Mathematics Techniques with Machine learning Implementation for Facial Recognition

by

Guy Mathias Gouaya

submitted in accordance with the requirements

for the degree of

DOCTOR OF PHILOSOPHY

In the subject

Applied Mathematics

In the

Department of Mathematical Sciences

At the

UNIVERSITY OF SOUTH AFRICA

Supervisor: Prof. Emile Franc Doungmo Goufo

Acknowledgements

First of all, let me take this opportunity to send my gratitude to my supervisor Prof E.F Doungmo Goufo for his valuable support, comments and understanding toward this hard journey.
I would like to thank the UNISA school of Science and in particular the department of mathematical Sciences for offering me the opportunity to study toward their institution.
I would like to express my sincere appreciation and gratitude to my master supervisor, Prof. Yorick Hardy. Your guidance, inspiration, and mentorship have played a pivotal role in shaping my academic journey. Thank you, Prof. Hardy, for introducing me to the fascinating concept of tensor decomposition and for broadening my understanding of its applications, particularly in the field of image processing, which sparked my curiosity to explore new avenues in research and innovation.
I am profoundly grateful to my wife, Isabelle Pauline Tassi, for her unwavering support and boundless motivation. You have been the driving force behind this achievement. Without you, none of this would have been possible. I will always cherish and stand by you with love and support.
I am deeply indebted to my family, both collectively and individually. In particular, I want to express my gratitude to my elder brother, Mr. Joseph Pango, who has always unwaveringly believed in my intellectual capabilities. Additionally, I am immensely grateful to my elder sisters, Prisque Mougah and Octavie Djaleu, as well as my younger sister, Monique Noualeu, who have consistently provided support, motivation, and concern during all of my challenges. I cannot overlook the support and encouragement of my elder brothers, Jean Baptist Yomeni, Hilaire Pangop, and Nicolas Feumekeu. Furthermore, I want to express my appreciation to all my nephews and nieces for placing their trust in me and inspiring me to work even harder to be a role model for them. It is my sincere hope that my dedication will serve as an inspiration to each and every one of you.

I cannot forget to express my appreciation to all my friends for their unwavering motivation and understanding during times when I couldn't be with them due to my study commitments. Your support and understanding have meant the world to me, and I am truly grateful for your friendship and encouragement.

My thoughts are with my late mother, Marie Kamani, who passed away this year while I was immersed in my studies. You will always hold a special place in my heart, and your memory will serve as a constant source of motivation for me. Your life was a shining example of hard work, honesty, and resilience, and I am forever grateful for the values you instilled in me. Though

you may no longer be with us physically, your spirit will continue to guide and inspire me every day.
Special thanks to Mr. Aubin Alawa and Mr. Eric Kameni for warmly welcoming me to this country and providing invaluable guidance as I navigated my path. Your kindness and support have made a significant difference in my journey, and I am deeply grateful to have you both in my life.
Lastly but certainly not least, I extend my heartfelt gratitude to my aunt, Monique Kameni, whose unwavering support and encouragement have propelled me forward in life. Additionally, I am deeply thankful to my mother-in-law, Genevieve Tcheumbou Kameni, for her endless support and guidance. Your belief in me has been a constant source of strength, and I am truly grateful for your presence in my life.

DEDICATION

With heartfelt dedication, I humbly dedicate this PhD to my three cherished sons: Ashley Bryan Kameni Gouaya, Aaron Pangop Gouaya, and Math Tcheumbou Gouaya. This academic milestone serves as both a testament to my journey and a challenge to inspire theirs. As I entrust them with this legacy, I implore them to surpass not only my educational achievements and scientific contributions but also to forge their own paths of excellence and innovation. May this challenge ignite within them an unwavering commitment to knowledge, and a boundless determination to leave an indelible mark upon the world.

Abstract

The rapid evolution of facial recognition technology has elevated its significance across diverse applications, ranging from security systems to human-computer interaction. This thesis focuses on the intricate challenges faced by facial recognition systems, particularly emphasizing the impact of facial occlusion heightened by the widespread use of face masks during the COVID-19 pandemic. The study advances the field by exploring dimension reduction techniques, encompassing established methods such as Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA) and Auto-Encoter-based, alongside innovative approaches hybrid methodologies such as PCA-Autoencoder and LDA-Autoencoder. Notably, the study introduces Higher-Order Singular Value Decomposition (HOSVD) as a novel avenue for dimension reduction in facial recognition.

The examination of facial occlusion yields nuanced insights into the challenges faced by recognition systems in real-world scenarios. Techniques developed in response aim to effectively mitigate the adverse effects of facial occlusion, ensuring precision and reliability in identification processes, by developing face mask datasets adequate for the study.

In the dimension reduction realm, the study meticulously evaluates traditional and innovative techniques. PCA and LDA are scrutinized for effectiveness, while Autoencoder-based methods prove instrumental in facial feature extraction and dimension reduction. The innovative hybrid methodologies, PCA-Autoencoder and LDA-Autoencoder, demonstrate synergistic potential by capitalizing on the strengths of individual techniques. Tensor decomposition (HOSVD) emerges as a novel mathematical approach, providing a fresh perspective on dimension reduction strategies.

The findings of this research significantly contribute to the theoretical foundations and practical applications of facial recognition technology. Recommendations for future research include further exploration of diverse facial occlusion scenarios, real-time adaptive systems, and the integration of deep learning architectures to enhance dimension reduction methodologies. As technology advances, this thesis stands as a catalyst for ongoing innovation, fostering a deeper understanding of the intricate dynamics inherent in facial recognition systems.

***Keywords*—** Face mask, Face occlusion, Dimension reduction, Principal Component Analysis(PCA), Linear Discriminant Analysis(LDA), AutoEncoder, PCA-Auto-Encoder, LDA-Auto-Encoder, Face recognition, Machine learning, Tensor decomposition, High Order Singular Value Decomposition(HOSVD).

Name: Guy Mathias Gouaya
Student Number: 42700825
I declare that Mathematics Techniques with Machine learning Implementation for Facial Recognition is my own work, and that all the sources that I have used or quoted have been indicated and acknowledged by mean of complete references.
I further declare that I have not submitted this work or part of this work for any examination, at UNISA or another higher education institute before.

Signed in Johansburg the 21 November2023
GM Gouaya

[illegible]

List of Symbols

U: vector U
A: matrix

$\det(A)$: determinant of the matrix A

$a_{i,j}$: row i and column j entry of a matrix

$A_{i,j}$: Block matrix from the matrix A

$(A)_{i,j}$: row i and column j entry of a matrix A

A^T: transpose of the matrix A

$[1;n]$: set of integer from 1 to n

log_n logarithm of base n

A^*: complex conjugate transpose of the matrix A

$A \otimes B$: Kronecker-product of the matrices A and B

$Max(x, y)$: the maximum between x and y

$dim(U)$ dimension of the vector space U

$U \times V$: Cartesian product of the two vector spaces U and V

$\mathrm{vec}(A)$: vector operation of the matrix A

$||X||$: norm of the vector X

$\mathcal{A}$: tensor

$C = A * B$: Hadamard product of A and B

$||.||$: doc product

$d = ||W - W_m||^2$: norm

$\langle \mathcal{A}, \mathcal{B} \rangle$: scalar product of two tensor

$\sqrt{\langle \mathcal{A}, \mathcal{A} \rangle}$: frobenuis norm of a tensor

$A_{(n)}$: n mode unfolding of a $\mathcal{A}$

V^* : dual space of subspace V

List of Figures

2.1 Different face occlusions scenario. 9
2.2 Sample of Faces mask. 10
2.3 Sample from 5 different persons on the different datasets 15
2.4 Plot of the accuracy variation with the number of components 22
2.5 Class balance. Showing all the 40 subjects with their sample number for each, which are unequal 23

3.1 Flower type and percent specialization 26
3.2 Flower type and percent specialization 28
3.3 Flower type and percent specialization 29
3.4 Table of Different Dimension Techniques. 30
3.5 Accuracy percentage of machines learning classification on LFW after PCA for DR . . 33
3.6 Accuracy percentage of machines learning classification on LFW after PCA for DR . . 34
3.7 Accuracy percentage of machines learning classification on LFW after PCA for DR . . 37
3.8 Accuracy percentage of machines learning classification on LFW after PCA for DR . . 38
3.9 Accuracy percentage of machines learning classification on LFW after PCA for DR . . 38
3.10 Sample of Neural Network Diagram 40
3.11 Flower type and percent specialization 41
3.12 Flower type and percent specialization 42
3.13 Flower type and percent specialization 43
3.14 Auto Encoder Diagram 44
3.15 Accuracy percentage of machines learning classification on LFW after PCA for DR . . 46
3.16 Accuracy percentage of machines learning classification on LFW after PCA for DR . . 46
3.17 Accuracy percentage of machines learning classification on LFW after PCA for DR . . 47
3.18 Accuracy percentage of machines learning classification on LFW after PCA for DR . . 47
3.19 Accuracy percentage of machines learning classification on LFW after PCA for DR . . 59
3.20 Accuracy percentage of machines learning classification on LFW after PCA for DR . . 59
3.21 Accuracy percentage of machine learning classification on LFW after PCA for DR . . 60
3.22 Accuracy percentage of machines learning classification on LFW after PCA for DR . . 60
3.23 Accuracy percentage of machines learning classification on LFW after PCA for DR . . 61
3.24 Accuracy percentage of machines learning classification on LFW after PCA for DR . . 61
3.25 Accuracy percentage of machines learning classification on LFW after PCA for DR . . 62
3.26 Accuracy percentage of machines learning classification on LFW after PCA for DR . . 62
3.27 Accuracy percentage of machines learning classification on LFW after PCA for DR . . 63

List of Tables

2.1 Table of the maximum accuracy with the different classification models on the three created datasets. 21

3.1 Accuracy table for Olivetti. 57
3.2 Accuracy Table for LFW. 57

5.1 Different Classification Techniques. 78
5.2 Confusion Matrix Structure 92

list of publication

Submitted papers

[1]GM. Goauaya, and FE.Doungmo Goufo: ”The impact of COVID-19 face masks on facial recognition with PCA denoising: ” Submitted for reviewing.

[2]GM Gouaya and FE. Doungmo Goufo: ”Enhancing Facial Recognition models Accuracy through Dimension Reduction Techniques.” Submitted for reviewing.

Paper in Preparation

[3] GM Gouaya and FE. Doungmo Goufo: ” Deconstructing the multi factor variation of face recognition via tensor decomposition, case of HOSVD.” In progress.

Contents

Acknowledgement **i**

DEDICATION **iii**

Abstract **v**

List of Symbols **ix**

List of Publication **xv**

1 Introduction **1**

1.1 Introduction 1

1.2 Significance 2

1.3 Research Problem 4

1.3.1 Facial Occlusion 4

1.3.2 Dimension Reduction Techniques 4

1.4 Objectives of the Study 4

1.4.1 Facial Occlusion 4

1.4.2 Dimension Reduction Techniques 4

1.5 Research Questions 5

1.6 Organization of the thesis 5

2 Occlusion and Face Recognition: Case of face mask **7**

2.1 Introduction 7

2.2 Face recognition and occlusion 7

2.3 Mathematics description of face occlusion 8

2.4 Facial recognition with face mask 12

2.4.1 Facial recognition with face mask pre-COVID-19 12

2.4.2 Facial recognition with face mask post COVID-19 12

2.4.3 COVID 19 and facial recognition 13

2.5 Face-mask datasets 14

2.5.1 Image augmentation description 14

2.5.1.1 Rotation 15

2.5.1.2 Translation 16

2.5.1.3 Scaling 17

2.5.1.4 Shearing 18

2.5.1.5 Flipping 19

2.5.1.6 Gaussian Noise 20
2.6 Conclusion 21

3 Dimension Reduction 25
3.1 introduction 25
3.2 Significance 27
3.3 application 28
3.4 Dimension Reduction Techniques 29
3.4.1 PCA 31
3.4.1.1 PCA for Dimension Reduction 31
3.4.1.2 PCA for Face Recognition 32
3.4.1.3 Advantages of PCA 32
3.4.1.4 Disadvantages of PCA 32
3.4.2 Linear Discriminant Analysis (LDA) 33
3.4.2.1 Between-Class Variance 34
3.4.2.2 Within-Class Variance 35
3.4.2.3 Lower Dimensional Space 35
3.4.2.4 Advantages of LDA 36
3.4.2.5 Disadvantages of LDA 36
3.4.3 Auto-Encoder 39
3.4.3.1 Neural Network 39
3.4.3.2 NN Activation Functions 40
3.4.3.3 Auto-Encoder for Dimension Reduction 42
3.4.3.4 Advantages of Auto-Encoder 44
3.4.3.5 Disadvantages of Auto-Encoder 45
3.4.3.6 Comparison of PCA, LDA, and Auto-Encoder 48
3.4.4 Dimension Reduction based on Hybrid Techniques 49
3.4.5 PCA-Auto-Encoder 49
3.4.5.1 PCA-Auto-Encoder Pseudo Code 50
3.4.5.2 Advantages of PCA-Auto-Encoder 50
3.4.5.3 Disadvantages of PCA-Auto-Encoder 52
3.4.6 LDA-Auto-Encoder 53
3.4.6.1 LDA-Auto-Encoder Pseudo Code 53
3.4.6.2 Advantages of LDA-Auto-Encoder 54
3.4.6.3 Disadvantages of LDA-Auto-Encoder 55
3.5 Experiment 56
3.5.1 PCA-AutoEncoder 57
3.5.2 LDA-AutoEncoder 58
3.5.3 Experimental Validation 58

4 Tensor decomposition with face recognition 65
4.1 Introduction 65
4.2 Tensor definition 66
4.3 Bilinear maps 67
4.3.1 Tensor products of bilinear maps 68

4.4 Higher Order Singular Value Decomposition (HOSVD) 69
4.4.1 Similarity of HOSVD SVD 70
4.5 HOSVD and Dimension Reduction 72
4.5.1 Reducing the core tensor 72
4.5.2 Reducing the dimension of the factor matrix 73
4.5.3 Dimension reduction with HOSVD for face images 73
4.6 HOSVD and face recognition 73
4.6.1 Data Representation 74
4.6.2 Factorization 74
4.6.3 Dimension Reduction 74
4.6.4 Reconstruction 74
4.6.5 Recognition 74

5 Machine Learning Classification Methods and Metrics evaluation 77
5.1 Introduction 77
5.2 Classification methods 77
5.2.1 Logistic Regression 78
5.2.2 Support Vector Machine 79
5.2.3 Linear Discriminant Analysis 80
5.2.4 K-Nearest Neighbour 81
5.2.5 Decision Tree 81
5.2.6 Random Forest 82
5.2.7 XGboost 83
5.2.8 Naive Bayers 84
5.2.9 Multilayer Perceptron 85
5.3 Evaluation Metrics in Facial Recognition 86
5.3.1 Accuracy: 86
5.3.1.1 Mathematics of Accuracy 86
5.3.1.2 Advantage of Accuracy 86
5.3.1.3 Disadvantage of Accuracy 87
5.3.2 Precision: 87
5.3.2.1 Mathematics of Precision 87
5.3.2.2 Advantage of Precision 88
5.3.2.3 Disadvantages of Precision 88
5.3.3 Recall (Sensitivity): 88
5.3.3.1 Mathematics of Recall 89
5.3.3.2 Advantage of Recall 89
5.3.3.3 Disadvantage of Recall 89
5.3.4 Specificity 90
5.3.4.1 Mathematics of Specificity 90
5.3.4.2 Advantage of Specificity 90
5.3.4.3 Disadvantage of Specificity 90
5.3.5 F1 Score: 91
5.3.5.1 Mathematics of F1 Score 91
5.3.5.2 Advantages of F1 Score 91

5.3.5.3 Disadvantages of F1 Score 91
5.3.6 Confusion Matrix: 92
5.3.6.1 Structure of Confusion matrix 92
5.3.6.2 Advantage of Confusion Matrix 92
5.3.6.3 Disadvantage of confusion Matrix 93
5.4 conclusion 93

6 Conclusion 95
6.1 Summary of Findings 95
6.2 Contributions to Knowledge 95
6.2.1 Facial Occlusion 95
6.2.2 Dimension Reduction Techniques 96
6.3 Limitations and Challenges 96
6.4 Recommendations for Future Research 96
6.4.1 Facial Occlusion 96
6.4.2 Dimension Reduction Techniques 96
6.5 Conclusion 96

A Python Code 99
A.1 Data Augmentation 99
A.2 Converting data to npy format 100
A.3 PCA for dimension Reduction on the face mask data 103

B Classification process 105

Chapter 1

Introduction

1.1 Introduction

The security of individuals has become a major concern for various institutions. This has prompted states and organization to explore new and efficient security systems. Various research groups are actively addressing this concern, with a predominant focus on developing biometric techniques for person identification, enabling decisions on whether to grant access to public or private spaces, airports, companies, and more. In recent years, the advancements in technology have elevated the importance of face recognition in diverse array of applications, including security, human-computer interfaces, access control, and multimedia communications [50]. Face recognition, viewed as a pivotal step, finds application in personal identity verification, video surveillance, facial expression analysis, and gender classification. Formulating the face recognition problem, Jafri and Hamid [41], posed the question: given an input face image and a database of known individuals' face images, how can we verify or determine the identity of the person in the input image? Numerous research endeavors are actively addressing this intricate problem.
Many face recognition methods rely on linear algebra techniques, with notable examples including PCA (Principal Component Analysis), LDA (Linear Discriminant Analysis), and ICA (Independent Component Analysis). These techniques leverage mathematical principles to extract essential features from facial data, facilitating the identification and verification of individuals in various applications.
PCA (Principal Component Analysis) stands out as a widely used technique for dimension reduction. It operates as a statistical procedure employing an orthogonal transformation to convert a set of potentially correlated variables into a collection of values representing linear uncorrelated variables, known as principal components or, at times, principal modes of variation. This transformation can be accomplished through eigenvalue decomposition of a data covariance (or correlation) matrix or via Singular Value Decomposition (SVD) of the data matrix. PCA is instrumental in identifying patterns within the data, enabling the highlighting of similarities and differences. Additionally, it is utilized for data compression, achieved by reducing the number of dimensions without sacrificing crucial information. Notably, Eigenfaces, a popular PCA technique, was introduced by Sirovich and Kirby in 1987 [49] and later applied by Turk and Pentland [55] for face detection and recognition. For further details on the development of Eigenfaces, refer to [50].

1.2 Significance

Access control techniques play a crucial role in addressing security challenges, particularly in scenarios like airport entrances, verification for access to private and public places, and various account access methods. While existing solutions often rely on smart cards, plastic cards, passwords, tokens, and keys, these methods present certain challenges. Passwords or PIN codes can be forgotten by users or easily accessed by third parties. Plastic cards, smart cards, tokens, and keys can be lost or stolen, posing security risks. An effective solution to mitigate these challenges involves utilizing human biological traits.

Biometric techniques, including face recognition, fingerprints, finger geometry, hand geometry, hand vein, palm, iris, retina, and voice are widely regarded as highly reliable. Numerous research efforts in biometrics focus on these physiological characteristics. However, face recognition stands out for several advantages over other biometric methods. Some of these advantages include:

I Non-Voluntary Action

Unlike other biometric technologies that require a user's voluntary action, such as placing a hand on a rest for fingerprinting or hand geometry detection, face recognition can be performed passively. Users do not need to take explicit actions; their face images can be acquired from a distance by a camera.

II Damage Resilience

Hand and finger systems may be rendered ineffective if the epidermal tissue is damaged (cracked, bruised, etc.). In contrast, facial recognition remains robust even in the presence of such damage.

III Equipment Sensitivity

Technologies like iris and retina recognition often require expensive equipment and are sensitive to any body motion. In contrast, reliable face recognition algorithms, coupled with appropriate preprocessing of images, can compensate for noise and variations in orientation, scale, and illumination.

IV Susceptibility to External Facts

Voice recognition is susceptible to background noises in public places and auditory fluctuations on phone lines or tape recordings. On the other hand, face recognition can be effectively implemented with appropriate algorithms, compensating for noise and variations.

V Forgery Concerns:

Signatures can be forged or modified, posing a risk to security. Facial recognition, when implemented with reliable algorithms, is less susceptible to forgery.

VI Health Risks:

Technologies requiring multiple individuals to use the same equipment for capturing biological characteristics. This can expose users to the transmission of germs and impurities. In contrast, face recognition is non-intrusive and does not carry such health risks.

Topic focusing on big data, particularly in image processing and signal processing, has captured the attention of numerous researchers. Here is a brief overview of some relevant studies:

I Tensor Decomposition for Big Data Improvement

Mathematical tools such as tensor decomposition, including techniques like singular value decomposition, have demonstrated significant potential for improving big data applications in image and signal processing. Tensor decomposition, with its ability to reduce the rank of tensors and consequently data volume, has been explored in various articles, including [1, 43, 50, 51, 53]. A comprehensive survey on tensors can be found in [45].

II Comparison of Face Recognition Techniques

Research has extensively compared different face recognition techniques, particularly PCA (Principal Component Analysis) and LDA (Linear Discriminant Analysis). Studies, such as [48, 60], have found that PCA provides a simple and efficient algorithm but can be time-consuming. On the other hand, LDA, while overcoming some limitations of PCA, faces singularity problems and illumination challenges. Neural network techniques have been proposed to enhance LDA's performance.

III Diverse Techniques in Face Recognition:

Kumar and Kaur introduced various face recognition techniques in [48]. For further exploration of face recognition methods, additional sources include [5, 13, 73]. These studies contribute to the understanding and advancement of face recognition technologies.

Integrating mathematical techniques with machine learning approaches offers a promising way to enhance the robustness and efficiency of facial recognition system. Leveraging mathematical principles alongside machine learning algorithms has the potential to address challenges such as variability in facial expressions, lighting conditions, pose variations, and, notably, facial occlusion due to face masks, which gained prominence during the COVID-19 pandemic. This study seeks to explore innovative solutions from the dimension reduction perspective that combine mathematical techniques and machine learning to overcome these challenges and advance facial recognition technology.

1.3 Research Problem

Despite the progress made in facial recognition technology, there are persistent challenges that require innovative solutions. Traditional methods may struggle to handle complex scenarios, including facial occlusion, specially, the case of masked face of which one of the major problem is the lack of dataset. Furthermore, there is a need to explore advanced mathematical techniques combined with machine learning techniques for dimension reduction to improve accuracy base, reliability, and adaptability in facial recognition systems.

The primary research problem addressed in this dissertation is twofold:

1.3.1 Facial Occlusion

Facial recognition systems face significant challenges in scenarios of facial occlusion, particularly when individuals wear face masks. Traditional methods struggle to accurately identify and authenticate individuals under these circumstances, emphasizing the need for innovative solutions and datasets.

1.3.2 Dimension Reduction Techniques

While dimension reduction techniques such as Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and Autoencoder have been extensively investigated, there is a continued need to explore their integration and innovation. This research aims to advance the understanding of dimension reduction by proposing and evaluating novel hybrid techniques, including PCA-Autoencoder and LDA-Autoencoder. Additionally, the exploration of tensor decomposition, specifically Higher-Order Singular Value Decomposition (HOSVD), introduces a novel perspective on dimension reduction in facial recognition.

1.4 Objectives of the Study

The research objectives are detailed to address the multifaceted challenges posed by facial occlusion and dimension reduction techniques:

1.4.1 Facial Occlusion

Investigate the impact of facial occlusion, particularly face masks, on the performance of facial recognition systems.
Develop a process to artificial produce dataset of faces with mask, dataset with faces without mask and and dataset with faces consisting of mask and no mask. All the face from the same people to find the impact of masked face on the accuracy of face mask.
Find an alternative solution to mitigate the challenges posed by facial occlusion and improve accuracy and reliability in identification.

1.4.2 Dimension Reduction Techniques

Evaluate the effectiveness of traditional dimension reduction techniques, including PCA and LDA, in the context of facial recognition.

Explore and implement Autoencoder-based dimension reduction techniques for facial feature extraction.
Propose and assess innovative hybrid techniques, namely PCA-Autoencoder and LDA-Autoencoder, to leverage the strengths of multiple methods.
Investigate tensor decomposition, specifically HOSVD, as an alternative approach for dimension reduction in facial recognition.
Investigate the mathematics background of classical facial recognition classification as well as the metric evaluation

1.5 Research Questions

This study seeks to answer the following key research questions:

How can mathematical techniques for image augmentation and dimension reduction, particularly in the context of facial occlusion, be effectively integrated into facial recognition systems?
What dimension reduction techniques, including PCA, LDA, Autoencoder, and hybrid approaches, are most effective for facial feature extraction and identification?
How does tensor decomposition, specifically HOSVD, contribute to dimension reduction in facial recognition?
How do machine learning classification models perform when applied to the proposed methodology, and what metrics are most relevant for evaluation?

1.6 Organization of the thesis

The remainder of this thesis is structured as follows:
Chapter 2 dives into a comprehensive review of existing literature on facial recognition techniques, with a focus on challenges posed by facial occlusion, particularly face masks before, during and beyond the COVID-19 pandemic. In this chapter, we use mathematics tool with machine learning to develop three datasets to address the shortage of public dataset in this area and we show how PCA dimension reduction can help in the accuracy of face mask.
In Chapter 3, an in-depth explanation of dimension reduction techniques, including PCA, LDA, Autoencoder, and innovative hybrid techniques (PCA-AutoEncoder, LDA-AutoEncoder) are detailed.
We use Chapter 4 for the exploration of tensor decomposition methods, specifically Higher-Order Singular Value Decomposition (HOSVD), for dimension reduction in facial recognition.
We continue in Chapter 5, the mathematics application of machine learning classification models and evaluation metrics on the proposed methodology are discussed.
Finally, Chapter 6 is intended to the summary of the study's contributions, limitations, and recommendations for future research.
By addressing the outlined objectives, this thesis aims to provide valuable insights into the synergies between mathematics and machine learning for advancing facial recognition technology, particularly in the context of facial occlusion and challenges introduced by the COVID-19 pandemic with the development of technique on dimension reduction.

Chapter 2

Occlusion and Face Recognition: Case of face mask

2.1 Introduction

Facial recognition systems can encounter performance challenges when dealing with various factors such as lighting conditions, face pose, changes in facial appearance, and occlusions, among others. In 2021, Huang et al.[39], highlighted a specific challenge posed by the widespread use of face masks during the COVID-19 pandemic. They noted that face recognition techniques, which are crucial for identification purposes, faced significant difficulties when confronted with masked faces. This presented significant dilemmas for authentication applications that rely on face recognition, including face access control, facial recognition gates, and face authentication, among others. One notable example is in the context of public security checks at railway stations, where security systems based on face recognition often struggled to recognize individuals wearing masks.This underscores the importance of addressing the impact of face masks on facial recognition accuracy, which is a key focus of this chapter.
Due to the COVID-19 pandemic, the mandatory use of face masks has become a common practice, and even after the restrictions were lifted, many people have continued to wear masks as a habit. Some individuals wear masks for health reasons, such as allergy protection or filtering the air in polluted environments. However, others, including criminals, have taken advantage of masks to avoid easy identification. All this can be found on the following link`https://www.health.com/`. This situation poses a significant challenge to traditional facial recognition systems, as face masks create a scenario of facial occlusion. This face mask has come with some challenges, one of them is explained in here.
The widespread use of face masks has raised concerns about an increase in thefts and robberies, as masks can effectively conceal most of a person's face. This concealment makes it difficult for both humans and facial recognition systems to identify individuals, even if they are known to others [30]. This highlights the need to investigate the impact of face masks on the accuracy of conventional facial recognition systems and develop strategies to address this challenge effectively.

2.2 Face recognition and occlusion

As discussed in section 2.1, facial recognition encounters significant challenges when faced with occluded images. Occlusion can occur due to various factors, including sunglasses, hats, masks, cellphones, or hands, just to name some few. Among these occlusion types, face masks are considered as one of

the most challenging occlusions, as they cover a substantial portion of the face, approximately 60%, including the mouth, nose, and chin. Several works related to face occlusion have been conducted and documented in articles such as [37, 59].
To understand how face is occluded, we start by listing all the different facial occlusions at the best of our knowledge.

I Eyeglasses Occlusion: Eyeglasses occlusion happens by person wearing glasses, in this case, the face is partially cover on their eyes, eyebrows, and parts of the forehead, cheeks, and temples.

II Hair Occlusion: Hair occlusion occurs from long hair, bangs, or hairstyles covering parts of the forehead, eyes, or cheeks, obstructing facial features. It can also come from a person who has a beard, moustache, or other facial hair that obscures parts of the mouth, cheeks, and chin.

III Hat Occlusion: Hat or cap wearing can covers the forehead, hairline, eyebrows, and parts of the eyes and cheeks.

IV Hand Occlusion: Hand occlusion happens when a person touches or covers their face with their hand(s), leading to occlusions around the cheeks, mouth, nose, and even eyes.

V Object Occlusion: If a person holds an object such as book, phone, tea cup, pen, etc., in front of their face, it usually causing occlusions in the facial region.

VI Environmental Occlusion: Environmental occlusion happens when the face is partially obscured by objects, shadows, or other environmental factors.

VII Shadow Occlusion:

VIII Pose Occlusion: Pose occlusion occurs when a person's head is in an extreme pose, tilting, or rotation, leading to self-occlusions of facial features.

IX Mask Occlusion:

Wearing face mask will most of the time covers one nose mouth or chin, hence partially or completely obstructing those facial features.

In figure 2.1 from [82], one can see some pictures of occluded faces with different cases.

We are in the view that there are less mathematics literatures related to face occlusion. Hence, we try to close this cap with the following section.

2.3 Mathematics description of face occlusion

Using some mathematics operation such as Hadamard product[35], on computer vision, we can superpose two different images together. In our case, we use the Hadamard matrix product to construct our occluded face.
Before going into the mathematics construction of face occlusion, we first start by giving a definition of the Hadamard product and some theorems that can be use for our purpose.

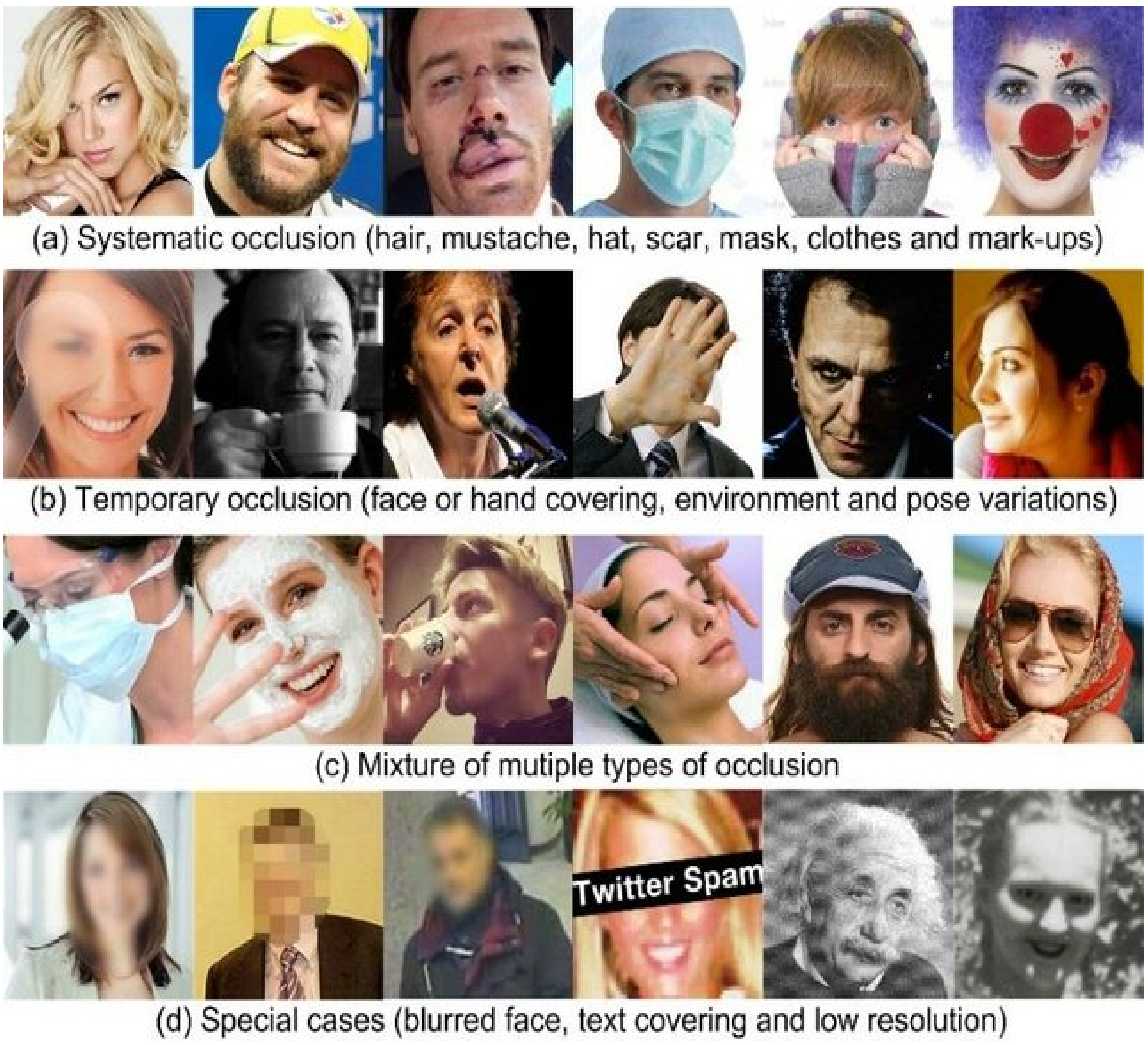

(a) Systematic occlusion (hair, mustache, hat, scar, mask, clothes and mark-ups)
(b) Temporary occlusion (face or hand covering, environment and pose variations)
(c) Mixture of mutiple types of occlusion
Twitter Spam
(d) Special cases (blurred face, text covering and low resolution)

Figure 2.1: Different face occlusions scenario.

