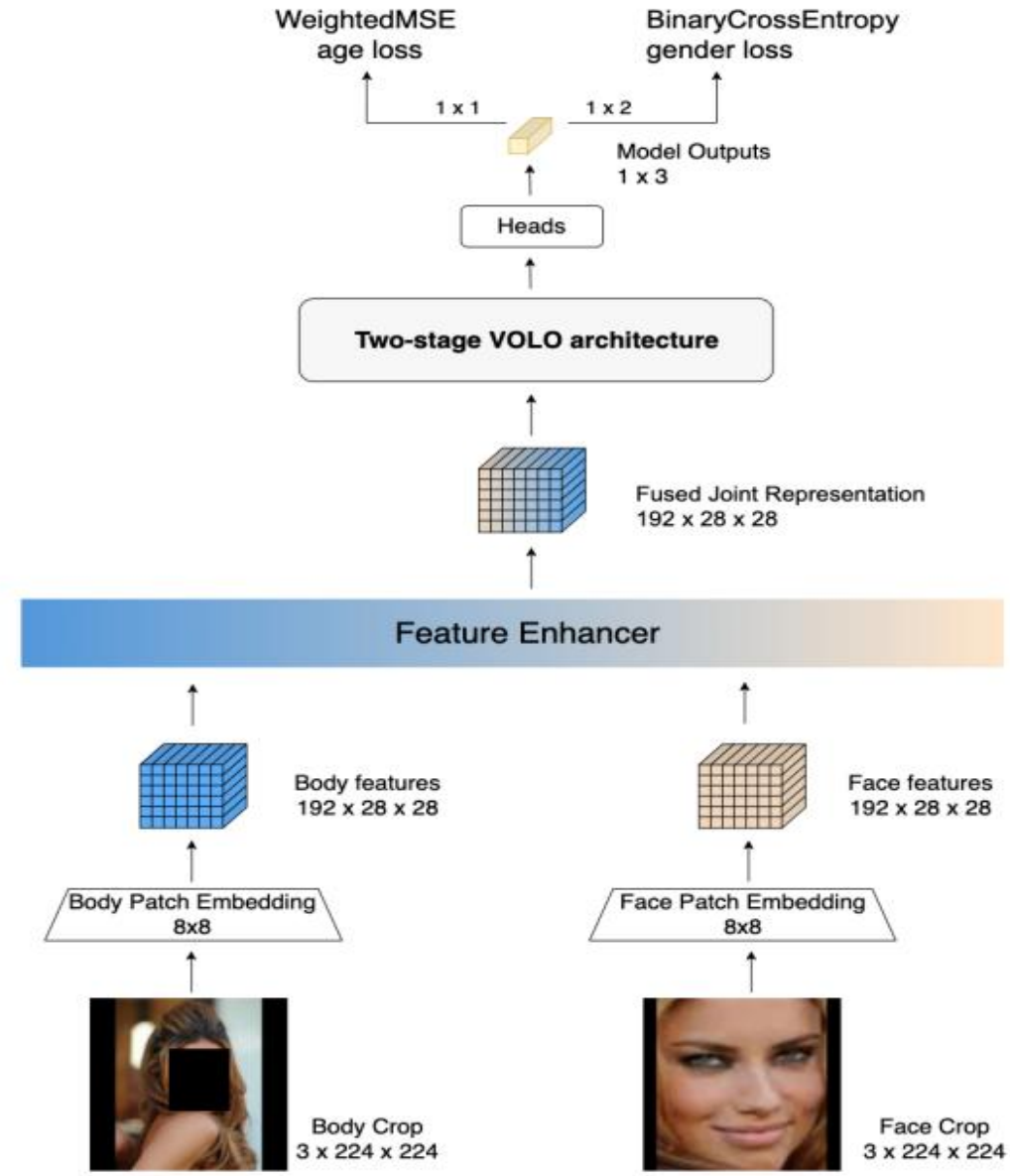

Fig. 7: The architecture of MiVOLO-V2. [22]

The approach taken by the state-of-the-art is particularly useful in scenarios where the face is partially or completely occluded, and there is variability in conditions, pose complexities, and image quality. It has given very accurate results in age and gender estimation tasks.

Fig. 8: An image showing the MiVOLO-V2 recognizing the age and gender. [22]

It has been recognized as the current state-of-the-art model for gender prediction on the Adience dataset [28]. Additionally, it has demonstrated high accuracy in age prediction, contributing to the advancement of age estimation techniques.

The MiVOLO-V2 model represents a significant advancement in age and gender estimation tasks, offering state-of-the-art performance on the Adience Gender benchmark and demonstrating the effectiveness of its multi-input transformer architecture and that is because it can identify a person using even just a body [fig.9].

Fig. 9: An image showing how this method works wihtout a face [37]

About the challenges and limitations of of this state of the art model, we know that MiVOLO-V2, encountered refusal to process images with significant losses over 21%, and that was occasionally of course, also according to their paper, the fine-tuning of a Multimodal Large Language Model (MLLM) is quite expensive, which makes it not practical for production compared to a specialized model, and the computational requirements, which can limit their practicality and accessibility in certain scenarios. [22]

ViT-hSeq on the other hand, is a stat of-the-art for age estimation with an accuracy of 84.91%, using age groups unlike MiVOLO-V2 that gives a specific age directly.

It was trained using the Adience dataset, which includes a total amount of 19370 individual images, labeled with their respective age-groups and genders as follows, 8 groups for age [(0-2), (4-6), (8-12), (15-20), (25-32), (38-43), (48-53), (60,100)] and 3 labels for gender [f,m,u], meaning female, male and unidentified respectively, using a hybrid model that employs the intra-attention-based knowledge gained on sub-characteristics of facial attributes, which is later clubbed with spatialpattern processing capabilities of BiLSTM (hybrid-Sequencer). [31]

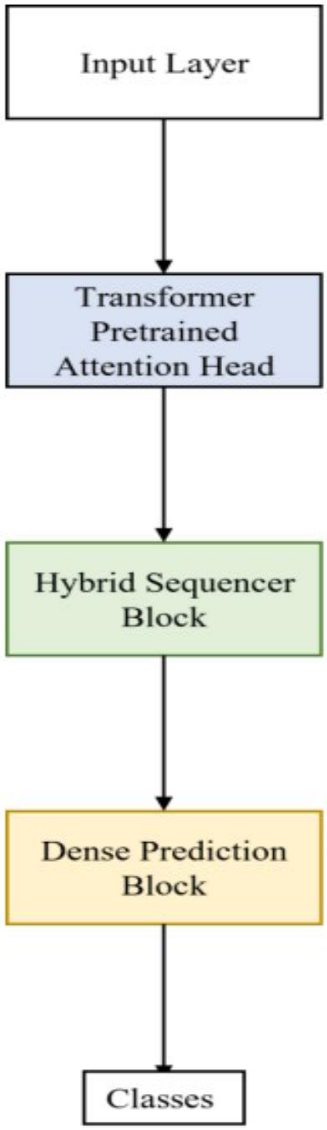

Fig. 10: The architecture used to predict the classes. [31]

4.2. Ethnicity/Race Literature Review

Race estimation refers to the process of identifying or categorizing individuals into different racial groups. Race estimation has seen advancements in recent years, particularly with the use of deep learning techniques.

Race estimation with deep networks [23] is centered on addressing the issue of ethnicity identification for 4 ethnic groups. BUPT Equalised Face dataset was used to train the deep convolutional network 'R-Net' which in fact achieves a very superior accuracy of 97%, the model was then texted using 'UTKFace' and 'CFD' datasets and

compared to VGG16 for race estimation, which demonstrated its robustness for its usage in unconstrained environments.

Some research [39] in ethnicity classification used Efficient CNN models and focused on the central part of the face alone, comparing to the results of 2 distinct CNN models, using a wait-and-see test on the MORPH and FERET datasets. The classification on 4 classes gave an accuracy of 85% on Model A and 86% on Model B from their study. [39]

Another research [40] to classify ethnicity based on images, used the IDL-ERCFI technique, which is based on smart DL. This technique uses facial landmarks to align the images before sending them to the network. The proposed model in this research uses an exclusion network as a feature extractor. Since the retrieved features are high-dimensional, the feature reduction procedure uses principal component analysis (PCA) technique, which is effective in overcoming the "curse of dimensionality". furthermore, the racial classification procedure was by using the kernel optimal maximum likelihood learning machine (KELM), with the parameters of the KELM model tuned using glowworm swarm optimization (GSO). [40]

The state-of-the-art work trained R-Net on the BUPT Equalised Face dataset [38], which contains approximately 1.3 million images in an unrestricted environment. The model achieved a state-of-the-art accuracy of 97% for race identification of four broad racial groups: Caucasian, African, Asian, and Indian. [23]

Fig. 11: The 4 classes used for prediction and training. [23]

The work starts by taking a face rotating it, till the horizontal line of the eyes is estimated, then crops the image to keep only the features of the face, such as the nose, face shape, eyes, eyebrows and mouth then it gets re-sized to a size of 40x40 and converted to gray-scale then classified.

Fig. 12: An image showing how the face gets rotated [23]

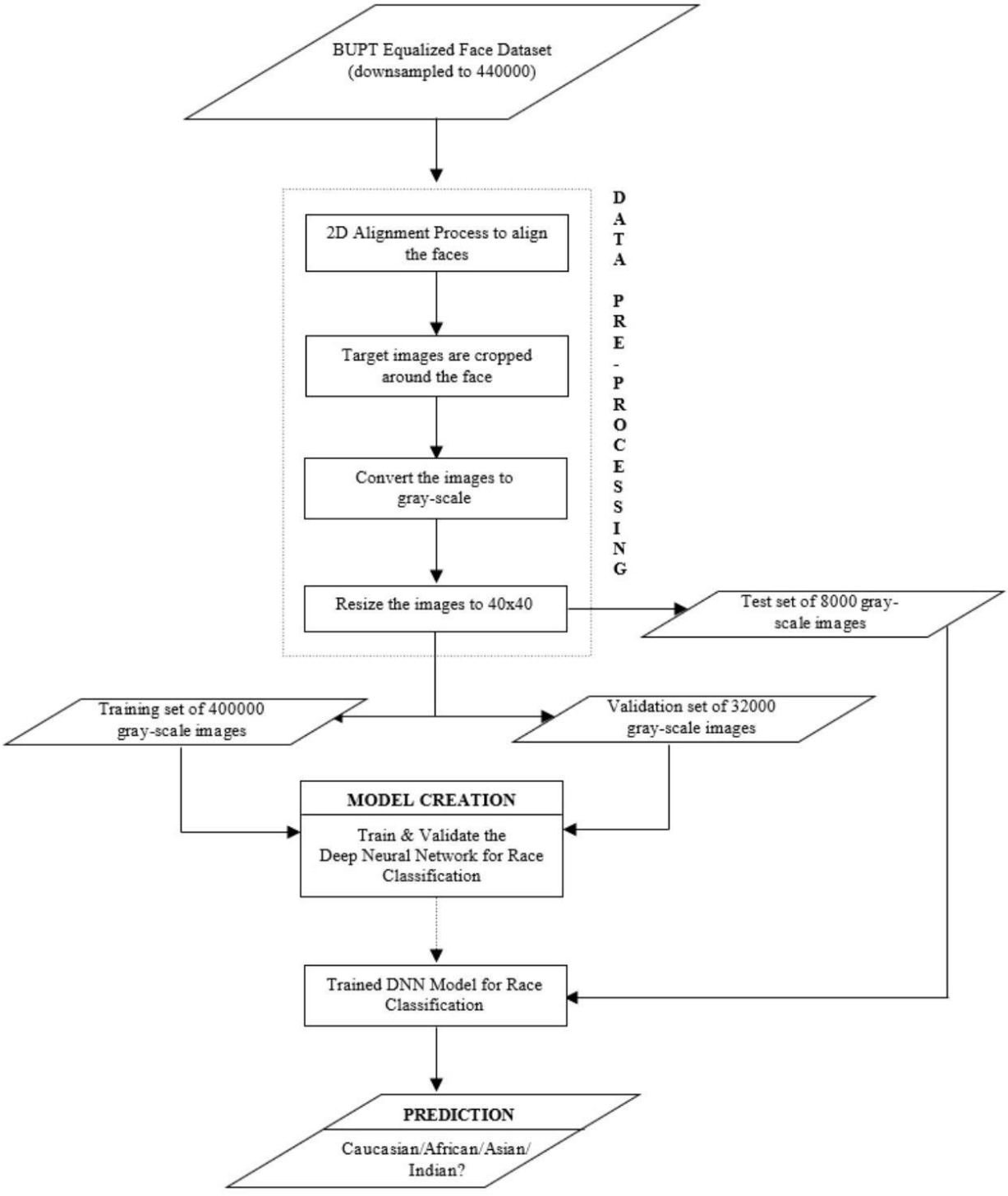

Fig. 13: An image showing the architecture of the ethnicity recognition model [23]

The model was also evaluated on other datasets, namely UTK and CFD, and compared with a fine-tuned VGG16 model for race estimation.

Race and ethnicity is a concept defined by humans about looks and cultures and how the people dress, making it harder for machines to accurately identify all of the races and ethnicities without a margin of error or bias, especially that there is an increased growth of the mixed-race population that further complicates the identification of a certain race or ethnicity, as it blurs the boundaries between the races that exist now.

The selection of training samples also plays a significant and crucial role in determining the performance of the model in real life scenarios, trying to avoid all biases and discrimination. Meaning, The selection of data samples to train should be picked very wisely and carefully.

5. Conclusion

In this chapter, we have given an overview of the concepts of age, gender, ethnicity and facial recognition, and presented the latest developments in the state of the arts of the said subjects and a bit of history about facial recognition, which made it clear that there is room for improvements, upon improving more the accuracy and robustness of the models especially in the field of ethnicity recognition.

In the next chapter, we will be moving on to talk about the dataset we used to create our deep learning models and the models themselves, and the development while showing the training and explaining the process of it and then seeing the results and evaluating them and the models themselves.

Chapter 3: Data, Models, Evaluation & Results

1. Introduction

After all that we've mentioned in the previous chapter, we concluded that age, gender and ethnicity are important tasks in the field of computer vision and deep learning. These tasks involve analyzing facial images to determine the age, gender and ethnicity/race of people.

The main objectives of this study are to estimate age, gender and ethnicity from images using AI and machine learning. These objectives will be achieved by proposing models based on deep learning using CNNs.

The proposed models will be separate for each task, with one model for age estimation, another for gender recognition, and the third for ethnicity detection, allowing the accuracy of each task to be optimized independently.

2. Data Collection & Training

2.1. Data Requirements

Training data is the foundation of any successful machine learning and deep learning model. To accurately estimate age, gender and ethnicity, we need several requirements to ensure the effectiveness and accuracy of the models.

- **Data diversity**
    - Age diversity: The data should include diverse ages/age groups, ranging from infants to the elderly. This helps in training the model to recognize subtle differences between the different age groups.
    - Gender diversity: The data should have a roughly equal distribution of males and females to ensure that the model can accurately recognize gender differences.
    - Ethnic diversity: The data should include images of people of different races (whites, black, asians, …etc) to ensure that the model can handle racial differences correctly.

- **Data volume and quantity**

To develop a reliable and accurate model, the dataset should be large enough to cover all possible categories. Generally, the larger the dataset, the better the model's generalization and prediction accuracy.

- **Image quality**

The images should cover a variety of lighting conditions and shooting environments (such as indoor and outdoor settings, natural and artifical lighting) to ensure the model can handle different conditions.

- **Dataset choices**

To do the trainings of our project here, lots of datasets can be used, such as Adience dataset, IMDB dataset..etc, and we used a certain dataset called UTKFace, one of the most well known datasets out there, other than Adience itself being the most well known for age and gender.

It was specifically chosen because it has all of the things we need to work on, from age, gender and ethnicity, to a lot of variety in the data, such as multiple ethnic groups and ages and close to perfect percentage of gender distribution.

We also used a dataset that we created by images of people we know and people we found on the internet from actors and actresses to stars and footballers, and it is just to help out the bias of UTKFace so we don't find the need to actually talk about it since it is only less than 500 images.