Definition 1 *Let A and B be two matrices of the same size $m \times n$, the Hadamard product of A and B is given by*

$$C = A * B,$$

where $C_{ij} = A_{ij}B_{ij}$ are the element wise multiplication.

Theorem 1 *Let A and B be two $m \times n$ matrices, Hadamard product of A and B is commutative:*

$$A * B = B * A$$

Figure 2.2: Sample of Faces mask.

Proof. Let A and B be two $m \times n$ matrices,

$$\begin{aligned} A * B &= A_{ij}B_{ij} && (2.1)\\ &= B_{ij}A_{ij} && (2.2)\\ &= B * A. && (2.3) \end{aligned}$$

Which completes the proof.
♣

Theorem 2 *The Hadamard product has an identity element which is given by the matrix with all its entries equal to* 1.
For any $m \times n$ matrix A, if the $m \times n$ matrix P is such that $p_{ij} = 1$ for all i and j then

$$A * P = P * A = A.$$

Proof. The proof follows directly from the definition of Hadamard product and the fact that 1 is and identity element in the simple multiplication. ♣

Theorem 3 *Let A be any $m \times n$ matrix, then A has an Hadamard inverse, that we denote $\hat{A}$ iff for any $i \in 1, m$ and $j \in 1, n$ we have $A_{ij} \neq 1$.*

Proof.
($\Rightarrow$) Let A_{ij} be any $m \times n$ matrix. Assuming that A has an Hadamard inverse say $\hat{A}_{ij}$, hence for any $i = 1, 2..., m$ and any $j = 1, 2..., n$ we have that $A_{ij}\hat{A}_{ij} = 1$ therefore $\hat{A}_{ij} = \frac{1}{A_{ij}}$, which is only possible if $A_{ij} \neq 0$.
($\Leftarrow$) Let A_{ij} be any $m \times n$ matrix. Suppose $\forall A_{ij}$, we have $A_{ij} \neq 0$. Hence $\exists p$ such that $A_{ij}p = 1$. Hence we can find a $m \times n$ matrix say $\hat{A}_{ij}$, such that $A_{ij} * \hat{A}_{ij} = 1$, $\forall i = 1, 2, ..., n$ and $\forall j = 1, 2, .., m$. Therefore, any nonsingular matrix has an Hadamard inverse.
This concludes the proof. ♣

Now, let handle the occlusion image itself. Let set our images as:
F is our original face image represented as a matrix. Each element $F(i, j)$ of the matrix corresponding to the pixel intensity at position (i, j) in the image.

M be the binary mask also represented as a matrix, where $M(i, j) = 1$ indicates an none-occluded pixel, and M(i, j) = 0 indicates an occluded pixel. The binary mask has the
same size as the original face image.

F_O the occluded face image. Mathematically, it is obtained by element-wise multiplication (Hadamard product) of the original face image F and the binary occlusion mask M.

$$F_O(i, j) = F(i, j) * M(i, j)$$

In the equation above, $F_O(i, j)$ represents the pixel intensity of the occluded face image at position (i, j). The Hadamard product (*) between $F(i, j)$ and $M(i, j)$ results in $F_O(i, j)$, where occluded regions ($M(i, j) = 0$) in the binary mask suppress the corresponding pixels in the original face image, while none-occluded regions ($M(i, j) = 1$) retain their original pixel intensities of the original face image.
It follow from the above description that to artificially perform a face occlusion one just need to choose an appropriate occlusion matrix $M(i, j)$ and using Hadamard product with the original $f(i, j)$ face matrix to get the modify image. Here is a simple example of how the Hadamard product will affect the image. Let

$$F = \begin{bmatrix} 255 & 30 & 10 & 27 \\ 8 & 20 & 10 & 25 \\ 40 & 125 & 75 & 35 \\ 254 & 30 & 15 & 33 \end{bmatrix}$$

be an example of the matrix representation of a face image.

$$M = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

be an example of the matrix representation of the occlusion object(face mask, sunglasses, hand, cup etc.).
Then applying the Hadamard product with this F and M will give us F_O as

$$F_O = \begin{bmatrix} 255 & 30 & 10 & 27 \\ 8 & 20 & 10 & 25 \\ 40 & 125 & 75 & 35 \\ 254 & 30 & 15 & 33 \end{bmatrix} * \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 255 & 30 & 10 & 27 \\ 0 & 0 & 10 & 0 \\ 0 & 125 & 75 & 35 \\ 254 & 0 & 15 & 33 \end{bmatrix}$$

Since we are mostly interested with the face mask, we will construct our M to be a masked face. Having explain how face mask as an occlusion can be build. In the next section, we discuss the face mask occlusion with facial recognition.

2.4 Facial recognition with face mask

Given the challenging nature of face masks as occlusions for facial recognition systems, our attention was drawn to the following questions:

1. To what extent can face masks impact the accuracy of traditional facial recognition systems?
2. Are there any research studies related to facial recognition with face masks?

The second question is addressed in two distinct periods: pre-COVID-19 and post-COVID-19. This partition allows us to explore the evolution of research on facial recognition in the context of face masks and assess the impact of the COVID-19 pandemic on this field of study.

2.4.1 Facial recognition with face mask pre-COVID-19

Even before the WHO made face masks mandatory during the COVID-19 pandemic, masked faces had garnered attention from industries and research groups. For instance, the Japanese company NEC was actively working on a face recognition system designed for individuals wearing masks NEC link.
In a BBC newsletter BBC link, dated March 25, 2021, titled "Facial recognition beats the Covid-mask challenge" authored by James Clayton, a North America technology reporter, it was reported that NEC had achieved an impressive 99% accuracy rate in recognizing faces even when individuals were wearing masks. This marked a significant advancement in overcoming the challenges posed by face masks to facial recognition technology.
Prior to the COVID-19 pandemic, one of the primary challenges faced by facial recognition systems was the scarcity of relevant datasets. In a publication by Adjabi et al [3] in 2020 , a list of 23 well-known datasets related to facial recognition was provided in chronological order. Notably, among these datasets, only one dataset contained images of individuals wearing face masks. This dataset, named the AR dataset, was created by Martinez and Benavente and made available to the public in 1998 [54]. However, this dataset only contained nine images of each individual wearing masks, making it challenging to thoroughly evaluate the impact of masked faces on facial recognition. Consequently, it is evident that prior to the COVID-19 pandemic, limited research and datasets were available for facial recognition with masked faces. Some of the notable works in this area can be found in [26, 29, 68]. These articles provide valuable insights into the challenges and approaches related to facial recognition in the context of masked faces before the COVID-19 pandemic.

2.4.2 Facial recognition with face mask post COVID-19

As discussed in the introduction, the mandatory requirement of wearing face mask has come with many issues. This requirement presented a significant challenge to traditional face recognition systems, leading to a surge in research on masked face recognition. A selection of relevant works in this area can be found in: [36, 6, 30, 23, 37, 44, 34, 39].
This research area has gained importance due to the practical need for accurate facial recognition even when individuals are wearing masks. It highlights the adaptability and resilience of technology in the face of evolving challenges.
This research will likely continue to evolve, as it has implications for addressing other facial occlusions and given that wearing face masks has become a habit for many individuals.

Following the COVID-19 pandemic, one of the immediate challenges that researchers and industries base of face recognition faced, was the availability of datasets containing images of individuals wearing face masks. One solution to this challenge was proposed by Anwar and Raychowdhury [6], by providing the public with a tool call MaskTheFace. This tool can be use to effectively put mask on large face dataset. In the research presented in [36], the authors aimed to build a machine learning model based on Convolutional Neural Networks (CNN). Their approach involved using the machine to discard the masked region of the face and utilizing the CNN to extract the most relevant features from the remaining region, which in this case, primarily included the eyes and the forehead region. Their model was thoroughly tested on datasets such as RMFRD, SMFRD, and MFDD, as detailed in [77].
This process of discarding a portion of the masked face represents one of three traditional approaches to addressing face occlusion. The other two methods include the local matching approach, which focuses on comparing the similarity between matching processes, and the restoration approach, where efforts are made to use a gallery to restore the occluded regions.
On March 25, 2021, BBC News, through their North America technology reporter, published a report from the National Institute of Standards and Technology (NIST). The report highlighted that NIST had tested 89 commercial facial recognition algorithms and found error rates ranging from 5% to 50% when matching faces with digitally applied mask photos of the same person. This underscores the importance of being cautious about the accuracy rates of the machine learning models employed in facial recognition systems.
These findings emphasize the ongoing challenges in achieving accurate facial recognition, especially in scenarios involving face masks and occlusions.

2.4.3 COVID 19 and facial recognition

Occlusion due to masked faces cover the frontal face. This area contains rich features, including the nose and the mouth[37]. Face mask makes higher inter-class similarity and inter-class variation due to covering a large area of the face which tricks the facial verification process of face recognition system [77].
During the COVID-19 pandemic, touching one's nose, mouth, or eyes after touching a surface contaminated with the coronavirus can be a way for transmission of the virus [21]. Therefore, relying on identification based on fingerprint and other contact biometric systems becomes very risky. This leaves us with no choice but to use contactless biometric systems, most of which will be facial recognition due to its user-friendliness and high accuracy.
Considering this and taking into account the recommendations of the WHO, which at the time required everyone to wear face masks during the COVID-19 pandemic, it becomes evident that masked faces pose some conflicting challenges. On one hand, face recognition is one of the appropriate systems for various security applications such as CCTV, ATM, passport checking, railway checking, etc., during the COVID-19 pandemic due to its contactless nature. On the other hand, masked faces, which were compulsory during the pandemic, can drastically reduce the accuracy of face recognition systems.
Hence, our interest lies in examining the impact of face masks on facial recognition. This problem can find improvement by using some dimension reduction techniques to extract the most important components of the faces. Before delve in to dimension reduction in the next chapter, we start by examining the question of dataset with face mask.

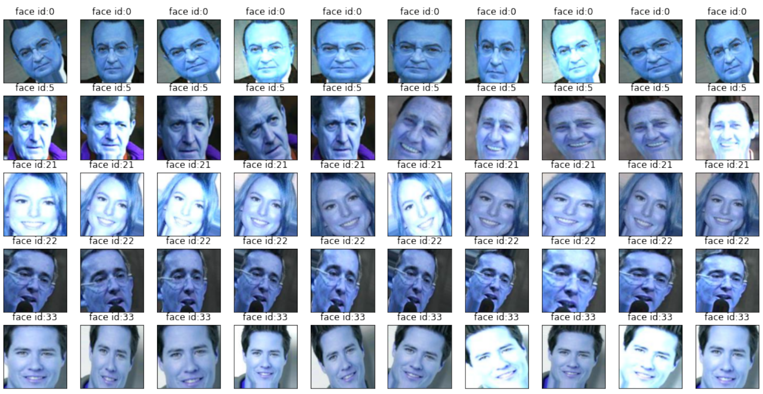
2.5 Face-mask datasets

The challenge of data availability, especially in the context of facial recognition with masked faces, is a significant concern in the field of machine learning. Despite the importance of addressing this issue, researchers have faced limitations, as existing datasets may not be suitable for solving specific problems, such as the accurate recognition of faces with masks.

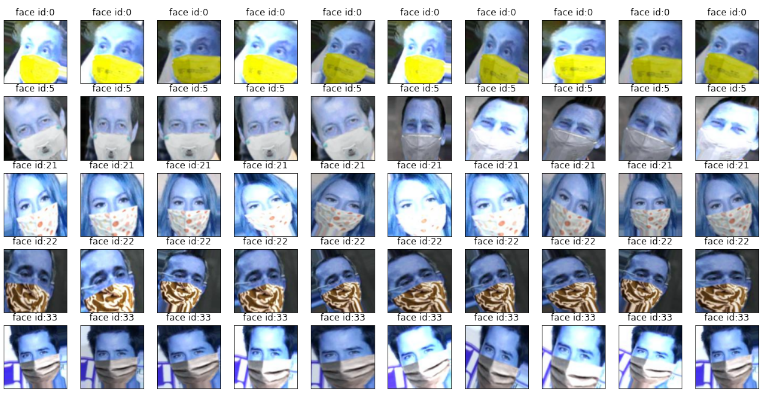
With the heightened relevance of masked faces post-Covid-19, the inadequacy of available datasets became more apparent. Researchers encountered challenges in obtaining data that aligns with the specific aspects they aimed to investigate, emphasizing the need for more comprehensive datasets tailored to the nuances of facial recognition in the presence of masks.
Due to the scarcity of suitable existing datasets, we were compelled to create our own dataset for facial recognition, particularly focusing on individuals wearing masks. We leveraged images from MLFW [75], selecting 40 distinct individuals. For each person, we obtained two images without a mask and two images with a mask. However, facing challenges in finding individuals with both masked and unmasked images, we employed Python code for data augmentation. This process yielded 880 images from the original 80 unmasked images, resulting in each person having 22 distinct images. The dataset, stored in a numpy file, became our repository for individuals without masks.
Figure 2.3(a) showcases 50 images from this dataset, featuring 5 randomly selected individuals, each represented by 10 images.
Following a similar approach, we created a dataset for individuals wearing masks using images of the same individuals. This dataset of masked faces underwent the same augmentation process, resulting in 22 images for each of the 40 individuals. Figure 2.3(b) provides a glimpse of this dataset, displaying images of five different individuals, each represented by 10 images.
Our third dataset amalgamates images of the same individuals from the unmasked and masked datasets. This compilation allows us to examine and analyze how machine learning models perform when presented with a mix of images with and without masks. Figure 2.3(c) provides a glimpse of this dataset, displaying images of five different individuals, each represented by 10 images.
Before delving to the face recognition, we examine the structure of our data with the help of python we could do some visualization on our data. We have the display of the class balance of each of our three datasets, which is provided in Figures 2.4. From these figures, it is evident that the datasets exhibit an imbalance in terms of class distribution. Note that class imbalance occurs when the number of instances in each class is significantly different, and this can potentially impact the Performance of machine learning models.
Class imbalance can introduce bias during the training of machine learning models, as the algorithm may become more inclined to predict the majority class, neglecting the minority class. To mitigate this issue and ensure fair model training, techniques such as oversampling the minority class or under sampling the majority class can be employed.

2.5.1 Image augmentation description

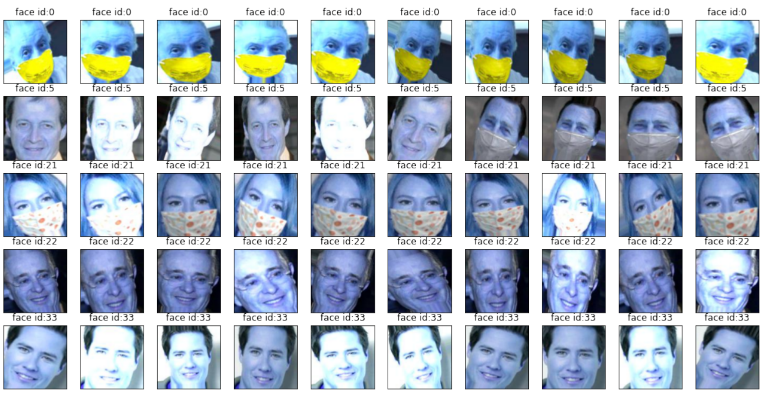
We felt that for the completeness sake, it is very important to give some explanations of our image augmentation implementation with some Mathematics descriptions of the concept used for it.
To expand the dataset and generate additional face images, we can apply various image augmentation techniques to each of the 80 original face images. Let us consider the following image augmentation



(a) Unmasked faces data



(b) Masked facees data



(c) Mixed faces data

Figure 2.3: Sample from 5 different persons on the different datasets

techniques:
Now we give some Mathematical details of the different transformation that can be used in image augmentation.

2.5.1.1 Rotation

Rotating a face image involves applying a transformation to the image to change its orientation. In the context of two-dimensional image processing, image rotation can be represented using mathematical concepts like transformations and matrices.
Transformation Matrix for 2D Rotation: The transformation matrix for rotating a 2D point (x, y) by

an angle θ counter clockwise around the origin is given by:

$$R(\theta) = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix}.$$

where, $cos(\theta)$ and $sin(\theta)$ represent the cosine and sine of the angle θ, respectively.
Image Rotation Algorithm: To rotate a face image, we apply the same transformation to each pixel in the image. The rotated image, denoted as I_R, is obtained by multiplying the transformation matrix $R(\theta)$ with the coordinates of each pixel in the original image I.
Let (x, y) represent the coordinates of a pixel in the original image I, and (x_R, y_R) represent the coordinates of the corresponding pixel in the rotated image I_R.

$$(x_R, y_R) = R(\theta)(x, y).$$

The rotation operation is performed for each pixel (x, y) in the original image, and the resulting coordinates (x_R, y_R) are used to obtain the pixel value in the rotated image I_R.
Interpolation: During image rotation, the new pixel coordinates (x_R, y_R) may not correspond to exact integer positions in the original image. Therefore, interpolation techniques, such as nearest neighbour interpolation or bilinear interpolation, are used to estimate the pixel values for non-integer coordinates.
Center of Rotation: In some cases, the rotation is performed around a point other than the origin $(0, 0)$. To rotate an image around a specific point (cx, cy), we first translate the image so that the centre of rotation becomes the origin, apply the rotation, and then translate it back to its original position.

$$(x_R, y_R) = R()(x - cx, y - cy) + (cx, cy).$$

Here, (cx, cy) represents the coordinates of the centre of rotation.
Image rotation is a fundamental operation in image processing and computer vision. It is often used for various tasks, including image alignment, feature extraction, and data augmentation. By using the transformation matrix for rotation and appropriate interpolation techniques, we can efficiently rotate face images to achieve the desired orientation.

2.5.1.2 Translation

Translating a face image involves shifting the image along the x and y axes to change its position. In the context of two-dimensional image processing, image translation can be represented using mathematical concepts like transformations matrices.
Translation Matrix for 2D Image: The transformation matrix for translating a 2D point (x, y) by distances dx and dy along the x and y axes, respectively, is given by:

$$T(dx, dy) = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix}.$$

Here, dx represents the translation distance along the x-axis, and dy represents the translation distance along the y-axis.
Image Translation Algorithm: To translate a face image, we apply the same transformation to each pixel in the image. The translated image, denoted as I_T, is obtained by multiplying the transformation

matrix $T(dx, dy)$ with the homogeneous coordinates$(x, y, 1)$ of each pixel in the original image I.
Let (x, y) represent the coordinates of a pixel in the original image I, and (x_T, y_T) represent the coordinates of the corresponding pixel in the translated image I_T.

$$(x_T, y_T, 1) = T(dx, dy)(x, y, 1).$$

The translation operation is performed for each pixel (x, y) in the original image, and the resulting coordinates (x_T, y_T) are used to obtain the pixel value in the translated image I_T.
Interpolation: Similar to image rotation, the new pixel coordinates (x_T, y_T) may not correspond to exact integer positions in the original image. Therefore, interpolation techniques, such as nearest neighbour interpolation or bilinear interpolation, are used to estimate the pixel values for non-integer coordinates.
Translation Center: The translation center represents the origin of the translation operation. By default, the translation is applied with respect to the origin $(0, 0)$. However, in some cases, the translation can be performed with respect to a specific point (cx, cy) as the translation centre.

$$(x_T, y_T, 1) = T(dx, dy)(x - cx, y - cy, 1) + (cx, cy, 1).$$

Here, (cx, cy) represents the coordinates of the translation centre.
Image translation is a fundamental operation in image processing and computer vision. It is often used for various tasks, including image alignment, data augmentation, and image registration. By using the translation matrix and appropriate interpolation techniques, we can efficiently translate face images to achieve the desired position.

2.5.1.3 Scaling

Scaling a face image involves resizing the image either larger or smaller. In the context of two-dimensional image processing, image scaling can be represented using mathematical concepts like transformations and matrices.
Scaling Matrix for 2D Image: The transformation matrix for scaling a 2D point (x, y) by factors sx and sy along the x and y axes, respectively, is given by:

$$S(sx, sy) = \begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Here, sx represents the scaling factor along the x-axis, and sy represents the scaling factor along the y-axis. If sx and sy are both greater than 1, the image will be enlarged (zoomed in). If sx and sy are both less than 1, the image will be reduced (zoomed out).
Image Scaling Algorithm: To scale a face image, we apply the same transformation to each pixel in the image. The scaled image, denoted as I_S, is obtained by multiplying the transformation matrix $S(sx, sy)$ with the homogeneous coordinates $x, y, 1)$ of each pixel in the original image I.
Let (x, y) represent the coordinates of a pixel in the original image I, and (x_S, y_S) represent the coordinates of the corresponding pixel in the scaled image I_S.

$$(x_S, y_S, 1) = S(sx, sy)(x, y, 1).$$

The scaling operation is performed for each pixel (x, y) in the original image, and the resulting coordinates (x_S, y_S) are used to obtain the pixel value in the scaled image I_S.
Interpolation: When scaling an image, the new pixel coordinates (x_S, y_S) may not correspond to exact integer positions in the original image. Therefore, interpolation techniques, such as nearest neighbour interpolation or bilinear interpolation, are used to estimate the pixel values for non-integer coordinates.
Scaling Center: By default, the scaling is applied with respect to the origin $(0, 0)$. However, in some cases, the scaling can be performed with respect to a specific point (cx, cy) as the scaling centre.

$$(x_S, y_S, 1) = S(sx, sy)(x - cx, y - cy, 1) + (cx, cy, 1).$$

Here, (cx, cy) represents the coordinates of the scaling centre.
Image scaling is a common operation in image processing and computer vision. It is used for various purposes, including resizing images for display or analysis, adjusting image sizes in datasets, and performing data augmentation for training machine learning models. By using the scaling matrix and appropriate interpolation techniques, we can efficiently scale face images to achieve the desired size.

2.5.1.4 Shearing

Shearing (also known as skewing) a face image involves transforming the image by shifting its rows or columns. In the context of two-dimensional image processing, image shearing can be represented using mathematical concepts like transformations and matrices.
There are two types of shearing commonly used in image processing: horizontal shearing and vertical shearing.
Horizontal Shearing: Horizontal shearing involves shifting each row of the image by a certain amount proportional to its y-coordinate. The transformation matrix for horizontal shearing is given by:

$$S_H(dx) = \begin{bmatrix} 1 & dx & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Here, dx represents the horizontal shearing factor. When dx is positive, the image is sheared to the right, and when dx is negative, the image is sheared to the left.
Vertical Shearing: Vertical shearing involves shifting each column of the image by a certain amount proportional to its x-coordinate. The transformation matrix for vertical shearing is given by:

$$S_V(dy) = \begin{bmatrix} 1 & 0 & 0 \\ dy & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Here, dy represents the vertical shearing factor. When dy is positive, the image is sheared downwards, and when dy is negative, the image is sheared upwards.
Image Shearing Algorithm: To shear a face image, we apply the corresponding shearing transformation matrix to each pixel in the image. The sheared image, denoted as I_s, is obtained by multiplying the appropriate shearing matrix (S_H or S_V) with the homogeneous coordinates $(x, y, 1)$ of each pixel in the original image I.
For horizontal shearing:

$$(x_s, y_s, 1) = S_H(dx)(x, y, 1).$$

For vertical shearing:

$$(x_s, y_s, 1) = S_V(dy)(x, y, 1).$$

The shearing operation is performed for each pixel (x, y) in the original image, and the resulting coordinates (x_s, y_s) are used to obtain the pixel value in the sheared image I_s.
Interpolation: Similar to image rotation and scaling, the new pixel coordinates (x_s, y_s) may not correspond to exact integer positions in the original image. Therefore, interpolation techniques, such as nearest neighbour interpolation or bilinear interpolation, are used to estimate the pixel values for non-integer coordinates.
Image shearing is a geometric transformation used in image processing and computer vision to create various effects or correct distortions. By using the appropriate shearing matrix and interpolation techniques, we can efficiently shear face images to achieve the desired shearing effect.

2.5.1.5 Flipping

Flipping a face image involves mirroring or reversing the image along a specified axis. There are two types of flipping commonly used in image processing: horizontal flipping and vertical flipping.
Horizontal Flipping: Horizontal flipping, also known as left-right flipping, involves reversing the image from left to right. It means that the pixels on the left side of the image become the pixels on the right side, and vice versa.
The transformation matrix for horizontal flipping is given by:

$$H_F = \begin{bmatrix} -1 & 0 & w-1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Here, w represents the width of the image.
Vertical Flipping: Vertical flipping, also known as up-down flipping, involves reversing the image from top to bottom. It means that the pixels on the top side of the image become the pixels on the bottom side, and vice versa.
The transformation matrix for vertical flipping is given by:

$$V_F = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & h-1 \\ 0 & 0 & 1 \end{bmatrix}.$$

Here, h represents the height of the image.

Image Flipping Algorithm: To flip a face image, we apply the corresponding flipping transformation matrix (H_F or V_F) to each pixel in the image. The flipped image, denoted as I_f, is obtained by multiplying the appropriate flipping matrix with the homogeneous coordinates $(x, y, 1)$ of each pixel in the original image I.

For horizontal flipping:

$$(x_f, y_f, 1) = H_F(x, y, 1).$$

For vertical flipping:

$$(x_f, y_f, 1) = V_F(x, y, 1).$$

The flipping operation is performed for each pixel (x, y) in the original image, and the resulting coordinates (x_f, y_f) are used to obtain the pixel value in the flipped image I_f.

Image flipping is a common operation in image processing and computer vision. It is used for various purposes, such as data augmentation, image alignment, and creating mirror images for artistic or visual effects. By using the appropriate flipping matrix, we can efficiently flip face images to achieve the desired mirroring effect.

2.5.1.6 Gaussian Noise

Adding Gaussian noise to a face image involves introducing random variations to the pixel values based on a Gaussian distribution. This is a common method to simulate real-world noise in images and assess the robustness of image processing algorithms. Mathematically, Gaussian noise is often represented as a random variable with a Gaussian (normal) distribution.

Here's how you can mathematically describe the process of adding Gaussian noise to a face image:

Gaussian Noise Model: Let's denote the original pixel value of a pixel at position (i, j) in the face image as $I(i, j)$. The pixel value after adding Gaussian noise can be modelled as a random variable:

$$I_N(i, j) = I(i, j) + N(i, j).$$

Here, $N(i, j)$ represents the Gaussian noise added to the pixel (i, j).

Gaussian Distribution: The Gaussian noise $N(i, j)$ is often assumed to follow a Gaussian (normal) distribution with a mean of 0 and a standard deviation (σ) that controls the magnitude of the noise. The probability density function (PDF) of the Gaussian distribution is given by:

$$f(x) = (\frac{1}{(\sigma\sqrt{2}})) \exp \frac{-(x-\mu)^2}{(2\sigma^2)}.$$

Here, μ is the mean of the distribution (which is typically 0 for Gaussian noise), σ is the standard deviation, and e denotes the exponential function.

Applying Gaussian Noise: To add Gaussian noise to a face image, for each pixel (i, j), you generate a random value from the Gaussian distribution with mean 0 and standard deviation $sigma$, and then add this value to the original pixel value $I(i, j)$:

$$I_N(i, j) = I(i, j) + R(0, \sigma).$$

The random value from Gaussian distribution $(R(0, \sigma))$ represents a random sample from the Gaussian distribution with mean 0 and standard deviation σ.

Adding Gaussian noise introduces variations to the pixel values that mimic real-world noise. The magnitude of the noise is controlled by the standard deviation parameter σ. This process is used for various purposes, such as testing the robustness of algorithms or training machine learning models to be resistant to noisy input data.

Accuracy percentage with the corresponding component			
Model	**Unmasked face data**	**Masked face data**	**Mixed face data**
MLP	85%, 5	85%, 6	81%, 3
RF	90%, 4	87%, 7	81%, 3
SVM	92%, 4	86%, 2	82%, 2
DT	46%, 1	48%, 3	50%, 2
KNN	79%, 2	78%, 2	75%, 2
XBOOST	71%, 7	67%, 2	69%, 3
LR	84%, 5	84%, 7	78%, 6
LDA	84%, 36	82%, 26	79%, 16
NVB	80%, 21	78%, 13	78%, 10

Table 2.1: Table of the maximum accuracy with the different classification models on the three created datasets.

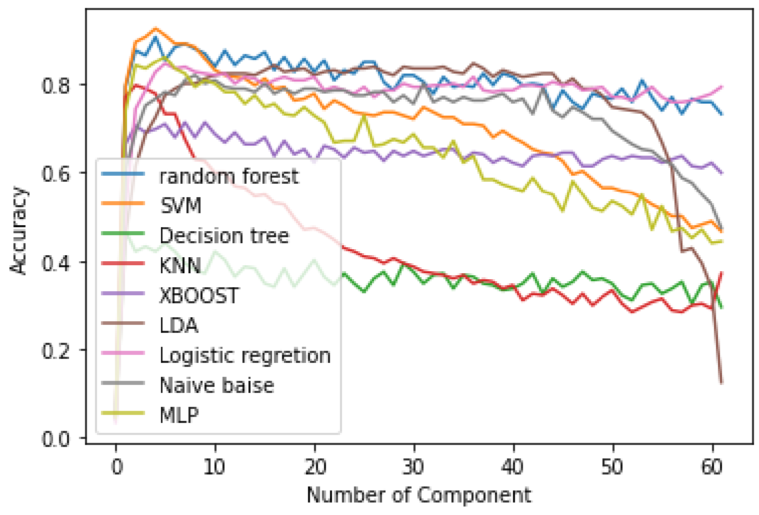
2.6 Conclusion

To consolidate our theory on the impact of face mask on the accuracy of face recognition, we used Google Colab to perform our Python implementation. We selected nine machine learning classifiers that are discussed in chapter 5, to determine the accuracy of each model on the three different datasets that we generated. Before implementing the classifiers for face recognition, we first apply PCA for the dimension reduction on the data.This guide us to see how the variation of the number of components impact on the accuracy of the classification. Thi can be seen in figures 2.5, where one can actually observe that depending of the number of component, the accuracy change.