- **UTKFace dataset**

UTKFace boasts over 20,000 images containing faces captured "in the wild", meaning they depict real-world scenarios with variations in pose, lighting, expression, and image resolution. [24]

Each image comes with annotations for age, gender (male or female), and ethnicity. This allowed us to train and evaluate algorithms that can automatically estimate these attributes from facial features. [24]

The dataset also offers pre-aligned and cropped face images. Additionally, facial landmark annotations (identifying 68 key points on each face) are included,

facilitating tasks like facial pose estimation. Whilst also being available free of charge and for non-commercial purposes. [24]

The dataset gender distribution is close to perfect with males having just 2.3% more images then women :

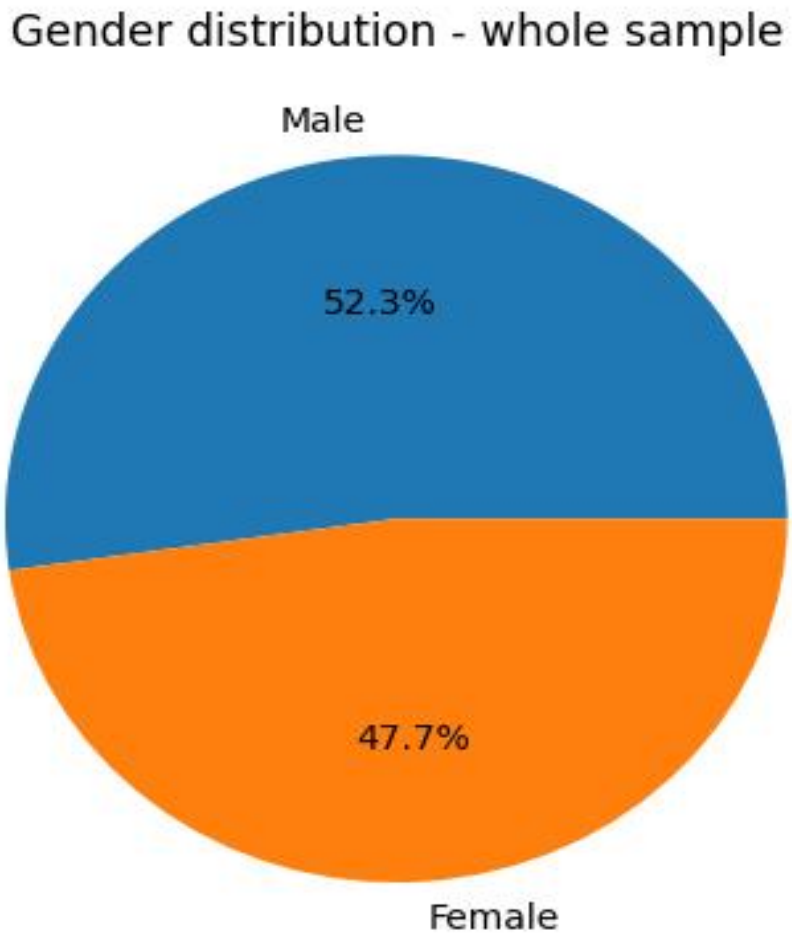

Fig. 14: The distribution of Males & Females in the dataset. [25]

Whilst the distribution of ethnicity is a bit concerning because it may have more white people than wanted, with them being close to half with 42.5%:

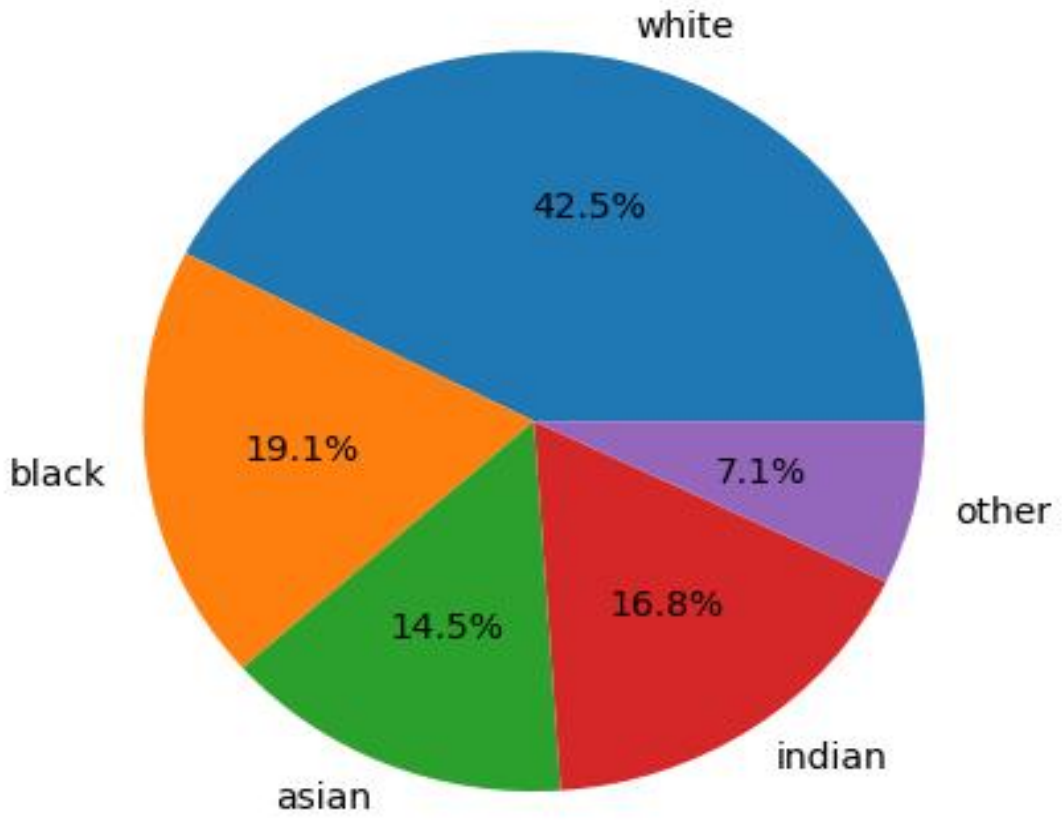

Fig. 15: The distribution of Ethnicities in the dataset. [25]

Age distribution has a lot of people between their 20s and 30s too and very few people above 90:

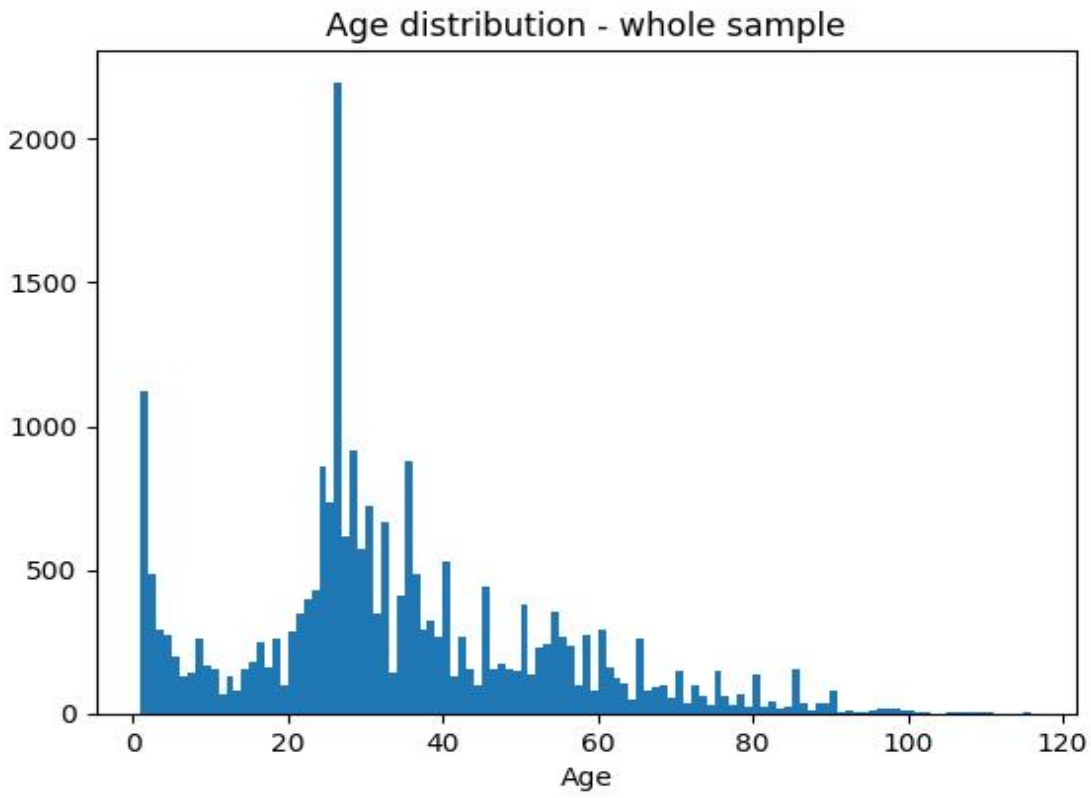

Fig. 16: A graph showing the age distribution of the dataset. [25]

- **Applications and Limitations**

Algorithms can be trained to predict a person's age based on facial features. This has applications in demographics analysis, age verification for online services, and potentially even personalized healthcare. identify the gender of a person from their face. This can be used in security systems or targeted advertising.

Algorithms can also be trained to categorize faces based on ethnicity. However, this is a complex task due to social and cultural factors influencing how ethnicity is categorized.

While UTKFace is a valuable resource it still has its limitations. As with many datasets sourced from the internet, there's a potential for bias in the data. The images might not represent the global population demographics accurately, and might not have a balanced representation of all ethnicities. potentially impacting the performance of algorithms trained on UTKFace.

2.2. Model Proposals and Training

Training and model proposal-wise, we've managed to train all the models with personalized architectures except age which was a pre-trained model that was perfected, meaning, everything was trained from scratch except for age training which was transfer learning, taking help from keras, we've managed to create architectures that mimic VGG16 and VGG19 and we've tried multiple architectures too in order to create the best models for us, that we'd be discussing bit by bit as we move on.

- **Age Training**

Age training here was a hardship to accomplish, it took us most of the time of our work, From trying RestNet50 architecture to train on our dataset, then finding out it wouldn't work because of the usage of the old 'ktrain' library, that gave us a model with the old files of (.h5) and (.preproc), that cannot be imported to the current Tensorflow library, then we managed to create multiple model architectures to try, unfortunately for us, all of the models gave less than 50% accuracy, not forgetting the usage of many other pre-trained models like 'EffecientNetB0' and 'VGG16'.

With all the research that we did to get age recognition to a percentage that would be acceptable in our eyes, we found a pre-trained simple personalized model already trained on another dataset called 'Facial-age' from a study website for machine learning called skillcate[15], with this architecture [Fig. 17]:

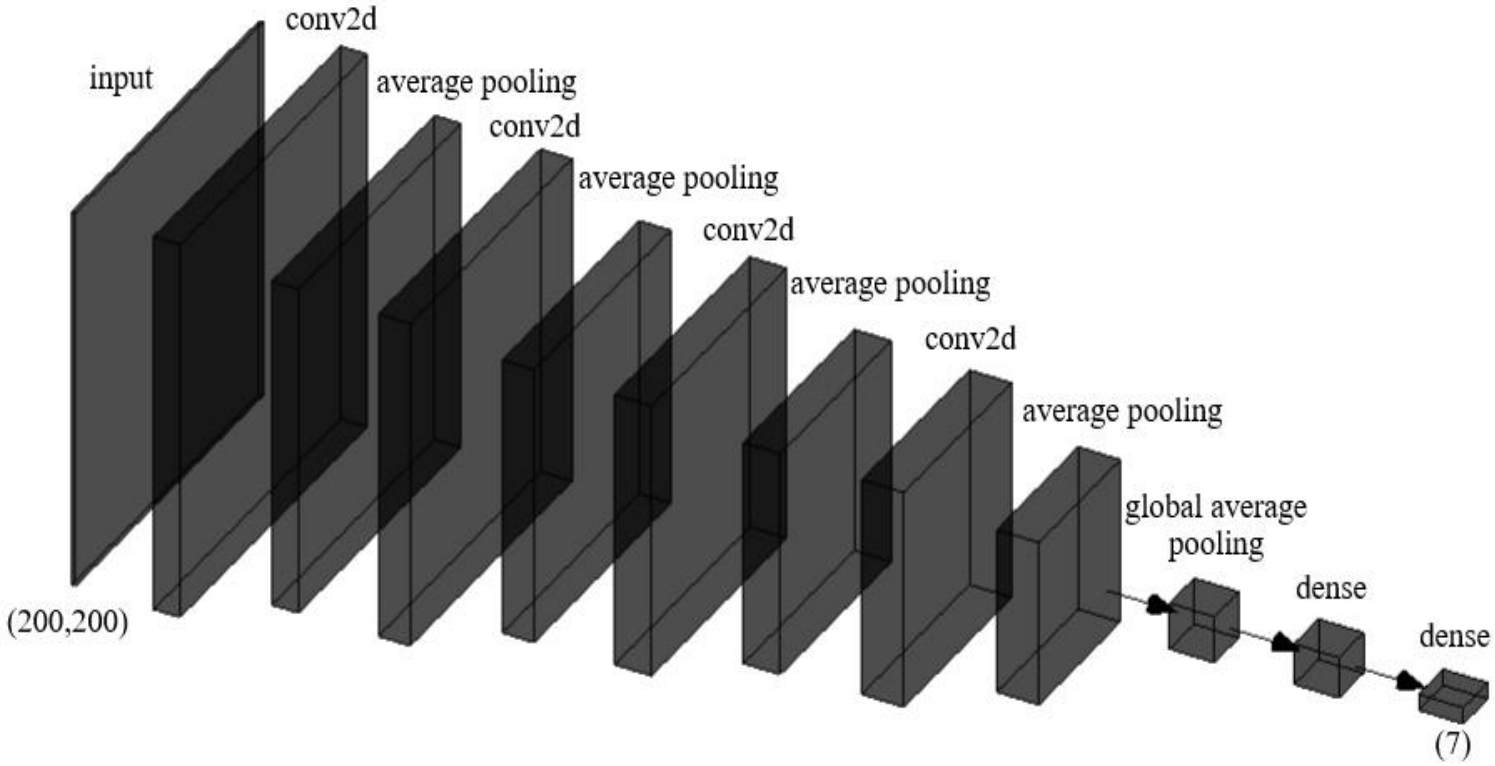

Fig. 17: The architecture of the model used for age training

Also we used a classification function to classify the dataset we have into age groups.

```
[14]: def class_labels_reassign(age):

          if 1 <= age <= 2:
              return 0
          elif 3 <= age <= 9:
              return 1
          elif 10 <= age <= 20:
              return 2
          elif 21 <= age <= 27:
              return 3
          elif 28 <= age <= 45:
              return 4
          elif 46 <= age <= 65:
              return 5
          else:
              return 6
```

Fig. 18: A Code showing a function to return age groups

This function is to classify images into different groups corresponding to which place they belong age-wise using the UTKFace dataset, then using our handmade dataset.

```
path = "C:/Users/HeeBe/Desktop/ML-Ethnicity-Detection-master/ML-Ethnicity-Detection-master/UTKFace/"
pixels = []
age = []
gender = []

i=0
for img in os.listdir(path):
    i=i+1
    ages = class_labels_reassign(int(img.split("_")[0]))
    img = cv2.imread(str(path)+"/"+str(img))
    img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    img=cv2.resize(img,(200, 200))
    pixels.append(np.array(img))
    age.append(np.array(ages))

pixels = np.array(pixels)
age = np.array(age,np.uint64)
len(age)
```

Fig. 19: A Code showing how the data was read and stored

We then took the path of UTKFace that we have stored on our machine, then proceeded to go into every image taking the first part of its name which is the Age and then changing into a grayscale image with one dimension, while resizing it to the amount of pixels we want and putting the target as age, knowing that the features are pixels. We also did the same to the dataset that we created.

```
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()
```

Fig. 20: A Code showing the compiling and how to display the architecture

The first line here compiles the model we imported to set it up for the training process.

The loss function used is 'sparse_categorical_crossentropy', which compares the model's predictions to the actual labels and calculates the error, and it is suitable for multi-class classification problems where labels are integers like ours.

The optimizer used is 'adam', which is quite popular and it adapts the learning rate for each parameter.

The metrics used is 'accuracy', which evaluates the model's performance on a separate validation dataset, here specifically, we are calculating the percentage of correctly classified examples.

The second line displays a summary of the model's architecture.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 198, 198, 32)	320
average_pooling2d (AveragePooling2D)	(None, 99, 99, 32)	0
conv2d_1 (Conv2D)	(None, 97, 97, 64)	18,496
average_pooling2d_1 (AveragePooling2D)	(None, 48, 48, 64)	0
conv2d_2 (Conv2D)	(None, 46, 46, 128)	73,856
average_pooling2d_2 (AveragePooling2D)	(None, 23, 23, 128)	0
conv2d_3 (Conv2D)	(None, 21, 21, 256)	295,168
average_pooling2d_3 (AveragePooling2D)	(None, 10, 10, 256)	0
global_average_pooling2d (GlobalAveragePooling2D)	(None, 256)	0
dense (Dense)	(None, 132)	33,924
dense_1 (Dense)	(None, 7)	931

Tab. 3: A table showing the summary of age model

It has 4 convolutional layers with increasing numbers of filters, each filter learns to detect specific patterns in the image.

The model consists of 4 average pooling layers after every convolutional layer, they are used to reduce the image size, making the model more efficient.

Global average pooling is used here to take the average of all elements in each feature channel, resulting in a single vector.

At the end, the usage of 2 dense layers, to combine the extracted features and make the final predictions, with the last dense layer having 7 neurons each neuron meaning a different age group.

```
model_path='age75.keras'
checkpointer = ModelCheckpoint(model_path, monitor='loss',verbose=1,save_best_only=True,
                               save_weights_only=False, mode='auto',save_freq='epoch')
callback_list=[checkpointer]
```

Fig. 21: A Code showing the callback_list and the model name to be saved

The first line here shows what the model will be saved as and where.

The second line shows a type of callback function used in ‘keras’ that goes by the name of ‘ModelCheckpoint’, and it is used to monitor the training process and save the model at specific points.

The parameters used in the ModelCheckpoint are:

- monitor=’loss’: which is used to monitor the loss function during the training.
- verbose=1: which enables verbose output during training, telling us when the model is saved.
- save_best_only=True: this makes the callback save the model only when the monitored metric (‘loss’ in our case) improves, and would overwrite the previous model in the model_path we specified.
- save_weights_only=False: this specifies saving the entire model instead of just the weights.

- mode='auto': this automatically selects the direction to monitor for improvement (to minimize the loss in our case).
- save_freq='epoch': this saves the model after every epoch.

Now the callback_list is a list containing the checkpointer object, and it would be passed to the model's training function to enable the callback functionality during the training.

```
save = model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=5,callbacks=[callback_list])
Epoch 1/5
556/556 ━━━━━━━━━━━━━━━━━━━━ 0s 432ms/step - accuracy: 0.9263 - loss: 0.2074
Epoch 1: loss improved from inf to 0.23365, saving model to age75.keras
556/556 ━━━━━━━━━━━━━━━━━━━━ 262s 471ms/step - accuracy: 0.9262 - loss: 0.2074 - val_accuracy: 0.6470 - val_loss: 1.3212
Epoch 2/5
556/556 ━━━━━━━━━━━━━━━━━━━━ 0s 434ms/step - accuracy: 0.9272 - loss: 0.1972
Epoch 2: loss improved from 0.23365 to 0.22657, saving model to age75.keras
556/556 ━━━━━━━━━━━━━━━━━━━━ 265s 476ms/step - accuracy: 0.9272 - loss: 0.1973 - val_accuracy: 0.6681 - val_loss: 1.1427
Epoch 3/5
556/556 ━━━━━━━━━━━━━━━━━━━━ 0s 448ms/step - accuracy: 0.9420 - loss: 0.1631
Epoch 3: loss improved from 0.22657 to 0.20858, saving model to age75.keras
556/556 ━━━━━━━━━━━━━━━━━━━━ 275s 495ms/step - accuracy: 0.9419 - loss: 0.1632 - val_accuracy: 0.6475 - val_loss: 1.1978
Epoch 4/5
556/556 ━━━━━━━━━━━━━━━━━━━━ 0s 449ms/step - accuracy: 0.9163 - loss: 0.2251
Epoch 4: loss did not improve from 0.20858
556/556 ━━━━━━━━━━━━━━━━━━━━ 274s 493ms/step - accuracy: 0.9163 - loss: 0.2251 - val_accuracy: 0.6575 - val_loss: 1.1768
Epoch 5/5
556/556 ━━━━━━━━━━━━━━━━━━━━ 0s 450ms/step - accuracy: 0.9427 - loss: 0.1596
Epoch 5: loss improved from 0.20858 to 0.18813, saving model to age75.keras
556/556 ━━━━━━━━━━━━━━━━━━━━ 276s 497ms/step - accuracy: 0.9427 - loss: 0.1596 - val_accuracy: 0.6470 - val_loss: 1.4017
```

Fig. 22: A Code showing the training of the age model

As we specified earlier, the model was pre-trained on another dataset called (Facial-age) for 60 epochs and we trained it for another 15 epochs to get somewhat stable results with an accuracy of 64% on the validation set and an accuracy of 94% on the training set and without forgetting to mention that every 5 iterations we save a model with a different name, this is to prevent the over-fitting that that may be visible here in the 75th iteration, that is why we had to create another dataset to also work with to help stabilize the over-fitting of this training accuracy here and somewhat we got better results on multiple people.

About the 'fit' method, it is a method on the 'model' object responsible for training the model.

'x_train' and 'y_train' are training data for the model to learn from, representing the input features and target labels respectively.

'x_test' and 'y_test' representing the input features and target labels for the validation data, while the performance of the model is evaluated on this data to avoid over-fitting.

'epochs' here specifies how many iterations our model will be trained for.

Finally, 'callback' containing 'callback_list' that we defined and talked about earlier, that tells the 'fit' method to use the functions in it.

- **Gender**

The training of gender wasn't as hard as training age, but we wouldn't lie and say it was easy, because we did have some problems stabilizing identification between males and females, we ended up making a simple yet efficient architecture model inspired from VGG networks and ResNet, using this architecture here:

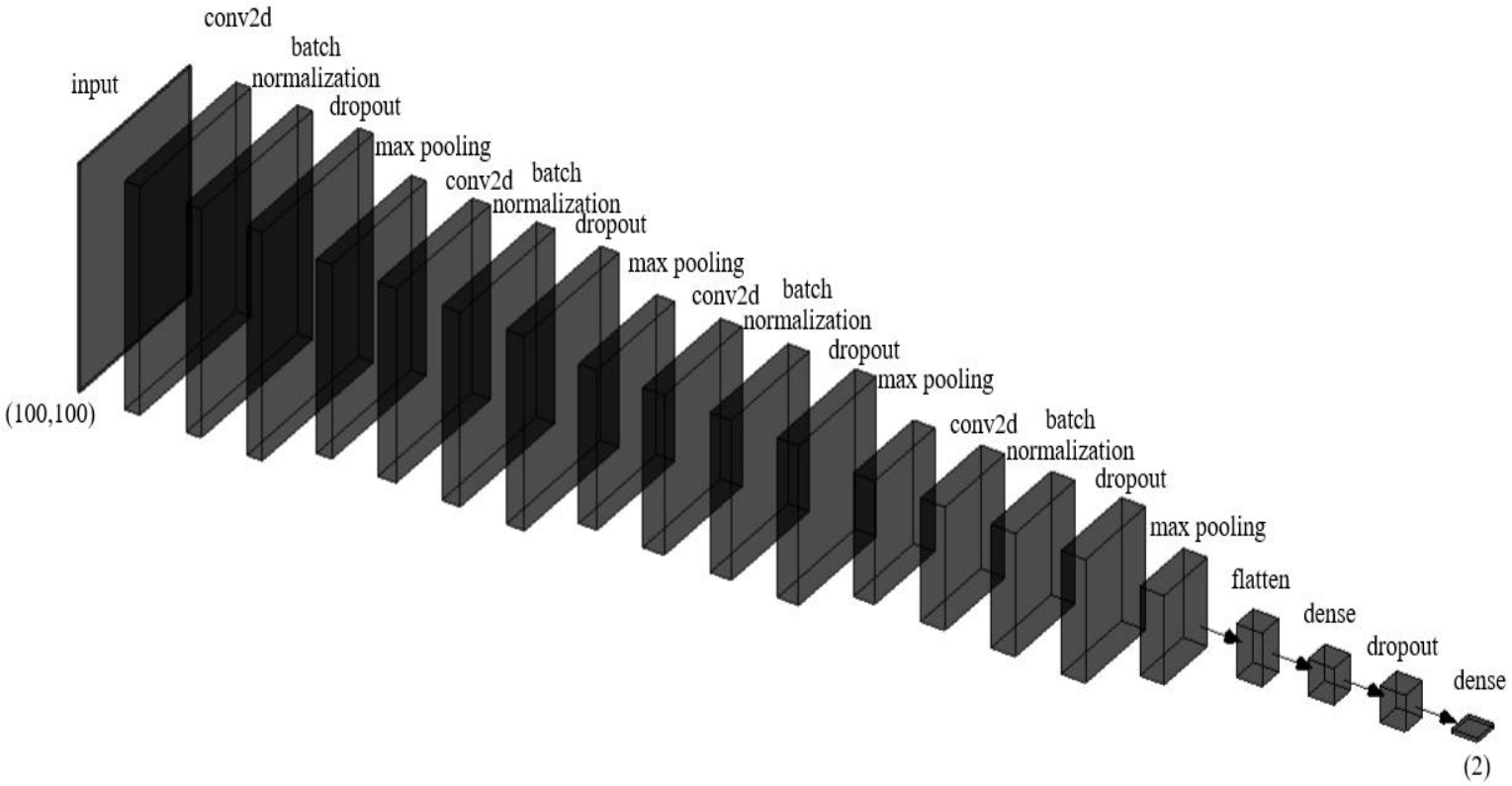

Fig. 23: The architecture of the gender model

And the code responsible for its creation is in [Fig. 24].

[4]:
```
  # Input layer for grayscale image
inputs = layers.Input(shape = (100,100,1))

  # Block 1 - Feature extraction with smaller filters for grayscale
x = layers.Conv2D(32, kernel_size=(3, 3), activation="relu", padding="same")(inputs)
x = layers.BatchNormalization(axis=-1)(x)  # Batch normalization for stability
x = layers.Dropout(0.2)(x)
x = layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2))(x)

  # Block 2 - Feature extraction with increased filters
x = layers.Conv2D(64, kernel_size=(3, 3), activation="relu", padding="same")(x)
x = layers.BatchNormalization(axis=-1)(x)
x = layers.Dropout(0.2)(x)
x = layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2))(x)

  # Block 3 - Feature extraction with increased filters
x = layers.Conv2D(128, kernel_size=(3, 3), activation="relu", padding="same")(x)
x = layers.BatchNormalization(axis=-1)(x)
x = layers.Dropout(0.2)(x)
x = layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2))(x)

  # Block 4 - Feature extraction with increased filters
x = layers.Conv2D(256, kernel_size=(3, 3), activation="relu", padding="same")(x)
x = layers.BatchNormalization(axis=-1)(x)
x = layers.Dropout(0.2)(x)
x = layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2))(x)

  # Flatten layer
x = layers.Flatten()(x)

  # Dense layers for classification
x = layers.Dense(512, activation="relu")(x)  # Increased hidden units for potentially more complex decision boundaries
x = layers.Dropout(0.3)(x)
outputs = layers.Dense(2, activation="softmax")(x)  # Softmax for multi-class classification
```

Fig. 24: A Code showing the architecture used

Here, we used 4 convolutional layers with increasing numbers of filters as we go deeper into the network.

4 ‘BatchNormalization’ layers after every convolutional layer which helps improve the training process by normalizing the activations of the previous layer, making the network less sensitive to initialization and improving training speed.

5 dropout layers after every ‘BatchNormalization’ layer, this layer randomly drops a certain percentage of activations during training, helping to avoid and prevent over-fitting by forcing the network to learn more robust features that are not dependent on specific neurons.

And 4 pooling layers after every dropout layer except the last one, and they are used to reduce the image size to decrease the model's complexity and prevent over-fitting.

And a flatten layer that is used to transform the extracted features from a multi-dimensional tensor (in our case just one) into a single long vector, which is suitable for dense layers, which we have 2 of, and these layers take the flattened features from the previous layer and make predictions about the image category.

Now using the same technique and code we used for age training to compile the model and the same callback functions. We go immediately to fit it and train it.

```
[7]: save = model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=10,callbacks=[callback_list])

Epoch 1/10
556/556 [==============================] - ETA: 0s - loss: 0.7492 - accuracy: 0.7351
Epoch 1: loss improved from inf to 0.74922, saving model to age1model.keras
556/556 [==============================] - 155s 277ms/step - loss: 0.7492 - accuracy: 0.7351 - val_loss: 0.3923 - val_accuracy: 0.8358
Epoch 2/10
556/556 [==============================] - ETA: 0s - loss: 0.3432 - accuracy: 0.8516
Epoch 2: loss improved from 0.74922 to 0.34323, saving model to age1model.keras
556/556 [==============================] - 155s 278ms/step - loss: 0.3432 - accuracy: 0.8516 - val_loss: 0.3504 - val_accuracy: 0.8448
Epoch 3/10
556/556 [==============================] - ETA: 0s - loss: 0.2927 - accuracy: 0.8734
Epoch 3: loss improved from 0.34323 to 0.29266, saving model to age1model.keras
556/556 [==============================] - 154s 277ms/step - loss: 0.2927 - accuracy: 0.8734 - val_loss: 0.3055 - val_accuracy: 0.8660
Epoch 4/10
556/556 [==============================] - ETA: 0s - loss: 0.2669 - accuracy: 0.8873
Epoch 4: loss improved from 0.29266 to 0.26691, saving model to age1model.keras
556/556 [==============================] - 154s 277ms/step - loss: 0.2669 - accuracy: 0.8873 - val_loss: 0.3370 - val_accuracy: 0.8487
Epoch 5/10
556/556 [==============================] - ETA: 0s - loss: 0.2441 - accuracy: 0.8958
Epoch 5: loss improved from 0.26691 to 0.24411, saving model to age1model.keras
556/556 [==============================] - 154s 277ms/step - loss: 0.2441 - accuracy: 0.8958 - val_loss: 0.2623 - val_accuracy: 0.8865
Epoch 6/10
556/556 [==============================] - ETA: 0s - loss: 0.2268 - accuracy: 0.9057
Epoch 6: loss improved from 0.24411 to 0.22683, saving model to age1model.keras
556/556 [==============================] - 154s 277ms/step - loss: 0.2268 - accuracy: 0.9057 - val_loss: 0.2994 - val_accuracy: 0.8780
Epoch 7/10
556/556 [==============================] - ETA: 0s - loss: 0.2117 - accuracy: 0.9121
Epoch 7: loss improved from 0.22683 to 0.21167, saving model to age1model.keras
556/556 [==============================] - 154s 277ms/step - loss: 0.2117 - accuracy: 0.9121 - val_loss: 0.3013 - val_accuracy: 0.8760
Epoch 8/10
556/556 [==============================] - ETA: 0s - loss: 0.1996 - accuracy: 0.9167
Epoch 8: loss improved from 0.21167 to 0.19961, saving model to age1model.keras
556/556 [==============================] - 154s 278ms/step - loss: 0.1996 - accuracy: 0.9167 - val_loss: 0.3188 - val_accuracy: 0.8765
Epoch 9/10
556/556 [==============================] - ETA: 0s - loss: 0.1885 - accuracy: 0.9211
Epoch 9: loss improved from 0.19961 to 0.18853, saving model to age1model.keras
556/556 [==============================] - 154s 277ms/step - loss: 0.1885 - accuracy: 0.9211 - val_loss: 0.3072 - val_accuracy: 0.8954
Epoch 10/10
556/556 [==============================] - ETA: 0s - loss: 0.1787 - accuracy: 0.9274
Epoch 10: loss improved from 0.18853 to 0.17872, saving model to age1model.keras
556/556 [==============================] - 154s 277ms/step - loss: 0.1787 - accuracy: 0.9274 - val_loss: 0.3038 - val_accuracy: 0.8863
```

Fig. 25: A Code showing the training of the gender model

Here, the training of the model to 10 epochs, knowing that we trained the model even more than this, and we didn't achieve a higher validation accuracy, which was stable at 88%, and the accuracy of the training data was 92% and evolved to be 96% and more using further training, and training the model using the other dataset that we created made the model better going up to 96% to each of them (training and validation accuracy).

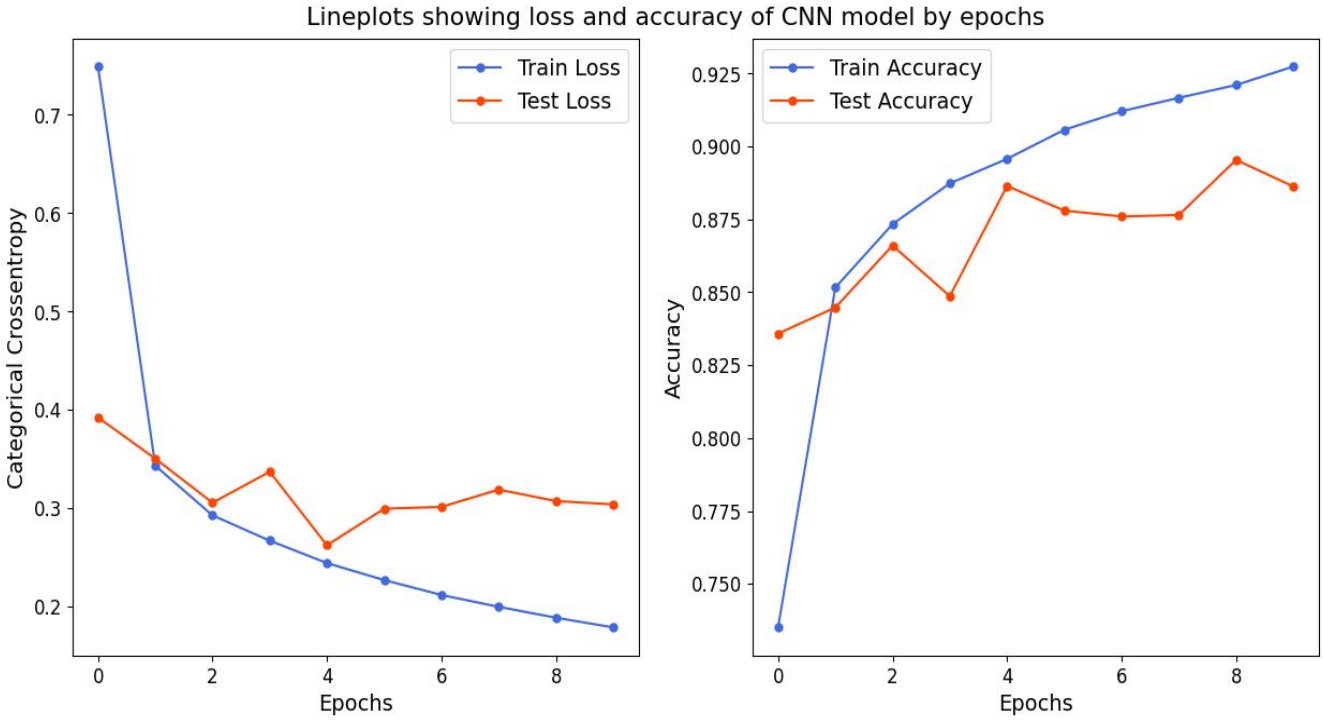

Fig. 26: A graph showing loss and accuracy of the model by epochs

Here is the graph to see how the accuracy and loss of the testing and training sets used was going till the 10th epoch.

Other than that we managed to create a somewhat stable gender estimation model even though it was a hard job to accomplish at first.

- **Ethnicity**

Ethnicity training wasn't the hardest nor was it the easiest, we tried some personalized architectures and we ended up with one that works nicely most of the times with 77% up to 80% accuracy on the validation samples and we trained it for

over 50 epochs saving a copy of the model every 5 epochs, using this architecture [fig. 27]:

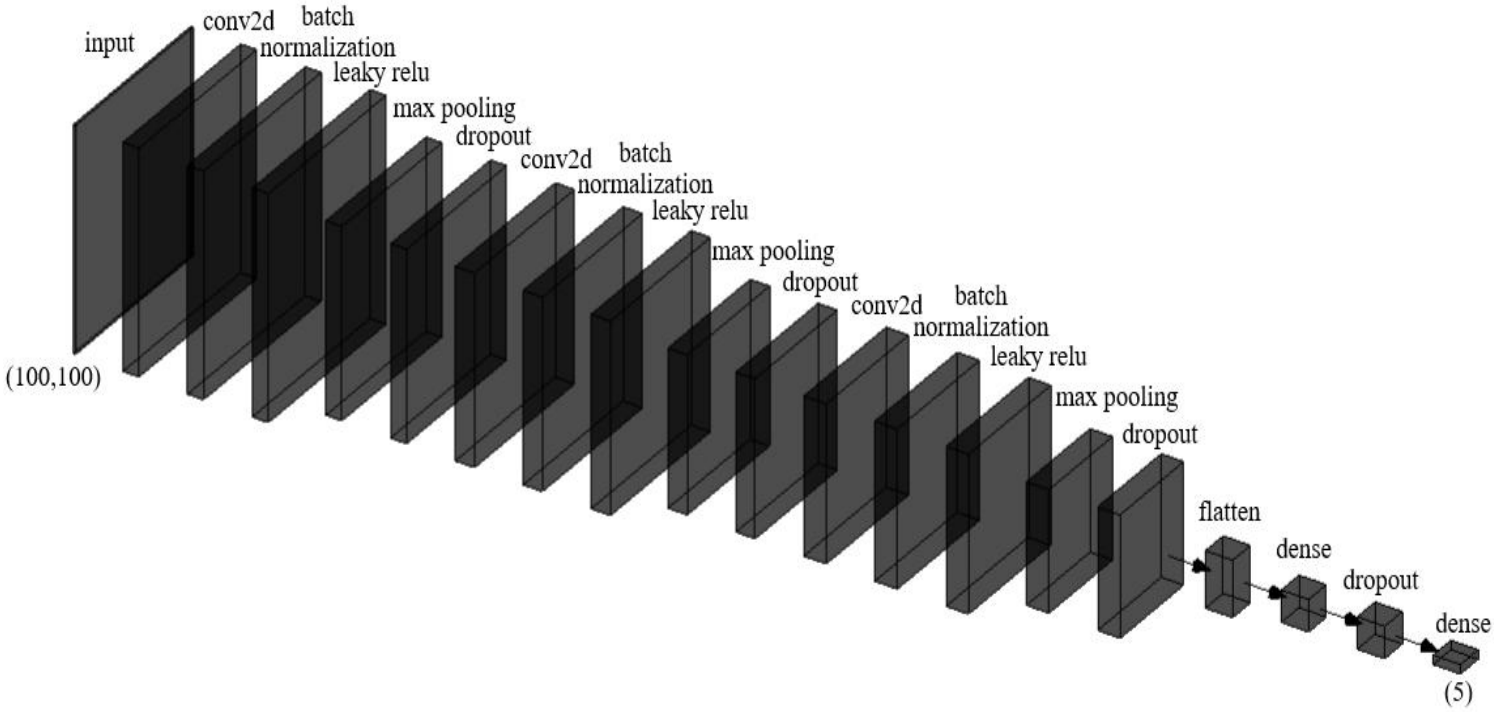

Fig. 27: The architecture of the ethnicity model

Using this architecture we created the model with the code on [Fig. 28]:

```
# Define the CNN architecture
def build_race_classifier():
  model = Sequential()
  model.add(Conv2D(32, (3, 3), padding='same', activation='relu', input_shape=(100, 100, 1)))
  model.add(BatchNormalization())
  model.add(LeakyReLU(alpha=0.1))  # Leaky ReLU for better gradient flow
  model.add(MaxPooling2D((2, 2)))
  model.add(Dropout(0.25))  # Dropout for regularization

  model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))
  model.add(BatchNormalization())
  model.add(LeakyReLU(alpha=0.1))
  model.add(MaxPooling2D((2, 2)))
  model.add(Dropout(0.25))

  model.add(Conv2D(128, (3, 3), padding='same', activation='relu'))
  model.add(BatchNormalization())
  model.add(LeakyReLU(alpha=0.1))
  model.add(MaxPooling2D((2, 2)))
  model.add(Dropout(0.25))

  model.add(Flatten())
  model.add(Dense(256, activation='relu'))
  model.add(Dropout(0.5))  # Higher dropout for fully-connected layers
  model.add(Dense(5, activation='softmax'))  # 5 denotes the number of race classes

  # Compile the model with Adam optimizer and categorical crossentropy loss
  model.compile(loss='sparse_categorical_crossentropy', optimizer=Adam(0.001), metrics=['accuracy'])
  return model

# Create the model instance
model = build_race_classifier()
```

Fig. 28: A code showing the architecture used for the model as a code

The code in the snippet shows the model's architecture starting by defining a function that we name 'build_race_classifier', then we used 'sequential' from keras to stack the layers sequentially.