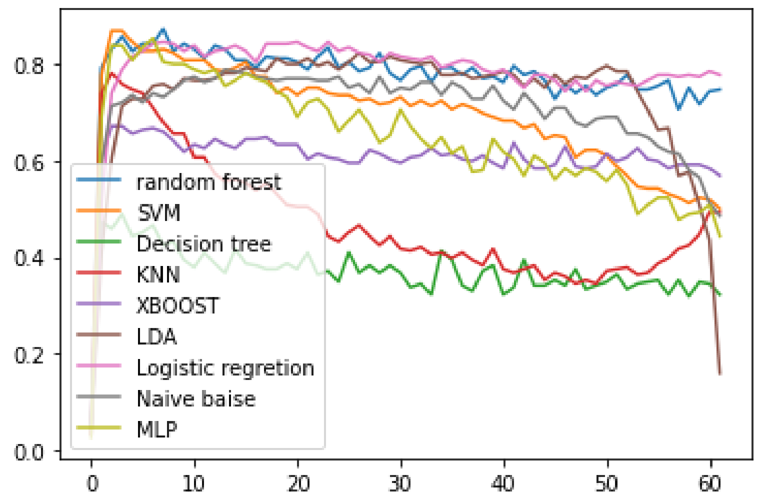
The presented plots offer in-depth insights into the performance dynamics of various machine learning classifiers across our three datasets. Notably, the observed trends in accuracy unveil a discernible decrement when transitioning from the dataset exclusively featuring individuals without masks to the dataset encompassing faces adorned with masks. This decrement underscores the challenge posed by facial masks in accurate facial recognition tasks. Intriguingly, the dataset amalgamating both masked and unmasked images consistently exhibits the lowest overall accuracy among the three datasets. This suggests that the inclusion of masked faces introduces complexities that impact model accuracy negatively.

A noteworthy aspect illuminated by the visualizations is the accuracy variation across principal components (PCs) obtained through PCA dimension reduction. The accuracy experiences an initial rapid ascent, achieving its zenith after a few principal components, followed by a gradual decline. This observed pattern aligns with the intended function of PCA to distil and retain the most influential components for enhancing model performance. The significance of this trend lies in its implication for model training; it emphasizes the criticality of capturing the right components for effective facial recognition.

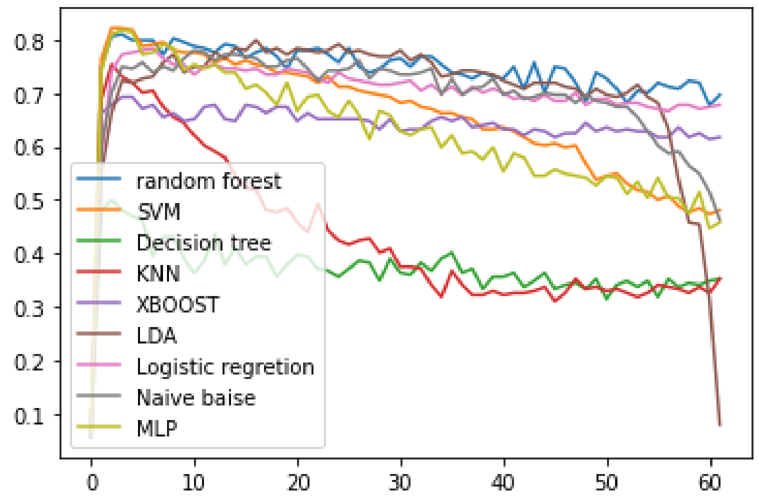
These findings collectively underscore the substantial impact of facial masks on recognition accuracy and emphasize the intricate relationship between dataset composition, PCA dimension reduction, and classifier performance. This nuanced understanding is vital for designing robust facial recognition models, particularly in the context of real-world scenarios where masked faces are prevalent.



(a) Unmasked faces data



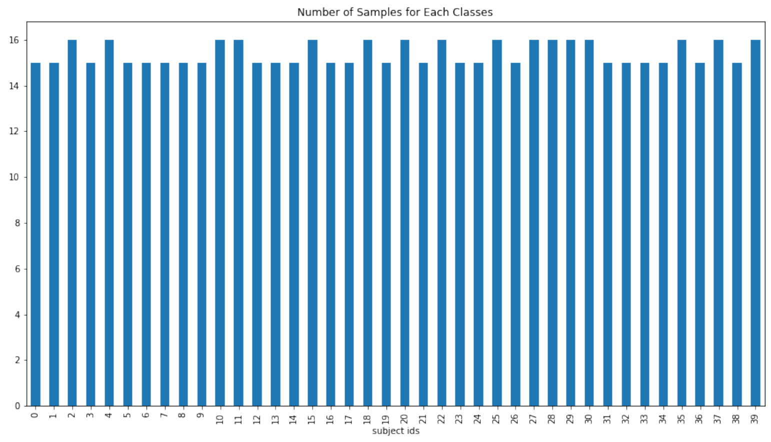
(b) Masked facees data data



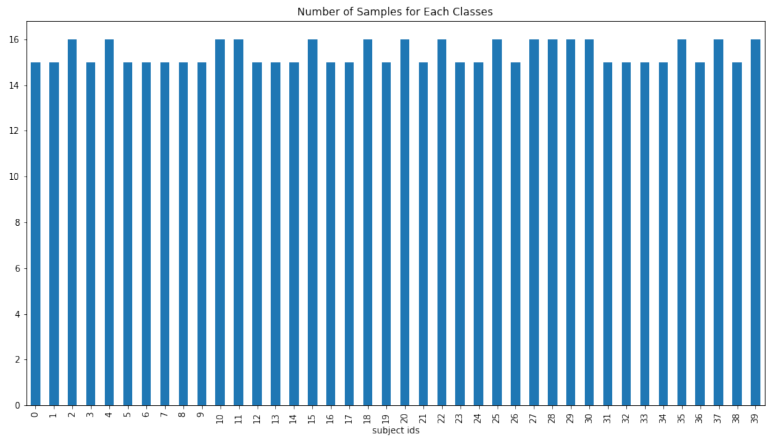
(c) Mixed faces

Figure 2.4: Plot of the accuracy variation with the number of components

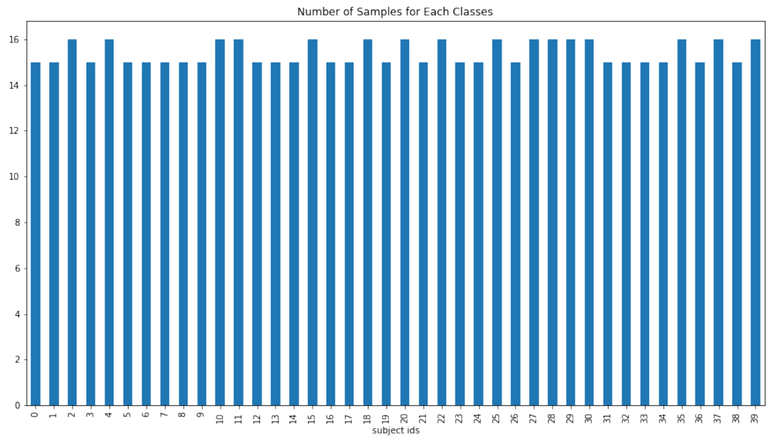
Table 1 presents the evaluation results of nine machine learning models in the context of facial recog-



(a) Unmasked faces data



(b) Masked facees data data



(c) Mixed faces

Figure 2.5: Class balance. Showing all the 40 subjects with their sample number for each, which are unequal

nition under different datasets without masks, with masks, and a combination of both. The highest accuracy achieved by each model and the corresponding optimal number of principal components, obtained through PCA dimension reduction, are detailed.

In Table 1, the Support Vector Machine (SVM) model demonstrates superior performance on unmasked data, achieving an accuracy of 92% with only 4 principal components. On the other hand,

Random Forest (RF) outperforms other models on masked face data, achieving an 87% accuracy with 7 principal components. Interestingly, SVM slightly outperforms RF on mixed face data. Consequently, we recommend SVM and RF as suitable models for our datasets. Its worth noting that Decision Tree (DT) exhibits the lowest performance among the models. Although it requires very few components to reach its maximum accuracy, its overall performance is not deemed suitable for our dataset.
The presented accuracy values and optimal components provide valuable insights into the performance of each model under different conditions, shedding light on the impact of face masks on facial recognition accuracy.

Hence the interests that we paid on looking to the impact for face mask on facial recognition. This problem can find improvement by using some dimension reduction techniques to gain the most important component of the faces, in figures 2.5, one can see how the accuracy of the model varied with the number of component using PCA for faces without mask, faces with mask and faces with both mask and without mask. Hence, we discuss Dimensional Reduction in the next chapter.

Chapter 3

Dimension Reduction

3.1 introduction

There is no machine learning implementation without data; it forms the bedrock upon which artificial intelligence (AI) and machine learning (ML) algorithms thrive. Data serves as the fuel that powers the learning and decision-making capabilities of these systems. Given its paramount significance, it is frequently asserted that data is the most critical component in the realm of AI and ML. Consequently, a profound understanding of what constitutes data and its inherent structure becomes imperative for any machine learning practitioner or researcher.
Facial recognition technology has evolved significantly in recent years, playing a vital role in various domains such as security systems, access control, and human-computer interaction. The accuracy and efficiency of facial recognition algorithms depend heavily on the representation of facial features in a lower-dimensional space. Traditional dimensionality reduction methods, such as Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA), have been widely employed for this purpose. However, they may have limitations in capturing complex and non-linear relationships present in facial data.

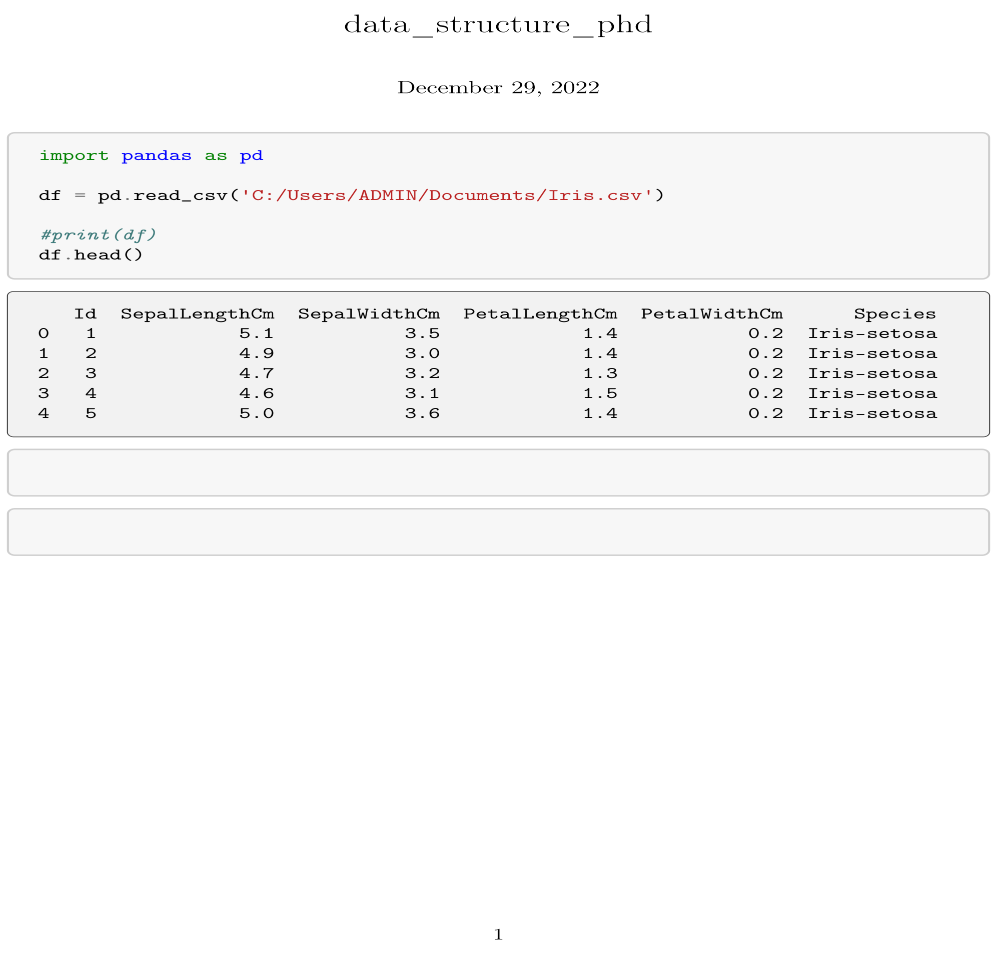
Definition 2 *Data refers to raw facts, observations, or measurements that are typically numerical or categorical. In the context of machine learning, data is the foundation on which algorithms are trained and patterns are learned.*

Definition 3 *A dataset is a collection of data, often organized in tabular form, where rows represent individual examples (instances) and columns represent features or attributes. Datasets are used to train, validate, and test machine learning models.*

Definition 4 *A feature is an individual measurable property or characteristic of a phenomenon being observed. In the context of machine learning, features are the variables used to make predictions or discover patterns.*

Definition 5 *In supervised learning, the label or target is the variable that the model aims to predict.*

When dealing with datasets, a feature is equivalent to a column. The number of columns in a dataset determines the number of features it has. Figure 3.1 likely contains code outputting the dimensions of the Iris dataset, a popular choice for classification in machine learning. The dataset information is sourced from [25].



data_structure_phd

December 29, 2022

```
import pandas as pd

df = pd.read_csv('C:/Users/ADMIN/Documents/Iris.csv')

#print(df)
df.head()
```

```
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
0   1            5.1           3.5            1.4           0.2  Iris-setosa
1   2            4.9           3.0            1.4           0.2  Iris-setosa
2   3            4.7           3.2            1.3           0.2  Iris-setosa
3   4            4.6           3.1            1.5           0.2  Iris-setosa
4   5            5.0           3.6            1.4           0.2  Iris-setosa
```

1

Figure 3.1:
Head of a data showing its data dimension

One may then be very concerned about the quality of the feature that its data has, since it can be seen as its building block. It is then obvious that we need to know the number of features (dimension of the data); this awareness is essential as it directly influences the complexity, interpretability, and computational efficiency of any analytical or modelling process. The sheer volume of features can pose challenges, leading to what is commonly known as the "curse of dimensionality," where the increased number of features can lead to sparsity, increased computational requirements, and a heightened risk of overfitting. Hence the need to manage and optimize the dimensionality of the data. therefore, the concept of dimension reduction comes into play. This is where the below definition becomes instrumental

Definition 6 *The term 'dimension reduction' refers to the systematic process of reducing the number of features or variables in a dataset while preserving its essential information. This technique is indispensable in scenarios where the original feature space is overly large, as it aims to mitigate the* *adverse effects associated with high dimensionality.*

When performing dimension reduction, the machine learning model selects the most important components of the feature space, preserves them, and drops the other non-important components feature. From what preceded, one has that, dimension reduction is very crucial for some machine learning models. We deem important to elaborate further on the significance of dimension reduction in the next section.

3.2 Significance

Dimension reduction holds significant importance for both machine learning practitioners and researchers in the field of data science. It serves as a cornerstone in tackling various challenges within these domains. Let's delve into a brief overview of some of these critical issues.

In today's era of big data, the sheer volume of datasets continues to grow exponentially, necessitating extensive storage resources. Dealing with high-dimensional datasets poses a considerable challenge due to the increased storage requirements. One viable solution to mitigate this challenge involves employing dimension reduction techniques before storing the data or implementing models.
In practical scenarios, it is advisable to visually and explore the structure of your data before determining the most appropriate model. Yet, visualizing data becomes increasingly challenging as the dimensionality exceeds three. Hence, the application of dimension reduction proves invaluable for facilitating effective data visualization.
In domains like image recognition, astronomy, video surveillance, medical imaging, and multimedia, datasets are often extensive. Data scientists and machine learning practitioners frequently encounter the need to compress these images while preserving vital information. In such scenarios, dimension reduction proves indispensable for achieving superior image compression without sacrificing critical details.
In the realm of machine learning, encountering a scenario where a model excels on training data but falters on testing data is termed overfitting. Dimension reduction emerges as a viable strategy to mitigate this issue effectively.
Given the importance of time efficiency in programming, research has demonstrated that employing dimension reduction techniques can significantly reduce the compilation time of specific code segments.

In the year 1957, R. Bellman in his book title Dynamic Programming [10] first introduced the term curse of dimensionality. This terminology nowadays in machine learning refer to the increasing of the error in a model when the features of the data is high. One can logically see that dimension reduction will be a solution for this type of problem.
Certainly, dimension reduction can also be used to eliminate redundant features and reduce noise in data, enhancing the quality of the data and improving the performance of machine learning models.

In machine learning, one may get across the situation where a model gives very good accuracy on training data but gives bad accuracy on testing data. This situation is known as over fitting. In practice, one can solve this over fitting problem with dimension reduction.

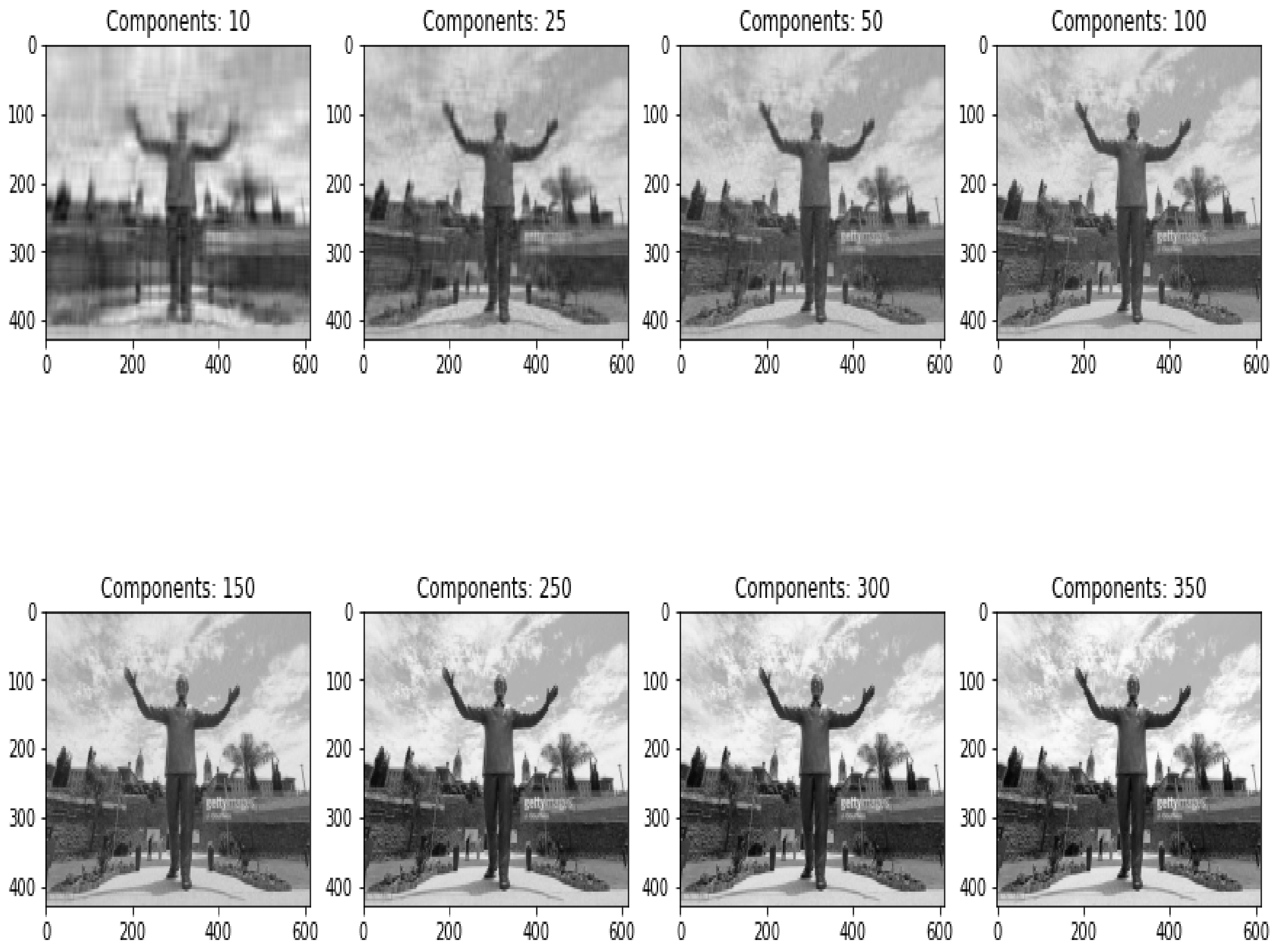


Figure 3.2:
Sample of data compresion showing same image with different componet after PCA.

Dimension reduction serves multiple purposes, including the removal of redundant features and noise from datasets.

Having discussed some of the importance of dimension reduction, it is equivalently interesting to understand some applications of this machine learning concept.

3.3 application

The practical applications of dimension reduction are evident across various problem domains. In the following sections, we delve into some notable examples.

I In their work [67], Narendra Singh and colleagues elucidate the advantages of employing PCA for dimensionality reduction in Customer Relationship Management (CRM).

II One of the significant steps in text classification, is dimension reduction. This is explained in section 2.1 of [66] which is introduced by: High dimensionality is one of the challenging problems for text data classification. In the above article, Sngh et al explain how useful dimension reduction

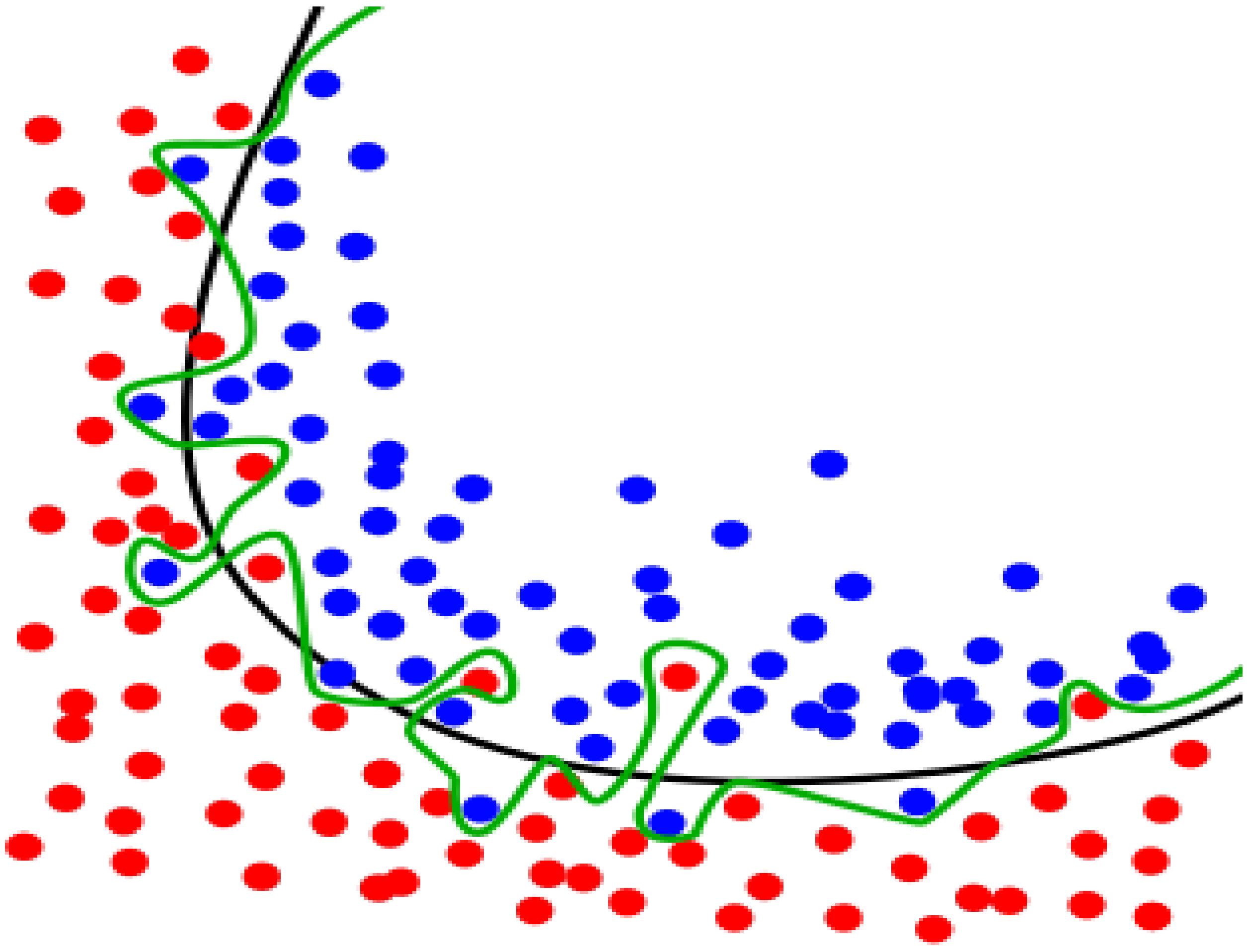


Figure 3.3:
Sample of over fitting data

is for this common real word problem.

III To gain insights into the utilization of dimension reduction in image and video retrieval, Wu et al. [79] discuss its significance in their work. The abstract begins as follows: 'Dimensionality reduction methods are of interest in applications such as content based image and video retrieval.'

IV Dimension reduction can be applied in instruction detection. This can be seen in [20] where the authors clearly state that: 'Dimensionality reduction is crucial when data mining techniques are applied for intrusion detection.'

V As mentioned in the previous section, medical imaging encounters the challenge of high dimensionality. For a good diagnostic to be done when using image by medical practitioner, they often need the image to be segmented. Hence to have a good interpretation this image segmentation needs to go through some dimension reduction process before being used.

Now that we have gained a better understanding of dimension reduction and its applications, it is intriguing to explore the techniques used to achieve the aforementioned results. Moreover, dimension reduction can enhance the performance of various machine learning models.

In the remainder of this article, we will utilize dimension reduction techniques such as PCA, LDA, and Autoencoder to assess the accuracy of facial recognition models. Additionally, we will introduce our innovative hybrid techniques aimed at further enhancing the accuracy of facial recognition models.

3.4 Dimension Reduction Techniques

Dimension reduction techniques are commonly categorized into two primary groups:

1. Feature Selection: This approach involves selecting the most relevant features from the original dataset while discarding the less important ones. By retaining only the most informative features, feature selection aims to simplify the model without altering the underlying feature space drastically. This process helps in reducing computational complexity and mitigating the risk of overfitting. Various methods such as filter methods, wrapper methods, and embedded methods are employed to evaluate and select features based on their individual importance and contribution to the predictive model.

2. Feature Extraction: Unlike feature selection, feature extraction involves transforming the original features into a new set of features through mathematical transformations. These transformations aim to capture the essential information present in the original features while discarding redundant or irrelevant information. Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and t-Distributed Stochastic Neighbor Embedding (t-SNE) are some common techniques used for feature extraction. By creating a compact representation of the data, feature extraction not only reduces dimensionality but also facilitates visualization and interpretation of complex datasets.

Both feature selection and feature extraction are crucial components of dimensionality reduction techniques, each offering unique advantages depending on the nature of the dataset and the objectives of the analysis. The choice between these methods often depends on factors such as computational resources, interpretability of the transformed features, and the desired performance of the predictive model. Figure 3.4 gives some of the dimension reduction techniques.
In what follows, we delve into the mathematical theory behind dimensional reduction techniques, focusing specifically on three established methods: PCA, LDA, and Autoencoder. We provide detailed explanations of how these techniques function, including the underlying mathematical principles. Additionally, we supplement our explanations with graphs generated using the Python programming language to illustrate the practical application of these techniques.

Towards the end of the section, we introduce our innovative techniques, PCA-Autoencoder and LDA-Autoencoder. Through our implementation in Python for face recognition after dimension reduction, we demonstrate that our novel approaches surpass the performance of the three traditional methods.

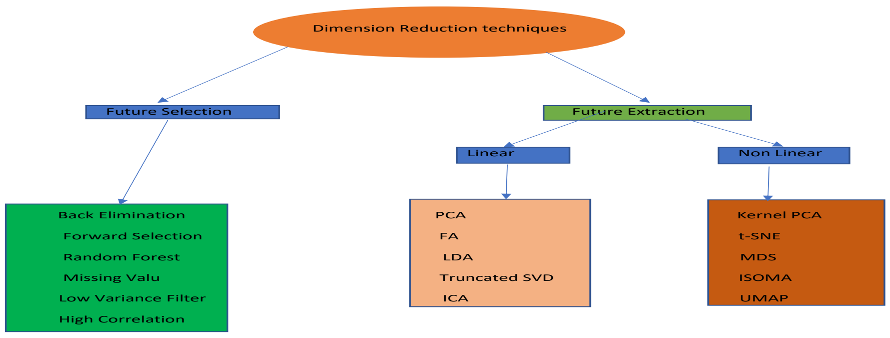


Figure 3.4: Table of Different Dimension Techniques.

3.4.1 PCA

Principal Component Analysis (PCA) stands out as one of the prominent machine learning algorithms employed for dimension reduction. This technique delves into statistical insights within the data and utilizes an orthogonal transformation to convert a set of possibly correlated variables into a collection of linearly uncorrelated variables known as principal components or principal modes of variation.

To achieve this, one can proceed with eigenvalue decomposition of a data covariance (or correlation) matrix or Singular Value Decomposition (SVD) of the data matrix. PCA is valuable for identifying patterns in the data, highlighting similarities and differences, and compressing data by reducing dimensions while retaining essential information.

One widely recognized PCA application is Eigenfaces, developed by Sirovich and Kirby in 1987 [49] and later employed by Turk and Pentland for face detection and recognition [55]. A detailed development on Eigenfaces can be found in [55].

PCA has found applications in various fields such as dimensionality reduction [70], face and gesture recognition [14], motion analysis and synthesis [64], clustering [55], and dimensionality reduction in [40].

3.4.1.1 PCA for Dimension Reduction

PCA, a fundamental mathematical technique, is widely used by data scientists and machine learning practitioners for reducing data dimensions, facilitating visualization [7, 52], noise removal [55], and enhancing processing time and accuracy. The following outlines the mathematical theory behind this machine learning application.

Let's explore the Eigenface process for dimension reduction in facial images. Assuming an image represented by a matrix M of size $n \times m$, the process unfolds as follows:

I **Matrix of Observations:** Given a dataset of P images of size $x \times y$ denoted as $I_1, I_2, \cdots, I_P$, reshape each image into a $1 \times xy$ row vector, constructing the matrix of observations X of size $M \times N$.

II **Covariance Matrix:** Normalize the matrix of observations, $\overline{X}$, by mean normalization and calculate the covariance matrix Z as:

$$Z = \frac{1}{M-1} \overline{X}^T \overline{X}.$$

III **Principal Component:** Obtain principal components by computing eigenvalues λ_i of Z and the corresponding eigenvectors v_i. Each component v_i is reshaped into an eigenface V_i of size $x \times y$.

IV **Dimensionality Reduction:** Approximate a given matrix A using the first k eigenfaces to achieve dimensionality reduction:

$$A \approx \sum_{i=1}^{N-k} \alpha_i V_i.$$

3.4.1.2 PCA for Face Recognition

For face recognition, given an input image vector $U \in \Re^n$ and the mean image vector M from the database, calculate the weight of the k^{th} eigenface as:

$$w_k = V_k^T (U - M),$$

forming a weight vector $W = [w_1, w_2, \cdots, w_k, \cdots, w_n]$. Compare W with weight vectors W_m of images in the database, finding the Euclidean distance $d = ||W - W_m||^2$. If $d < \epsilon$, then the m^{th} entry in the database is the corresponding one; otherwise, U may be an unknown face.

3.4.1.3 Advantages of PCA

PCA offers several advantages:

I **Simplicity:**
PCA is conceptually and implementable straightforward, making it accessible for various tasks. Its intuitive nature allows for easy understanding and application in diverse fields.

II **Data Compression:**
By reducing data dimensionality while retaining most of the variance present in the original dataset, PCA facilitates data compression. This reduction in dimensionality leads to decreased storage requirements and accelerates the processing time of models, making it particularly useful for handling large datasets efficiently.

III **Uncorrelated Features:**
Principal components extracted through PCA are orthogonal to each other. This orthogonality ensures that the new set of features is uncorrelated, simplifying subsequent analyses and enhancing the stability and robustness of the model.

IV **Interpretability:**
PCA provides a clear interpretation of the data by showcasing the original combination of features

in terms of their contributions to the principal components. This interpretability enhances the understanding of the underlying structure of the data and aids in making informed decisions during the model-building process.

These advantages collectively make PCA a powerful and versatile tool for dimensionality reduction, enabling efficient data analysis, visualization, and modelling across various domains.

3.4.1.4 Disadvantages of PCA

While PCA offers advantages, it has limitations:

I **Linearity:**
PCA operates under the assumption of linearity, which means it may not effectively capture complex nonlinear relationships present in the data. As a result, when the underlying relationships between variables are nonlinear, PCA may not provide an optimal representation of the data, potentially leading to information loss or distorted representations.

II **Global Structure:**
One of the limitations of PCA is its inability to preserve the global structure or class separability of the data, particularly in classification problems. While PCA focuses on maximizing variance along principal components, it does not explicitly consider class labels or the global structure of the data. Consequently, important discriminative information may be lost during dimensionality reduction, potentially affecting the performance of classification algorithms.

Figure 3.5 and 3.6 show the scatter plot and projection respectively of Olivetti face data after performing PCA for dimension reduction. One can see how the data are grouped from the scatter plot which also reveal some outlier. Similarly the projection graph help us visualize the most significant patterns and relationships, it accentuate the difference between groups.
These limitations highlight the importance of considering the characteristics of the data and the specific goals of the analysis when choosing dimensionality reduction techniques. In scenarios where the data exhibits nonlinear relationships or when preserving class separability is crucial, alternative methods such as nonlinear dimensionality reduction techniques or supervised dimensionality reduction approaches may be more appropriate. One such technique may be Linear Discriminant Analysis(LDA).

3.4.2 Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) is a machine learning algorithm commonly employed for multiclass data classification problems. It can be likened to logistic regression, a popular algorithm for single-class classification. For a detailed comparison between these algorithms, refer to [15] and [18]. LDA is also widely used for visualization and serves as a popular supervised machine learning algorithm for dimension reduction. For more information on different supervised learning algorithms for dimension reduction, consult [16, 46]. LDA techniques for dimension reduction find applications in Biometrics [63, 56], Bioinformatics [28], and Chemistry [69, 57]. This section focuses specifically on LDA for dimension reduction, delving into its mathematical foundations.

LDA for dimensionality reduction is based on the projection of the original data matrix onto a lower-dimensional space. The process involves answering key questions:
What are the distances between the means of different classes (between-class variance or between-class

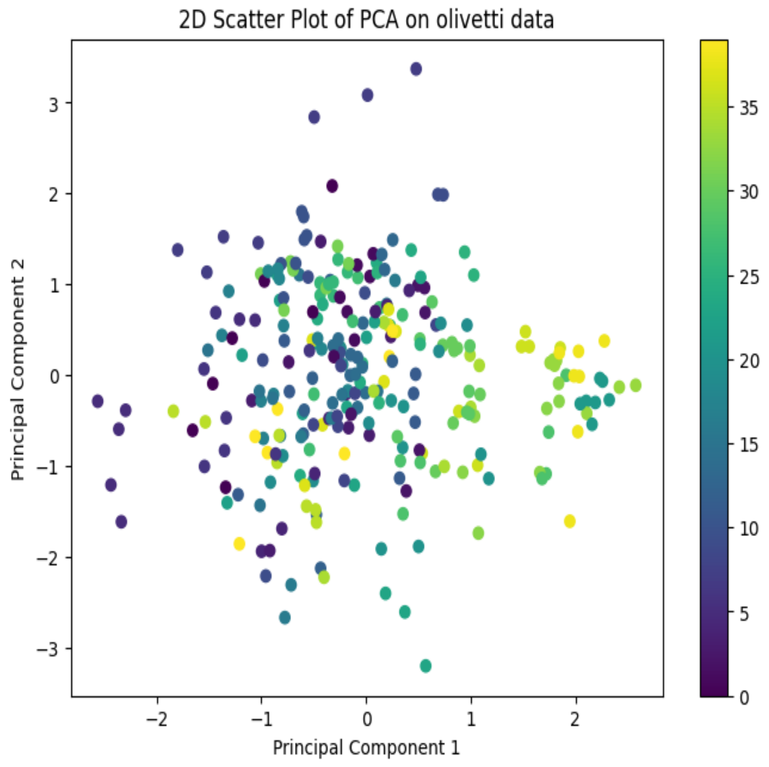


Figure 3.5:
Scatter plot of the ollivetti dataset after PCA dimension reduction

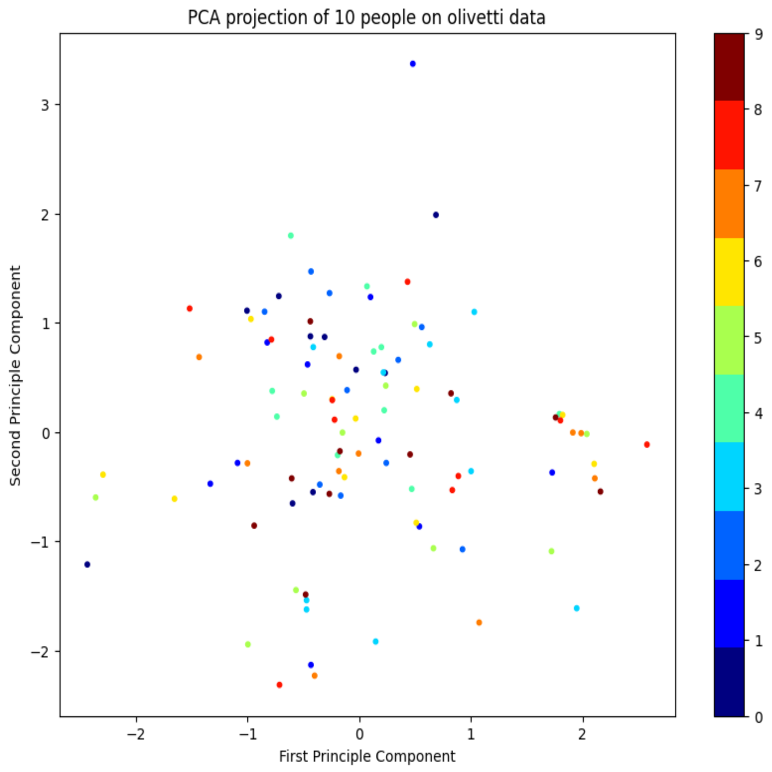


Figure 3.6:
The PCA projection of 10 subject on ollivetti dataset

matrix)?
What are the distances between the mean of the data matrix and the samples of the matrix (within-class variance or within-class matrix)?
These concepts guide the construction of the lower-dimensional space by maximizing the between-class variance and minimizing the within-class variance.

3.4.2.1 Between-Class Variance

The between-class variance of the i-th class (A_{B_i}) is the distance between the mean of the i-th class (μ_i) and the overall mean (μ). Given the original data matrix $X = (x_1, x_2, \cdots, x_N)$, where x_i represents the i-th observation or sample, each with M features, and N is the total number of observations, we project our data matrix on a $1 \times M$ dimension to obtain W. The representation of all projections for each class mean can be calculated as $m_i = W^T \mu_i$. The total mean projection is calculated as $m = W^T \mu$. Considering p classes, each with n_j features, we calculate the between-class variance A_{B_i} as $W(m_i - m)^2 W^T$, and the total between-class variance is given by $A_B = \sum_{i=1}^{p} n_i A_{B_i}$.

3.4.2.2 Within-Class Variance

The within-class variance of the i-th class (A_{W_i}) is the difference between the mean and the samples of that class.
we proceed to calculate the centring projection data by computing: $d_j = W_j \mu_j$. Thus the within-class variance can be computed as follow:

$$A_{W_j} = \sum_{i=1}^{n_j} (x_{ij} - \mu_j)(x_{ij} - \mu_j)^T.$$

Now, let us use the projection for any sample and the mean for any class; we have the following equality:

$$\begin{aligned}
\sum_{x_i \in W_j j=1 \ldots p} (W^T x_i - m_j)^T &= \sum_{x_i \in W_j j=1 \cdots p} (W^T x_{ij} - W^T \mu_j) && (3.1)\\
&= \sum_{x_i \in W_j j=1 \cdots p} W^T (x_{ij} - \mu_J)^2 W && (3.2)\\
&= \sum_{x_i \in W_j j=1 \cdots p} W^T (x_{ij} - \mu_j)(x_{ij} - \mu_j) && (3.3)\\
&= \sum_{x_i \in W_j j=1 \cdots p} W^T A_{W_j} W. && (3.4)
\end{aligned}$$

Hence the total within-class variance is given by

$$A_W = \sum_{i=1}^{p} S_{W_i}.$$

3.4.2.3 Lower Dimensional Space

Having computed the between-class variance A_B and the within-class variance A_W, the optimization problem becomes $\arg\max_W \frac{W^T A_B W}{W^T A_W}$. The maximizer is given by the largest eigenvector of A

$$\arg\max_{W} \frac{W^T A_B W}{W^T A_W}.$$

Theorem 4 *Suppose A_W is nonsingular. The maximizer of this problem is given by the largest eigenvector of A_W^{-1}, i.e.,*
$A_W^{-1} A_B W = \lambda W$ Or $A_W W = \lambda A_B W$.

One then has to compute the eigenvalues $(\lambda_1, \lambda_2, \cdots, \lambda_M)$ and the eigenvectors $(V = \{V_1, V_2, \cdots, V_M\})$ of $W = A_W^{-1} A_B$.
Since the eigenvalues are scalar, and the eigenvectors are non-zero vectors, our approximation can be resolved. We have that the eigenvectors represent the directions of the new space, and the eigenvalues are the scaling factors, lengths, or magnitudes of the eigenvectors. Therefore, the axes of the LDA are represented by each eigenvector and are associated with its eigenvalue. Thus, the higher the eigenvalue, the better it can discriminate between different classes, i.e., increase the between-class variance and decrease the within-class variance of each class; hence meeting the LDA requirement.
One then uses the k highest eigenvalues with their corresponding eigenvectors to construct the lower-dimensional space. Let's call this matrix L_k, given by $L_k = \{V_1, V_2, \cdots, V_k\}$, and its dimension is $M \times k$. Since the original data matrix is of dimension $N \times M$, we then obtain our reduced data matrix by computing $Y = XV_k$.
This concludes the dimension reduction process using Linear Discriminant Analysis.

3.4.2.4 Advantages of LDA

LDA offers several advantages:

I **Supervised Dimension Reduction:**
LDA is particularly well-suited for classification problems because it is a supervised technique that takes class labels into account during dimensionality reduction. By leveraging class information, LDA aims to find a projection that maximizes the separation between classes, thereby enhancing the discriminative power of the reduced feature space.

II **Maximizes Class Separation:**
A primary objective of LDA is to maximize the separation between classes. Unlike PCA, which focuses solely on maximizing variance, LDA explicitly considers class information to find a projection that optimally separates different classes in the data. This makes LDA particularly effective for feature extraction and classification tasks, where maximizing class separability is crucial for model performance.

III **Interpretability:**
The impact of maximizing class separability in LDA results in components that are easily interpretable. Each linear discriminant derived from LDA represents a direction in the feature space that maximizes class discrimination. As a result, the components obtained from LDA provide insights into the underlying structure of the data and the discriminative features that contribute to class separation, enhancing interpretability and aiding in model understanding.

These advantages make LDA a valuable tool for tasks such as classification, pattern recognition, and feature extraction, particularly when class discrimination and interpretability are essential considerations.

3.4.2.5 Disadvantages of LDA

While LDA (Linear Discriminant Analysis) offers several advantages, it also comes with its own set of disadvantages:

I **Assumes Gaussian Distributions:**
One of the primary assumptions of LDA is that the data within each class follows a Gaussian distribution. However, in real-world scenarios, this assumption may not always hold true. When the data deviates significantly from Gaussian distributions in one or more classes, LDA's performance can be compromised. In such cases, the decision boundaries derived by LDA may not accurately represent the underlying structure of the data, leading to suboptimal classification results.

II **Sensitivity to Outliers:**
LDA is highly sensitive to outliers and noise in the dataset, posing significant challenges to its robustness and performance. Outliers can disrupt the estimation of class means and covariances, which are crucial parameters in LDA's decision rule. This sensitivity stems from LDA's reliance on the assumption of multivariate normality within each class. Outliers can distort this assumption, leading to biased parameter estimates and suboptimal decision boundaries. Moreover, noisy data can introduce additional variance, potentially resulting in overfitting or poor generalization performance of LDA models.

Figure 3.7, 3.8 and 3. 9 represent the confusion matrix, the scatter plot and the class separation respectively of the olivetti face data after LDA dimension reduction. In the scatter plot, one see how the LDA help to group the data, The confusion matrix help to se that they are few outlier after the LDA dimension reduction.
These limitations highlight the importance of assessing the suitability of LDA for a given dataset, considering factors such as the distributional assumptions of the data and the presence of outliers or noise. In scenarios where these assumptions are violated or the data contains significant outliers, alternative techniques or preprocessing methods may be necessary to ensure robust and reliable analysis results.
Using a nonlinear method may assist one in overcoming the limitations of these two techniques in many situations where the dataset is not suitable for linearity. Hence we are now delving into a nonlinear technique, namely Auto-Encoder.

3.4.3 Auto-Encoder

In this section, we delve into a deep learning method for dimensionality reduction known as the Neural Network machine learning algorithm Auto-Encoder (AE). This unsupervised artificial neural network leverages both feature extraction and feature selection techniques to compress high-dimensional data into a lower dimension.

The introduction of Chapter 14 of [31] defines Auto-Encoder as follows: an Auto-Encoder is a neural network that is trained to attempt to copy its input to its output. Internally, it has a hidden layer h that describes a code used to represent the input. The network may be viewed as consisting of two parts: an encoder function $h = f(x)$ and a decoder that produces a reconstruction $r = g(h(x))$.
Before going into details on the Auto-encoder process himself, it is important to lay some groundwork with the mathematics description of Neural Network which is the back born of the method.

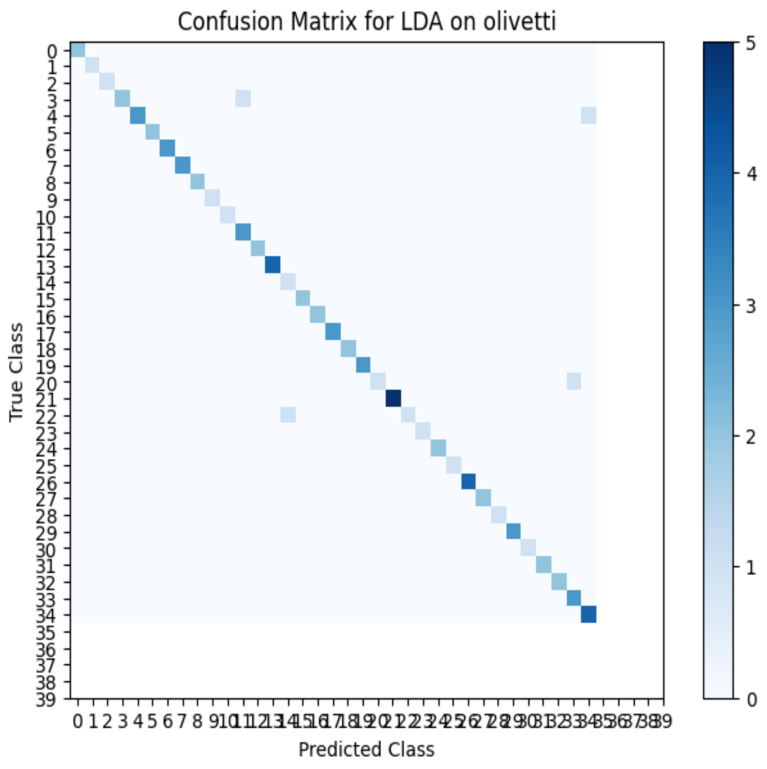


Figure 3.7:
Confusion matrix of ollivetti dataset after LDA dimension reduction

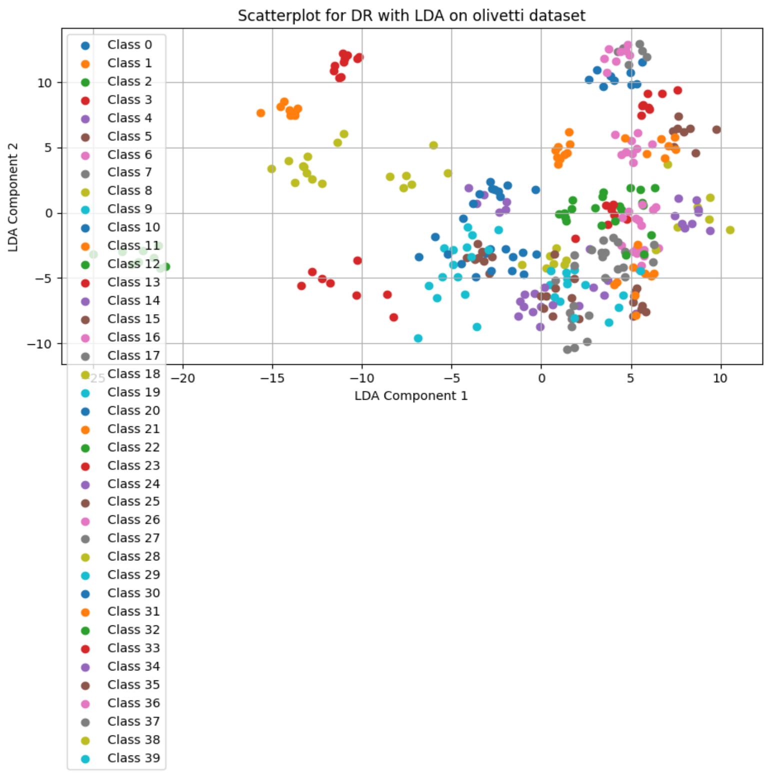


Figure 3.8:
Scatter plot of the ollivetti dataset after LDA dimension reduction

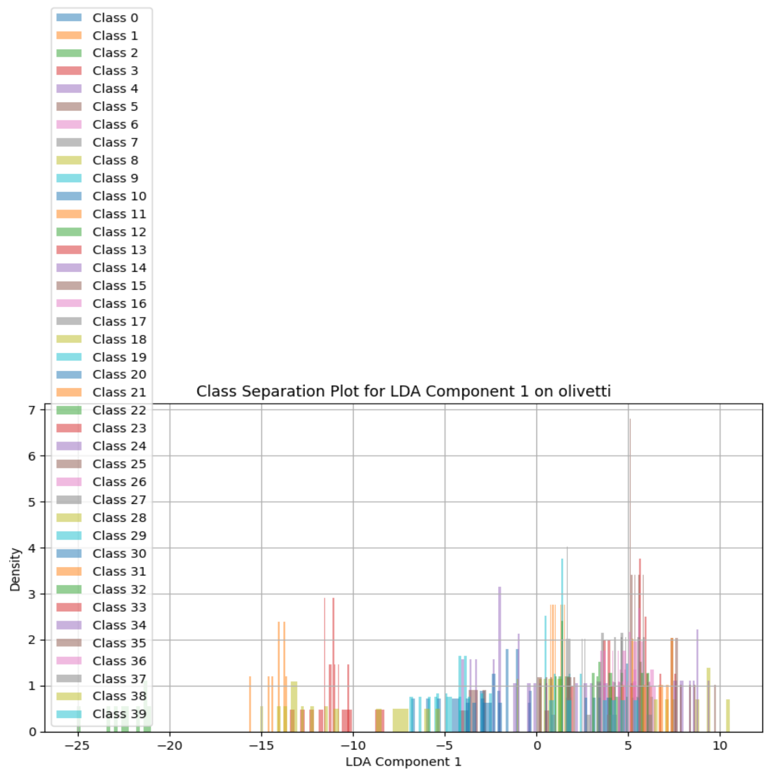


Figure 3.9:
LDA class separation on ollivetti

3.4.3.1 Neural Network

A neural network is a mathematical model derived from the architecture and functioning of the human brain, interconnecting artificial neurons organized into layers. Each layer has a specific role in processing and transforming input information, which, in machine learning, consists of initial data used to produce desired outputs.

Neural networks play a fundamental role in various machine learning tasks, including data compression, pattern recognition, regression, and classification.

Neural networks are a fundamental concept in machine learning and are widely been used for tasks like data compression[58, 74, 80], pattern recognition[2, 11], regression[8, 24], classification [27, 62, 78, 81].

Before delving further, let explore some basic concepts crucial for understanding neural networks:

I **Neuron or Node:** The basic processing unit that receives one or more inputs and computes a weighted sum to produce an output.

II **Layer:** A group of neurons working together at a specific depth of the neural network. Layers include the input layer, hidden layers (responsible for processing and transforming information), and the output layer.

III **Activation Function:** A function that decides if a neuron should be activated (fired), introducing non-linearity to the network.

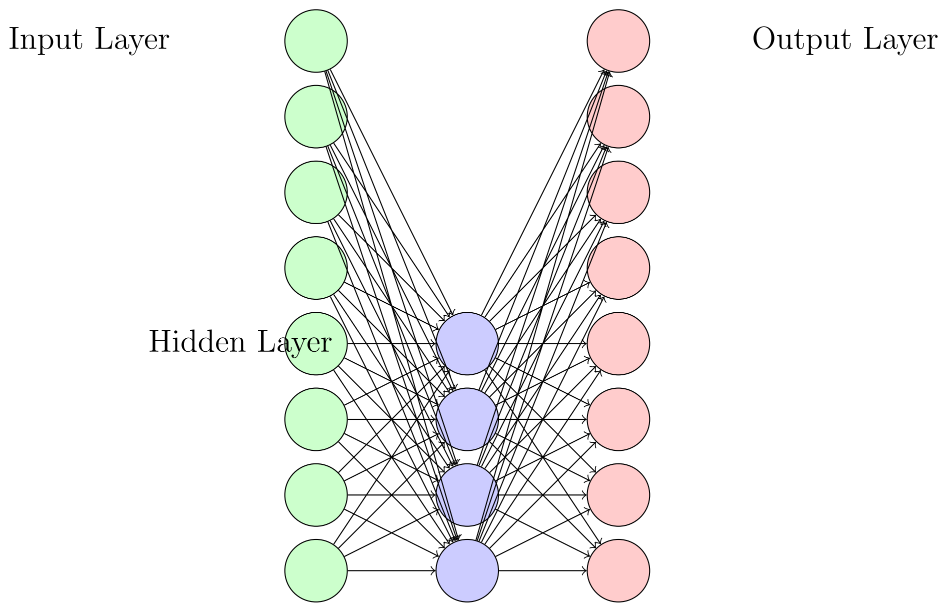


Figure 3.10: Sample of Neural Network Diagram

IV **Feedforward Propagation:** The process of moving information from the input layer to the output layer.

V **Backpropagation:** A process used to adjust weights and biases, minimizing the difference between predicted outputs and actual targets.

VI **Loss Function:** A function that measures the difference between predicted and actual outputs, used for optimization.

VII **Architecture:** The structure of the neural network, determined by the number of layers, neurons per layer, and interconnections.

3.4.3.2 NN Activation Functions

Understanding the activation functions used in neural networks is crucial. Here are three commonly used ones:

I **Sigmoid Function:** Converts input values to a range between 0 and 1, making it suitable for mapping arbitrary input values to a probability-like range.

$$f(z) = \frac{1}{1 + e^{-z}}.$$

An example of sigmoid function is defined in figure 3.11.

The sigmoid function is used primarily in the context of feedforward neural networks, where it is applied to the weighted sum of inputs and bias for each neuron. It transforms the output of each neuron to introduce non-linearity and allows the network to learn complex relationships in the data.

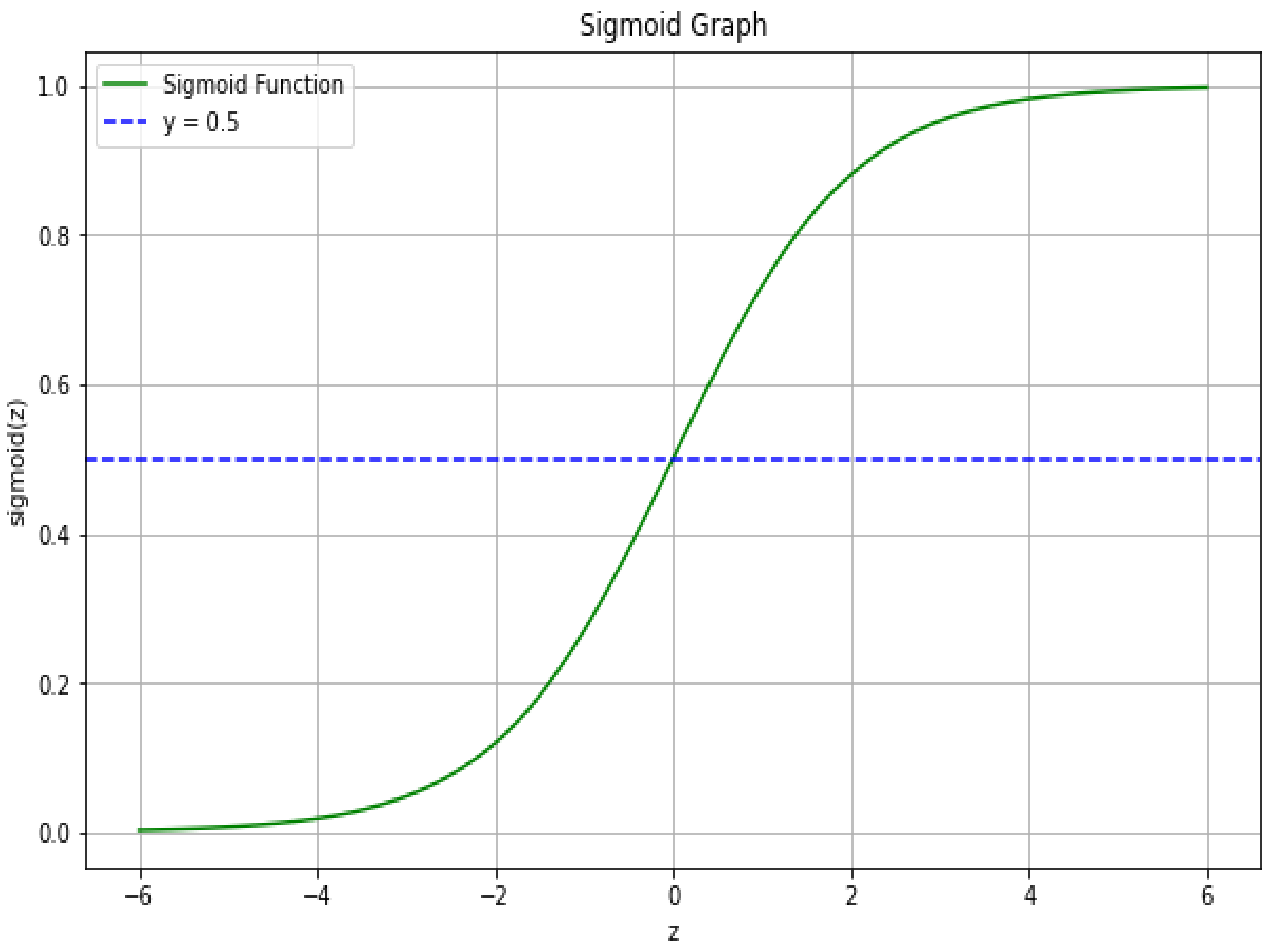


Figure 3.11:
Example of Sigmoid

However, one drawback of the sigmoid function is that its gradients can saturate (become very small) for very large or very small inputs, leading to slow learning during backpropagation.

II **ReLU (Rectified Linear Unit):** Introduces non-linearity and effectively mitigates the vanishing gradient problem. It outputs the input for positive values and 0 for negative values.

$$ReLU(z) = \max(0, z).$$

The ReLU function is computationally efficient and addresses the vanishing gradient problem, as it doesn't saturate for positive inputs. However, it has a limitation known as the "dying ReLU" problem, where neurons can become inactive (output zero) for all inputs during training, leading to those neurons not contributing to learning. This issue has led to the development of variations of ReLU, such as Leaky ReLU, Parametric ReLU (PReLU), and Exponential Linear Unit (ELU), which aim to address the dying ReLU problem while preserving the non-linear properties of ReLU.

III **Softmax Function:** Converts a vector of real-valued numbers into a probability distribution. It

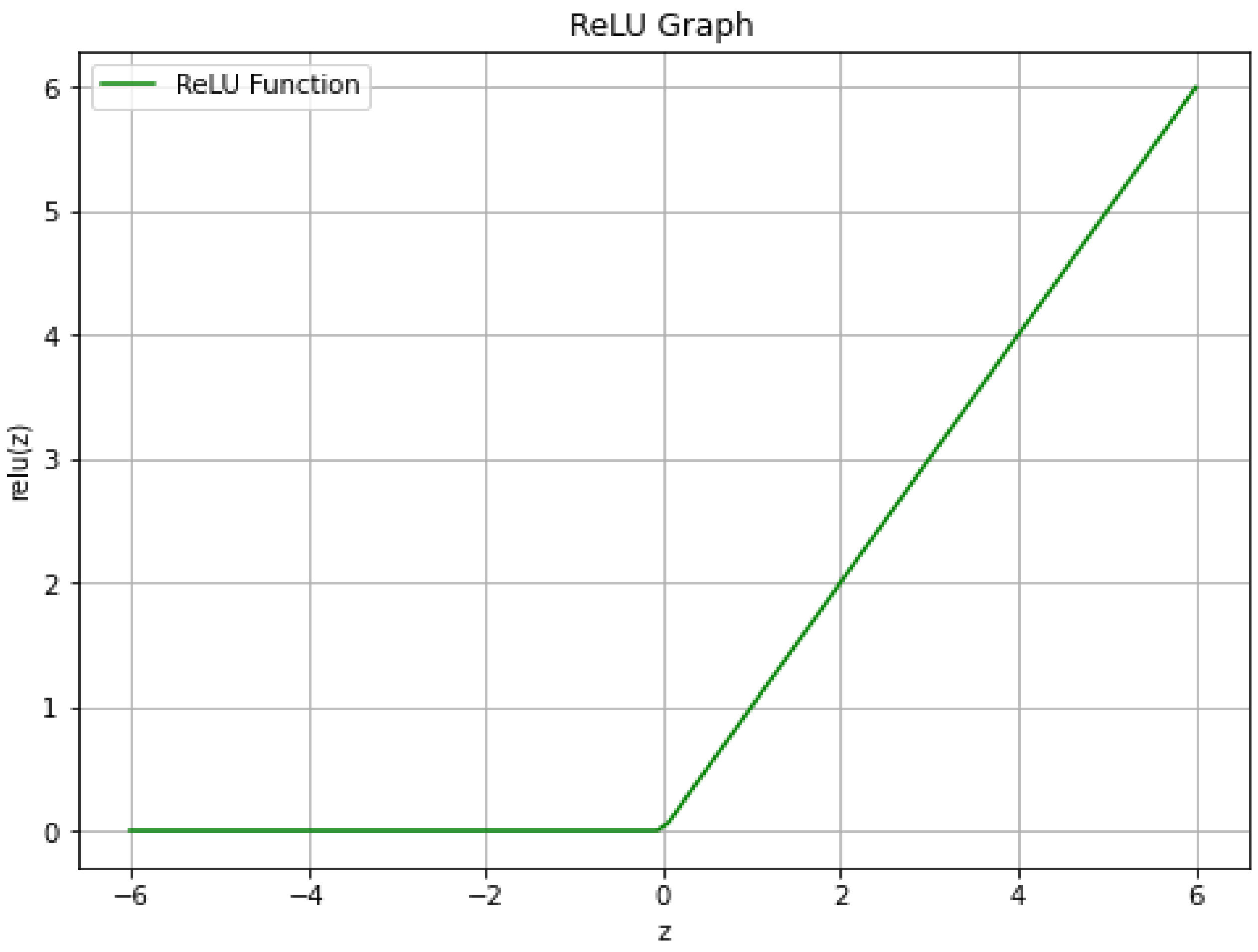


Figure 3.12:
Example of a ReLu Graph

is commonly used in multi-class classification tasks.

$$S(u) = \frac{e^{u_i}}{\sum_{j=1}^{n} e^{u_j}}.$$

The softmax function takes an input vector and transforms each element into a probability value between 0 and 1. The transformed values represent the probabilities of the input belonging to different classes, and they sum up to 1. It essentially ”softens” the input values to represent probabilities, allowing for better differentiation between classes.

The softmax function has several desirable properties:

It converts arbitrary real-valued scores into a valid probability distribution.
It emphasizes the largest values in the input while suppressing the smaller ones.
It is differentiable, making it suitable for backpropagation during gradient-based optimization.
One important use of the softmax function is in the output layer of neural networks for multi-class classification tasks. The activation of the softmax function in the output layer produces class probabilities, and the predicted class corresponds to the class with the highest probability.