We used 3 convolutional layers with increasing filter numbers, all of them use the 'same' padding and a kernel size of '(3,3)' to maintain the dimensions of the image, and a 'relu' activation that adds non-linearity.

After every convolutional layer, there is a 'BatchNormalization' layer, which improves the training stability.

After every ‘BatchNormalization’ layer, we added a leaky relu layer, that allows for a small positive gradient for negative inputs, helping the training.

And after each leaky relu layer, there is a max pooling layer, that is used to capture spatial information and reduce the size of the images.

We also have 4 dropout layers, that randomly drop the desired amount of activations during training to prevent over-fitting (0.25 after each max pooling, 0.5 after the before-last dense layer).

A flatten layer to transform the 2D feature maps into a one dimensional vector.

And 2 dense layers, one with 256 units and a ‘relu’ activation function to perform classification, and the other with 5 units (the nunmber of classes we have) and a ‘softmax’ actiavtion function that outputs probabilites for each race/ethnicity.

The model compiling is the same as the compiling of age and gender recognition models, the only addon here is that, ‘adam’ optimizer has a learning rate of 0.001.

Finally, the function ‘build_race_classifier’ returns the model to us and then we built it using the last line shown in the snippet.

```
model.summary()
```

```
Model: "sequential_8"
```

Layer (type)	Output Shape	Param #
conv2d_25 (Conv2D)	(None, 100, 100, 32)	320
batch_normalization_25 (BatchNormalization)	(None, 100, 100, 32)	128
leaky_re_lu_23 (LeakyReLU)	(None, 100, 100, 32)	0
max_pooling2d_22 (MaxPooling2D)	(None, 50, 50, 32)	0
dropout_28 (Dropout)	(None, 50, 50, 32)	0
conv2d_26 (Conv2D)	(None, 50, 50, 64)	18,496
batch_normalization_26 (BatchNormalization)	(None, 50, 50, 64)	256
leaky_re_lu_24 (LeakyReLU)	(None, 50, 50, 64)	0
max_pooling2d_23 (MaxPooling2D)	(None, 25, 25, 64)	0
dropout_29 (Dropout)	(None, 25, 25, 64)	0
conv2d_27 (Conv2D)	(None, 25, 25, 128)	73,856
batch_normalization_27 (BatchNormalization)	(None, 25, 25, 128)	512
leaky_re_lu_25 (LeakyReLU)	(None, 25, 25, 128)	0
max_pooling2d_24 (MaxPooling2D)	(None, 12, 12, 128)	0
dropout_30 (Dropout)	(None, 12, 12, 128)	0
flatten_7 (Flatten)	(None, 18432)	0
dense_15 (Dense)	(None, 256)	4,718,848
dropout_31 (Dropout)	(None, 256)	0
dense_16 (Dense)	(None, 5)	1,285

Tab. 4: A table showing the summary of ethnicity model

This table here, shows the model architecture better as a table.

Using the same method to compile the model. We finally move on to the training of the model using 'model.fit', while saving a new copy of the model every 5 epochs to try on different photos later to choose the best performance.

The snippet here shows the first 5 epochs of the training and we ended up preferring the usage of the model that has 77% accuracy on the validation set and 77% accuracy on the training set, to then retrain it on the dataset that we created to perfect it even more hoping for the best, not negating that the 77% validation and training set accuracy model has in some cases worse accuracy at finding other groups such as blacks and asians then the models that we've trained for more than 50 epochs but that one was the most less-biased model yet till we added our dataset to improve it, and we did see an improvement thankfully.

```
save = model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=5,callbacks=[callback_list])
```

```
Epoch 1/5
556/556 ━━━━━━━━━━━━━━━━━━━━ 0s 264ms/step - accuracy: 0.4072 - loss: 3.0958
Epoch 1: loss improved from inf to 1.76064, saving model to ethnicity.keras
556/556 ━━━━━━━━━━━━━━━━━━━━ 158s 280ms/step - accuracy: 0.4073 - loss: 3.0934 - val_accuracy: 0.5519 - val_loss: 1.1308
Epoch 2/5
556/556 ━━━━━━━━━━━━━━━━━━━━ 0s 260ms/step - accuracy: 0.5300 - loss: 1.1816
Epoch 2: loss improved from 1.76064 to 1.16208, saving model to ethnicity.keras
556/556 ━━━━━━━━━━━━━━━━━━━━ 153s 275ms/step - accuracy: 0.5300 - loss: 1.1816 - val_accuracy: 0.6298 - val_loss: 0.9683
Epoch 3/5
556/556 ━━━━━━━━━━━━━━━━━━━━ 0s 254ms/step - accuracy: 0.5785 - loss: 1.0640
Epoch 3: loss improved from 1.16208 to 1.05447, saving model to ethnicity.keras
556/556 ━━━━━━━━━━━━━━━━━━━━ 150s 269ms/step - accuracy: 0.5785 - loss: 1.0640 - val_accuracy: 0.5382 - val_loss: 1.0819
Epoch 4/5
556/556 ━━━━━━━━━━━━━━━━━━━━ 0s 266ms/step - accuracy: 0.6291 - loss: 0.9802
Epoch 4: loss improved from 1.05447 to 0.97265, saving model to ethnicity.keras
556/556 ━━━━━━━━━━━━━━━━━━━━ 158s 284ms/step - accuracy: 0.6291 - loss: 0.9802 - val_accuracy: 0.7233 - val_loss: 0.8521
Epoch 5/5
556/556 ━━━━━━━━━━━━━━━━━━━━ 0s 269ms/step - accuracy: 0.6545 - loss: 0.9229
Epoch 5: loss improved from 0.97265 to 0.92867, saving model to ethnicity.keras
556/556 ━━━━━━━━━━━━━━━━━━━━ 159s 286ms/step - accuracy: 0.6545 - loss: 0.9229 - val_accuracy: 0.7110 - val_loss: 0.8416
```

Fig. 29: A code showing the training of the model

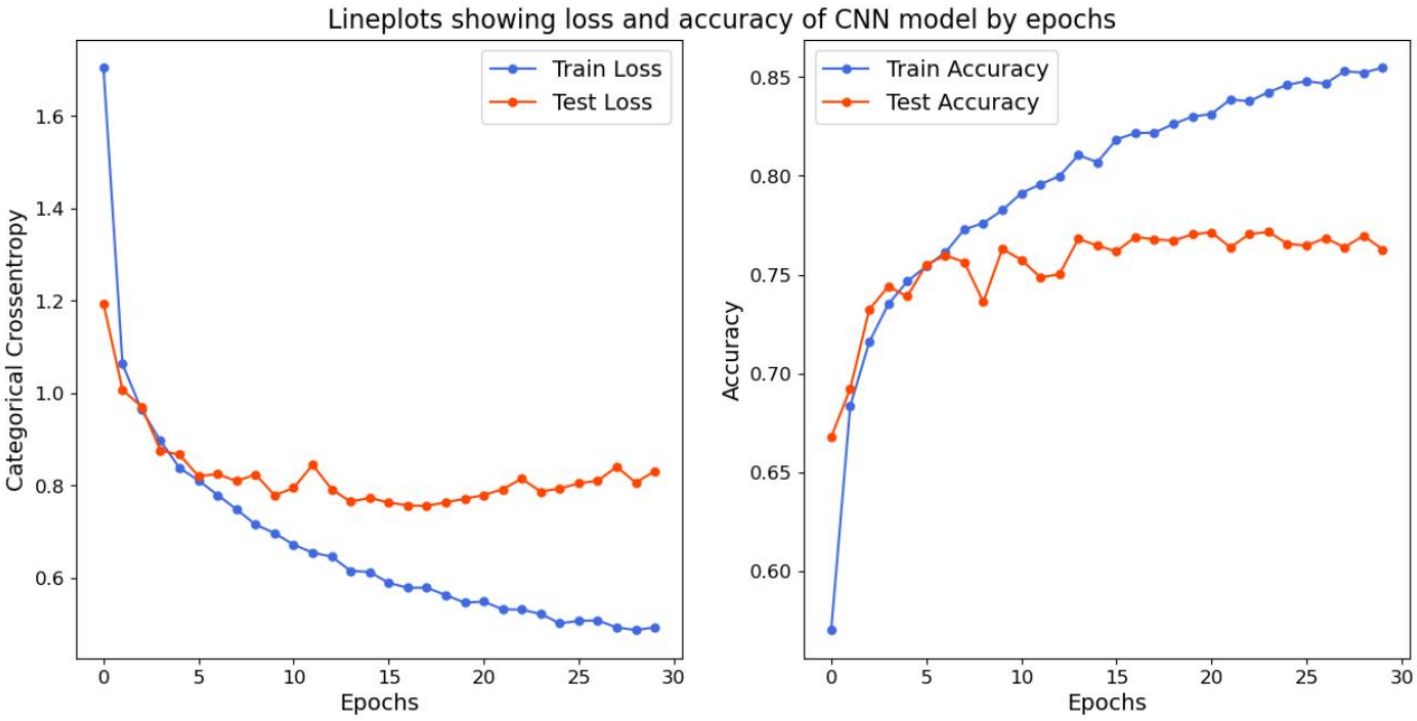

Fig. 30: A graph showing the loss and accuracy of the first 5 epochs

This graph shows the first 5 epochs and how they were going.

We saved lots of models in ethnicity, even more than gender and age to try and find the best model and the less discriminatory one, because manually doing it seemed to us the best way to actually validate it alongside the accuracy of it in both the validation and training data without forgetting the loss.

3. Results & Evaluation:

After completing the training phase, we are moving to the implementation phase to see the results and evaluate it, which involves trying out the model in the real world using ourselves and also using some pictures found on the internet to see if it would work correctly for the most part then evaluating them.

3.1. Results

Firstly, when it comes to the implementation of our model we used 2 specific codes, one to try on images and one to try on ourselves using the camera of the computer.

Also, it is essential to present some visual results from the trained models to see and evaluate their performances next.

- **Implementing using images**

We tried on many photos and we're going to show them after explaining the models that we are using. So, some of the models that we thought are good at the moment are:

- mds1: Age trained for 10 epochs on UTKFace and perfected on our dataset for 4 epochs, Gender trained for 20 epochs on UTKFace then 6 times on half of our dataset and retrained for another 1 epoch on our dataset after we enlarged it, ethnicity trained for 25 epochs on UTKFace only.
- mds2: Age trained one time more than mds1's model, Gender trained 2 more times over the mds1's model, Ethnicity trained 6 times over mds1's model on our dataset then one time on our enlarged dataset.
- mds3: Age trained for 15 epochs on UTKFace, Gender trained for 20 epochs on UTKFace, Ethnicity Trained 3 times over mds2's model on our dataset.

Now we are going to show some of the images that we tried these models on and evaluate them which is better than which:

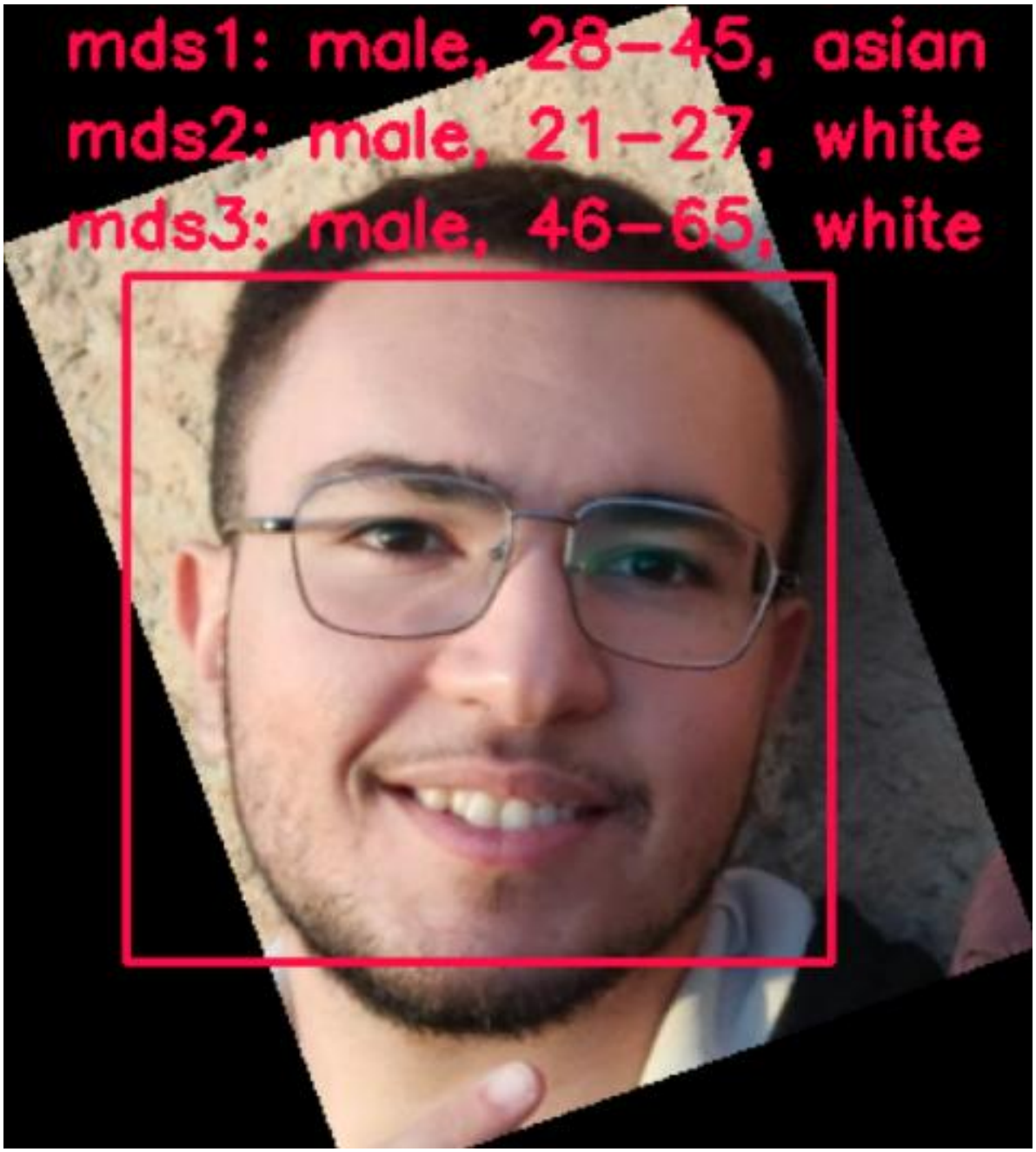

Fig. 31: An image showing the predictions of the chosen models

We can see here that the models chosen on 2 are better at recognizing (Zehani Mohamed Hached) with accurately guessing everything.

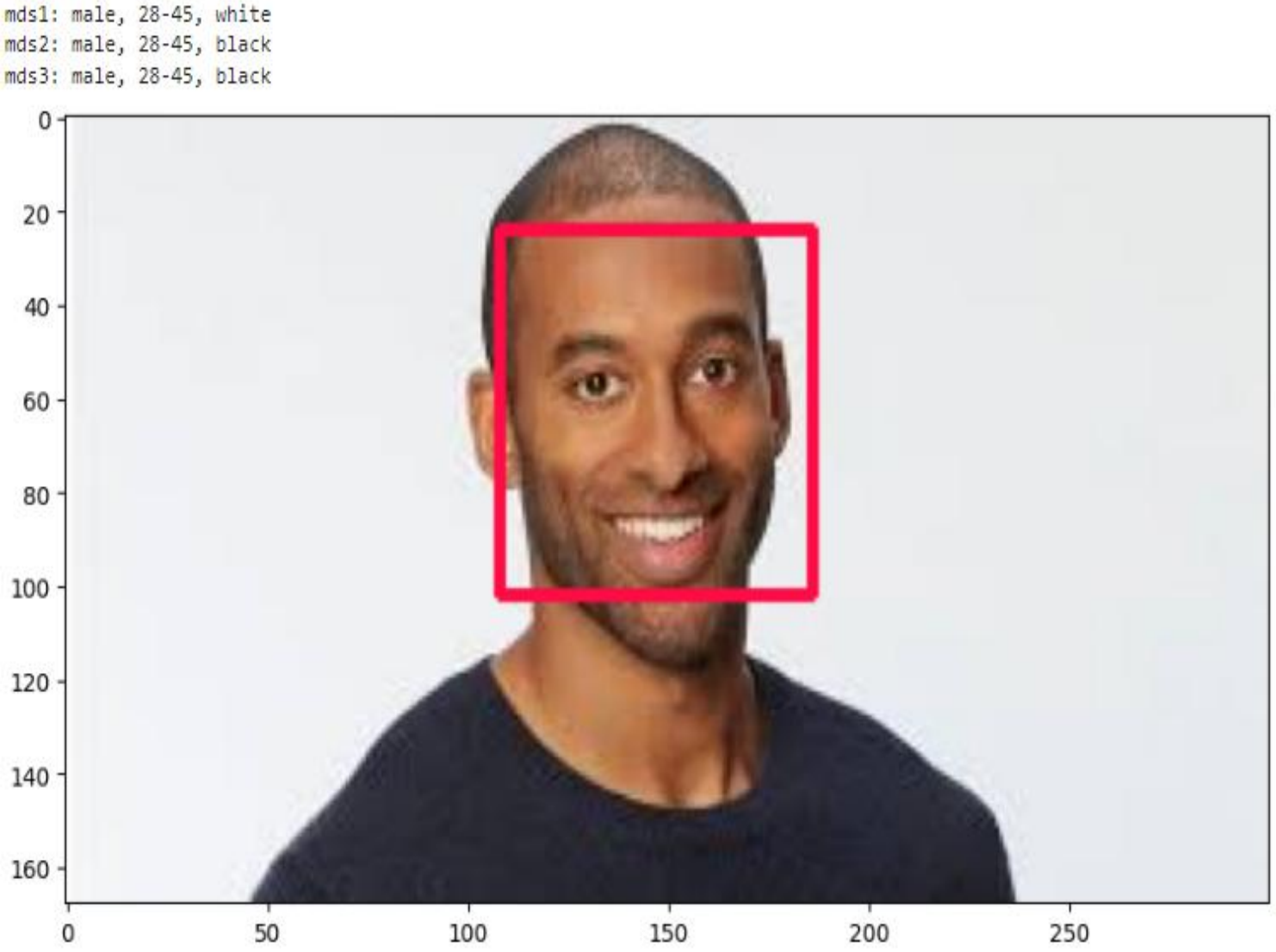

Fig. 32: An image showing another prediction of the models

As we see, the models chosen in mds2 are still guessing accurately in this photo.