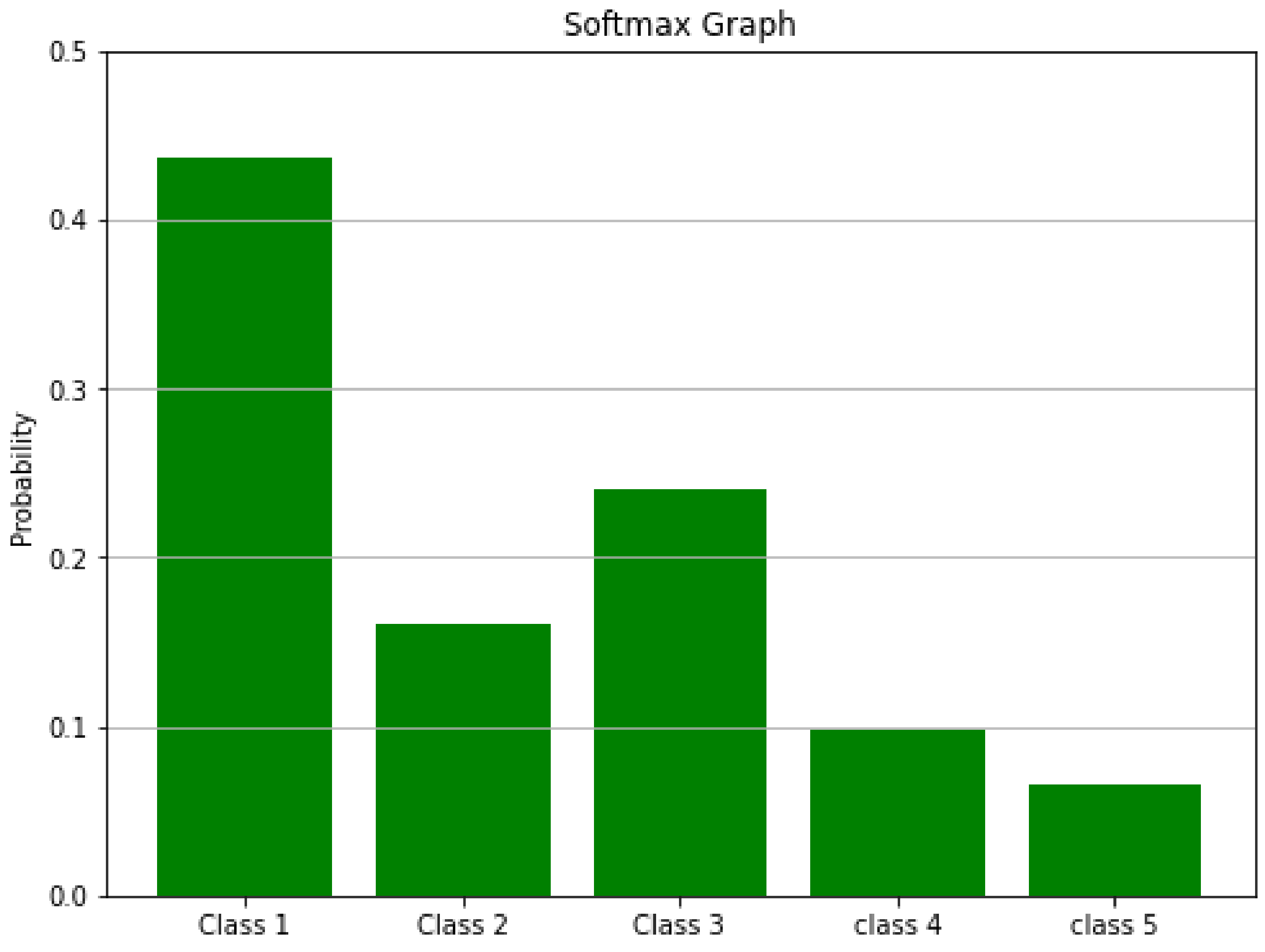


Figure 3.13:
Example of a Softmax Graph

However, it is important to note that the softmax function tends to amplify the differences between input values. In cases where there are significant differences in input values, the softmax output can be highly skewed towards the largest value. This can lead to numerical stability issues in cases of extreme values. Techniques like subtracting the maximum value from each input element (known as softmax normalization) can help address these issues.

3.4.3.3 Auto-Encoder for Dimension Reduction

Auto-Encoder, a deep learning technique derived from the mathematical structure of a neural network, is employed for dimensionality reduction. Comprising an Encoder and a Decoder, it transforms input data into an approximation of the original data.

I Encoder

The Encoder takes input data (e.g., face images) and compresses it. Mathematically, it is defined as:

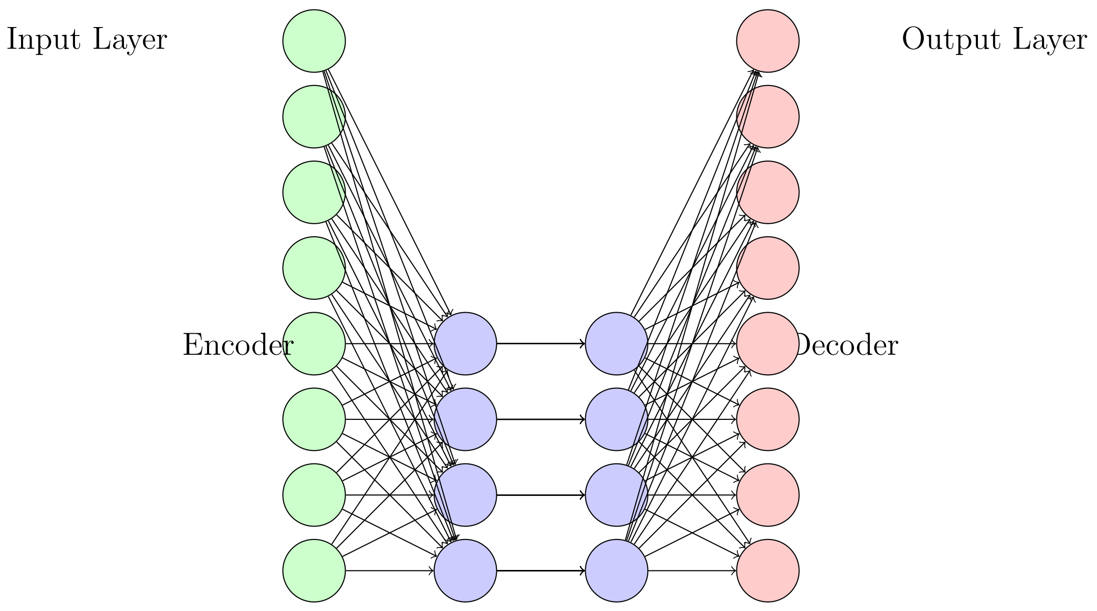


Figure 3.14: Auto Encoder Diagram

$$f(z) = \sigma(W_e z + b_e),$$

where W_e is the weight matrix, b_e is the bias vector, and σ is the activation function.

II Decoder

The Decoder reconstructs the input data from the compressed representation:

$$\hat{z} = \sigma(W_d \cdot h + b_d),$$

where W_d is the weight matrix, b_d is the bias vector, and $\hat{z}$ is the reconstructed input.

The Mean Square Error (MSE) is often used to quantify the discrepancy between the reconstructed and original data, expressed as:

$$L = \frac{1}{N} \sum_{i=1}^{N} ||z_i - \hat{z}_i||_2^2.$$

The loss function is minimized by adjusting weights and biases through backpropagation and optimization algorithms.

3.4.3.4 Advantages of Auto-Encoder

This method has some advantages that are enumerated here.

I **Nonlinear Transformations:**
Auto-Encoders excel at capturing nonlinear relationships present in complex data. Unlike linear methods such as PCA, which are limited to capturing linear correlations between features, Auto-Encoders leverage neural network architectures to learn nonlinear transformations of the input

data. This capability allows them to model intricate patterns and dependencies within the data, making them particularly effective for tasks involving complex, high-dimensional datasets.

II **Feature Learning:**
Auto-Encoders have the ability to automatically learn relevant features from the input data, thereby reducing the reliance on manual feature engineering. By iteratively encoding and decoding the input data through hidden layers, Auto-Encoders extract meaningful representations that capture the underlying structure of the data. This feature learning process is unsupervised, meaning it does not require labelled data, making Auto-Encoders versatile and applicable to a wide range of domains and tasks.

III **Versatility:**
Auto-Encoders are highly versatile and can be applied to various tasks beyond dimension reduction. In addition to dimensionality reduction, Auto-Encoders are commonly used for de-noising noisy data, reconstructing corrupted or missing information, and generating synthetic data samples. Furthermore, they have demonstrated effectiveness in tasks such as anomaly detection, feature extraction, and image compression. This versatility makes Auto-Encoders a valuable tool in data preprocessing, representation learning, and generative modelling across different domains.

These advantages highlight the strengths of Auto-Encoders as powerful tools for learning compact representations of complex data, facilitating various machine learning tasks, and addressing challenges associated with nonlinear relationships and feature learning.

3.4.3.5 Disadvantages of Auto-Encoder

As other method, Auto-Encoder come with some drawback.

I **Complexity:**
Designing and training Auto-Encoders can be complex compared to other dimensionality reduction techniques. Auto-Encoders typically involve designing and fine-tuning the architecture of neural networks, which requires expertise in deep learning concepts and techniques. Additionally, selecting appropriate hyperparameters, such as the number of layers, neurons per layer, and activation functions, can significantly impact the performance of the Auto-Encoder. Moreover, optimizing the training process, including choosing suitable optimization algorithms and regularization techniques, adds to the complexity of implementation. Consequently, the complexity involved in designing and training Auto-Encoders may pose challenges, particularly for users with limited experience in deep learning.

II **Data Size:**
Auto-Encoders often require large amounts of data to effectively learn meaningful representations, which can pose challenges for implementation in terms of both time and space. Training Auto-Encoders on small datasets may lead to overfitting, where the model learns to memorize the training examples rather than capturing meaningful patterns and relationships within the data. Moreover, the computational resources and time required to train Auto-Encoders increase with the size of the input data, making them less suitable for applications with limited computational resources or time constraints. Additionally, storing and processing large datasets can strain memory and computational resources, further complicating the implementation of Auto-Encoders, especially in resource-constrained environments.

These limitations underscore the importance of considering factors such as computational complexity, data requirements, and resource constraints when deciding whether to use Auto-Encoders for dimensionality reduction or other tasks. Despite these challenges, Auto-Encoders remain powerful tools for learning compact representations of complex data and addressing nonlinear relationships and feature learning tasks.

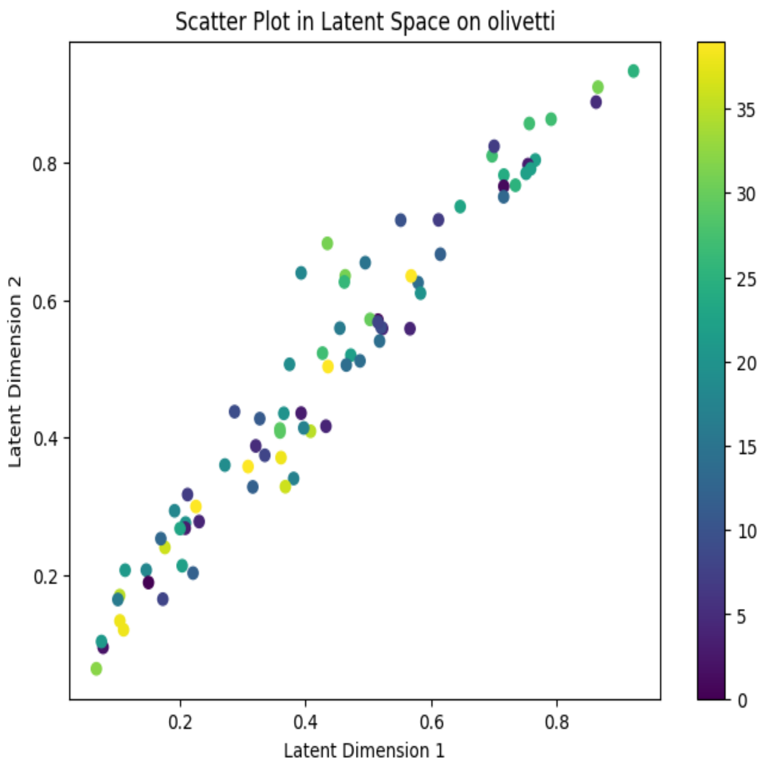


Figure 3.15:
Scatter plot of olivetti after AE dimension reduction

The transformation of Auto-encoder on olivetti dataset is illustrated with some visualization graph:
Figure 3.15 is the Scatter Plot after Auto-Encoder Transformation
shows how the data points are distributed in the reduced feature space obtained through the Auto-Encoder. shows the clusters or patterns, this reveal the structures in the dataset. The scatter plot shows us the insights into the relationships between different facial features.
Figure 3.16 is the Outlier Analysis after Auto-Encoder Transformation:
It help one to identify outliers or instances that deviate significantly from the general patterns in the data. Few outliers suggesting that the Auto-Encoder has successfully reduced the impact of noise or anomalies in the dataset.

Figure 3.17 and 3.18 is the Transformation Comparison: Original vs. After Auto-Encoder:
:
Compare the original images with the reconstructed images obtained after the Auto-Encoder dimension reduction. Observe how facial features are altered or retained in the transformed images. Difference Analysis. Identify differences between the original and transformed faces, focusing on details that have been enhanced or suppressed by the dimension reduction process. Assess whether the transformation maintains facial identity while reducing dimensionality.

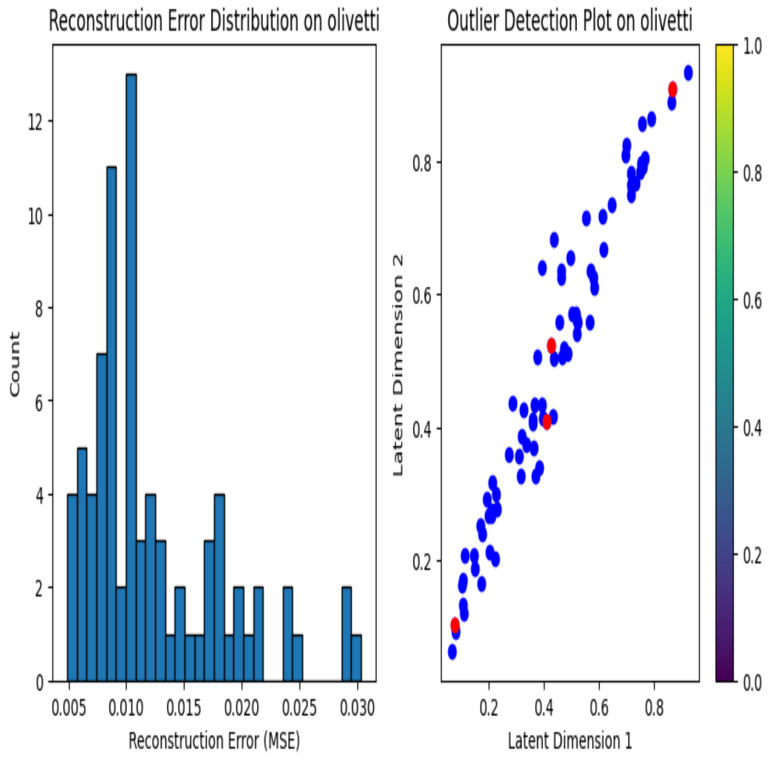


Figure 3.16:
Auto encoder reconstruction error with oulier detection on Olivetti dataset

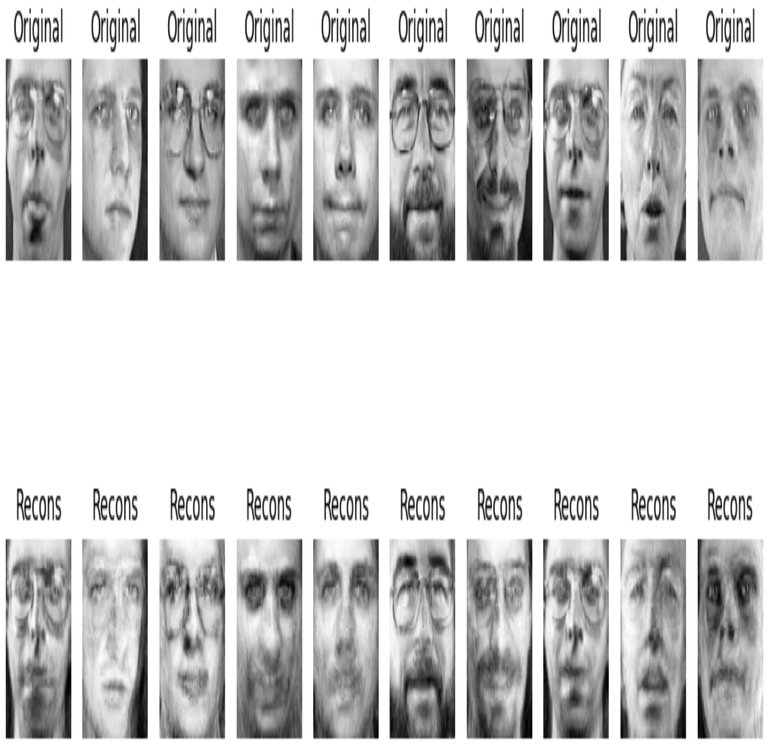


Figure 3.17:
Sample of olivetti dtaset image reconstruction with AE

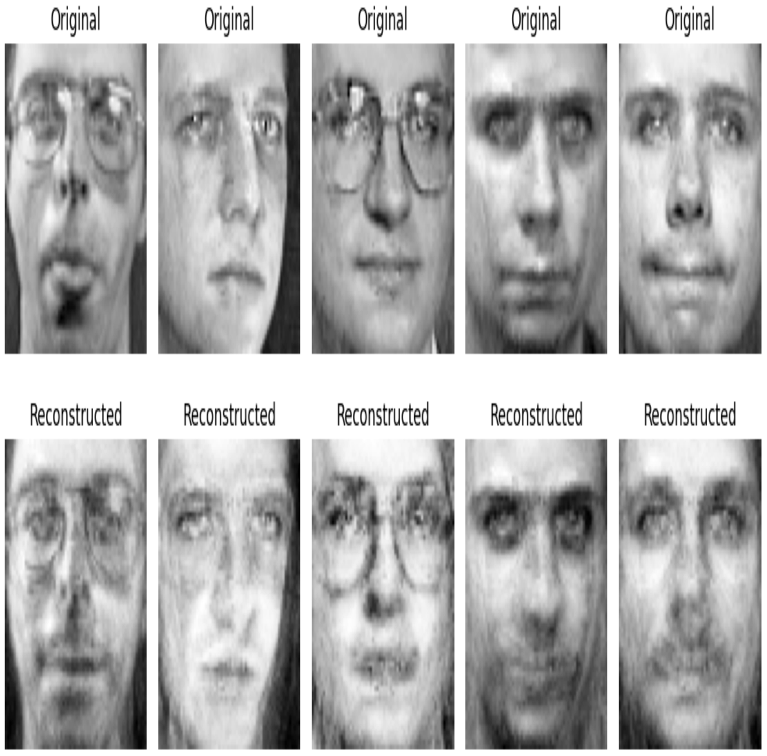


Figure 3.18:
Sample of image reconstruction with AE

These visualizations collectively provide a comprehensive view of the impact of the Auto-Encoder on the Olivetti dataset:
Clustering Effect: Evaluate how well the Auto-Encoder captures patterns and clusters in the reduced feature space.
Outlier Reduction: Confirm the reduction in outliers, indicating improved data representation and noise reduction.

By combining scatter plots and visual comparisons, we gain valuable insights into both the structural changes in the feature space and the visual impact on the facial images after the Auto-Encoder transformation. These figures contribute to understanding the quality and effectiveness of the dimensionality reduction process for the Olivetti dataset.

In the ensuing section, our exploration delves into a meticulous comparative analysis, drawing parallels between the Auto-Encoder and a spectrum of other prevalent dimensionality reduction techniques. This undertaking is pivotal, aiming to unravel the nuanced intricacies inherent in each method and elucidate the distinct strengths and vulnerabilities they bring to the fore.

3.4.3.6 Comparison of PCA, LDA, and Auto-Encoder

PCA and LDA are both linear transformation methods. Despite their similarity, PCA is an unsupervised technique, whereas LDA is a supervised approach to reduce dimension.
PCA is indifferent to class labels. It condenses the feature set without relying on the output. Its objective is to identify directions of maximum variance in the dataset. In extensive feature sets, re-

dundancies often exist, where some features are duplicated or exhibit high correlation with others. PCA's task is to pinpoint such highly correlated or duplicated features and generate a new feature set with minimal inter-feature correlation or, equivalently, a feature set exhibiting maximum variance between features. As this variance is independent of the output, PCA does not consider output labels in its process.

In contrast to PCA, LDA aims to reduce the dimensions of the feature set while preserving the discriminative information among output classes. LDA endeavors to identify decision boundaries around each class cluster, projecting data points to new dimensions in a manner that maximizes the separation between clusters and minimizes the distance between individual points within a cluster and their respective centroids. The ranking of these new dimensions is determined by their effectiveness in maximizing the inter-cluster distance and minimizing the intra-cluster distance between data points and their centroids. These newly derived dimensions constitute the linear discriminants of the feature set.

For uniformly distributed data, LDA typically outperforms PCA. However, in the presence of highly skewed or irregularly distributed data, it is recommended to opt for PCA. This is because LDA has the potential to exhibit bias toward the majority class in such scenarios.

Use PCA when aiming to diminish data dimensionality while retaining maximum variance. Suitable for tasks such as data compression, visualization, and situations with a substantial number of correlated features.

Opt for LDA when working with labelled data, and the goal is to reduce dimensionality while enhancing class separation. Particularly beneficial for classification tasks where clear class discrimination is crucial.

Choose Auto-Encoder when dealing with intricate data featuring nonlinear relationships, and the objective is to capture complex features or patterns. Well-suited for tasks like image denoising, anomaly detection, and when ample data are available for effective training.

In conclusion, the versatility of PCA is evident as it can be applied to both labelled and unlabelled data, given that it operates independently of output labels. Conversely, LDA relies on output classes to identify linear discriminants, making it dependent on labelled data for its application.

After meticulously considering the advantages, disadvantages, and comparisons of the three techniques we studied above, we were inspired to leverage the strengths of each to construct two hybrid techniques for dimension reduction.

3.4.4 Dimension Reduction based on Hybrid Techniques

To enhance the performance of machine learning models, we devised two hybrid dimensionality reduction algorithms by leveraging the strengths of existing techniques. Our inspiration stemmed from the downstream applicability of each previously explained method. Consequently, we constructed a PCA-Auto-Encoder technique and an LDA-Auto-Encoder technique. Through empirical validation

on facial recognition tasks using two distinct benchmark datasets, we ascertained that our formulations surpassed existing approaches. This validation was particularly crucial, given the significance of effective dimension reduction in facial recognition tasks, where capturing essential facial features while minimizing computational complexity is essential.
In what follows, we discuss these two algorithms.

3.4.5 PCA-Auto-Encoder

Leveraging the strengths of both PCA and Auto-Encoder allows for a sequential dimension reduction approach. This is achieved by initiating the process with PCA and subsequently utilizing the approximated dimension reduction obtained from PCA to achieve a final dimension reduction of the data using an Auto-Encoder. Below, we outline the sequential steps of this process:
Starting with the input data z:

Determine the desired number of principal components to be retained.
Train PCA for Dimension Reduction:

Apply PCA to the data and train the model for dimension reduction based on the fixed number of principal components. Retain and Save Reduced Data:

Save the reduced data obtained from PCA with the fixed components. This reduced dataset is denoted as z_{PCA}.
Utilizing the reduced data z_{PCA} as the input:

Auto-Encoder Dimension Reduction:
Employ the Auto-Encoder to perform a subsequent dimension reduction on z_{PCA}.
Obtain Final Reduced Data:

The objective of the Auto-Encoder at this stage is to further reduce dimensionality and reconstruct the non-linear output data. The result is the final reduced dataset.

The final reduced dimensional data (z_{final}) after applying PCA and the Auto-Encoder can now be used for downstream tasks such as face recognition, clustering visualization, just to name a few.

3.4.5.1 PCA-Auto-Encoder Pseudo Code

Algorithm 1 designs the PCA-AutoEncoder that we constructed.

3.4.5.2 Advantages of PCA-Auto-Encoder

The combination of these two techniques for dimension reduction has several advantages that can be described by:

I Linear Initialization for Auto-Encoders:
Auto-Encoders with a linear encoder and decoder can be thought of as linear transformations.

Algorithm 1 PCA-Auto-Encoder for Dimension Reduction

```
 1: procedure PCA-AUTO-ENCODER($\mathbf{X}$, $k$, $m$)
 2:     Input:
 3:     Load the dataset $\mathbf{X}$ with $p$ features and $N$ data points
 4:     Set lower dimension $k$ (to be returned after PCA)
 5:     Set the bottleneck layer size $m$ (for the Auto-Encoder)
 6:     Output:
 7:     Lower-dimensional representations using PCA and Autoencoders
 8:     # Reduction with PCA
 9:     Find the mean vector $\mu$ of $\mathbf{X}$
10:     Center the data: $\mathbf{X}_{\text{centered}} = \mathbf{X} - \mu$
11:     Find the covariance matrix: $\mathbf{C} = \frac{1}{N}\mathbf{X}^T_{\text{centered}}\mathbf{X}_{\text{centered}}$
12:     Do the eigenvalue decomposition on $\mathbf{C}$ to obtain the eigenvectors $\mathbf{V}$ and eigenvalues
13:     Arrange eigenvectors by descending eigenvalues and select the top $k$ eigenvectors to
    form the projection matrix $\hat{\mathbf{X}} \in \mathbb{R}^{p \times k}$
14:     Get the lower-dimensional subspace with the projection: $\mathbf{X}_{\text{PCA}} = \mathbf{X}_{\text{centered}} \cdot \hat{\mathbf{X}}$
15:     # Further Reduction with Auto-Encoder
16:     Initialize an Auto-Encoder model with an encoder network that maps $\mathbf{X}_{\text{PCA}}$ to a further
    lower-dimensional representation $\mathbf{X}_{\text{final}}$ of size $m$ and a decoder network
17:     Train the Autoencoder with $\mathbf{X}_{\text{PCA}}$ as input data to learn the non-linear transformation
    and further reduce dimensionality
18:     Use a mean squared error (MSE) loss to measure the difference between the input $\mathbf{X}_{\text{PCA}}$
    and the reconstructed output $\mathbf{X}_{\text{AE}}$
19:     Optimize the Autoencoder's weights and biases using an optimization technique
20:     After training, the encoder part of the Auto-Encoder will provide the lower-dimensional
    representation $\mathbf{X}_{\text{final}}$ for each data point
21:     Output:
22:     The lower-dimensional representation obtained using PCA: $\mathbf{X}_{\text{PCA}}$
23:     The lower-dimensional representation obtained using Auto-Encoders: $\mathbf{X}_{\text{final}}$
24: end procedure
```

Using PCA as an initial transformation ensures that the Auto-Encoder starts with a linear approximation of the data. This can be advantageous when the data's principal components capture most of the variance.

II Dimension Reduction Cascade:
In some scenarios, you may want to perform a two-stage dimensionality reduction. PCA can be used as the first stage to reduce dimensionality substantially. Auto-Encoders, which are capable of capturing non-linear relationships, can then be applied to further reduce the dimension or learn more intricate features. This two-stage approach can be particularly beneficial when dealing with high-dimensional data with non-linear variations.

III Noise Reduction:
When the data contains noise or irrelevant features, PCA can help eliminate some of this noise by retaining only the most significant components. Auto-Encoders can then be applied to denoise the data and capture informative features.

IV Improved Representation Learning:
Auto-Encoders are effective at learning hierarchical and non-linear representations. By initializing Auto-Encoders with the linearly transformed data obtained from PCA, you can build upon the compact representations generated by PCA and further enhance the quality of the learned features.

V Combining Linear and Non-Linear Transformations:
PCA provides a linear transformation, while Auto-Encoders can capture non-linear relationships. By combining these transformations, you can create a more flexible and expressive representation of the data, accommodating both linear and non-linear patterns.

VI Data Exploration and Visualization:
PCA can be used for data exploration and visualization. By applying PCA initially, you can reduce the dimension for visualization purposes, and then apply Auto-Encoders to uncover complex structures or patterns in the reduced-dimensional space.

3.4.5.3 Disadvantages of PCA-Auto-Encoder

This hybrid dimension reduction technique also has some drawbacks, which can be described by:

I Loss of Nonlinear Patterns:
PCA is a linear technique that focuses on capturing linear relationships in the data. If the data has complex nonlinear patterns, PCA may not capture them effectively. Auto-Encoders are better at capturing nonlinear patterns, but combining them with PCA might still limit the ability to fully model complex data structures.

II Overlapping Information:
Both PCA and Auto-Encoders methods aim to capture the most important features, but they do so differently. When combined, they may be overlapping information or redundancy in the features they select, which can lead to inefficiency.

III Complexity:
Combining PCA and Auto-Encoders can increase the complexity of the overall dimension re-

duction pipeline. This might make it harder to tune hyperparameters and understand the inner workings of the system.

IV Difficulty in Interpretation:
The combined model might produce lower-dimensional representations that are difficult to interpret because it involves a mixture of linear and nonlinear transformations.

V Increased Computational Overhead:
Running both PCA and Auto-Encoders can be computationally expensive, especially for large datasets. This can slow down the dimension reduction process.

VI Hyperparameter Tuning:
Combining two techniques often requires additional hyperparameter tuning to make them work well together. Finding the right set of hyperparameters can be time-consuming and challenging.

VII Limited Generalization:
The combination of PCA and Auto-Encoders might be highly specific to the dataset it was designed for. It may not generalize well to other datasets or tasks.

VIII Data Preprocessing Challenges:
Preprocessing data to be suitable for both PCA and Auto-Encoders can be tricky. It's important to ensure that the data preprocessing steps are consistent for both techniques.

IX Loss of Model Transparency:
Combining PCA and Auto-Encoders may result in a less interpretable model. Understanding the transformed features can be more challenging.

X Dependent on Data Characteristics:
The effectiveness of combining PCA and Auto-Encoders depends on the specific characteristics of the dataset. What works well for one dataset may not work as effectively for another.

3.4.6 LDA-Auto-Encoder

Capitalizing on the strengths of LDA and Auto-Encoder, we successfully integrated these two techniques for dimension reduction. Our approach involves initiating dimension reduction on the input data using LDA, followed by Auto-Encoder based dimension reduction on the LDA-derived reduced data. This process provides the advantage of leveraging both supervised and unsupervised approaches. The sequential steps of this combined technique are as follows:
Given the data z comprising features and class labels y:

Determine the desired number of LDA components to be retained.

Apply LDA to the data to maximize class separability, effectively reducing dimension based on the fixed number of LDA components.
Save the reduced data obtained from LDA with the fixed components. This reduced dataset is denoted as z_{lda}.
Use z_{lda} as Input for Auto-Encoder:
Utilize the reduced data z_{lda} as the input for the Auto-Encoder.
Apply the Auto-Encoder to perform further dimension reduction on z_{lda}. Obtain Final Reduced Data:

The ultimate reduced dimensional data, denoted as z_{final}, is obtained after the sequential application of LDA and the Auto-Encoder. These refined data are now ready for application in various tasks, such as face recognition, as exemplified in our case. Additionally, one may consider downstream tasks like clustering visualization using the reduced data. The versatility of z_{final} makes it suitable for a range of applications, benefiting from the enhanced features extracted through the combined strengths of LDA and Auto-Encoder.

3.4.6.1 LDA-Auto-Encoder Pseudo Code

The algorithm for this technique is:

Algorithm 2 LDA-Auto-Encoder for Dimension Reduction

```
Require:
    Labeled face dataset with $c$ classes and $n$ samples.
    Desired lower-dimensional representation dimension $k$.
Ensure:
    Lower-dimensional representations for face images.
 1: procedure COMBINE LDA AND AUTO ENCODERS(face Dataset, $k$)
 2:     Compute class means $\mu_i$ for each class $i$
 3:     Calculate within-class scatter matrix $S_w$
 4:     Calculate between-class scatter matrix $S_b$
 5:     Eigen decomposition of $S_w^{-1}S_b$
 6:     Sort eigenvectors by corresponding eigenvalues
 7:     Select top $k$ eigenvectors as transformation matrix
 8:     Apply LDA transformation to face Dataset
 9:     Design Auto Encoder architecture
10:     Initialize encoder and decoder networks
11:     Train Auto Encoder to minimize MSE loss
12:     Fine-tune Auto Encoder using LDA-transformed data
13:     Use Auto Encoder output as lower-dimensional representations
14: end procedure
```

3.4.6.2 Advantages of LDA-Auto-Encoder

The combination of LDA and Auto-Encoder for dimension reduction provides several advantages. Here are some of them:

I Preservation of Discriminative Information:
LDA is specifically designed to maximize class separability, making it effective for supervised dimension reduction. By using LDA in combination with an Auto-Encoder, you can retain the discriminative information necessary for classification or recognition tasks.

II Improved Feature Extraction:
LDA focuses on extracting features that best discriminate between classes, while Auto-Encoders are capable of capturing nonlinear and fine-grained patterns in data. Combining these techniques can lead to more informative and descriptive lower-dimensional representations.

III Dimensionality Reduction and Feature Learning:
LDA reduces dimension by selecting a subset of the most informative dimensions. Auto-Encoders

can further reduce the dimension and perform feature learning by capturing complex relationships within the data.

IV Enhanced Robustness to Noise:
LDA can be sensitive to noisy data, but Auto-Encoders can help filter out noise by learning to represent the underlying structure of the data. This combination can lead to more robust feature extractions.

V Improved Generalization:
The combination of LDA and Auto-Encoders can create features that generalize well to new and unseen data. This is especially valuable in tasks like face recognition, where the system needs to recognize individuals it hasn't seen during training.

VI Reduced Overfitting:
Auto-Encoders can act as regularization, helping to reduce overfitting in LDA by capturing essential information without modelling noise or outliers.

VII Flexible Modelling:
Auto-Encoders can adapt to various data distributions and nonlinearity, which complements LDA's linearity. This flexibility can make the combined technique suitable for a wide range of applications.

VIII Interpretable Features:
LDA provides features that are linear combinations of the original variables, making them interpretable. This is particularly useful in applications where understanding the significance of features is important.

IX Robust to Imbalanced Data:
LDA can help mitigate issues related to class imbalance by focusing on class separation. Combined with Auto-Encoders, it can address challenges posed by imbalanced datasets.

X Improved Classification Performance:
When the lower-dimensional representations are used as input for classifiers (e.g., support vector machines, neural networks), the combination of LDA and Auto-Encoders often leads to enhanced classification performance.

XI Reduced Computational Complexity:
By first reducing dimension using LDA and then applying Auto-Encoders, you may reduce the computational complexity of training the Auto-Encoders, making it more efficient for large datasets.

3.4.6.3 Disadvantages of LDA-Auto-Encoder

While LDA-Auto-Encoder comes with several advantages, it also has its drawbacks:

I Complexity of Implementation:
Combining two different techniques requires additional coding and integration work, which can be complex, especially for those less familiar with both methods.

II Increased Computational Overhead:
Running both LDA and Auto-Encoders can be computationally expensive, especially for large datasets. The additional complexity can slow down the dimension reduction process.

III Hyperparameter Tuning:
Both LDA and Auto-Encoders have their own hyperparameters. Combining them may require additional tuning to make them work well together, which can be time-consuming and challenging.

IV Data Preprocessing Challenges:
Ensuring that data preprocessing is consistent for both techniques can be tricky. Mismatched preprocessing can lead to suboptimal results.

V Loss of Model Transparency:
Combining LDA and Auto-Encoders may result in a less interpretable model. Understanding the transformed features can be more challenging, especially when nonlinear transformations are involved.

VI Difficulty in Interpretation:
The combined model might produce lower-dimensional representations that are difficult to interpret because it involves a mixture of linear and nonlinear transformations.

VII Sensitivity to Hyperparameters:
The combined model's performance may be sensitive to the choice of hyperparameters, and finding the right set of hyperparameters can be non-trivial.

VIII Limited Generalization:
The combination of LDA and Auto-Encoders might be highly specific to the dataset it was designed for. It may not generalize well to other datasets or tasks.

IX Dependent on Data Characteristics:
The effectiveness of combining LDA and Auto-Encoders depends on the specific characteristics of the dataset. What works well for one dataset may not work as effectively for another.

X Risk of Overfitting:
Combining two powerful techniques can potentially lead to overfitting if not carefully controlled and regularized.

XI Increased Dimensionality:
In some cases, combining LDA and Auto-Encoders may lead to higher dimensionality instead of reducing it. This can happen when both techniques retain a large number of features.

In summary, while LDA-Auto-Encoder offers significant advantages in terms of feature preservation, robustness, and improved generalization, users should be mindful of the complexities and challenges associated with its implementation, computational requirements, and potential limitations in certain scenarios. Careful consideration and experimentation with hyper-parameters and data characteristics are essential to realizing the full potential of this hybrid dimension reduction technique.

3.5 Experiment

In this experimental section, we embark on a comprehensive exploration of facial recognition by implementing five distinct dimension reduction techniques on two diverse datasets. Leveraging the power and flexibility of Python, we aim to rigorously evaluate the performance of our novel dimension reduction algorithm in the context of facial recognition. The integration of cutting-edge dimensionality reduction methods provides a nuanced lens through which we can scrutinize and compare the efficacy of our approach against established techniques. By conducting these experiments on two distinct datasets, we seek to unravel the algorithm's adaptability and generalizability across varied data structures, thereby contributing valuable insights to the field of facial recognition and dimensionality reduction. This experimental endeavour not only sheds light on the algorithm's performance but also underscores the practical implications of its application in real-world scenarios. Through meticulous experimentation and analysis, we endeavour to push the boundaries of understanding in the realm of facial recognition and pave the way for advancements in dimensionality reduction methodologies.
Table3.1 provides the maximum accuracy of facial recognition after 5 dimension reduction techniques on the Olivetti dataset.

Accuracy percentage on Olivetti Dataset					
Classification Methods	PCA	LDA	Auto Encoder	PCA-AutoEncoder	LDA-AutoEncoder
MLP	94,16	95,00	75,00	96,25	98,75
RF	91,66	93,00	85,00	95,00	98,75
SVM	93,30	96,00	27,50	90,00	96,25
DT	68,33	67,00	56,25	58,75	70,00
KNN	89,66	95,00	88,12	95,31	100
XBooST	75,83	74,66	74,16	74,16	74,16
NVB	83,33	87,50	73,75	86,25	90,00
LR	94,16	96,25	85,00	97,50	98,75

Table 3.1: Accuracy table for Olivetti.

Table 3.2 provides the maximum accuracy of facial recognition after 5 dimension reduction techniques on the LFW dataset.

3.5.1 PCA-AutoEncoder

The PCA-AutoEncoder method introduces a sequential dimension reduction approach, combining the linear transformations of PCA with the non-linear capabilities of AutoEncoders. The process involves initiating dimension reduction with PCA and subsequently employing an AutoEncoder on the PCA-derived data. This hybrid approach aims to capture both linear and non-linear relationships in the facial data, providing a more comprehensive representation.

The PCA-AutoEncoder algorithm, presented in Algorithm 1, outlines the sequential steps of this process. Advantages and disadvantages of this hybrid technique are discussed, highlighting its potential

Accuracy percentage on LFW Dataset					
Classification Methods	PCA	LDA	Auto Encoder	PCA-AutoEncoder	LDA-AutoEncoder
MLP	83,97	62,40	80,62	82,17	52,32
RF	62,79	57,75	63,95	63,56	55,42
SVM	93,20	63,17	71,31	75,19	53,87
DT	43,92	49,22	48,22	48,06	55,03
KNN	61,24	63,56	72,13	72,03	99,70
XBooST	71,05	70,54	70,54	70,54	70,54
NVB	74,41	63,17	59,68	74,67	54,26
LR	79,32	70,54	79,45	80,62	50,77

Table 3.2: Accuracy Table for LFW.

benefits, such as improved representation learning and noise reduction, along with challenges like the loss of non-linear patterns and increased computational overhead.

3.5.2 LDA-AutoEncoder

The LDA-AutoEncoder technique combines Linear Discriminant Analysis (LDA) and AutoEncoders to capitalize on the strengths of supervised and unsupervised dimensionality reduction. The algorithm involves initiating dimension reduction with LDA, followed by an AutoEncoder-based reduction on the LDA-derived data. This approach aims to preserve discriminative information while capturing non-linear patterns, offering advantages such as improved feature extraction and enhanced robustness to noise.
The LDA-AutoEncoder algorithm, presented in Algorithm 2, outlines the steps of this combined technique. The advantages and disadvantages include preservation of discriminative information, reduced overfitting, and improved generalization, along with challenges like increased computational overhead and difficulty in interpretation.

3.5.3 Experimental Validation

To assess the effectiveness of the proposed hybrid dimension reduction techniques, we conducted empirical validation on facial recognition tasks using two benchmark datasets: Olivetti Faces and Labeled Faces in the Wild (LFW). Classification accuracy results for various machine learning models, including Multi-Layer Perceptron (MLP), Random Forest (RF), Support Vector Machine (SVM), Decision Tree (DT), k-Nearest Neighbors (KNN), XGBoost (XBooST), Naive Bayes (NB), and Logistic Regression (LR), were obtained and compared across different dimensionality reduction techniques.
Tables 3.1 and 3.2 present the maximum accuracy percentages achieved by each method on the Olivetti Faces and LFW datasets, respectively. Additionally, Figures 3.21 to 3.27 provide visual representations of the results, allowing for a more intuitive analysis.
Furthermore, our comprehensive experiment underscores the efficacy of our innovative dimension reduction techniques, PCA-Auto-Encoder and LDA-Auto-Encoder, showcasing their superior performance in face recognition. In contrast to traditional approaches like PCA, LDA, and Auto-Encoder, our methods exhibit heightened accuracy and efficiency on olivetti dataset in general and in the case of LFW dataset it outperform wit KNN model, marking a significant advancement in the realm of

dimension reduction for facial recognition applications.

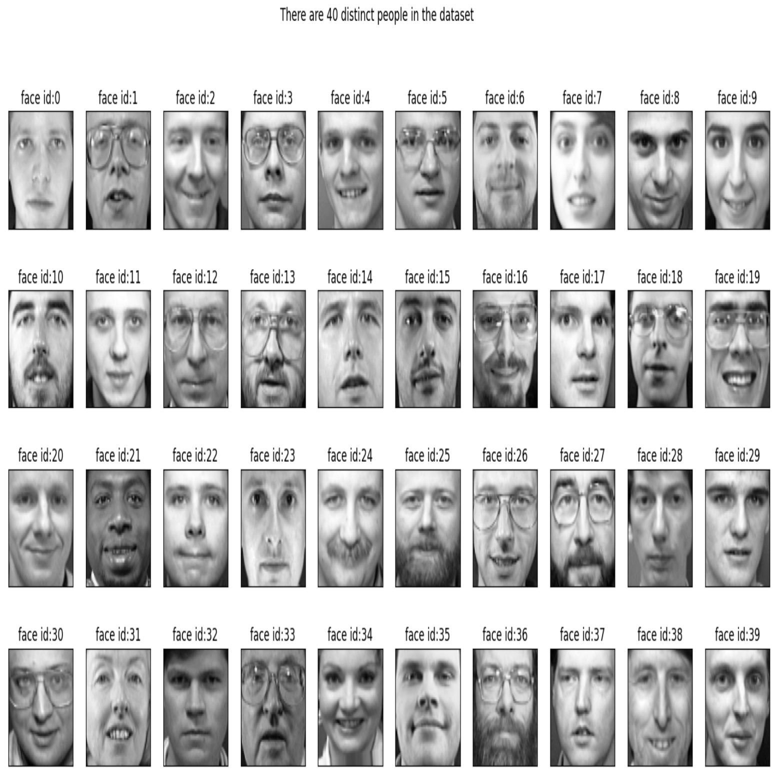


Figure 3.19:
Some saples of ollivetti dataset

3.6 Conclusion

Conclusively, our in-depth investigation into hybrid dimension reduction techniques, particularly PCA-AutoEncoder and LDA-AutoEncoder, unequivocally establishes their superiority in elevating the performance of facial recognition systems. Through rigorous empirical validation on benchmark datasets, the inherent advantages of these techniques stand out when compared to traditional methods such as PCA, LDA, and AutoEncoder.

The detailed analysis of accuracy results, coupled with a nuanced exploration of the strengths and limitations of each technique, presents invaluable insights for both researchers and practitioners within the field of facial recognition. The proposed hybrid approaches not only address current challenges but also open up promising avenues for future advancements in dimensionality reduction techniques, suggesting potential applications that extend beyond the realm of facial recognition

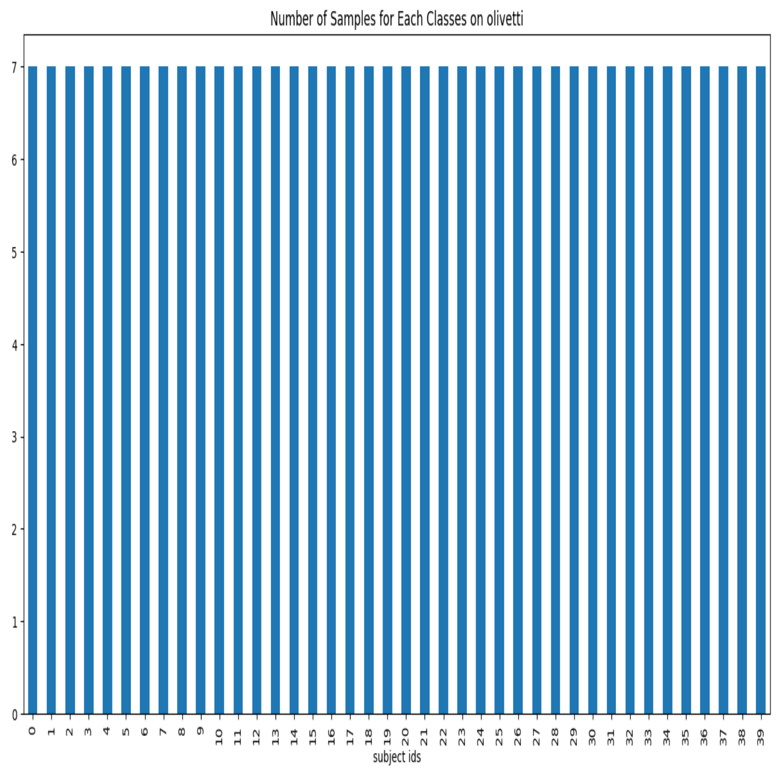


Figure 3.20:
Classes Ballance of the ollivetti Dataset

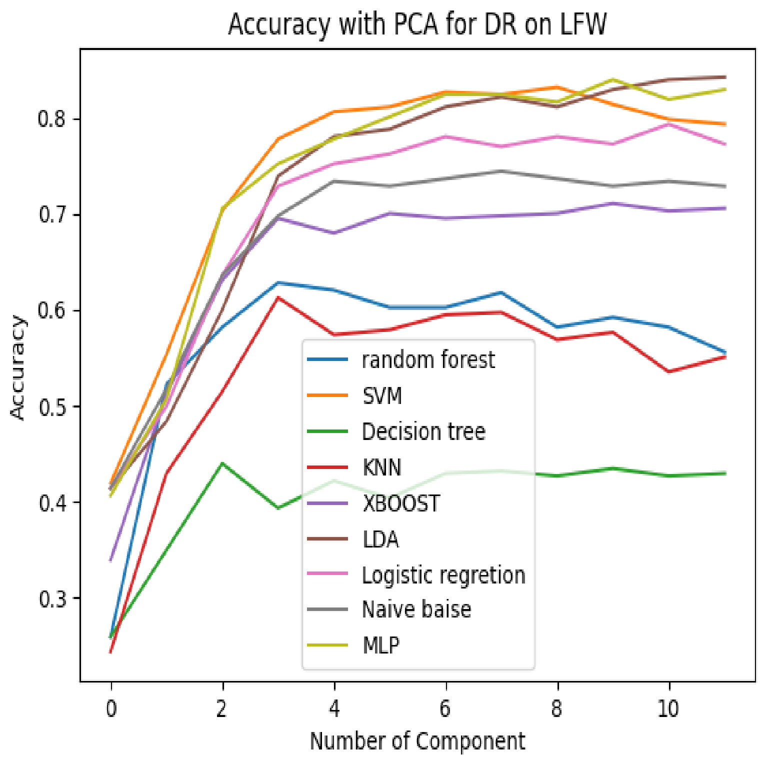


Figure 3.21: Accuracy percentage of machine learning classification on LFW after PCA for DR

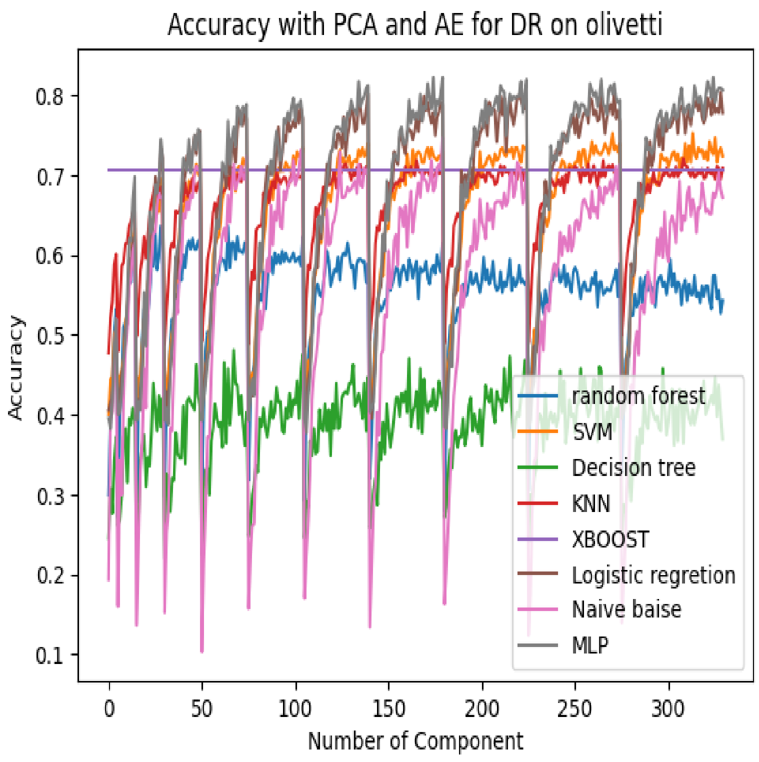


Figure 3.22:
Accuracy percentage of machines learning classifications on Olivetti after PCA and AE for DR

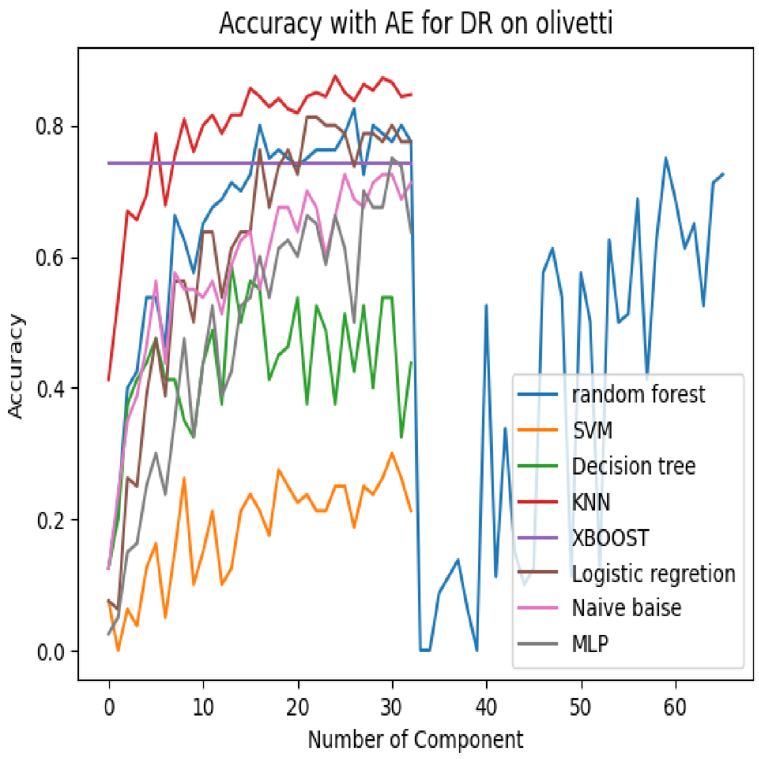


Figure 3.23:
Accuracy percentage of machines learning classifications on Olivetti after AE for DR

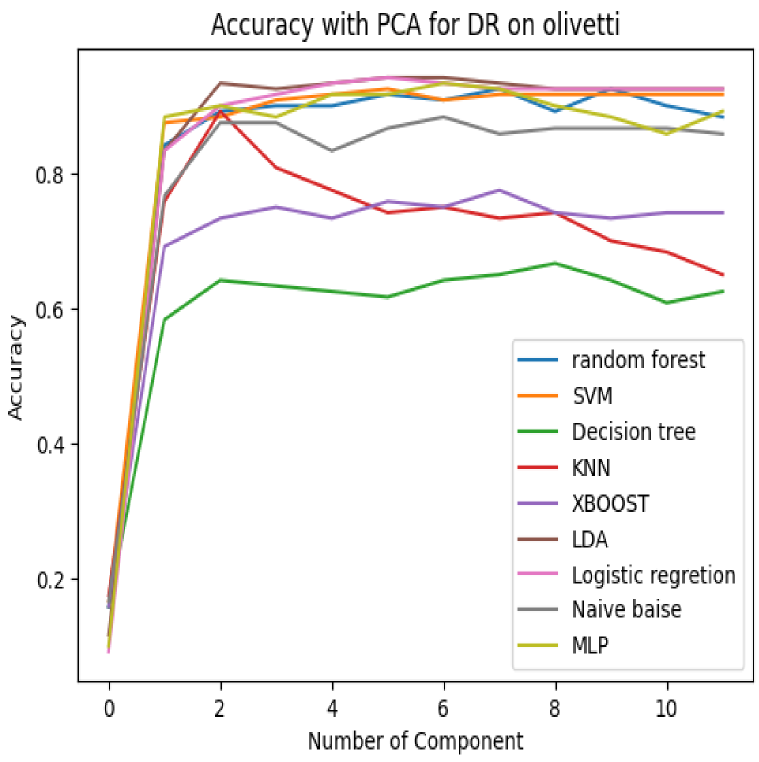


Figure 3.24:
Accuracy percentage of face recognition classification on Olivetti dataset with PCA for DR

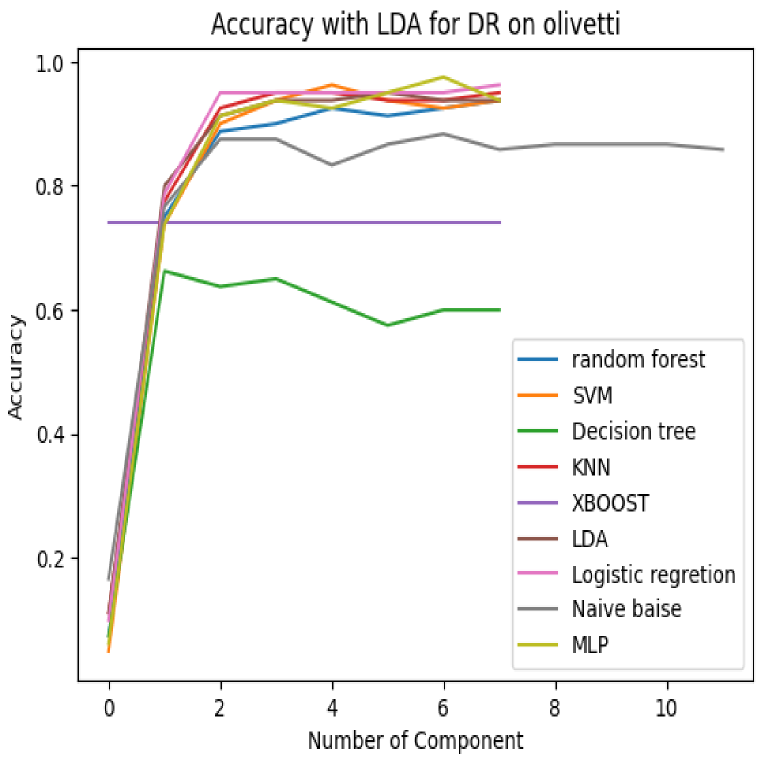


Figure 3.25:
Accuracy percentage of face recognition classification on Olivetti dataset with LDA for DR

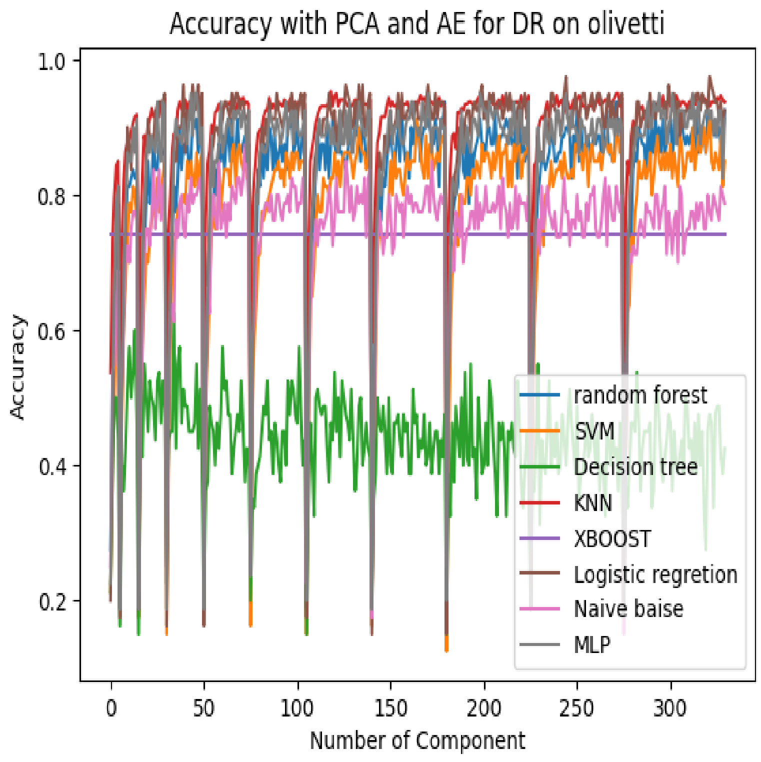


Figure 3.26:
Accuracy percentage of face recognition classification on Olivetti dataset with LDA and AE for DR

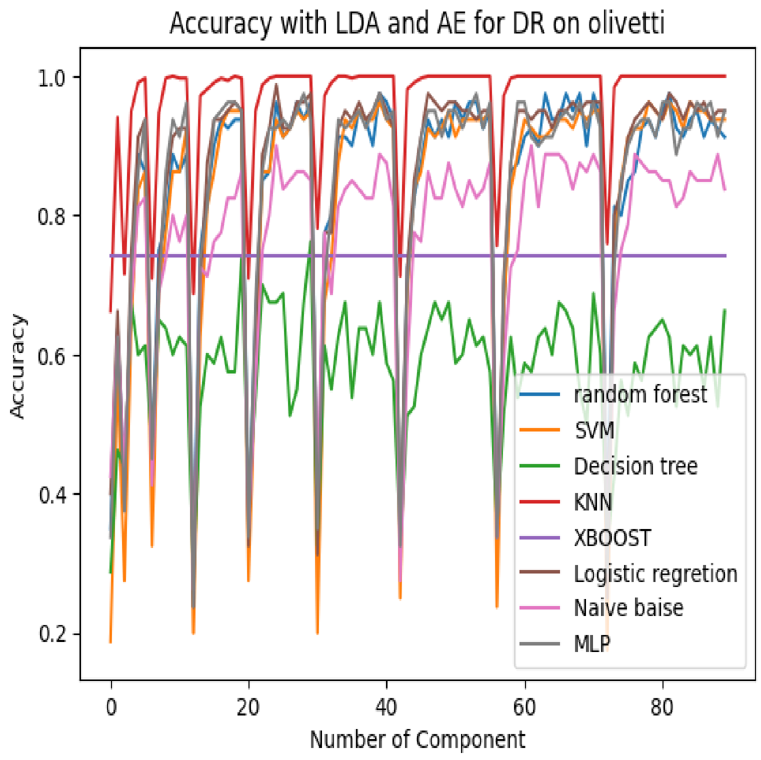


Figure 3.27:
Accuracy percentage of face recognition classification on Olivetti dataset with LDA and AE for DR

Chapter 4

Tensor decomposition with face recognition

4.1 Introduction

Most of the traditional machine learning models that have been used for facial recognition are getting their mathematics structure from linear algebra, taking in consideration that the person identity is the only factor that are supposed to variate. But in the real face recognition scenario, one has to come across other variations such as view angle, lightning, face pose, face expression and so forth. One then has that face recognition can come across many complexity factors. Hence making face recognition a multiple analysis problem, therefore, requesting someone to think above the linear algebra perspective.

This multidimensional perspective challenge is not restricted only to face recognition, since big data have become a reality in most applications. In the abstract of [17] one can read: "The widespread use of multi-sensor technology and the emergence of big data has highlighted the limitation of standard flat-view matrix models and the necessity to move towards more versatile data analysis tools." Hence, this leaded to our research of alternative method to improve the traditional linear algebra or neural network technique on facial recognition that we presented in the previews chapters.

One mathematics alternative to this problem is Multilinear algebra which has a potential to disentangle constituent factors that can be found in some face data, this was stated by Vasilescu and Demetri in [72] in their introduction as: mulltiinear algebra, the algebra of higher-order tensors, has a potential mathematics framework for analysing ensembles of images resulting from the interaction of any number of underlying factors.

Using multilinear algebra which is the algebra of higher-order matrix, one will be offered with powerful and sophisticated tools to approach a multi factor model of representation of images. The Higher Order Singular Value Decomposition(HOSVD) algorithm for face recognition makes use of third order tensor which from multilinear algebra property can provide us with the possibility for improvement. Using the mathematical property of the HOSVD one may be able to make an impact on the image processing problem and in particular with face recognition algorithm.

In what follows, we will dive into the concept of HOSVD which is one among many tensor decomposition techniques(works on tensor decomposition can be found in [9, 12, 19, 38, 42, 45, 65, 72]). We use the HOSVD to address the multi factor challenge of face recognition. We first give some basic concepts an definition of tensor, secondly we discuss some mathematical tools useful for our decomposition, then we explain how we construct HOSVD. We also explain how we use HOSVD to do dimension reduction and lastly we introduce the application of HOSVD on facial recognition.

4.2 Tensor definition

To work with multi factors data, one needs to understand the concept of tensor which is very important in big data. We will now provide some definitions in the area of tensor with some mathematics concepts that will be used for HOSVD.

Definition 7 *A tensor is a multidimensional array of data where its elements are referred by using multiple indexes:* $\mathcal{A} \in \mathbb{C}^{I_1 \times I_2 \times \cdots \times I_N}$.

It is worth noticing that a tensor of order 1 is a vector and a tensor of order 2 is a matrix. So when dealing with order above two, we will need more than linear algebra to manipulate the given tensor. As we always find the norm of two vectors it is important to understand how one can find the norm of two tensors. This is done with the help of the Frobenuis norm.