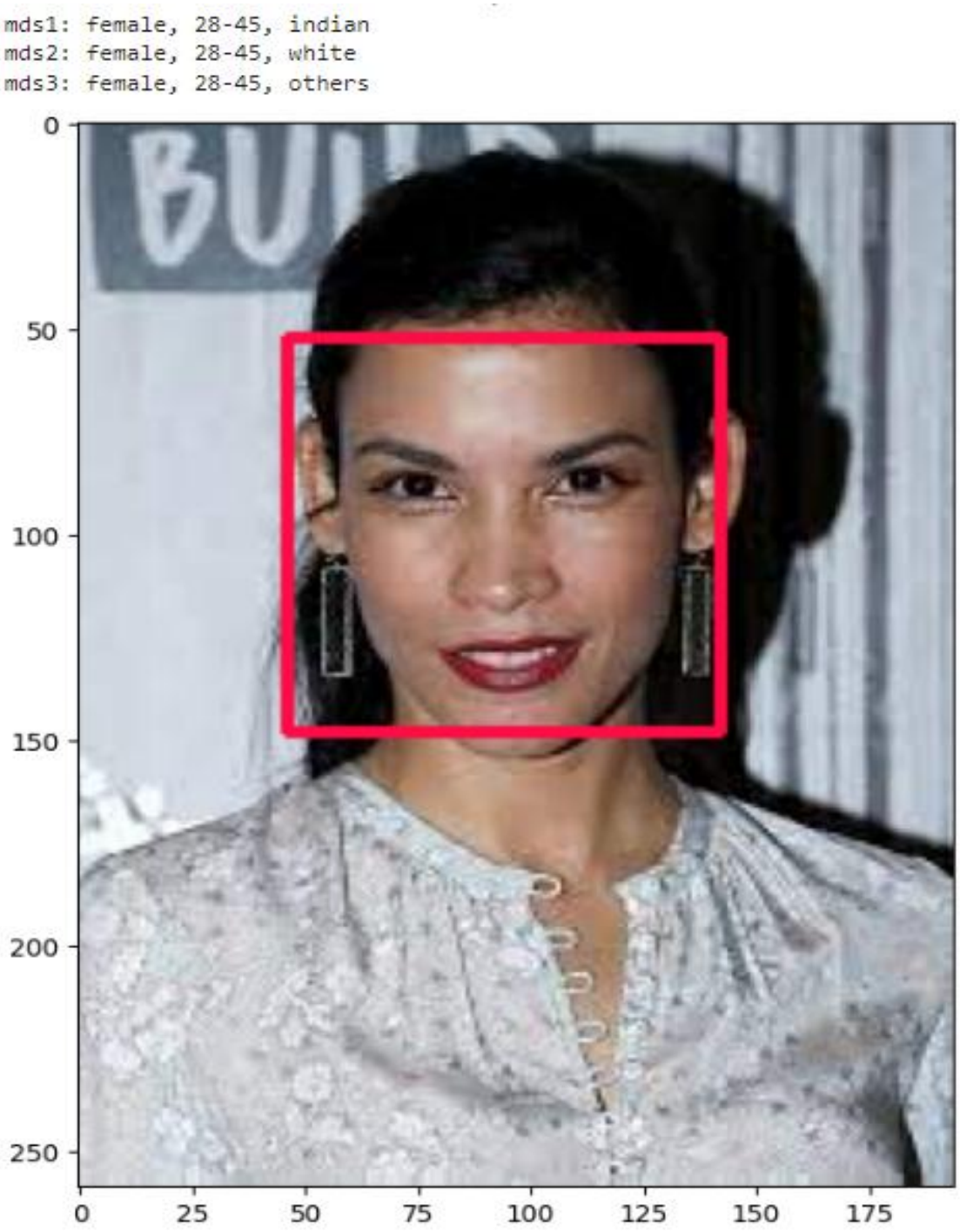

Fig. 33: An image with a different prediction

Here, we can see that the models chosen in 2 were barely correct, because this person belongs to the 5th ethnic group "others", and the third model's ethnicity was on point.

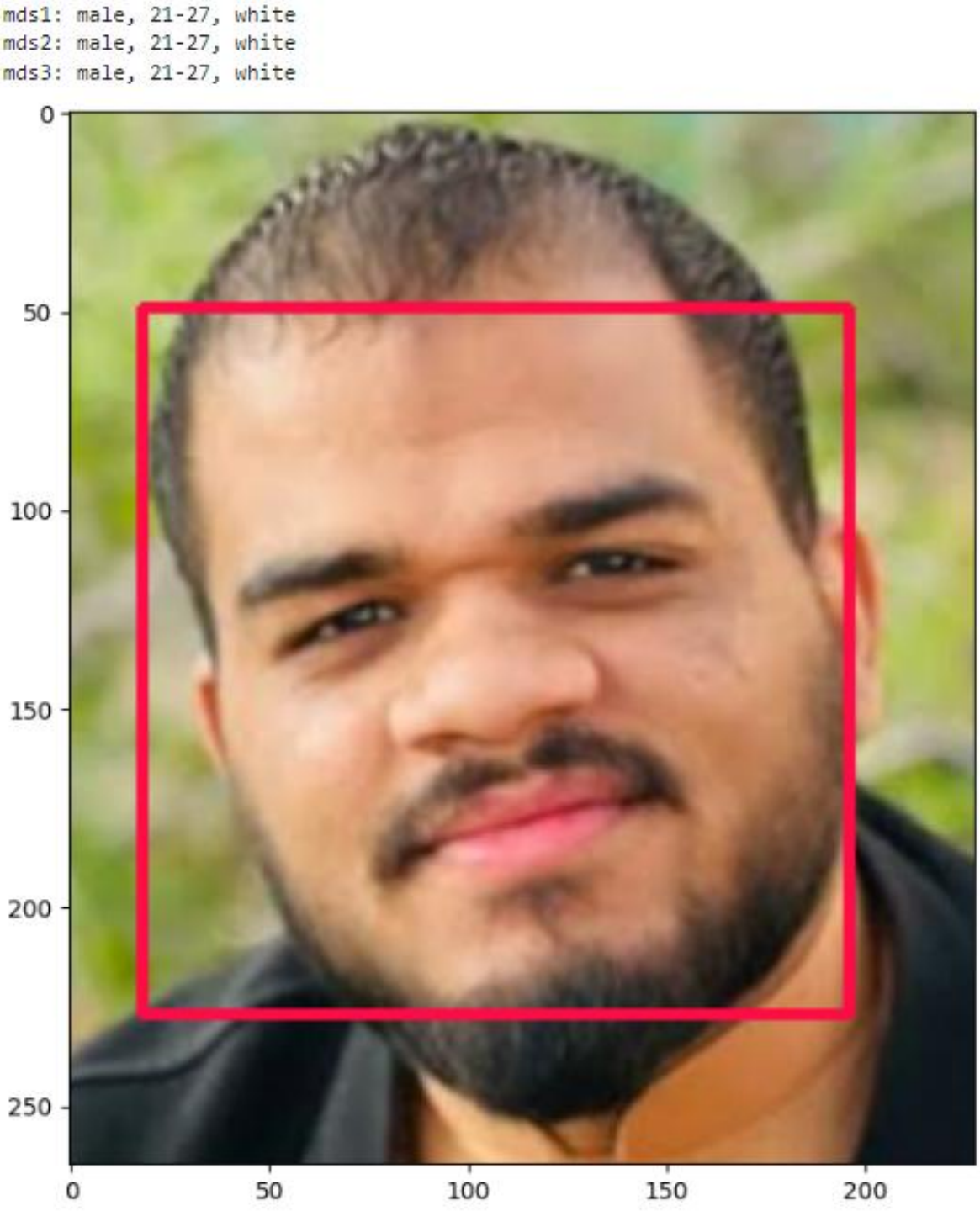

Fig. 34: An image showing another prediction of the models

All of these models were close enough guessing (Doucene Salah Eddine), with the ethnicity preferably being either “others” or ”black”, for us to consider it correct.

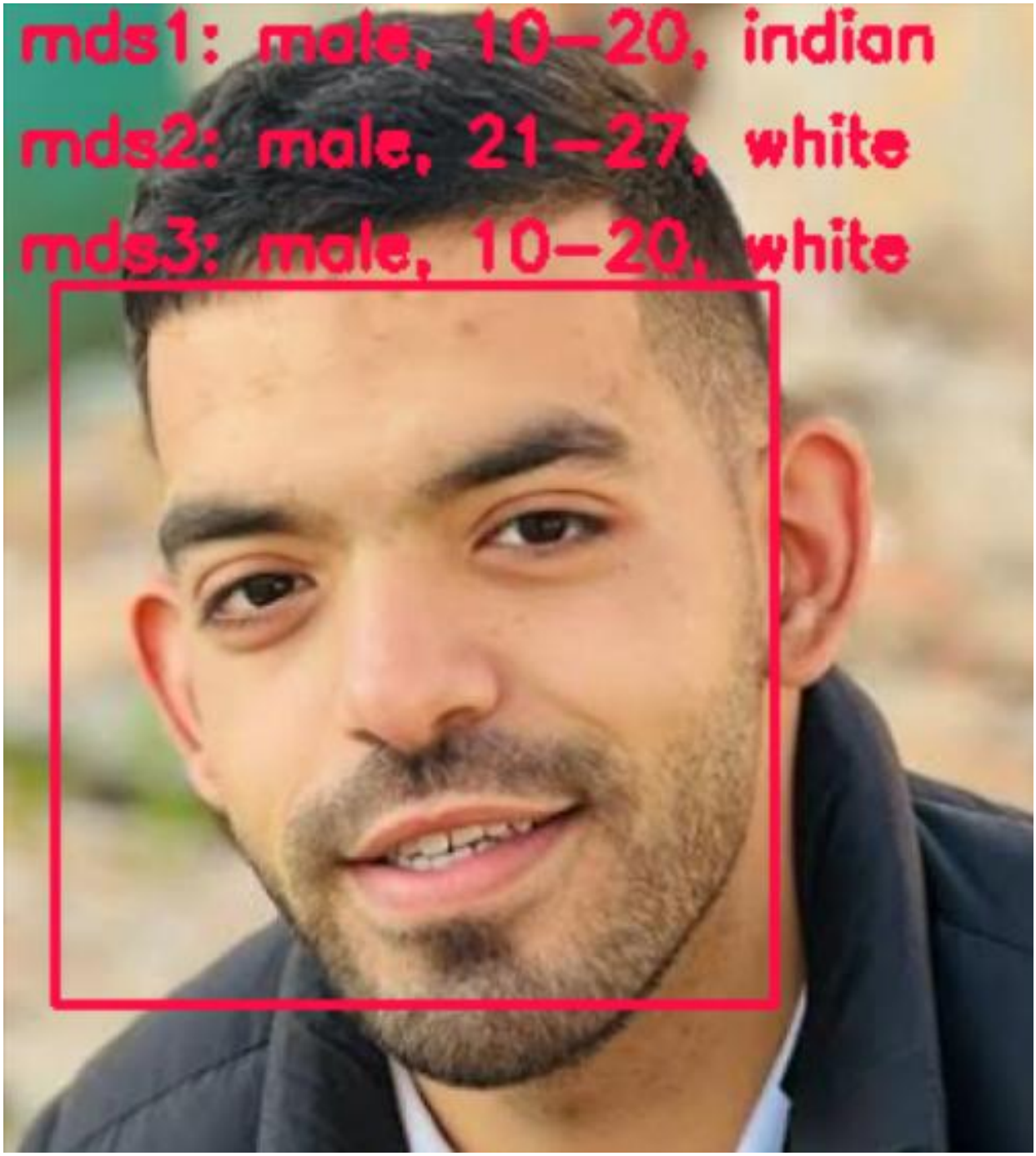

Fig. 35: An image showing another prediction of the models

Here, we can see that mds2 models did better than all the rest.

So, from all what we've seen we can probably say that using 'mds2' with the ethnicity from 'mds3' would make us the preferred model to use for prediction.

- **Using the webcam**

Using the models we chose above us from mds2 and the ethnicity of mds3, we can make these predictions :

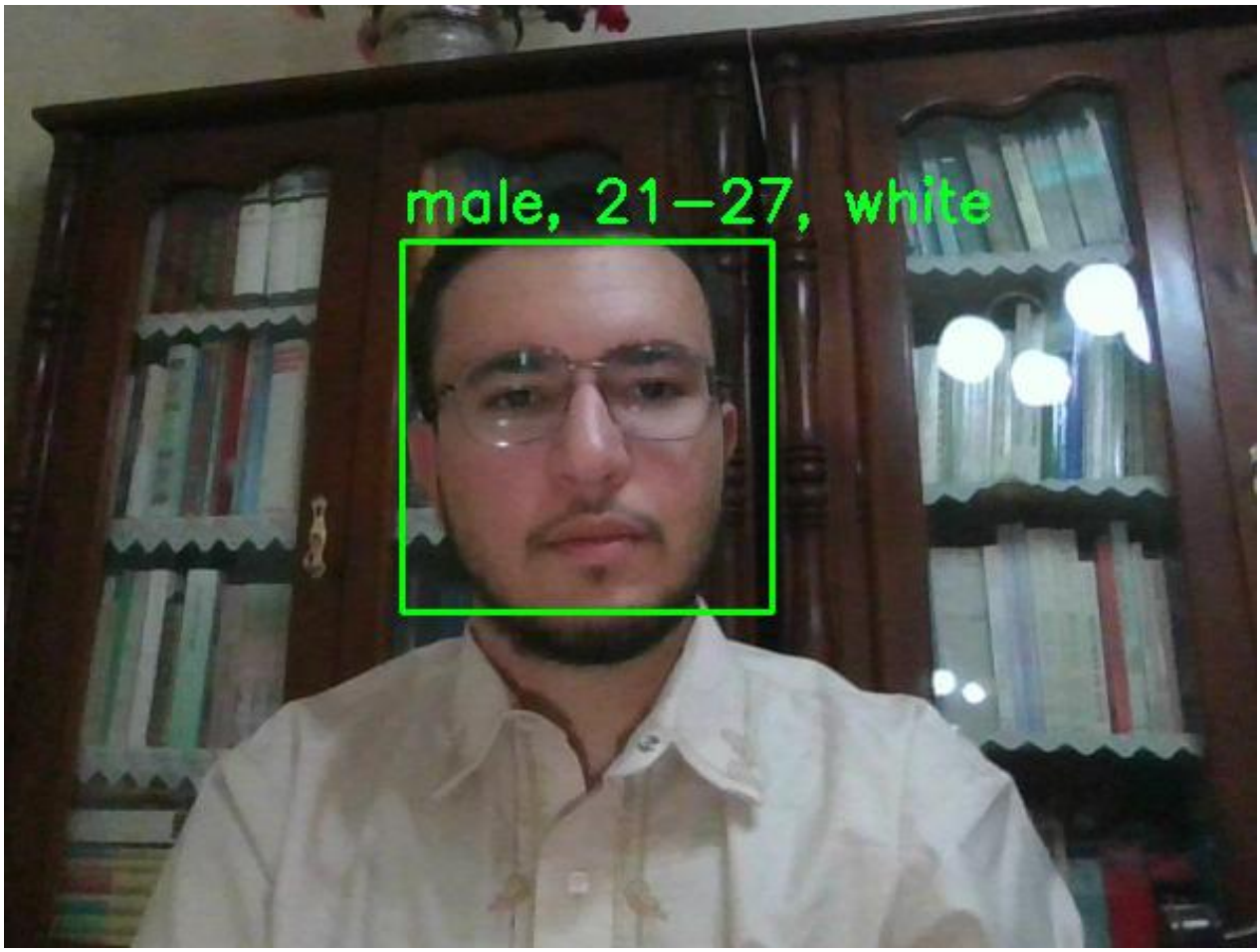

Fig. 36: An image showing the predictions using the chosen models

Here, we can safely say that these models were correct guessing the age, gender and ethnicity of 'Hached'.

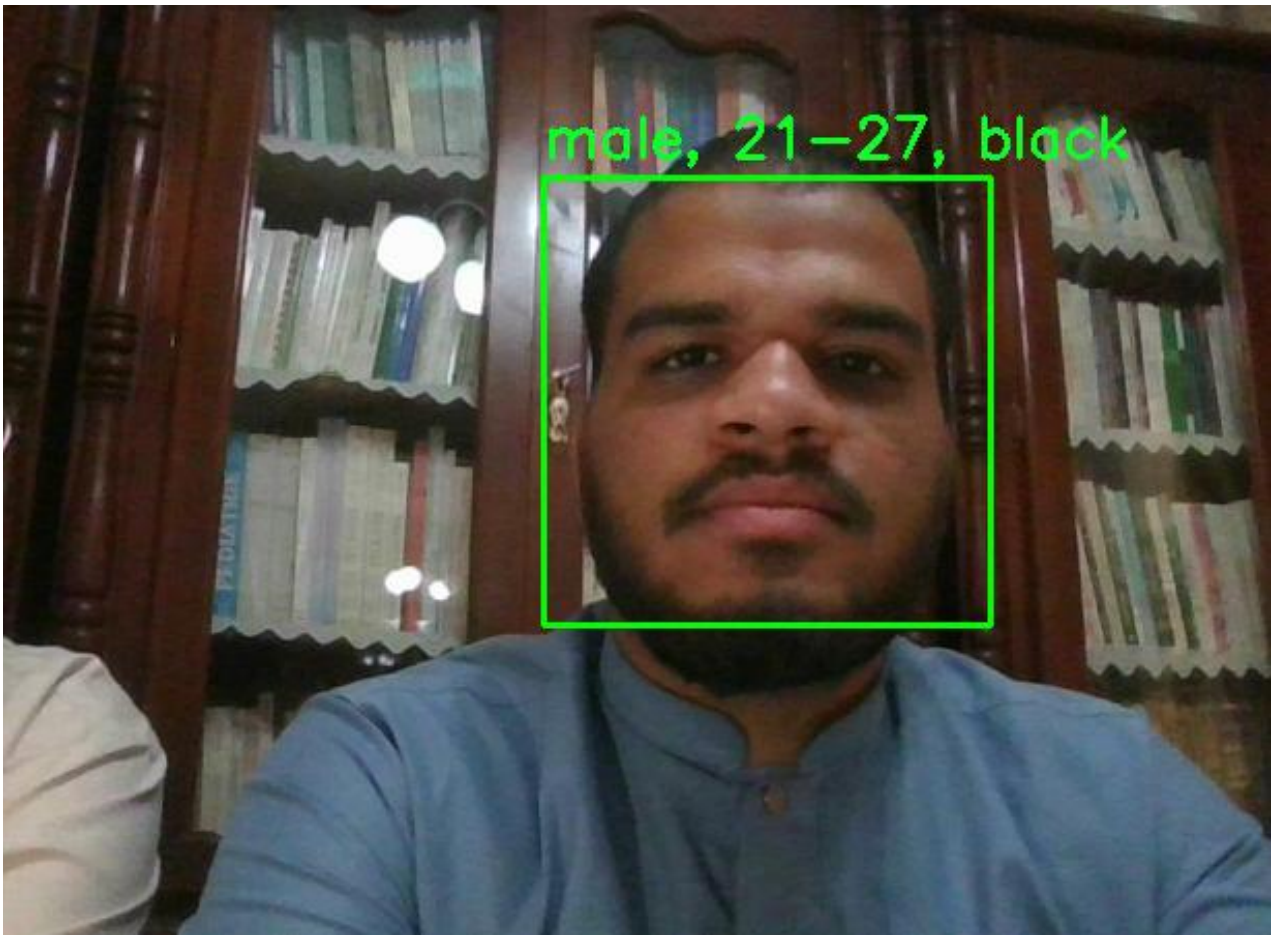

Fig. 37: Another image showing the predictions using the chosen models

Here, also ‘Salah’ was correctly predicted.