Definition 8 *The scalar product of two tensors is given by:*

$$\langle \mathcal{A}, \mathcal{B} \rangle = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} b_{i_1,i_2,\cdots,i_N} a_{i_1,i_2,\cdots,i_N}.$$

We compute the Frobenuis norm by:

$$||\mathcal{A}|| = \sqrt{\langle \mathcal{A}, \mathcal{A} \rangle}.$$

Definition 9 *[51] Let* $\mathcal{A} \in \mathbb{C}^{I_1 \times I_2 \times \cdots \times I_N}$ *be an Nth-order tensor. The n-th matrix unfolding of* $\mathcal{A}$ *is* $A_{(n)} \in \mathbb{C}^{I_n \times (I_{n+1} I_{n+2} \cdots I_N I_1 I_2 \cdots I_{n-1})}$ *which contains the element* $a_{i_1 i_2 \cdots i_N}$ *at the position with row* i_n *and column number equal to*

$$\begin{aligned}
& (i_{n+1} - 1) I_{n+2} I_{n+3} \cdots I_N I_1 I_2 \cdots I_{n-1} \\
+ \quad & (i_{n+2} - 1) I_{n+3} I_{n+4} \cdots I_N I_1 I_2 \cdots I_{n-1} \\
+ \quad & \cdots + (i_N - 1) I_1 I_2 \cdots I_{n-1} + (i_1 - 1) I_2 I_3 \cdots I_{N-1} \\
+ \quad & (i_2 - 1) I_3 I_4 \cdots I_{n-1} \\
& \vdots \\
+ \quad & i_{n-1}.
\end{aligned}$$

These are some examples of an order-3 tensor matrix unfolding.
Let $\mathcal{A} \in \mathbb{C}^{3 \times 2 \times 3}$, we have:

1. Unfolding 1

$$\mathbf{A_1} = \begin{bmatrix} a_{111} & a_{112} & a_{113} & a_{121} & a_{122} & a_{123} \\ a_{211} & a_{212} & a_{213} & a_{221} & a_{222} & a_{223} \\ a_{311} & a_{312} & a_{313} & a_{321} & a_{322} & a_{323} \end{bmatrix}.$$

2. Unfolding 2

$$\mathbf{A_2} = \begin{bmatrix} a_{111} & a_{112} & a_{113} & a_{211} & a_{212} & a_{213} & a_{311} & a_{312} & a_{313} \\ a_{121} & a_{122} & a_{123} & a_{221} & a_{222} & a_{223} & a_{321} & a_{322} & a_{323} \end{bmatrix}.$$

3. Unfolding 3

$$\mathbf{A_3} = \begin{bmatrix} a_{111} & a_{121} & a_{211} & a_{221} & a_{311} & a_{321} \\ a_{112} & a_{122} & a_{212} & a_{222} & a_{312} & a_{322} \\ a_{113} & a_{123} & a_{213} & a_{223} & a_{313} & a_{323} \end{bmatrix}.$$

Definition 10 *[32] The n-mode product of a tensor $\mathcal{A} \in \mathbb{C}^{I_1 \times I_2 \times \cdots \times I_N}$ by a matrix $U \in \mathbb{C}^{J_n \times I_n}$ denoted by $\mathcal{A} \times_n U$ is an $(I_1 \times I_2 \times \cdots \times I_{n-1} \times J_n \times I_{n+1} \times \cdots \times I_N)$-tensor of which the entries are given by:*

$$(\mathcal{A} \times_n U)_{i_1 i_2 \cdots i_{n-1} j_n i_{n+1} \cdots i_N} = \sum_{i_n=1}^{I_n} a_{i_1 i_2 \cdots i_{n-1} i_n i_{n+1} \cdots i_N} u_{j_n i_n}.$$

4.3 Bilinear maps

Going forward, we need to understand the mathematical concept of bilinear map since tensor decomposition can be viewed as a bilinear manipulation. Hence we give some definition and details on this concept.

Definition 11 *[32]*
Let U, V and W be three vector spaces over a field K, a map $f : U \times V \to W$ is said to be bilinear if and only if for any $u \in U$, $v \in V$ and $\lambda \in K$:

$$\begin{aligned} f(u_1 + \lambda u_2, v) &= f(u_1, v) + \lambda f(u_2, v); \\ f(u, v_1 + \lambda v_2) &= f(u, v_1) + \lambda f(u, v_2). \end{aligned}$$

Usually, the set $\mathrm{Bil}(U, V, W)$ represents the vector space of all bilinear maps $f : U \times V \to W$.

Definition 12 *Let V be any subspace over a field K, its dual space is the set of all linear maps $\varphi : V \to K$ and we denote it by V^*.*

One can also note that if the dual space V^* is fitted with the addition and scalar multiplication:

$$(\varphi + \psi)(x) := \varphi(x) + \psi(x);$$

$$(k\varphi)(x) := k\varphi(x),$$

then, it is also a vector space.

Definition 13 *[22]*
The rank of the bilinear map $f : U \times V \to W$ is the smallest non-negative integer r such that

$$f(u,v) = \sum_{i=1}^{r} x_i(u) y_i(v) z_i,$$

with $x_i \in U^*$, $y_i \in V^*$ and $z_i \in W$ for $i = 1, \cdots, r$. We write $R(f) = r$.

In the tensor manipulation, one will write

$$F = \sum_{i=1}^{r} x_i \otimes y_i \otimes z_i \in U^* \otimes V^* \otimes W.$$

Note that when we set I as the identity of W, we have the equality:

$$F(u \otimes v \otimes I) = f(u,v).$$

4.3.1 Tensor products of bilinear maps

Given two bilinear maps $f_i : U_i \times V_i \to W_i$, $i = 1, 2$ where U, V, W are finite dimensional $K-$vector spaces. And taking into consideration the canonical isomorphism that was defined in[32] by:

$$\begin{aligned} Bil(U_1, V_1; W_1) \otimes Bil(U_2, V_2; W_2) &\cong U_1^* \otimes V_1^* \otimes W_1 \otimes U_2^* \otimes V_2^* \otimes W_2; \\ &\cong U_1^* \otimes U_2^* \otimes V_1^* \otimes V_2^* \otimes W_1 \otimes W_2; \\ &\cong (U_1^* \otimes U_2^*) \otimes (V_1^* \otimes V_2^*) \otimes (W_1 \otimes W_2); \\ &\cong (U_1 \otimes U_2)^* \otimes (V_1 \otimes V_2)^* \otimes (W_1 \otimes W_2); \\ &\cong Bil((U_1 \otimes U_2), (V_1 \otimes V_2); W_1 \otimes W_2). \end{aligned}$$

The bilinear map $f_1 \otimes f_2 \in Bil(U_1, V_1; W_1) \otimes Bil(U_2, V_2; W_2)$ can be defined by

$$f_1 \otimes f_2 : (U_1 \otimes U_2) \times (V_1 \otimes V_2) \to W_1 \otimes W_2.$$

Note that this can be uniquely determined by

$$(f_1 \otimes f_2)(x_1 \otimes x_2, y_1 \otimes y_2) = f_1(x_1, y_1) \otimes f_2(x_2, y_2).$$

$f_1 \otimes f_2$ is called the tensor product of f_1 and f_2 [[22] pp 41]. When taking into consideration the definition of the tensor of a bilinear map, we have that:

$$f_i(x_i, y_i) = \sum_{\rho=1}^{R(f_i)} u_\rho^{(i)}(x_i) v_\rho^{(i)}(y_i) w_\rho^{(i)} \quad (i = 1, 2),$$

then from [32] , we have

$$\begin{aligned} f_1(x_1, y_1) \otimes f_2(x_2, y_2) &= \sum_{\rho=1}^{R(f_1)} \sum_{\sigma=1}^{R(f_2)} \Big((u_\rho^{(1)}(x_1) u_\sigma^{(2)}(x_2) v_\rho^{(1)}(y_1) v_\sigma^{(2)}(y_2) \Big) w_\rho^{(1)} \otimes w_\sigma^{(2)} \\ &= (f_1 \otimes f_2)(x_1 \otimes x_2, y_1 \otimes y_2); \end{aligned}$$

Which then lead us to the following proposition.

Proposition 1 *The rank of a tensor product is less or equal to the product of the rank.*

$$R(f_1 \otimes f_2) \leq R(f_1) \cdot R(f_2)$$

We can now give the following definition and proposition.

Definition 14 *We say that a bilinear map is concise when the following conditions hold.*

1. *The left kernel*
   $\{u \in U \mid f(u,v) = 0 \ \forall v \in V\} = \{0\}$.
2. *The right kernel*
   $\{v \in V \mid f(u,v) = 0 \ \forall v \in V\} = \{0\}$.
3. *Span*
   $\{f(u,v), u \in U, v \in V\} = W$.

Proposition 2 *[32]*
If a bilinear map $f : U \times V \to W$ is concise, then $R(f) \geq \max(dim(U), dim(V), dim(W))$.

Proof.
We do this by contradiction
Suppose

$$\left(f(u,v) = \sum_{i=1}^{r(F)} x_i(u) y_i(v) z_i \right),$$

where $x_i \in U^*, y_i \in V^*$ and $z_i \in W$. If $\mathrm{R}(f) < \dim(U)$, then $\{x_1, x_2, \cdots, x_{r(F)}\}$ does not form a basis for U^*. Hence $\exists u \neq 0$ such that $x_i(u) = 0$ for all x_i. Hence the left kernel of f will be non-zero. This contradicts the first condition of conciseness. Hence proving the left kernel condition.

One can use similar argument with V to prove $R(f) \geq \dim(\mathrm{V})$, which is the right kernel condition.
Now let us prove the third condition which deals with the span.
Suppose $R(f) < \dim(W)$, hence the dimension of the image of $f(u,v)$ will be less than the dimension of the space W. Hence our span condition is contradicted.
The proof is completed.
♣

All these concepts are part and parcel for the minimum rank approximation which are crucial for tensor decomposition.

4.4 Higher Order Singular Value Decomposition (HOSVD)

The tensor unfolding, together with the singular value decomposition(SVD) which is one of the orthogonal matrix decomposition are the tools that one needs to understand for the construction of HOSVD.

Given a A, $m \times n$ matrix over $\mathbb{R}$, we define its singular value decomposition (SVD) by

$$A = U\Sigma V^T.$$

Here, U is a $m \times m$ and orthogonal matrix, while V is a $n \times n$ and orthogonal matrix, and very importantly, $\sum = \text{diag}(\sigma_1, \sigma_2, \cdots, \sigma_r)$ is a "diagonal" matrix whom the σ_i are in descending order. One can also use the croneker product with U, V and the $e_{i,k}$ basis matrix to do a second order tensor decomposition of A as

$$A = \sum_{j=1}^{r} \sigma_j (Ue_{j,m}) \otimes (Ve_{j,n})^T.$$

Note that here r is the number of non-zero singular values of A which represent the rank of A.
This equation is a minimum rank tensor decomposition of A.
Now, let construct the HOSVD.

Definition 15 *[51] The Higher Order Singular Value Decomposition is computed by*

$$\mathcal{A} = \mathcal{S} \times_1 U_{(1)} \times_2 U_{(2)} \times_3 \cdots \times_n U_{(n)},$$

where $\mathcal{A}$ is a n-th order tensor and $U_{(j)} (j = 1, \cdots, n)$ are orthogonal matrices.
While this definition recaptures many properties of the singular value decomposition, it is important to note that higher order singular value decomposition does not necessarily provide a decomposition with minimal rank.

The HOSVD computation in practice can be done as follow:

1. Unfolding matrices:
   Compute the matrix unfolding A_k for $k \in [1, N]$

2. Singular value decomposition:
   Do the singular value decomposition of j-th unfolding $A_{(n)}$ of $\mathcal{A}$ for all $n \in [1; N]$ and save the $U^{(j)}$ for each of them.

3. Core tensor:
   Find the core tensor $\mathcal{S}$ by computing:

$$\mathcal{S} = \mathcal{A} \times_1 U^{(1)^H} \times_2 U^{(2)^H} \times_3 \cdots \times_n U^{(n)^H}.$$

4. Final tensor decomposition computation:
   Now with all these, compute the HOSVD as defined above by:

$$\mathcal{A} = \mathcal{S} \times_1 U_{(1)} \times_2 U_{(2)} \times_3 \cdots \times_n U_{(n)}.$$

4.4.1 Similarity of HOSVD SVD

Some properties of the singular values σ_j of A have analogues for the tensor $\mathcal{S}$ of $\mathcal{A}$.

Proposition 3 *For any α, β; then, $\mathcal{S}_{i_k=\beta}$ and $\mathcal{S}_{i_k=\alpha}$ are orthogonal for all possible values of $k \in [1; n]$, α and β provided that $\alpha \neq \beta$. Here orthogonality is in the usual Euclidean sense.*

Proof. The Euclidean inner product $\langle \mathcal{S}_{i_n=\alpha}, \mathcal{S}_{i_n=\beta} \rangle$ is given by

$$\langle \mathcal{S}_{i_n=\alpha}, \mathcal{S}_{i_n=\beta} \rangle \tag{4.1}$$

$$= \sum_{i_1} \sum_{i_2} \cdots \sum_{i_{k-1}} \sum_{i_{k+1}} \cdots \sum_{i_n} \sum_{j_k} \mathcal{S}_{i_1, i_2 \cdot \alpha \cdots i_n} \mathcal{S}^*_{i_1, i_2 \cdots \beta \cdots i_n} \tag{4.2}$$

$$= \sum_{i_1} \sum_{i_2} \cdots \sum_{i_{k-1}} \sum_{i_{k+1}} \cdots \sum_{i_n} \sum_{j_k} \left(\mathcal{A} \times_1 U^{(1)^T} \times_2 U^{(2)^T} \cdots \times_k U^{(k)^T} \cdots \times_n U^{(n)^T} \right)_{i_1, i_2, \cdots, \alpha, \cdots, i_n} \tag{4.3}$$

$$\times \left(\mathcal{A} \times_1 U^{(1)^T} \times_2 U^{(2)^T} \cdots \times_n U^{(n)^T} \cdots \times_N U^{(N)^T} \right)^*_{i_1, i_2, \cdots, \beta, \cdots i_n} \tag{4.4}$$

$$= \sum_{i_1} \sum_{i_2} \cdots \sum_{i_{k-1}} \sum_{i_{k+1}} \cdots \sum_{i_n} \sum_{j_k} \left(a_{i_1, i_2, \cdots, i_n} U^{(1)^T}_{j_1 i_1} U^{(2)^T}_{j_2 i_2} \cdots U^{(n-1)^T}_{J_{n-1} i_{n-1}} U^{(n)^T}_{j_n \alpha} U^{(k+1)^T}_{j_{k+1} i_{k+1}} \cdots U^{(n)^T}_{j_n i_n} \right) \tag{4.5}$$

$$\times \qquad \left(a^*_{i_1, i_2, \cdots, i_N} U^{(1)}_{j_1 i_1} U^{(2)}_{j_2 i_2} \cdots U^{(n-1)}_{J_{n-1} i_{n-1}} U^{(n)}_{j_n \alpha} U^{(n+1)}_{j_{n+1} i_{n+1}} \cdots U^{(N)}_{j_N i_N} \right) \tag{4.6}$$

$$= \sum_{i_1} \sum_{i_2} \cdots \sum_{i_{n-1}} \sum_{i_{n+1}} \cdots \sum_{i_N} \sum_{j_n} a_{i_1, i_2, \cdots, i_N} a^*_{i_1, i_2, \cdots, i_N} U^{(1)^T}_{j_1 i_1} U^{(1)}_{j_1 i_1} U^{(2)^T}_{j_2 i_2} U^{(2}_{j_2 i_2} \cdots \tag{4.7}$$

$$U^{(n-1)^T}_{j_{n-1} i_{n-1}} U^{(n-1)}_{j_{n-1} i_{n-1}} U^{(n)^T}_{j_n \alpha} U^{(n)}_{j_n \beta} \cdots U^{(N)^T}_{j_N i_N} U^{(N)}_{j_N i_N} \tag{4.8}$$

Since $U^{(n)}$ are orthogonal for all n, we have: $\sum_{j_n} U^{(n)^T}_{j_n \alpha} U^{(n)}_{j_n \beta} = 0$ for all $\alpha \neq \beta$. Hence $\langle \mathcal{S}_{i_n=\alpha}, \mathcal{S}_{i_n=\beta} \rangle = 0 \quad$ for all $\alpha \neq \beta$.

Let:

$$W^{(n)} = \left(U^{(n-1)} \otimes \cdots \otimes U^{(N)} \otimes U^{(1)} \otimes \cdots \otimes U^{(n-1)} \right)^T V^{(n)},$$

since $U^{(n)}$ And $V^{(n)}$ are orthogonal matrices for all n, it follows that W is an orthogonal matrix. Now we write

$$\mathbf{W} = \begin{bmatrix} W^{(1)} \\ W^{(2)} \\ \vdots \\ W^{(N)} \end{bmatrix},$$

where $W^{(i)}$ for all i are the row vectors. First we find that:

$$\begin{aligned} & \text{vec} \left[e^T_{\alpha, I_n} \Sigma^{(n)} \left(U^{(n+1)} \otimes ... \otimes U^{(N)} \otimes U^{(1)} \otimes \cdots \otimes U^{(n-1)} \right)^T V^{(n)} \right]^T \\ = \; & \text{vec} \left[e^T_{\alpha, I_n} \sum^{n} W \right]; \\ = \; & \text{vec} \left[\sigma_{(\alpha)} W^{(\alpha)} \right]^T; \\ = \; & \sigma_{(\alpha)} \text{vec} \left[W^{(\alpha)} \right]^T; \\ = \; & \sigma_{(\alpha)} W^{(\alpha)^T}. \end{aligned}$$

In the similar way, we have:

$$\operatorname{vec}\left[e^T_{\beta,I_n}\sum^n\left(U^{(n+1)}\otimes ...\otimes U^{(N)}\otimes U^{(1)}\otimes ...\otimes U^{(n-1)}\right)^T V^{(n)}\right]^T = \sigma_{(\alpha)}W^{(\beta)^T}.$$

♣

One has the same proposition for the SVD since any two columns of the singular diagonal are all orthogonal.

Proposition 4 *Similar to the decreasing order of singular values in the SVD, we find the decreasing order*

$$||\mathcal{S}_{i_k=1}|| \geq ||\mathcal{S}_{i_k=2}|| \geq ... \geq ||\mathcal{S}_{i_k=n}|| \geq 0,$$

for all $k \in [1;n]$, for the HOSVD.

This can then give us the room to use HOSVD for dimension reduction.

4.5 HOSVD and Dimension Reduction

When dealing with big data, the likelihood of having these stored as a tensor is very high. Hence one will need to do some preprocessing tensor decomposition work on those type of data before performing the initial task on hand. We will be interested on using the HOSVD to perform dimension reduction. This can be very helpful to reduce the space for the storage, to accelerate the running time of the algorithm and sometimes the performance of the algorithm. Some previous research have been done on this before [47, 76, 4, 61].

The goal of dimension reduction using HOSVD is to approximate a higher-dimensional tensor with a lower-dimensional representation while preserving as much as possible, the important information. This is achieved by retaining a subset of the most important modes and their associated factors.
Let us consider a tensor $\mathcal{A}$ of dimension $I \times I \times ... \times I_n$, where n is the number of dimensions. In the previous sections, we saw that HOSVD factorizes this tensor as:

$$\mathcal{A} = \mathcal{S} \times_1 U_{(1)} \times_2 U_{(2)} \times_3 \cdots \times_n U_{(n)},$$

where:
$\mathcal{S}$ is the core tensor of dimensions $R \times R \times \cdots \times R_n$ (where $R_i I_i$), representing the interactions between the factors.
U, U, $\cdots$, U_n are orthogonal matrices of dimensions $I_i \times R_i$, capturing the mode-specific information.
One may well use this equation to perform some dimension reduction in the tensor $\mathcal{A}$ ether by reducing the dimension of the core tensor $\mathcal{S}$ or by reducing the dimension of each factor matrix U_i.

4.5.1 Reducing the core tensor

To perform the dimension reduction of a tensor by reducing the dimension of the core tensor, we have to decrease the number of interactions captured between factor matrices. This can be done by fixing the limit value for all R_i by setting $R_i \leq I_i$ for all i. Since the core tensor is obtained from the U_i which are from the ith mode matrix unfolding.

This reduces the complexity of the decomposition, and results in a more compact representation of the data.

4.5.2 Reducing the dimension of the factor matrix

By reducing the dimensions of one or more factor matrices, we capture fewer features from certain modes, effectively reducing the dimensionality of the tensor. For example, if we set $R_k \leq I_k$ for some k, the mode-k factor matrix captures fewer features, leading to dimension reduction.
The challenge lies in selecting the appropriate dimensions to reduce while minimizing information loss. One common approach is to choose the top-k largest singular values for a certain mode's factor matrix. This retains the most important features in that specific mode. Hence leading to the most important information for that specific mode.

It is worth noticing that the degree of dimension reduction affects the quality of approximation.
We use Frobenius norm of the difference between the original tensor $\mathcal{A}$ and the reconstructed tensor $\hat{\mathcal{A}}$ as an approximation error metric.

4.5.3 Dimension reduction with HOSVD for face images

Taking a dataset of face images with multiple factor parameter, we load these face images in a tensor and use HOSVD to perform dimension reduction in these images.

Now your dataset consists of face images, represented as a tensor $\mathcal{A}$ of dimensions $N \times H \times W$, where N is the number of images, and H and W are the height and width of the images.

We use HOSVD to factorize the face image tensor by:

$$\mathcal{A} = \mathcal{S} \times_1 U_2 \times_2 U_2,$$

with:
$\mathcal{S}$ been the core tensor capturing inter-mode relationships.
U been the factor matrix for the mode-1 (N) interactions.
U being factor matrix for the mode-2 $(H \times W)$ interactions.

As we have demonstrated above, we can use this decomposition to do the reduction. One may do this by reducing the dimension of the core tensor $\mathcal{S}$ which will give a compact representation and reduce the complexity of the decomposition of the face images. The other way is to reduce the complexity of one or more mode.
In our case we use PCA in the U_i. For example, we apply PCA to the columns of U_1 and/or U_2, to retain the top-k principal components. Then use this to do the projection into the original tensor and find the approximation data.

4.6 HOSVD and face recognition

HOSVD can be very useful when we come across face recognition with the challenge of face variation, Which under normal situation will be the case.

In what follows, we will consider the situation where we are given a dataset of face images under variation of face pose, illumination and expression.
Our aims are to set this data as a tensor of images and use tensor decomposition in our case HOSVD to undergo a full process of face recognition.
We realise that when dealing with variations in face pose, illumination, and facial expressions, representing face images as a tensor, will be more complex. Applying Higher-Order Singular Value Decomposition (HOSVD) for face recognition with these variations has to be subdivided in many process.

4.6.1 Data Representation

The tensor face image now includes multiple modes: N (number of images), H (height),W (width), P (pose), I (illumination), and E (expression). The tensor dimensions would be $N \times H \times W \times P \times I \times E$.
Hence, our face images are going to be loaded in a tensor with dimension $N \times H \times W \times P \times I \times E$

4.6.2 Factorization

HOSVD Factorization:
Apply HOSVD to factorize the complex tensor face image:

$$\mathcal{A} = \mathcal{S} \times_1 U_1 \times_2 U_2 \times_3 U_3 \times_4 U_4 \times_5 U_5 \times_6 U_6,$$

with $\mathcal{S}$ the Core tensor capturing inter-mode relationships and
$U_1, U_2, \cdots, U_6$ the factor matrices for each mode (N, H, W, P, I, E) interactions.

4.6.3 Dimension Reduction

Using dimension reduction technique PCA in the factor matrices U_4, U_5, and U_6 we can handle variation in pose, illumination and expression.
This helps in retaining the most dominant variation within each mode.
We do this by choosing to retain the top-k principal components in each factor matrix.

4.6.4 Reconstruction

Before diver in to the recognition, one have to use the reduction factor matrices to reconstruct the tensor. We do this with the equation:

$$\hat{\mathcal{A}} = \hat{\mathcal{S}} \times_1 \hat{U}_1 \times_2 \hat{U}_2 \times_3 \hat{U}_3 \times_4 \hat{U}_4 \times_5 \hat{U}_5 \times_6 \hat{U}_6$$

where $\hat{\mathcal{A}}$ is the new tensor.
Using the reduced core reduced tensor $\hat{\mathcal{S}}$ and reduced factor matrices $\hat{U}_1, \hat{U}_2, \cdots, \hat{U}_6$.

4.6.5 Recognition

Now one can apply face recognition machine learning model to the reconstructed tensor to do the classification.

Then, we are able to load any given data set of faces as a tensor of face images of which any mode represents a specific variation (pose, facial expression, illumination just to name some of them), we can well apply a HOSVD on it and use a projection to perform a better recognition rate.

Hence, with the groundwork laid, we can now delve into the intricacies of developing a robust face recognition algorithm utilizing the power of High-Order Singular Value Decomposition (HOSVD). This sophisticated technique offers a multi-dimensional approach to analysing facial data, enabling us to extract meaningful features for recognition with enhanced accuracy and efficiency.

Algorithm 3 HOSVD for Face Recognition

```
1:  procedure FaceRecognition(HOSVDFaceRecognition)
2:      Input: multi factors face Dataset
3:      Output: RecognitionModel
4:      Data Preparation:
5:      load the face images in a tensor.
6:      Mode Unfolding:
7:      Unfold the tensor along each mode to create matrices.
8:      HOSVD Decomposition:
9:      Apply HOSVD to the tensor.
10:
            $$\text{HOSVD(Tensor } X) \Rightarrow \{U_1, U_2, \ldots, U_N, S\}$$
11:     Dimension Reduction:
12:     Select a rank $R_n$ for each mode.
13:
            $$U_n^\wedge = U_n(:, 1 : R_n), \quad S^\wedge = S(1 : R_1, 1 : R_2, \ldots, 1 : R_n)$$
14:     Recognition Model:
15:     Use reduced-rank factor matrices and the core tensor.
16:
            $$\text{Face}_i\text{representation} = U1^\wedge \cdot U2^\wedge \cdot \ldots \cdot Un^\wedge \cdot S^\wedge$$
17:     Recognition:
18:     For face recognition, compare representations of the test face with those of known faces
        in the reduced space.
19:     Evaluation:
20:     Evaluate the recognition accuracy.
21: end procedure
```

Chapter 5

Machine Learning Classification Methods and Metrics evaluation

5.1 Introduction

One of the fundamental aspects of facial recognition systems is their ability to classify and identify individuals accurately. This chapter, focus on traditional machine learning classification techniques applied to facial recognition as well as their evaluation metrics.

Traditional machine learning methods have long been the cornerstone of facial recognition systems, providing robust and interpretable solutions. We will be interested on a diverse set of classification algorithms, each with its unique strengths and characteristics. From linear models to probabilistic approaches and ensemble methods, we aim to provide a comprehensive understanding of the landscape of traditional machine learning techniques and their application in facial recognition scenarios.

Assessing the performance of different classification models in machine learning requires a careful consideration of various evaluation metrics. In the second part of this chapter, we exploit the mathematics, advantages, and disadvantages of key machine learning evaluation metrics commonly applied to facial recognition tasks.

5.2 Classification methods

The aims of any facial recognition application is to be able to identify an unknown face from a dataset. This task requires from a machine learning practitioner either to develop or to use a traditional classification technique. Here we will be interested on existing traditional machine learning classification models. Table 5.1 provides us with some of these models.
Moving forward, we will discus linear models such as Logistic Regression, Support Vector Machines (SVM), and Linear Discriminant Analysis (LDA). These models form the foundation of many facial recognition systems, leveraging mathematical principles to delineate boundaries between different facial features.

Moving beyond linearity, we will be interested on nearest neighbour models especially K-Nearest Neighbors (KNN). This algorithms rely on the proximity of data points to make predictions, offering

Group	**Classification Techniques**
Linear Models	Logistic Regression, Support Vector Machines (SVM), Linear Discriminant Analysis (LDA)
Nearest Neighbor Models	K-Nearest Neighbors (KNN)
Tree-Based Models	Decision Trees, Random Forest, Gradient Boosting Algorithms (e.g., XGBoost, AdaBoost)
Probabilistic Models	Naive Bayes
Neural Network Models	Multilayer Percetron

Table 5.1: Different Classification Techniques.

simplicity and effectiveness in facial recognition tasks.

Tree-based models, including Decision Trees, Random Forest and XGBoost Algorithms, present a powerful framework for capturing complex relationships within facial data. These ensembles of methods leverage the collective intelligence of multiple learners to enhance classification accuracy.

We also explore probabilistic model such as Naive Bayes, which provides insight into the statistical foundations of facial recognition. Additionally, we discuss the application of neural network models, Multilayer Perceptron.

5.2.1 Logistic Regression

Logistic Regression(LR) is a supervised learning technique that establishes a nonlinear relationship between training instances and their known labels [83]. Logistic regression is an older machine learning algorithm that has garnered significant attention from researchers due to its various applications. This algorithm has been used in the context of facial recognition problems in [71, 33] and [83]. Below we describe how logistic regression works with its mathematics details.
Given a dataset D of images with each image of size $M \times N$ pixels, we will have $M \times N$ features. Let's denote the feature vector for a single image as $x = (x_1, x_2, \ldots, x_{M \cdot N})$.

The logistic regression model then estimates the probability that an input image belongs to the positive class (contains a face). This probability is calculated using the logistic (sigmoid) function:

$$P(y = 1|x) = \frac{1}{1 + e^{-z}}.$$

Here, $P(y = 1|x)$ represents the probability that the image x contains a face, and z is the linear combination of the features and model parameters:

$$z = w_0 + w_1x_1 + w_2x_2 + \ldots + w_{M \cdot N}x_{M \cdot N}.$$

In the above equation, w_0 is the bias term, and $w_1, w_2, \ldots, w_{M \cdot N}$ are the weight parameters associated with each feature.
The goal during the training phase is to find the optimal values for the weight parameters w that minimize the logistic regression model's error. This is typically done by minimizing the cross-entropy loss (log loss) over the training dataset:

$$J(w) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log(\hat{y}^{(i)}) + (1-y^{(i)})\log(1-\hat{y}^{(i)})\right]$$

Here, m is the number of training examples, $y^{(i)}$ is the true label for the i-th training example, $\hat{y}^{(i)}$ is the predicted probability that $y^{(i)} = 1$ given the features $x^{(i)}$, and w are the weights of the logistic regression model.
The optimization algorithm (e.g., gradient descent) is used to find the values of w that minimize $J(w)$. After training, you can use the trained logistic regression model to make predictions on new, unseen images. For a given input image, you calculate the probability

$$P(y = 1|x)$$

using the model. If this probability is greater than a threshold (α), you classify the image as containing a face (1); otherwise, you classify it as not containing a face (0).

5.2.2 Support Vector Machine

The acronym SVM stands for Support Vector Machine. This is a supervised machine learning model widely used for classification problems, although it can also be applied to specific regression problems.

This algorithm creates a decision boundary line or hyperplane that segregates the dataset into n-classes in a way that allows any new data point to be easily assigned to the correct class.

This is achieved by identifying the extreme points (vectors) that assist in creating the hyperplane. These extreme points are referred to as support vectors.

Using the structural risk method, one can find the optimal line that separates the classes. This optimal line is determined through a combination of the weighted elements of the data.

The hyperplane is described by the equation:

$$WZ + b = 0$$

with

$$W = \sum_{i=1} N_S \alpha_i y_i S_i,$$

where we have N_S support vectors, denoted as S_i, α_i as the weighted coefficient, and b as the constant term. Note that our data is in the form of X_i, y_i, with X_i being the training data, and $y_i = -1$ or $y_i = 1$ representing the label.

Given a dataset of N face images, each image, say X_i, can be represented as a vector. Now, consider that we have P different persons; we can view this as a P-class problem. Fortunately, using the principle of one versus all, we can assume this is a binary classification problem where one class consists of a set of images of the same person, and the other class includes the images of the remaining people. We can then build P hyperplanes for the P different individuals. These hyperplanes can assist in face recognition, as they can be used as classifiers.

5.2.3 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is a machine learning algorithm widely used for multi-class data classification problems. It can be compared to logistic regression, which is a popular machine learning algorithm for single-class classification. For a detailed comparison of these algorithms, one can refer to [15] and [18].

LDA is also widely used for visualization problems. Lastly, it is one of the popular supervised machine learning algorithms for dimensionality reduction. LDA techniques for dimensionality reduction are applied in many fields, such as biometrics, bioinformatics, and chemistry, to name just a few.

Formulation of LDA for face recognition:

Let's consider a dataset of N images $x_1, x_2, \cdots, x_N$ with p classes $y_1, y_2, \cdots, y_P$. We can derive the following information from the dataset.

1. The mean of each class

$$\mu_i = \frac{1}{N_i}\sum_{k=1}^{N_i} x_k,$$

   where N_i is the number of images in class i and x_k are the images of the class i.