Fig. 38: An image showing the models predicting us

The models that we chose did a good job predicting us here, exactly how and who we are.

3.2. Evaluation

Now that we've seen the results, we can start evaluating them and comparing the models used.

mdss \ models	Age	Gender	Ethnicity
mds1	MAE = 0.91 MSE = 2.87	MAE = 0.08 MSE = 0.08	MAE = 1.39 MSE = 3.55
mds2	MAE = 1.52 MSE = 4.46	MAE = 0.03 MSE = 0.03	MAE = 1.07 MSE = 3.27
mds3	MAE = 1.91 MSE = 5.82	MAE = 0.24 MSE = 0.24	MAE = 0.64 MSE = 1.82

Tab. 5: A table determining the MAE and MSE of every model used to get the results

mdss \ models	Age	Gender	Ethnicity
mds1	58%	91%	32%
mds2	30%	96%	57%
mds3	13%	76%	71%

Tab. 6: A table determining the accuracy of every model used to get the results

We used our own dataset to evaluate these models, and based on the comparison that can be visible when viewing [Tab. 5] and [Tab. 6], we can say that mds1's age, mds2's gender and mds3's ethnictity are the best ones to choose to work with till now.

Our system surely still needs work and perfecting, but it is working for the most part better than we have expected starting to work on this project, especially when we keep in mind that it was trained using only one dataset 'UTKFace', and then tuned a bit using a small dataset of less than 500 images taken and downloaded and cropped by us.

The accuracy of age coming at 58% to be improved, gender at 96% to be also improved and tuned away from over-fitting, and ethnicity at 71% to be perfected and made better, but we still have tried to go further, and in some cases the over-fitted models worked better, other than that these were the most ones to not count outliers, especially when it comes to identifying indians, and blacks, or finding males in a pixelated photo, or giving a somewhat acceptable age.

These models starting with age, have some error in them especially being stable when a face moves to identify it correctly because every time a face moves, a new age comes, until the face becomes stable, that it stabilizes on an output that it thinks is correct.

About gender on the other hand, it mostly is correct with bits of mistakes, when the face moves too.

Ethnicity though, have some trouble because it is hard to determine someone's race based on their face, and using grayscale, because face traits can be similar sometimes especially when it comes to khoisan people for example, by being blacks from the southern region of africa, yet they look asian-like, so the model would always put them as asians and many more ethnic groups. Sometimes it also forgets about indians and would classify them as whites, even though other groups would get benefited from the over-fitting of the models, the indians wouldn't at all, because indians were always classified correctly back when the model had a very low percentage of being accurate.

The system is acceptable overall, when tested on multiple age groups, multiple populations and multiple genders, and as can be seen in the photos earlier, it works okay, as it is still not managing to be always correct yet it is to be perfected and made better, and we hope that we'd be able to improve on the models and make them stable and better as they are already acceptable in our eyes.

The bias of our dataset was a bit obvious from what we've seen and discussed earlier, especially the fact that close to half of the dataset are white people and people in between the ages of 20 and 30. The only stable thing about it was that it had a close to perfect ratio of males and females with a percentage being close to 50% each in the favour of males, and that made us try to create a more stable dataset to work alongside

it, which we gladly did but we couldn't make it as big as UTKFace with less than 500 images in it.

And as we said before, we only used one dataset for all and a bit, with the labels of ages being 0 for ages between 1 and 2, 1 for ages between 3 and 9, 2 for ages between 10 and 20, 4 for ages between 21 and 27, 5 for ages between 28 and 45, 6 for ages between 46 and 65, and finally 7 for ages higher than 66.

The classes of gender were males and females as those are the only genders existing.

The classes of ethnicity were 5 classes because of the selection 'UTKFace' has, which are White , Black, Asian, Indian and Others.

Other than that the models work better than what we would've expected, so we're happy for making this possible and it doesn't mean that we're stopping here, we're surely fixing the biases and are going to add more images to our dataset so we can use it to improve upon and fix the models we have now.

4. Conclusion

Age, gender and ethnicity/race recognition are tasks that involve analyzing facial images to determine the age, gender and ethnicity/race of people.

There is one large dataset 'UTKFace' and a very small one and many deep models that have been worked on using that huge dataset alongside the small one, such as the models we have stated and showed above in our training work, that have shown acceptable yet promising results in age, gender and ethnicity/race classification.

We also have described the implementation phases and evaluated our system, to come to the following conclusions:

- The system is highly interactive: It can recognize more than one person at a time.
- The system has acceptable to good performance: Thanks to good recognition rates that is.

- The system is simple: Anyone with the simple knowledge of clicking on run of a python file can use it, no background knowledge required.
- Recognition is fast and instantaneous: So that age, gender and ethnicity can be determined simultaneously.

These features make the system effective and easy to use, enabling users to take advantage of its advanced features without any need of deep to any technical expertise.

General Conclusion

The thesis addresses the topic of automated recognition of age, gender and ethnicity from images using image processing and AI techniques.

We have developed a multi-stage system based on deep learning and image processing algorithms.

Age, gender and ethnicity recognition are very challenging tasks due to the challenges and limitations we faced, namely:

- The quality and availability of data: the dataset we used contained an acceptable amount of data, but there was an issue with its somewhat unfair segmentation, and it is important to ensure that the data is accurate and representative of the populations it has.
- Generalizing: the models present another challenge, which is that they may not handle specific data well, leading to biased and inaccurate results.
- Privacy and ethics: it is essential to obtain informed consent from individuals and respect their privacy.

In terms of our models though, we have made some good work leading to a somewhat acceptable to good models that surely can be improved upon.

Also, the recognition system exhibits biases due to the data trained on, leading us to a stranded variation of predictions over certain demographic groups more than others.

In the other hand, when it comes to future works, there are many areas that can be worked on later, such as integrating deep learning techniques by continuing to explore and optimize new deep learning algorithms to enhance the limits and effectiveness of our system.

In addition to expanding its scope to include other types of data such as video and audio to enhance its capabilities in real-world scenarios to be able to recognize a person even if their face is absent.

We also seek to address ethical concerns related to privacy and bias and design a system that meets the needs of specific applications such as security, surveillance and demographic data analysis.

Finally, we conclude this letter by emphasizing the growing importance of age, gender and ethnicity/race recognition technology in today's world and as this technology continues to be developed and refined, we expect it to play an important role in shaping our future.

Appendix

1. Introduction

In this section, we will provide an overview of the tools, languages and images of codes that weren't essential to use in the chapters above, but yet useful to know and look at or read.

2. Tools & Technical Details

Firsly, we used Google Colab and Jupyter Notebook using anaconda and visual basic too, programming everything using Python, on 2 computers of these specs :

- I5 12500, GTX 1060 6G, 16GB Ram, 512GB SSD
- i7-1165G7, Intel IRIS Xe Graphics, 16GB ram, 512GB SSD

We used 'Tensorflow' and 'keras' for everything even though we also used 'ktrain', but we didn't use it anywhere so we don't really have to mention it other than that we used it in the start of the project.

3. Code Documentation

3.1. Training Codes

We are going to show here the codes used to create the models and train them alongside the reading and storing of the dataset

- **Age**

```
[2]: import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import cv2
import os

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, AveragePooling2D, BatchNormalization, LeakyReLU

import matplotlib.pyplot as plt

from keras.models import Sequential,load_model,Model
from keras.layers import Conv2D,MaxPooling2D,AvgPool2D,GlobalAveragePooling2D,Dense,Dropout,BatchNormalization,Flatten,Input
from sklearn.model_selection import train_test_split


from tensorflow.keras.layers import Input,Activation,Add
from tensorflow.keras.regularizers import l2
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint
```

Fig. 39: A Code showing the libraries that we imported to use

```
[14]: def class_labels_reassign(age):

    if 1 <= age <= 2:
        return 0
    elif 3 <= age <= 9:
        return 1
    elif 10 <= age <= 20:
        return 2
    elif 21 <= age <= 27:
        return 3
    elif 28 <= age <= 45:
        return 4
    elif 46 <= age <= 65:
        return 5
    else:
        return 6
```

Fig. 40: A Code showing a function to return age groups

```
path = "C:/Users/HeeBe/Desktop/ML-Ethnicity-Detection-master/ML-Ethnicity-Detection-master/UTKFace/"
pixels = []
age = []
gender = []

i=0
for img in os.listdir(path):
    i=i+1
    ages = class_labels_reassign(int(img.split("_")[0]))
    img = cv2.imread(str(path)+"/"+str(img))
    img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    img=cv2.resize(img,(200, 200))
    pixels.append(np.array(img))
    age.append(np.array(ages))

pixels = np.array(pixels)
age = np.array(age,np.uint64)
len(age)
```

Fig. 41: A Code showing how the data was read and stored

```
x_train,x_test,y_train,y_test = train_test_split(pixels,age,random_state=100)
```

Fig. 42: A Code showing the train_test_split method

```
model = load_model("C:/Users/HeeBe/Desktop/ProjectAGE/age_model_pretrained.h5")
```

Fig. 43: A Code showing the loading of the pre-trained model

```
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()
```

Fig. 44: A Code showing the compiling and how to display the architecture

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 198, 198, 32)	320
average_pooling2d (AveragePooling2D)	(None, 99, 99, 32)	0
conv2d_1 (Conv2D)	(None, 97, 97, 64)	18,496
average_pooling2d_1 (AveragePooling2D)	(None, 48, 48, 64)	0
conv2d_2 (Conv2D)	(None, 46, 46, 128)	73,856
average_pooling2d_2 (AveragePooling2D)	(None, 23, 23, 128)	0
conv2d_3 (Conv2D)	(None, 21, 21, 256)	295,168
average_pooling2d_3 (AveragePooling2D)	(None, 10, 10, 256)	0
global_average_pooling2d (GlobalAveragePooling2D)	(None, 256)	0
dense (Dense)	(None, 132)	33,924
dense_1 (Dense)	(None, 7)	931

Fig. 45: A table showing the architecture of the model used

```
model_path='age75.keras'
checkpointer = ModelCheckpoint(model_path, monitor='loss',verbose=1,save_best_only=True,
                               save_weights_only=False, mode='auto',save_freq='epoch')
callback_list=[checkpointer]
```

Fig. 46: A Code showing the callback_list and the model name to be saved

```
save = model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=5,callbacks=[callback_list])
```

```
Epoch 1/5
556/556 ━━━━━━━━━━━━━━━━━━━━ 0s 432ms/step - accuracy: 0.9263 - loss: 0.2074
Epoch 1: loss improved from inf to 0.23365, saving model to age75.keras
556/556 ━━━━━━━━━━━━━━━━━━━━ 262s 471ms/step - accuracy: 0.9262 - loss: 0.2074 - val_accuracy: 0.6470 - val_loss: 1.3212
Epoch 2/5
556/556 ━━━━━━━━━━━━━━━━━━━━ 0s 434ms/step - accuracy: 0.9272 - loss: 0.1972
Epoch 2: loss improved from 0.23365 to 0.22657, saving model to age75.keras
556/556 ━━━━━━━━━━━━━━━━━━━━ 265s 476ms/step - accuracy: 0.9272 - loss: 0.1973 - val_accuracy: 0.6681 - val_loss: 1.1427
Epoch 3/5
556/556 ━━━━━━━━━━━━━━━━━━━━ 0s 448ms/step - accuracy: 0.9420 - loss: 0.1631
Epoch 3: loss improved from 0.22657 to 0.20858, saving model to age75.keras
556/556 ━━━━━━━━━━━━━━━━━━━━ 275s 495ms/step - accuracy: 0.9419 - loss: 0.1632 - val_accuracy: 0.6475 - val_loss: 1.1978
Epoch 4/5
556/556 ━━━━━━━━━━━━━━━━━━━━ 0s 449ms/step - accuracy: 0.9163 - loss: 0.2251
Epoch 4: loss did not improve from 0.20858
556/556 ━━━━━━━━━━━━━━━━━━━━ 274s 493ms/step - accuracy: 0.9163 - loss: 0.2251 - val_accuracy: 0.6575 - val_loss: 1.1768
Epoch 5/5
556/556 ━━━━━━━━━━━━━━━━━━━━ 0s 450ms/step - accuracy: 0.9427 - loss: 0.1596
Epoch 5: loss improved from 0.20858 to 0.18813, saving model to age75.keras
556/556 ━━━━━━━━━━━━━━━━━━━━ 276s 497ms/step - accuracy: 0.9427 - loss: 0.1596 - val_accuracy: 0.6470 - val_loss: 1.4017
```

Fig. 47: A Code showing the training of the age model

- **Gender**

```
[1]: import pandas as pd
     import numpy as np

     import matplotlib.pyplot as plt
     import cv2
     import os

     import tensorflow as tf
     import keras
     import matplotlib.pyplot as plt

     from keras.models import Sequential,load_model,Model
     from keras.layers import Conv2D,MaxPooling2D,AvgPool2D,GlobalAveragePooling2D,Dense,Dropout,BatchNormalization,Flatten,Input
     from sklearn.model_selection import train_test_split


     from tensorflow.keras.layers import Input,Activation,Add
     from tensorflow.keras.regularizers import l2
     from tensorflow.keras.optimizers import Adam
     from tensorflow.keras.callbacks import ModelCheckpoint
     from tensorflow.keras import layers
```

Fig. 48: A Code showing the libraries that we imported to use

[2]:
```
path = "C:/Users/hache/Desktop/UTKFace1"
pixels = []
age = []
gender = []

i=0
for img in os.listdir(path):
    i=i+1
    genders = img.split("_")[1]
    img = cv2.imread(str(path)+"/"+str(img))
    img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    img=cv2.resize(img,(100,100))
    pixels.append(np.array(img))
    gender.append(np.array(genders))

pixels = np.array(pixels)
gender = np.array(gender,np.uint64)
```

[3]:
```
x_train,x_test,y_train,y_test = train_test_split(pixels,gender,random_state=100)
```

Fig. 49: A Code showing the importation of data and how it was read and stored

[4]:
```
  # Input layer for grayscale image
inputs = layers.Input(shape = (100,100,1))

  # Block 1 - Feature extraction with smaller filters for grayscale
x = layers.Conv2D(32, kernel_size=(3, 3), activation="relu", padding="same")(inputs)
x = layers.BatchNormalization(axis=-1)(x)  # Batch normalization for stability
x = layers.Dropout(0.2)(x)
x = layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2))(x)

  # Block 2 - Feature extraction with increased filters
x = layers.Conv2D(64, kernel_size=(3, 3), activation="relu", padding="same")(x)
x = layers.BatchNormalization(axis=-1)(x)
x = layers.Dropout(0.2)(x)
x = layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2))(x)

  # Block 3 - Feature extraction with increased filters
x = layers.Conv2D(128, kernel_size=(3, 3), activation="relu", padding="same")(x)
x = layers.BatchNormalization(axis=-1)(x)
x = layers.Dropout(0.2)(x)
x = layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2))(x)

  # Block 4 - Feature extraction with increased filters
x = layers.Conv2D(256, kernel_size=(3, 3), activation="relu", padding="same")(x)
x = layers.BatchNormalization(axis=-1)(x)
x = layers.Dropout(0.2)(x)
x = layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2))(x)

  # Flatten layer
x = layers.Flatten()(x)

  # Dense layers for classification
x = layers.Dense(512, activation="relu")(x)  # Increased hidden units for potentially more complex decision boundaries
x = layers.Dropout(0.3)(x)
outputs = layers.Dense(2, activation="softmax")(x)  # Softmax for multi-class classification
```

Fig. 50: A Code showing the architecture used

[5]:
```
model = Model(inputs=inputs,outputs=outputs)
model.compile(optimizer="adam",loss=["sparse_categorical_crossentropy"],metrics=['accuracy'])
model.summary()
```

Fig. 51: A Code showing the model and its compiler and summary

```
[11]: model_path='gender.keras'
checkpointer = ModelCheckpoint(model_path, monitor='loss',verbose=1,save_best_only=True,
                               save_weights_only=False, mode='auto',save_freq='epoch')
callback_list=[checkpointer]
```

Fig. 52: A Code showing the mode name to be saved and the callback_list

```
[7]: save = model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=10,callbacks=[callback_list])

Epoch 1/10
556/556 [==============================] - ETA: 0s - loss: 0.7492 - accuracy: 0.7351
Epoch 1: loss improved from inf to 0.74922, saving model to age1model.keras
556/556 [==============================] - 155s 277ms/step - loss: 0.7492 - accuracy: 0.7351 - val_loss: 0.3923 - val_accuracy: 0.8358
Epoch 2/10
556/556 [==============================] - ETA: 0s - loss: 0.3432 - accuracy: 0.8516
Epoch 2: loss improved from 0.74922 to 0.34323, saving model to age1model.keras
556/556 [==============================] - 155s 278ms/step - loss: 0.3432 - accuracy: 0.8516 - val_loss: 0.3504 - val_accuracy: 0.8448
Epoch 3/10
556/556 [==============================] - ETA: 0s - loss: 0.2927 - accuracy: 0.8734
Epoch 3: loss improved from 0.34323 to 0.29266, saving model to age1model.keras
556/556 [==============================] - 154s 277ms/step - loss: 0.2927 - accuracy: 0.8734 - val_loss: 0.3055 - val_accuracy: 0.8660
Epoch 4/10
556/556 [==============================] - ETA: 0s - loss: 0.2669 - accuracy: 0.8873
Epoch 4: loss improved from 0.29266 to 0.26691, saving model to age1model.keras
556/556 [==============================] - 154s 277ms/step - loss: 0.2669 - accuracy: 0.8873 - val_loss: 0.3370 - val_accuracy: 0.8487
Epoch 5/10
556/556 [==============================] - ETA: 0s - loss: 0.2441 - accuracy: 0.8958
Epoch 5: loss improved from 0.26691 to 0.24411, saving model to age1model.keras
556/556 [==============================] - 154s 277ms/step - loss: 0.2441 - accuracy: 0.8958 - val_loss: 0.2623 - val_accuracy: 0.8865
Epoch 6/10
556/556 [==============================] - ETA: 0s - loss: 0.2268 - accuracy: 0.9057
Epoch 6: loss improved from 0.24411 to 0.22683, saving model to age1model.keras
556/556 [==============================] - 154s 277ms/step - loss: 0.2268 - accuracy: 0.9057 - val_loss: 0.2994 - val_accuracy: 0.8780
Epoch 7/10
556/556 [==============================] - ETA: 0s - loss: 0.2117 - accuracy: 0.9121
Epoch 7: loss improved from 0.22683 to 0.21167, saving model to age1model.keras
556/556 [==============================] - 154s 277ms/step - loss: 0.2117 - accuracy: 0.9121 - val_loss: 0.3013 - val_accuracy: 0.8760
Epoch 8/10
556/556 [==============================] - ETA: 0s - loss: 0.1996 - accuracy: 0.9167
Epoch 8: loss improved from 0.21167 to 0.19961, saving model to age1model.keras
556/556 [==============================] - 154s 278ms/step - loss: 0.1996 - accuracy: 0.9167 - val_loss: 0.3188 - val_accuracy: 0.8765
Epoch 9/10
556/556 [==============================] - ETA: 0s - loss: 0.1885 - accuracy: 0.9211
Epoch 9: loss improved from 0.19961 to 0.18853, saving model to age1model.keras
556/556 [==============================] - 154s 277ms/step - loss: 0.1885 - accuracy: 0.9211 - val_loss: 0.3072 - val_accuracy: 0.8954
Epoch 10/10
556/556 [==============================] - ETA: 0s - loss: 0.1787 - accuracy: 0.9274
Epoch 10: loss improved from 0.18853 to 0.17872, saving model to age1model.keras
556/556 [==============================] - 154s 277ms/step - loss: 0.1787 - accuracy: 0.9274 - val_loss: 0.3038 - val_accuracy: 0.8863
```

Fig. 53: A Code showing the training of the gender model

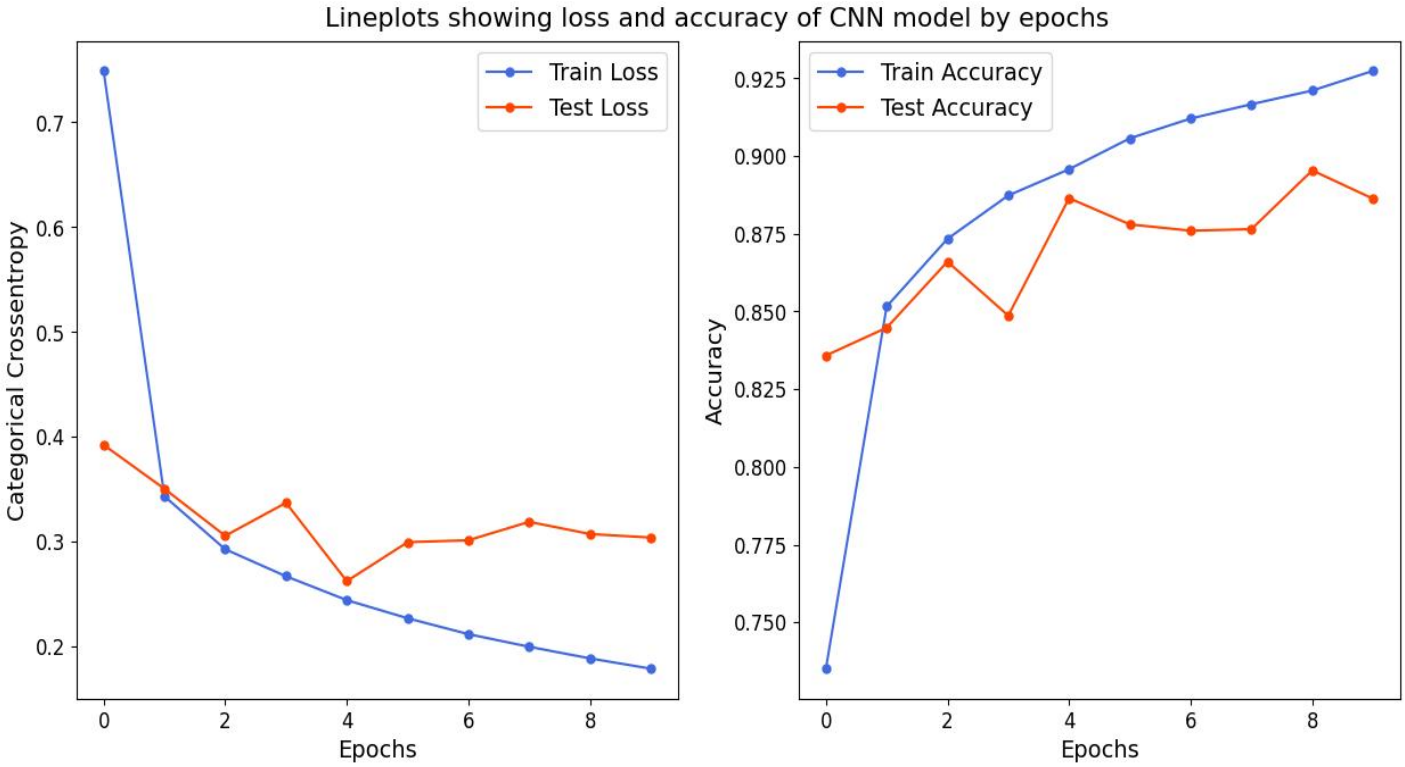

Fig. 54: A graph showing loss and accuracy of the model by epochs

- **Ethnicity**

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import cv2
import os

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, AveragePooling2D, BatchNormalization, LeakyReLU

import matplotlib.pyplot as plt

from keras.models import Sequential,load_model,Model
from keras.layers import Conv2D,MaxPooling2D,AvgPool2D,GlobalAveragePooling2D,Dense,Dropout,BatchNormalization,Flatten,Input
from sklearn.model_selection import train_test_split


from tensorflow.keras.layers import Input,Activation,Add
from tensorflow.keras.regularizers import l2
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint
```

Fig. 55: A code showing the libraries imported to work with

```
path = "C:/Users/HeeBe/Desktop/ML-Ethnicity-Detection-master/ML-Ethnicity-Detection-master/UTKFace/"
pixels = []
age = []
gender = []

i=0
for img in os.listdir(path):
    i=i+1
    genders = img.split("_")[2]
    img = cv2.imread(str(path)+"/"+str(img))
    img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    img=cv2.resize(img,(100,100))
    pixels.append(np.array(img))
    gender.append(np.array(genders))

pixels = np.array(pixels)
gender = np.array(gender,np.uint64)
```

Fig. 56: A code showing the importing of the data and how we read it and stored it

```
x_train,x_test,y_train,y_test = train_test_split(pixels,gender,random_state=100)
```

Fig. 57: A code showing the splitting of data

```
# Define the CNN architecture
def build_race_classifier():
  model = Sequential()
  model.add(Conv2D(32, (3, 3), padding='same', activation='relu', input_shape=(100, 100, 1)))
  model.add(BatchNormalization())
  model.add(LeakyReLU(alpha=0.1))  # Leaky ReLU for better gradient flow
  model.add(MaxPooling2D((2, 2)))
  model.add(Dropout(0.25))  # Dropout for regularization

  model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))
  model.add(BatchNormalization())
  model.add(LeakyReLU(alpha=0.1))
  model.add(MaxPooling2D((2, 2)))
  model.add(Dropout(0.25))

  model.add(Conv2D(128, (3, 3), padding='same', activation='relu'))
  model.add(BatchNormalization())
  model.add(LeakyReLU(alpha=0.1))
  model.add(MaxPooling2D((2, 2)))
  model.add(Dropout(0.25))

  model.add(Flatten())
  model.add(Dense(256, activation='relu'))
  model.add(Dropout(0.5))  # Higher dropout for fully-connected layers
  model.add(Dense(5, activation='softmax'))  # 5 denotes the number of race classes

  # Compile the model with Adam optimizer and categorical crossentropy loss
  model.compile(loss='sparse_categorical_crossentropy', optimizer=Adam(0.001), metrics=['accuracy'])
  return model

# Create the model instance
model = build_race_classifier()
```

Fig. 58: A code showing the architecture used for the model as a code

```
model.summary()
```

Model: "sequential_8"

Layer (type)	Output Shape	Param #
conv2d_25 (Conv2D)	(None, 100, 100, 32)	320
batch_normalization_25 (BatchNormalization)	(None, 100, 100, 32)	128
leaky_re_lu_23 (LeakyReLU)	(None, 100, 100, 32)	0
max_pooling2d_22 (MaxPooling2D)	(None, 50, 50, 32)	0
dropout_28 (Dropout)	(None, 50, 50, 32)	0
conv2d_26 (Conv2D)	(None, 50, 50, 64)	18,496
batch_normalization_26 (BatchNormalization)	(None, 50, 50, 64)	256
leaky_re_lu_24 (LeakyReLU)	(None, 50, 50, 64)	0
max_pooling2d_23 (MaxPooling2D)	(None, 25, 25, 64)	0
dropout_29 (Dropout)	(None, 25, 25, 64)	0
conv2d_27 (Conv2D)	(None, 25, 25, 128)	73,856
batch_normalization_27 (BatchNormalization)	(None, 25, 25, 128)	512
leaky_re_lu_25 (LeakyReLU)	(None, 25, 25, 128)	0
max_pooling2d_24 (MaxPooling2D)	(None, 12, 12, 128)	0
dropout_30 (Dropout)	(None, 12, 12, 128)	0
flatten_7 (Flatten)	(None, 18432)	0
dense_15 (Dense)	(None, 256)	4,718,848
dropout_31 (Dropout)	(None, 256)	0
dense_16 (Dense)	(None, 5)	1,285

Fig. 59: A code showing the architecture using ‘model.summary’

```
model_path='ethnicity.keras'
checkpointer = ModelCheckpoint(model_path, monitor='loss',verbose=1,save_best_only=True,
                               save_weights_only=False, mode='auto',save_freq='epoch')
callback_list=[checkpointer]
```

Fig. 60: A code showing the model saving path and the checkpointer

```
save = model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=5,callbacks=[callback_list])
```

```
Epoch 1/5
556/556 ━━━━━━━━━━━━━━━━━━━━ 0s 264ms/step - accuracy: 0.4072 - loss: 3.0958
Epoch 1: loss improved from inf to 1.76064, saving model to ethnicity.keras
556/556 ━━━━━━━━━━━━━━━━━━━━ 158s 280ms/step - accuracy: 0.4073 - loss: 3.0934 - val_accuracy: 0.5519 - val_loss: 1.1308
Epoch 2/5
556/556 ━━━━━━━━━━━━━━━━━━━━ 0s 260ms/step - accuracy: 0.5300 - loss: 1.1816
Epoch 2: loss improved from 1.76064 to 1.16208, saving model to ethnicity.keras
556/556 ━━━━━━━━━━━━━━━━━━━━ 153s 275ms/step - accuracy: 0.5300 - loss: 1.1816 - val_accuracy: 0.6298 - val_loss: 0.9683
Epoch 3/5
556/556 ━━━━━━━━━━━━━━━━━━━━ 0s 254ms/step - accuracy: 0.5785 - loss: 1.0640
Epoch 3: loss improved from 1.16208 to 1.05447, saving model to ethnicity.keras
556/556 ━━━━━━━━━━━━━━━━━━━━ 150s 269ms/step - accuracy: 0.5785 - loss: 1.0640 - val_accuracy: 0.5382 - val_loss: 1.0819
Epoch 4/5
556/556 ━━━━━━━━━━━━━━━━━━━━ 0s 266ms/step - accuracy: 0.6291 - loss: 0.9802
Epoch 4: loss improved from 1.05447 to 0.97265, saving model to ethnicity.keras
556/556 ━━━━━━━━━━━━━━━━━━━━ 158s 284ms/step - accuracy: 0.6291 - loss: 0.9802 - val_accuracy: 0.7233 - val_loss: 0.8521
Epoch 5/5
556/556 ━━━━━━━━━━━━━━━━━━━━ 0s 269ms/step - accuracy: 0.6545 - loss: 0.9229
Epoch 5: loss improved from 0.97265 to 0.92867, saving model to ethnicity.keras
556/556 ━━━━━━━━━━━━━━━━━━━━ 159s 286ms/step - accuracy: 0.6545 - loss: 0.9229 - val_accuracy: 0.7110 - val_loss: 0.8416
```

Fig. 61: A code showing the training of the model

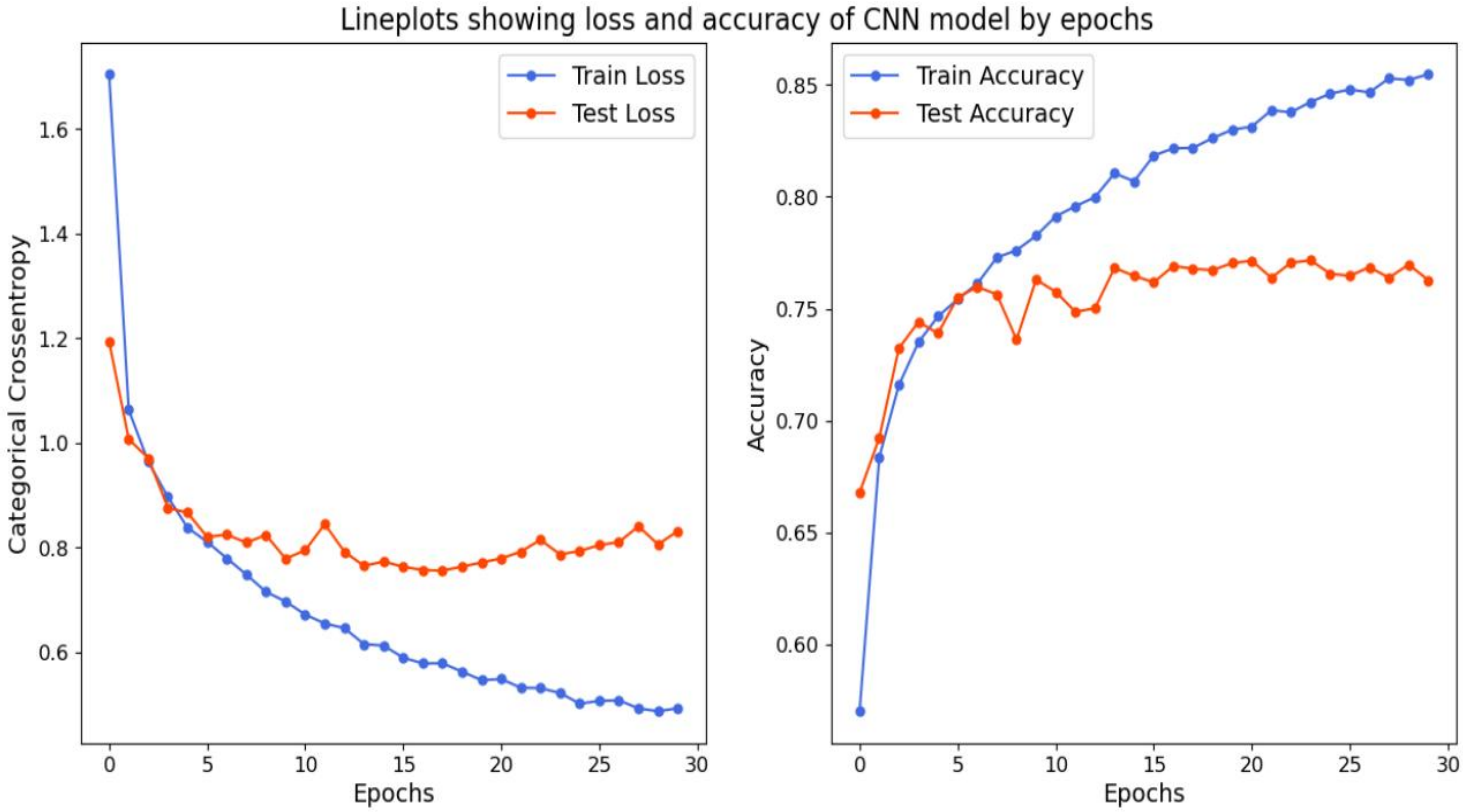

Fig. 62: A graph showing the loss and accuracy of the first 5 epochs

3.2. Implementation Codes

Here, we show the codes used to use the models trained in the codes earlier

- **Using Images**

```
from keras.models import load_model
from PIL import Image
import numpy as np
import cv2

#the following are to do with this interactive notebook code

%matplotlib inline
from matplotlib import pyplot as plt # this lets you draw inline pictures in the notebooks
import pylab # this allows you to control figure size
pylab.rcParams['figure.figsize'] = (10.0, 8.0) # this controls figure size in the notebook
```

Fig. 63: A code showing the importation of the libraries used

```
export_dir='C:/Users/HeeBe/Desktop/Project5_AgeGenderEmotion_Detection-20240418T122950Z-002/Project5_AgeGenderEmotion_Detection/age75.keras'
age_model = load_model(export_dir)

# summarize model.
age_model.summary()
```

Fig. 64: A code showing the loading of age model

```
# load and evaluate a saved model
export_dir='gender30epochs.h5'
gender_model = load_model(export_dir)

# summarize model.
gender_model.summary()
```

Fig. 65: A code showing the loading of gender model

```
# Labels on Age, Gender and Emotion to be predicted
export_dir='C:/Users/HeeBe/Desktop/ProjectAGE/ethnicity55.keras'
ethnicity_model = load_model(export_dir)

# summarize model.
ethnicity_model.summary()

age_ranges = ['1-2', '3-5', '6-8', '9-12', '13-15', '16-18', '19-25', '26-35', '36-50', '51-65', '66-75', '75+']
age_ranges1 = ['1-2', '3-9', '10-20', '21-27', '28-45', '46-65', '66-116']
gender_ranges = ['male', 'female']
ethnicity_ranges = ['white', 'black', 'indian', 'asian', 'others']
```

Fig. 66: A code showing the loading of ethnicity model and the ranges for each

```
img_path = "C:/Users/HeeBe/Desktop/yes/hap.jpg"
```

```
from IPython.display import Image
pil_img = Image(filename=img_path)
display(pil_img)
```

Fig. 67: A code showing the loading of an image and displaying it

```
test_image = cv2.imread(img_path)
gray = cv2.cvtColor(test_image,cv2.COLOR_BGR2GRAY)
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
faces = face_cascade.detectMultiScale(gray, 1.3, 5)

i = 0

for (x,y,w,h) in faces:
  i = i+1
  cv2.rectangle(test_image,(x,y),(x+w,y+h),(203,12,255),2)

  img_gray=gray[y:y+h,x:x+w]

  gender_img = cv2.resize(img_gray, (100, 100), interpolation = cv2.INTER_AREA)
  gender_image_array = np.array(gender_img)
  gender_input = np.expand_dims(gender_image_array, axis=0)
  output_gender=gender_ranges[np.argmax(gender_model.predict(gender_input))]

  ethnicity_img = cv2.resize(img_gray, (100, 100), interpolation = cv2.INTER_AREA)
  ethnicity_image_array = np.array(ethnicity_img)
  ethnicity_input = np.expand_dims(ethnicity_image_array, axis=0)
  output_ethnicity=ethnicity_ranges[np.argmax(ethnicity_model.predict(ethnicity_input))]

  age_image=cv2.resize(img_gray, (200, 200), interpolation = cv2.INTER_AREA)
  age_input = age_image.reshape(-1, 200, 200, 1)
  output_age = age_ranges1[np.argmax(age_model.predict(age_input))]


  output_str = str(i) + ": "+  output_gender + ', '+ output_age + ', '+ output_ethnicity
  print(output_str)

  col = (0,255,0)

  cv2.putText(test_image, str(i),(x,y),cv2.FONT_HERSHEY_SIMPLEX,1,col,2)

plt.imshow(cv2.cvtColor(test_image, cv2.COLOR_BGR2RGB))
```

Fig. 68: A code showing the classification of the image

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1: female, 28-45, white
<matplotlib.image.AxesImage at 0x21199c60a10>
```

Fig. 69: A demonstration of the classification

- **Using Webcam**

```
import os

from keras.models import load_model
from PIL import Image
import numpy as np
import cv2
import tensorflow
from tensorflow import keras

%matplotlib inline
from matplotlib import pyplot as plt
import pylab
pylab.rcParams['figure.figsize'] = (10.0, 8.0)
```

Fig. 70: A code showing the importation of libraries

```
ethnicity_model = load_model('ethnicity55.keras')
ethnicity_model.summary()
```

Fig. 71: A code showing how we imported the models

```
age_ranges = ['1-2', '3-5', '6-8', '9-12', '13-15', '16-18', '19-25', '26-35', '36-50', '51-65', '66-75', '75+']
age_ranges1 = ['1-2', '3-9', '10-20', '21-27', '28-45', '46-65', '66-116']
gender_ranges = ['male', 'female']
ethnicity_ranges = ['white', 'asian', 'indian', 'black', 'others']
```

Fig. 72: A code showing the ranges

```
test_image = cv2.VideoCapture(0)
output_str = ""
while True:
    ret, frame = test_image.read()

    gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)


    for (x,y,w,h) in faces:
        img_gray=gray[y:y+h,x:x+w]
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

        gender_img = cv2.resize(img_gray, (100, 100), interpolation = cv2.INTER_AREA)
        gender_image_array = np.array(gender_img)
        gender_input = np.expand_dims(gender_image_array, axis=0)
        output_gender=gender_ranges[np.argmax(gender_model.predict(gender_input))]

        ethnicity_img = cv2.resize(img_gray, (100, 100), interpolation = cv2.INTER_AREA)
        ethnicity_image_array = np.array(ethnicity_img)
        ethnicity_input = np.expand_dims(ethnicity_image_array, axis=0)
        output_ethnicity=ethnicity_ranges[np.argmax(ethnicity_model.predict(ethnicity_input))]

        age_image=cv2.resize(img_gray, (200, 200), interpolation = cv2.INTER_AREA)
        age_input = age_image.reshape(-1, 200, 200, 1)
        output_age = age_ranges1[np.argmax(age_model.predict(age_input))]


        output_str = output_gender + ', '+ output_age + ', '+ output_ethnicity
        print(output_str)
        cv2.putText(frame, output_str, (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (36,255,12), 2)
        col = (0,255,0)
        cv2.imshow('frame',frame)
        label = "{},{}".format(output_gender, output_age,output_ethnicity)

    k=cv2.waitKey(1)
    if k==ord('q'):
        test_image.release()
        cv2.destroyAllWindows()
        break
```

Fig. 73: A code showing how the webcam opens and how the model predict

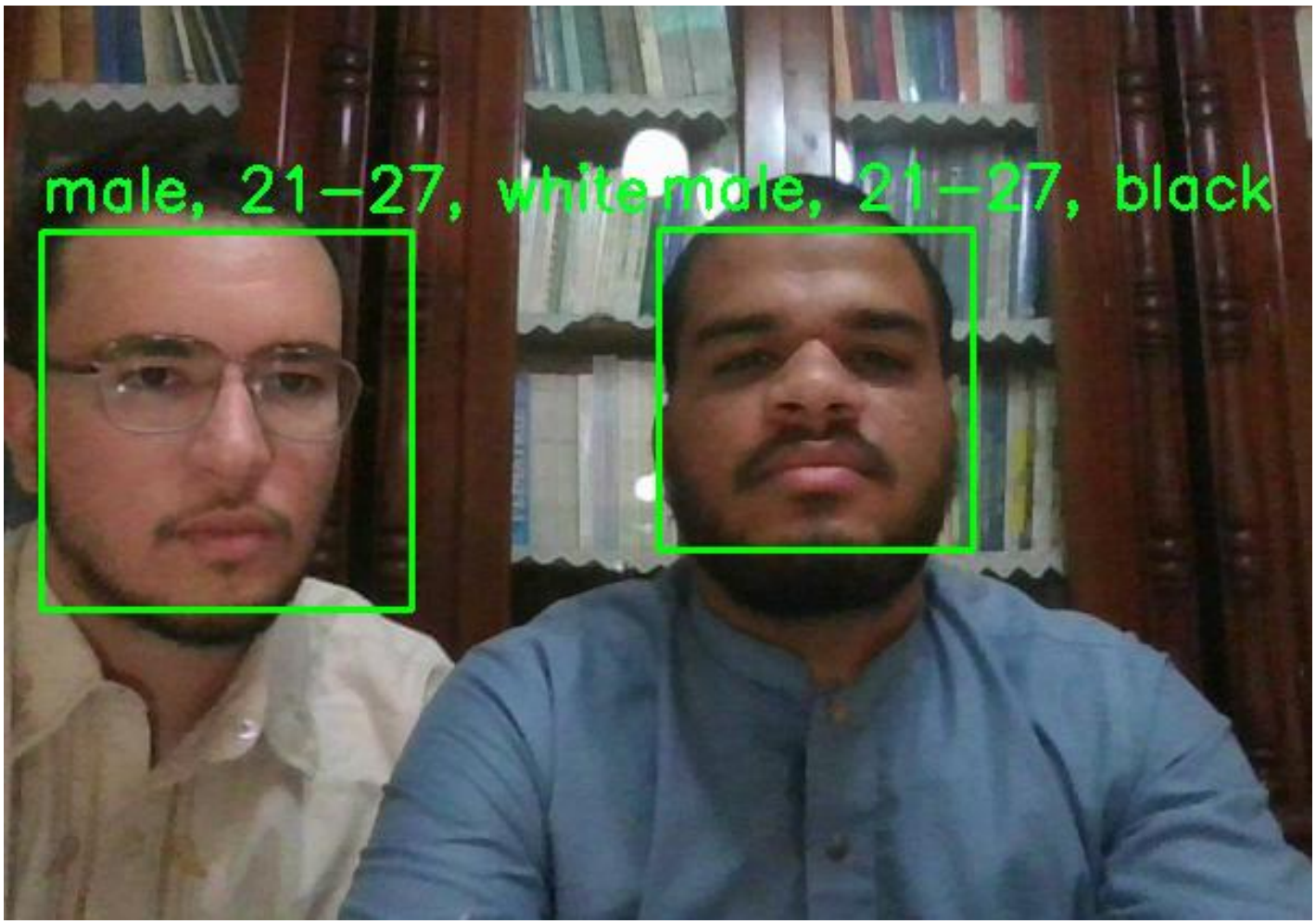

Fig. 74: A demonstration of the system classifying us

```
model = load_model("C:/Users/HeeBe/Desktop/Project5_AgeGenderEmotion_Detection-20240418T122950Z-002/Project5_AgeGenderEmotion_Detection/ethnicity37_2.keras")
```

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, accuracy_score
import numpy as np
ranges = ['0', '1', '2', '3', '4']
predictions = model.predict(pixels)
predictions[1]
```

```
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 55ms/step
array([0.2506535 , 0.08087955, 0.05555843, 0.3596056 , 0.2533029 ],
      dtype=float32)
```

```
p=[int(ranges[np.argmax(i)]) for i in predictions]
```

```
mae = mean_absolute_error(p, age)

# Calculate MSE
mse = mean_squared_error(p, age)
print(mae)
print(mse)
accuracy = accuracy_score(age, p)
print(accuracy)
```

```
0.6380208333333334
1.8203125
0.7109375
```

Fig. 75: The code to find the MAE, MSE, Accuracy of each model using our dataset

4. Conclusion

This section presented additional information and visuals about our work, including the images of our codes that we used, alongside the softwares and the machines providing a better insight of how everything was done, and giving a deeper understanding of our work.

Bibliography

[1] "Sistemas Expertos: Fundamentos, Metodologías y Aplicaciones", 2013, Badaró, Ibañez, Agüero, available on : https://dialnet.unirioja.es/servlet/articulo?codigo=4843871

[2] *"La competencia digital en el docente universitario"*, 2019, Yolvi Ocaña-Fernández, Luis Valenzuela-Fernández, John Morillo-Flores, available on : http://www.scielo.org.pe/pdf/pyr/v8n1/2310-4635-pyr-8-01-e455.pdf

[3] *"Higher Education in the AI Age"*, 2019, Yizhi Ma, Keng Siau, available on : https://www.researchgate.net/publication/333296294_Higher_Education_in_the_AI_Age

[4] *"Machine Learning for Secure Vehicular Communication: an Empirical Study"*, 2019, Matvej Mikael Yli-Olli, available on : https://www.researchgate.net/figure/A-Venn-Diagram-that-shows-how-each-subset-of-AI-is-related-to-one-another_fig10_334416564

[5] "استخدام تطبيقات الذكاء الاصطناعي في تحسين عملية اتخاذ القرار في المؤسسة الاقتصادية"، كادي سليمة، حيدة سعاد و هو متاح https://dspace.univ-adrar.edu.dz/jspui/handle/123456789/4730

[6] *"18 Cutting-Edge Artificial Intelligence Applications in 2024"*, 2024, Avijeet Biswal, available on : https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/artificial-intelligence-applications

[7] *"Top 20 Applications of Artificial Intelligence (AI) in 2024"*, 2024, lognoroy2000, available on :

[8] Available on : https://freecontent.manning.com/the-computer-vision-pipeline-part-4-feature-extraction/

[9] *"Image Recognition: Definition, Algorithms & Uses"*, 2022, Pragati Baheti, available on : https://www.v7labs.com/blog/image-recognition-guide

[10] *"Image Processing: Techniques, Types, & Applications"*, 2022, Rohit Kundu, available on : https://www.v7labs.com/blog/image-processing-guide

[11] *"convolutional neural network (CNN)"*, 2024, Lev Craig, Rahul Awati, available on : https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network

[12] *"Convolutional Neural Network"*, Thomas Wood available on : https://deepai.org/machine-learning-glossary-and-terms/convolutional-neural-network

[13] *"Convolutional Neural Networks, Explained", 2020, Mayank Mishra, available on : https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939*

[14] Available on : https://www.geeksforgeeks.org/introduction-convolution-neural-network/

[15] *"The History of Facial Recognition Technologies: How Image Recognition Got So Advanced"*, 2019, Claude Hochreutiner, available on : https://anyconnect.com/blog/the-history-of-facial-recognition-technologies

[16] *"A brief history of Facial Recognition"*, 2022, available on : https://www.nec.co.nz/market-leadership/publications-media/a-brief-history-of-facial-recognition/

[17] *"Impact of Age Groups on Fingerprint Recognition Performance"*, 2007, Shimon K. Modi, Prof. Stephen J. Elliot, Ph.D., Jeff Whetsone, College of Technology Purdue University, West Lafayette, U.S.A, available on : https://www.researchgate.net/publication/4257989_Impact_of_Age_Groups_on_Fingerprint_Recognition_Performance

[18] *"An Introduction to Age Invariant Face Recognition"*, 2021, Saurav Rai, available on : https://medium.com/geekculture/an-introduction-to-age-invariant-face-recognition-404e8d44ab48

[19] *"Gender recognition software'*, available on : https://visagetechnologies.com/gender-detection/

[20] Available on : https://excavating.ai/

[21] *"Study finds that artificial intelligence can determine race from medical images"*, 2022, Karen Olsen, available on : https://www.nibib.nih.gov/news-events/newsroom/study-finds-artificial-intelligence-can-determine-race-medical-images

[22] *"Beyond Specialization: Assessing the Capabilities of MLLMs in Age and Gender Estimation"*, Maksim Kuprashevich, Grigorii Alekseenko, Irina Tolstykh, available on : https://paperswithcode.com/paper/beyond-specialization-assessing-the-1

[23] *"Race estimation with deep networks"*, 2022, Mazida Akhtara Ahmed, Dr. Ridip Dev Choudhury, Mr. Kishore Kashyap, available on : https://www.sciencedirect.com/science/article/pii/S1319157820305589

[24] *"UTKFace"*, Yang Song, Zhifei Zhang, available on : https://susanqq.github.io/UTKFace/

[25] *"UTKFace_Age_Regression"*, zalewskij, Available on : https://github.com/zalewskij/UTKFace_Age_Regression

[26] *"Facial Recognition Technology - A Primer"*, Congressional Research Service; *"Facial Recognition: A Critical Look"*, Georgetown Law Center on Privacy & Technology

[27] *"Eigenfaces: Recovering Humans from Ghosts"*, 2018, Nev Acar, available on : https://towardsdatascience.com/eigenfaces-recovering-humans-from-ghosts-17606c328184

[28] Available on : https://www.kaggle.com/datasets/ttungl/adience-benchmark-gender-and-age-classification

[29] Available on :https://paperswithcode.com/sota/age-and-gender-classification-on-adience

[30] Available on : https://paperswithcode.com/sota/age-and-gender-classification-on-adience-age

[31] *"A Hybrid Transformer-Sequencer approach for Age and Gender classification from in-wild facial images"*, 2024, Aakash Singh, Vivek Kumar Singh, available on : https://paperswithcode.com/paper/a-hybrid-transformer-sequencer-approach-for

[32] *"MiVOLO: Multi-input Transformer for Age and Gender Estimation"*, 2023, Maksim Kuprashevich, Irina Tolstykh, available on : https://paperswithcode.com/paper/mivolo-multi-input-transformer-for-age-and

[33] *"Generalizing MLPs With Dropouts, Batch Normalization, and Skip Connections"*, 2021, Taewoon Kim, available on : https://paperswithcode.com/paper/generalizing-mlps-with-dropouts-batch

[34] *"Compacting, Picking and Growing for Unforgetting Continual Learning"*, 2019, Steven C. Y. Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, Chu-Song Chen, available on : https://paperswithcode.com/paper/compacting-picking-and-growing-for

[35] *"Fine-Grained Age Estimation in the wild with Attention LSTM Networks"*, 2018, Ke Zhang, Na Liu, Xingfang Yuan, Xinyao Guo, Ce Gao, Zhenbing Zhao, Zhanyu Ma, available on : https://paperswithcode.com/paper/fine-grained-age-estimation-in-the-wild-with

[36] *"ConvNets with Smooth Adaptive Activation Functions for Regression"*, 2017, Le Hou, Dimitris Samaras, Tahsin M. Kurc, Yi Gao, Joel H. Saltz, available on : https://paperswithcode.com/paper/convnets-with-smooth-adaptive-activation

[37] Available on : https://medium.com/@mvkuprashevich/mivolo-a-new-state-of-the-art-open-source-neural-network-for-age-and-gender-estimation-9074bb4780cb

[38] Available on : https://buptzyb.github.io/CBFace/?reload=true

[39] *"Classification of Ethnicity Using Efficient CNN Models on MORPH and FERET Datasets Based on Face Biometrics"*, 2023, Abdulwahid Al Abdulwahid, available on : https://www.mdpi.com/2076-3417/13/12/7288

[40] *"Intelligent deep learning based ethnicity recognition and classification using facial images"*, 2022, Gurram Sunitha, K. Geetha, S. Neelakandan, Aditya Kumar Singh Pundir, S. Hemalatha, Vinay Kumar, available on : https://www.sciencedirect.com/science/article/abs/pii/S0262885622000336

Abstract:

Age, gender, and ethnicity recognition technology analyzes faces using computer vision, but accuracy can be impacted by lighting, pose, and image quality. This report investigates existing methods, trains deep models similar to VGG/ResNet architectures and a pre-trained model from SkillCate, then evaluates and discusses the results. It aims to improve recognition accuracy despite these challenges.

Key words: Age, Gender and Ethnicity Recognition ; Computer Vision.

الملخص:

تقوم تقنية التعرف على العمر والجنس والعرق بتحليل الوجوه باستخدام الرؤية الحاسوبية، ولكن يمكن أن تتأثر الدقة بالإضاءة والوضعية وجودة الصورة. يبحث هذا التقرير في الأساليب الحالية، ويدرّب نماذج عميقة مشابهة لبنية VGG/ResNet ونموذج مُدرّب مسبقًا منSkillCate، ثم يقيّم النتائج ويناقشها. ويهدف إلى تحسين دقة التعرف على الرغم من هذه التحديات.

الكلمات المفتاحية: التعرف على العمر والجنس والعرق؛ الرؤية الحاسوبية.

Résumé:

La technologie de reconnaissance de l'âge, du sexe et de l'origine ethnique analyse les visages à l'aide de la vision par ordinateur, mais la précision peut être affectée par l'éclairage, la pose et la qualité de l'image. Ce rapport étudie les méthodes existantes, entraîne des modèles profonds similaires aux architectures VGG/ResNet et un modèle pré-entraîné de SkillCate, puis évalue et discute les résultats. Il vise à améliorer la précision de la reconnaissance malgré ces défis.

Mots clés : Reconnaissance de L'âge, Sexe et de L'origine Ethnique; Vision Par Ordinateur.