2. The mean of all dataset

$$\mu = \frac{1}{N}\sum_{k=1}^{N} x_k.$$

3. Scatter Matrices

   For class i we have

$$S_i = \sum_{k=1}^{N_i}(x_k - \mu_i)(x_k - \mu_i)^T.$$

   Within class scatter is given by

$$S_w = \sum_{i=1}^{p} S_i.$$

   Between class scatter is given by

$$S_b = \sum_{i=1}^{p} N_i(\mu_i - \mu)(\mu_i - \mu)^T.$$

Our goal now is to find the projection, denoted as W, which will transform any $x \in \mathbb{R}^m$ into a new space $z \in \mathbb{R}^n$. This can be expressed by the equation:

$$z = W^T x.$$

We can now have a new within class scatter and between class scatter as: $S_B = W^T S_b W$ and $S_W = W^T S_w W$. The optimal of our projection can then be defined as:

$$W_{opt} = argmax_W \frac{|S_B|}{|S_W|} \quad (5.1)$$

$$= argmax_W \frac{|W^T S_B W|}{|W^T S_W W|}. \quad (5.2)$$

$$(5.3)$$

By introducing Lagrange multipliers, we will have a generalized eigenvector problem with the following equation:

$$S_B W_i = \lambda_i S_W W_i.$$

5.2.4 K-Nearest Neighbour

K-Nearest Neighbor (KNN) is a machine learning algorithm that is used for both regression and classification. Like the Decision Tree, it is a non-parametric algorithm.

As we discussed previously, if X is the dataset comprising N images, we can represent each image X_i as a vector as follows:

$$\mathbf{X_i} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}.$$

Since KNN use the euclidean distance as the metric, let find the distance of two images X_i and X_j.

$$d(X_i, X_j) = \sqrt{\sum_{k=1}^{n} \left(x_k^i - x_k^j\right)^2}.$$

Given any image, we need to find its nearest neighbors in the dataset. We do this by fixing the number of neighbors that need to be saved for any image. This number is represented by K, hence the name K-Nearest Neighbor. Note that:

$$1 \leq K \leq N$$

Now that we can find the K nearest neighbors of each given image, we can use this to determine the class to which the image belongs and, consequently, perform face recognition. The class we select is the one that is dominant among the nearest neighbors that were selected.
Note that, as far as the choice of K is concerned, it is always recommended to choose an odd number for K.

5.2.5 Decision Tree

Decision Tree is a supervised machine learning model that can be used for regression, binary classification, as well as multi-class classification problems. Due to the fact that Decision Trees don't require any assumptions about the distribution of the independent variables or the functional relationships between them, one can view the Decision Tree model as a non-parametric algorithm.
We formulate Decision Trees as follows: Let $x_i \in R^n i = 1, \cdots, p$ be any training vector and $y \in R^p$ be a target vector associate with it.
A decision tree algorithm will recursively partition the feature of space in such a way that samples with same target values are put together.
Now, suppose the data at node m is represented by Q_m, and it has n_m samples. A tree consists of splitting any candidate $\theta = (y, t_m)$ where y is considered as a feature and t_m is the threshold at node m into $Q_m^{\text{left}}(\theta)$ and $Q_m^{\text{right}}(\theta)$ subsets. Typically, this is done with the decision boundary or threshold as follows:

$$Q_m^{left}(\theta) = (x, y)|x_y \leq t_m;$$

$$Q_m^{rigt}(\theta) = (x, y)|x_y \geq t_m.$$

We use an impurity function, denoted as $H()$, to compute the quality of a candidate split. This function is influenced by θ_m and the type of problem to be solved (regression or classification). It can be defined as

$$G(_m, \theta) = \frac{n_m^{left}}{n_m} H(Q_m^{left}(\theta)) + \frac{n_m^{right}}{n_m} H(Q_m^{right}(\theta)).$$

To optimize this, one has to choose the parameter that minimizes the impurity:

$$\theta^* = argmin_\theta G(Q_m, \theta).$$

This is done recursively for any subset $Q_m^{left}(\theta^*)$ and $Q_m^{rigth}(\theta^*)$ until we reach the maximum depth($nmin \leq min_sample$ or $n_m = 1$).
In our case, we are dealing with a classification problem; hence, the outcomes are taken from values $0, 1, \cdots, k-1$. For node m, we have:

$$P_{mk} = \frac{1}{m_n} \sum_{y \in Q_n} I(y = k),$$

as the proportion of class k observation in node m. If m is the terminal node, predict-probability for this region is set to p_{mk}.

We usually use one of these two impurity measures:
GINI:

$$H(Q_m) = \sum_k p_{mk}(1 - P_{mk}).$$

log loss or Entropy:

$$H(Q_m) = -\sum_k P_{mk} log(P_{mk}).$$

5.2.6 Random Forest

Random Forest (RF) is a supervised machine learning algorithm that shares the same fundamentals as decision trees. This algorithm constructs different decision trees on different samples and takes a majority vote to make classifications.
So, basically, given an image dataset D, RF works as follows:

1. Randomly split the data in to K subset.
2. Built decision trees in all the K subsets. We choose the feature that best separate the data base on some criterion such as GINI impurity or entropy.
3. Select the number of decision that you going to built
4. Repeat step 1 and 2.
5. Make the prediction with a new image using the majority vote principle on the multi decision trees.

Mathematically, this how the process can be describe.

1. Gini Impurity: Gini impurity measures the degree of disorder in a dataset. For a dataset D with K classes, the Gini impurity (Gini(D)) is calculated as:

$$Gini(D) = 1 - \sum_{i=1}^{k} (p_i)^2$$

   Where p_i is the proportion of samples in class i in dataset D.

2. Entropy: Entropy measures the level of uncertainty or disorder in a dataset. For a dataset D with K classes, the entropy (Entropy(D)) is calculated as:

$$Entropy(D) = -\sum_{i=1}^{k} p_i log_2(p_i)$$

   Where p_i is the proportion of samples in class i in dataset D.

3. Feature Selection: At each node of the decision tree, the algorithm selects the feature that minimizes the Gini impurity or entropy after the split. The exact formula for these calculations depends on the chosen criterion.

4. Voting: In the Random Forest ensemble, the final prediction for a test sample is made by aggregating the results of individual trees. This can be done by taking the mode of the predicted class labels from all the trees.

5. Bagging: Random Forest uses bootstrapped subsets of the training data to train individual trees. A bootstrapped sample is created by randomly selecting samples with replacement from the original dataset.

5.2.7 XGboost

XGBoost, which stands for Extreme Gradient Boosting, is a supervised machine learning method. This model uses trees to build the classifier. The trees are constructed sequentially in such a way that tree i reduces the errors of tree $i-1$. This means that each tree learns from the previous tree and updates the residual errors.
We can describe the boosting process as follows:

1. Initialise a model say F_0 to predict y given x with a residual say $(y - F_0(x))$.

2. Built a new model h_1 which is fitted to the residual of F_0.

3. Combining F_0 and h_1, one then finds the boosted model

$$F_1(x) \longleftarrow F_0(x) + h_1(x)$$

$$\sum \left(l\left(y_i, \hat{y}_i^{(t)}\right)\right) + \sum_{i=1}^{t} w\left(f_i\right).$$

We can do the boosting of F_1 to obtain F_2 by optimising the residual of F_1. One can then see that this process can go on to k iterations and we will have

$$F_k(x) \longleftarrow F_{k-1}(x) + h_k(x)$$

Note that in our loss function, we use Mean Square Error (MSE), resulting in exponential changes. Instead of fitting $h_m(x)$ to the residual, one can fit it to the gradient of the loss function or the step along which the loss occurs. This would make the process generic and applicable across all loss functions.
Gradient descent is very helpful when it comes to minimizing differentiable functions. The $h_k(x)$ predicts the mean residual at each terminal node of the tree. For gradient boosting, one computes the average gradient.
Now, we multiply $h_k(x)$ by a factor y, which helps account for the difference in the impact of each branch of the split. Unlike classical gradient descent, which reduces the error in the output at each iteration, gradient boosting assists in predicting the optimal gradient for the additive model.

The gradient boosting process is performed as follows:"

1. Initialize $F_0(x)$
$$F_0(x) = argmin \sum_{i=1}^{n} L\left(y_i, y\right)$$
2. Compute the gradient of the loss function
$$r_{ik} = -\alpha \left[\frac{\partial \left(L_{y_i}, F(x_i)\right)}{\partial F(x_i)} \right]$$
with α been the learning rate.
3. Feet $h_n(x)$ on the gradient obtained at each step.
4. The multiplication factor y_n for each terminal mode is derived and the boosted model $F_n(x)$ is defined
$$F_n(x) = F_{n-1}(x) + y_n h_n(x).$$

5.2.8 Naive Bayers

Naive Bayes (NB) is a classification machine learning model that is a collection of classification algorithms using Bayes' theorem. Bayes' theorem finds the probability of an event occurring given the probability of another event that has already occurred. The mathematical equation for Bayes' theorem is given as:
$$P(A|B) = \frac{P(B|A)P(A)}{P(B)},$$
where A and B are events and $P(B) \neq 0$
In this equation, we are finding the probability of event A knowing that event B is true. Hence, $P(A)$ is called the prior probability, and $P(A|B)$ is the posterior probability of B.
Now, let's consider a dataset of face images, denoted as $X = (x_1, x_2, x_3, \cdots, x_n)$, with labeled classes y for each person. Bayers theorem can be used as follows to find the probability of each class.
$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}.$$

Using the well known probability property of two independent evens A and B define as $P(A,B) = P(A)P(B)$ on the above equation, we will have:

$$\begin{aligned} P(y|(x_1, x_2, x_3, \cdots, x_n)) &= \sum_{i=1}^{n} P(y|x_i) && (5.4) \\ &= \frac{\sum_{i=1}^{n} P(x_i|y)}{\sum_{i=1}^{n} P(x_i)}. && (5.5) \end{aligned}$$

Since the denominator term of the equation will remain constant for all the input, one can remove it and get a simplify equation as follow

$$P(y|x_1, x_2, x_3, \cdots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y).$$

With this, we can now find the probability of any input for all the class variables y. Our classifier is simply the output with the highest probability, which is expressed by

$$y = agrmax_y P(y) \prod_{i=1}^{n} P(x_i|y).$$

Thus if one can calculate $P(y)$ and $P(X_i|y)$ then our classification problem is solved.
Using the Gaussian Naive bayers classifier, we will have a continuous values associated with each feature which is assumed to be distributed according to Gaussian distribution. Note that a Gaussian distribution is also called normal distribution and that his plot gives us a bell shape curve which is symmetric about the mean of the values.
Assuming that the likelihood of each feature is Gaussian, we can express our conditional probability with the equation:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp\left[-\frac{(x_i - \mu y)^2}{2\sigma_y^2} \right].$$

5.2.9 Multilayer Perceptron

Multilayer Perceptron (MLP) is a feedforward artificial neural network that uses backpropagation as its supervision method. To generate the output, this deep learning algorithm computes the weighted sum of the inputs plus the threshold weight. The equation is as follows:

$$z_i = \sum_{j=1}^{N} w_{ij} x_j + \theta_i.$$

Then the output of the neuron is given by Using the sigmoid function as the activation function:

$$y_i = \frac{1}{1 + \exp(-z_i)}.$$

The first layer is called the input layer, and the last layer is called the output layer. The output of the first layer is used as the input for the second layer, and its output is used as the input for the third layer, and so on until the last layer. The layers between the input layer and the output layer are referred to as hidden layers (in the case of three layers, we only have one hidden layer). Layers are connected from the lower layer to the upper layer, and there are no connections between neurons

within the same layer.
Now, let $f(x)$ be the activation function, w_{0j} be the synaptic weight of neuron j in the last hidden layer to the single output neuron 0, z_i be the i-th element of the input vector, z, and w_{ij} be the weight at layer i. The MLP will have the following equation:

$$F(z,w) = f\left(\sum_k w_{0k}\left(\sum_{jk} f\left(\cdots\left(\sum_{jl}\cdots\right)\right)\right)\right).$$

5.3 Evaluation Metrics in Facial Recognition

The efficacy of a facial recognition system is not solely determined by its ability to identify individuals correctly; rather, it extends to the system's capacity to navigate the intricate landscape of true positives, false positives, true negatives, and false negatives. As we navigate through the various evaluation metrics, we will encounter measures that emphasize precision, recall, accuracy, and discrimination capability. Each metric serves a unique purpose, offering a lens through which we can scrutinize different facets of a facial recognition model's performance.

5.3.1 Accuracy:

Facial recognition systems are tasked with the intricate challenge of identifying individuals based on facial features, making accuracy a pivotal metric in assessing their performance. Accuracy, in the context of facial recognition, is a fundamental metric that provides a global view of the system's correctness. It quantifies the proportion of correctly classified instances relative to the total number of instances, encapsulating both true positive and true negative predictions.

5.3.1.1 Mathematics of Accuracy

mathematically it is define by:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}.$$

For better understanding of this, we divide the prediction result in four classes namely:
True Positives(TP) which is the positive classes that are correctly predicted as positive.
False Positives(FP) that is a negative classes that are falsely predicted as positive.
True Negatives(TN) is a negative classes that are correctly predicted as negative. False Negatives(FN)which is positive classes that are falsely predicted as negative.
The accuracy score is then given by the formula:

$$\text{Accura}y = \frac{TP+TN}{TP+FP+TN+FN}.$$

5.3.1.2 Advantage of Accuracy

Accuracy metric has two majeure advantage which are:

1. Intuitiveness

Accuracy provides an intuitive and easily understandable measure of overall correctness. It resonates well with stakeholders and non-technical audiences, offering a clear and concise evaluation of the system's performance.

2. Comprehensive View

   As a global metric, accuracy offers a comprehensive view of the facial recognition system's ability to make correct predictions across all classes. It considers both positive and negative classifications, providing a balanced perspective.

5.3.1.3 Disadvantage of Accuracy

This metric evaluation also has its on downfall that are:

1. Sensitivity to Class Imbalance

   Accuracy may not be suitable for datasets with imbalanced class distributions. In scenarios where one class significantly outnumbers the others (e.g., a small number of positive instances in a large pool of negatives), accuracy might be skewed, as the model could achieve high accuracy by predominantly predicting the majority class.

2. Limited Information on Misclassifications

   While accuracy indicates the overall correctness, it does not provide insights into the types of errors made by the model. Misclassifications, whether false positives or false negatives, are treated equally in the accuracy calculation, potentially masking critical aspects of performance.

When interested of the accuracy of positive instances, accuracy metric will not be suitable and hence forcing one to look at Precision metric which will be the subject of the next section.

5.3.2 Precision:

Precision is a crucial metric in the realm of facial recognition, reflecting the system's ability to accurately identify positive instances. In the context of facial recognition, precision assesses the proportion of correctly identified individuals among those predicted as positive. It is particularly valuable when the cost associated with false positives is high, as precision provides insights into the accuracy of positive predictions.

5.3.2.1 Mathematics of Precision

Precision is calculated by dividing the number of true positives by the sum of true positives and false positives. This metric highlights the accuracy of positive predictions and serves as a valuable indicator of the system's precision in recognizing individuals of interest. Its formula is

$$\text{Precision} = \frac{TP}{TP + FP}.$$

5.3.2.2 Advantage of Precision

Precision has two important advantage that are:

1. Emphasis on Positive Predictions

   Precision focuses specifically on positive predictions, making it a targeted metric for evaluating the accuracy of identified individuals. This is especially crucial in applications where the consequences of false positives are significant, such as security or access control.

2. Useful in Imbalanced Datasets

   In scenarios where positive instances (e.g., individuals of interest) are rare compared to the negative instances (non-matching individuals), precision becomes particularly valuable. It provides an effective measure of the system's ability to make accurate positive predictions despite imbalanced class distributions.

5.3.2.3 Disadvantages of Precision

This metric evaluation also face some challenges like:

1. Neglects False Negatives

   Precision does not account for instances where positive individuals are incorrectly classified as negative (false negatives). In situations where the cost of missing positive instances is high, precision alone may not provide a comprehensive assessment.

2. Sensitivity to Class Imbalance

   Similar to accuracy, precision can be influenced by imbalanced datasets. In cases where the number of negative instances significantly outweighs positive instances, achieving a high precision may be easier, but it might not reflect the overall system performance.

   Precision stands as a key metric in evaluating the accuracy of positive predictions in facial recognition. Its emphasis on minimizing false positives makes it particularly relevant in applications where the cost of misidentification is high. However, a holistic evaluation, considering multiple metrics and understanding the interplay with recall, ensures a nuanced understanding of the facial recognition system's performance in real-world scenarios.

5.3.3 Recall (Sensitivity):

Recall, also known as sensitivity or true positive rate, is a critical metric in the evaluation of facial recognition systems. It measures the system's ability to correctly identify all positive instances, emphasizing the avoidance of false negatives. In the context of facial recognition, recall is particularly relevant when the cost of missing positive identifications is high, such as in security applications.

5.3.3.1 Mathematics of Recall

Recall is calculated by dividing the number of true positives (correctly identified individuals) by the sum of true positives and false negatives (positive individuals incorrectly identified as negative). This metric provides insights into the system's sensitivity to correctly identifying individuals of interest. The formula is

$$\text{Recall} = \frac{TP}{TP + FN}.$$

5.3.3.2 Advantage of Recall

Recall has the following advantage

1. Emphasis on Positive Instances

   Recall focuses on the system's ability to capture all positive instances, making it a crucial metric in applications where missing positive identifications is costly.

2. Suitability for Imbalanced Datasets

   In scenarios where positive instances (individuals of interest) are rare compared to negative instances, recall remains robust. It ensures that the system is effective in identifying positive instances even in the presence of imbalanced class distributions.

5.3.3.3 Disadvantage of Recall

This metric evaluation also has some downfall which are:

1. Potential Increase in False Positives
   While recall minimizes false negatives, it does not consider the number of false positives. In situations where the cost of false positives is high, a high recall may be achieved at the expense of an increase in false positives.

2. Trade-off with Precision

   Recall is often in tension with precision, forming a trade-off relationship. A system can achieve high recall by being less strict in its positive identifications, potentially leading to more false positives.

Recall plays a pivotal role in evaluating the effectiveness of a facial recognition system in capturing all positive instances. Its emphasis on minimizing false negatives makes it particularly relevant in scenarios where missing positive identifications has significant consequences. However, a holistic evaluation, considering multiple metrics and understanding the trade-offs with precision, is essential for a nuanced assessment of a facial recognition system's performance in real-world applications.

5.3.4 Specificity

Specificity is a crucial metric in the evaluation of facial recognition systems, particularly when the emphasis is on correctly identifying negative instances. It measures the system's ability to accurately reject non-matching individuals, offering insights into the model's performance in scenarios where false positives carry significant consequences.

5.3.4.1 Mathematics of Specificity

Specificity is calculated by dividing the number of true negatives by the sum of true negatives and false positives. This metric provides a measure of the system's precision in correctly rejecting non-matching instances. Its formula is

$$\frac{TN}{TN+FP}.$$

5.3.4.2 Advantage of Specificity

This metric present the following advantage.

1. Focus on Negative Predictions

   Specificity is designed to emphasize the accuracy of negative predictions, making it particularly relevant in applications where the consequences of false positives are significant.

2. Complement to Sensitivity

   While recall (sensitivity) focuses on the accurate identification of positive instances, specificity complements this by highlighting the system's proficiency in correctly rejecting negative instances.

5.3.4.3 Disadvantage of Specificity

Specificity has the below mentioned downfall.

1. Neglects False Negatives

   Specificity does not consider false negatives, instances where non-matching individuals are incorrectly identified as matching. In situations where the cost of missing positive identifications is high, a high specificity may not necessarily indicate overall system effectiveness.

2. Sensitivity to Class Imbalance

   Similar to other metrics, specificity can be influenced by imbalanced datasets. In scenarios where the number of negative instances significantly outweighs positive instances, achieving a high specificity may be easier but might not reflect the overall system performance.

Specificity plays a pivotal role in the evaluation of facial recognition systems, offering a targeted assessment of the model's ability to accurately reject non-matching individuals. As a complement to sensitivity, specificity provides a balanced perspective on the system's overall performance. However, a holistic evaluation, considering multiple metrics and understanding the trade-offs, ensures a nuanced assessment of a facial recognition system's effectiveness in real-world applications.

5.3.5 F1 Score:

The F1 score is a harmonic mean of precision and recall, providing a balanced evaluation of a facial recognition system's performance. This metric is especially valuable in situations where there is a need to strike a balance between minimizing false positives and false negatives. The F1 score considers both precision and recall, offering a comprehensive assessment that is particularly relevant in applications where the consequences of misclassifications are significant.

5.3.5.1 Mathematics of F1 Score

The F1 score is mathematically defined as the harmonic mean of precision and recall:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

It combines the precision and recall metrics, providing a single value that reflects the system's ability to balance accurate positive predictions with the avoidance of false negatives.

5.3.5.2 Advantages of F1 Score

This metric evaluation is characterised by the following advantages:

1. Balanced Evaluation

   The F1 score provides a balanced evaluation by considering both false positives and false negatives. This is particularly beneficial in scenarios where achieving a balance between precision and recall is crucial.

2. Suitability for Imbalanced Datasets

   In facial recognition tasks with imbalanced class distributions, where positive instances are rare compared to negatives, the F1 score offers a robust measure that is not overly influenced by the majority class.

5.3.5.3 Disadvantages of F1 Score

Like all metric, F1 Score also has one downfall which is the problem of equal weighting of Precision and Recall.

The F1 score equally weights precision and recall, which may not be suitable in all scenarios. In cases where the importance of precision and recall differs, other metrics or weighted combinations may be more appropriate.

The F1 score serves as a valuable metric in the evaluation of facial recognition systems, offering a balanced perspective on precision and recall. Its application is particularly relevant in scenarios where achieving equilibrium between minimizing false positives and false negatives is paramount. However, practitioners should be mindful of the equal weighting of precision and recall and consider the specific context and consequences of misclassifications in their facial recognition application.

The choice of using the F1 score should align with the specific goals and requirements of the facial recognition application. If the consequences of false positives and false negatives are significantly different, practitioners may need to prioritize precision or recall over the other.

5.3.6 Confusion Matrix:

The confusion matrix is a foundational tool in the evaluation of facial recognition systems. It provides a detailed breakdown of the model's predictions, categorizing instances into true positives, true negatives, false positives, and false negatives. This matrix serves as the basis for deriving various performance metrics and offers valuable insights into the strengths and weaknesses of a facial recognition model.

5.3.6.1 Structure of Confusion matrix

The confusion matrix for facial recognition is organized as follows:

Actual	Non-Matching	Matching
	True Negatives (TN)	False Positives (FP)
Non-Matching	TN	FP
Matching	False Negatives (FN)	True Positives (TP)

Table 5.2: Confusion Matrix Structure

5.3.6.2 Advantage of Confusion Matrix

Since this metric can be useful in the calculation of other metrics, it provides the following advantages:

1. Granular Insights

   Offers a detailed breakdown of correct and incorrect predictions, providing granular insights into the facial recognition model's performance.

2. Foundation for Multiple Metrics

   Serves as the foundation for deriving various performance metrics, allowing practitioners to choose metrics aligned with specific goals.

5.3.6.3 Disadvantage of confusion Matrix

The confusion matrix may be challenging to interpret for those unfamiliar with its structure, requiring a nuanced understanding of true positives, true negatives, false positives, and false negatives.

The confusion matrix is a cornerstone in the evaluation toolkit for facial recognition systems. Its detailed breakdown of predictions enables practitioners to derive a spectrum of performance metrics, providing a comprehensive understanding of the model's strengths and weaknesses. By interpreting the confusion matrix, practitioners can make informed decisions to optimize the facial recognition system for specific application requirements.

5.4 conclusion

In this chapter, we comprehensively examine the machine learning classification techniques for the implementation of facial recognition and the evaluation metrics. This can equip practitioners with the knowledge needed to assess facial recognition models effectively. The choice of the classification technique as well as the metrics should align with the specific goals and challenges posed by the facial recognition task at hand.

Chapter 6

Conclusion

6.1 Summary of Findings

This thesis undertook a comprehensive investigation into the intricate realm of facial recognition, specifically focusing on the challenges posed by facial occlusion and the advancements in dimension reduction techniques. Through a meticulous exploration of mathematical principles and machine learning applications, the study aimed to contribute to the valuable insights in the field.

In addressing facial occlusion, particularly exacerbated by face masks during the COVID-19 pandemic, the research unveiled the significant impact on the performance of facial recognition systems. By meticulously analysing and developing datasets useful for the investigation on face mask, the study has laid the foundation for adaptive dimension reduction in face occlusion for facial recognition systems.

The exploration of dimension reduction techniques yielded noteworthy findings. Traditional methods such as Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) showcased their effectiveness, while Auto-Encoder-based techniques demonstrated their capacity for facial feature extraction and dimension reduction. The introduction of innovative hybrid techniques, namely PCA-Autoencoder and LDA-Autoencoder, provided a synergistic approach, leveraging the strengths of individual methods. Additionally, tensor decomposition, specifically Higher-Order Singular Value Decomposition (HOSVD), emerged as a novel perspective on dimension reduction, offering insights for further exploration.

6.2 Contributions to Knowledge

The contributions of this thesis extend to both theoretical advancements and practical applications in the realm of facial recognition:

6.2.1 Facial Occlusion

The study provides a nuanced understanding of the challenges introduced by facial occlusion, especially in the context of face masks during the COVID-19 pandemic and beyond. Techniques developed to mitigate these challenges offer practical solutions for enhancing the accuracy and reliability of facial recognition systems in real-world scenarios.

6.2.2 Dimension Reduction Techniques

Traditional dimension reduction techniques (PCA, LDA) have been comprehensively evaluated, providing a benchmark for their efficacy in facial recognition. The introduction and evaluation of Auto-Encoder-based techniques, along with innovative hybrid approaches (PCA-Autoencoder, LDA-Autoencoder), contribute to the diversification of dimension reduction methodologies.
Tensor decomposition (HOSVD) presents a novel avenue for dimension reduction, expanding the spectrum of mathematical techniques applied in facial recognition.

6.3 Limitations and Challenges

While this research has made significant strides, it is essential to acknowledge certain limitations and challenges. The diversity in facial occlusion scenarios and the evolving nature of face mask usage presents ongoing challenges. Additionally, the generalization of findings to broader demographics and the scalability of proposed techniques requires further scrutiny.

6.4 Recommendations for Future Research

Building upon the foundations laid by this dissertation, there are several avenues for future research:

6.4.1 Facial Occlusion

Further investigation into diverse facial occlusion scenarios, encompassing varying mask types and usage patterns.
Exploration of real-time adaptive systems that dynamically respond to changing facial occlusion conditions.
Development of non-artificial face mask datasets.

6.4.2 Dimension Reduction Techniques

Continued exploration of innovative hybrid techniques and tensor decomposition for dimension reduction.
Integration of deep learning architectures to enhance the capabilities of Autoencoder-based methods.
Extension of the study to large-scale datasets and diverse demographic groups for comprehensive validation.

6.5 Conclusion

In conclusion, this thesis marks a significant contribution to the intersection of mathematics and machine learning in the domain of facial recognition. By addressing the challenges posed by facial occlusion and advancing dimension reduction techniques, the study provides a holistic perspective on the complexities of real-world scenarios. The findings offer practical implications for the development of facial recognition systems that are not only accurate and reliable but also adaptable to the evolving challenges presented by facial occlusion. As technology continues to evolve, this research serves as a catalyst for further exploration and innovation in the field, fostering a deeper understanding of the intricate dynamics of facial recognition.

Appendix A

Python Code

A.1 Data Augmentation

The following code uses the data augmentation technique that we described in chapter two to generate more faces for our created dataset.
The full code can be provided upon request.

```
from keras.preprocessing.image import ImageDataGenerator
from skimage import io
import numpy as np
import os
from PIL import Image
datagen = ImageDataGenerator(
rotation_range = 25,
shear_range = None,
zoom_range = 0.2,
brightness_range
image_directory = r'C:/Users/ADMIN/Documents/mixed_data_images/'

from keras.preprocessing.image import ImageDataGenerator
from skimage import io
import numpy as np
import os
from PIL import Image
datagen = ImageDataGenerator(
rotation_range = 25,
shear_range = None,
zoom_range = 0.2,

dataset = []
my_images = os.listdir(image_directory)
for i, image_name in enumerate(my_images):

image = io.imread(image_directory + image_name)
image = Image.fromarray(image, 'RGB')

dataset.append(np.array(image))
x = np.array(dataset)
i = 0
for batch in datagen.flow(x, batch_size=16,
save_to_dir= r'C:\Users\ADMIN\Documents\mixed_data_images_augmented',
save_prefix='image',
save_format='jpg'):
i += 1

break
print('check the new folder')
```

A.2 Converting data to npy format

```
import cv2
import glob
import numpy as np
import array as arr
#Train data
data_array=np.array('d',)
train = []
train_reshape = []
files = glob.glob ("C:/Users/ADMIN/Documents/Masked_face_data/*.jpg") # your image path
for myFile in files:
image = cv2.imread (myFile)
image_resize=cv2.resize(image,(120,120))

image_array=np.array(image_resize)
image_reshape=np.reshape(image_array, image_array.shape[0]*image_array.shape[1])
train.append (image_array)
train_reshape.append(image_reshape)
train_array=np.append(data_array, image_reshape)
print(image_array.shape)
print(image_reshape.shape)
pyplot.imshow(image_array)

pyplot.show()

'''files = glob.glob ("C:/Users/ADMIN/Documents/Masked_face_data/*.jpg")
for myFile in files:
image = cv2.imread (myFile)
train.append (image)
```

```
train_labels.append([0., 1.])
'''

#train_labels = np.array(train_labels,dtype='float64') #as mnist
#print(train.shape)
# convert (number of images x height x width x number of channels) to (number of
#train = np.reshape(train,train.[shape[0]*train.shape[1])
#train = np.reshape(train.shape[0]*train.shape[1])

# save numpy array as .npy formats
np.save('train',train)
np.save('train_reshape',train_reshape)
np.save('train_array', train_array)

#Test data
face_data = []
test_labels = []
files = glob.glob ("C:/Users/ADMIN/Desktop/face_data/*.jpg")
for myFile in files:
image_2 = cv2.imread (myFile)
image_2_resize=cv2.resize(image_2,(60,60))
image_2_resize = cv2.cvtColor(image_2_resize, cv2.COLOR_BGR2GRAY)
image_2_array=np.array(image_2_resize)
image_2_reshape=np.reshape(image_2_array,
face_data_reshape=np.append(image_2_reshape, image_reshape)

'''files = glob.glob ("/data/test/class2/*.png")
test.append (image)
pyplot.imshow(image_data_array)
pyplot.show()
print(image_data_array.shape)

test_labels = np.array(test_labels,dtype='float64') #as mnist
#test = np.reshape(test,[test.shape[0],test.shape[1]*test.shape[2]*test.shape[3]])

# save numpy array as .npy formats
np.save('data_face',face_data) # saves test.npy
np.save('face_data_reshape',face_data_reshape)
np.save('face_data_array', face_data_array)
print('completed')
print(face_data[0].shape)
image2=Image.fromarray(train[395])
#print(image2.shape)

pyplot.show()
```

```
#create the target of the data
data_target = []
for i  in range (1, 41):
for j in range (1, 11):
data_target.append(i)
#print(data_target)
len(data_target)

np.save('data_target', data_target)

face_data_target = []
for i  in range (1, 41):
for j in range (1, 31):
face_data_target.append(i)
#print(face_data_target)
print(len(face_data_target))

np.save('face_data_target', face_data_target)
```

Here we import the library for the face recognition after PCA dimensional reduction

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
import numpy as np
import pandas as pd
import os
from  matplotlib import pyplot as plt
import matplotlib.image as mpimg
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from numpy import loadtxt
from xgboost import XGBClassifier
from sklearn.neural_network import MLPClassifier
from sklearn import metrics
import plotly.express as px
from IPython.display import display
```

Here We load the file from the local coal machine directory.

```
from google.colab import files

uploaded = files.upload()

#for fn in uploaded.keys():
# print('User uploaded file "{name}" with length {length} bytes'.format(
#name=fn, length=len(uploaded[fn])))

unmasked_face_data=np.load("data_images.npy")
target=np.load("data_images_target.npy")
masked_face_data=np.load("masked_data_images.npy")
mixed_face_data=np.load("mixed_data_images.npy")
import pylab
#pylab.imshow(unmasked_face_data[9,:,:])
#pylab.imshow(masked_face_data[9,:,:])
pylab.imshow(mixed_face_data[459,:,:])
print(target[9])
print(unmasked_face_data.shape)
print(masked_face_data.shape)
print(mixed_face_data.shape)
print(target.shape)


from sklearn.model_selection import train_test_split
X_train1, X_test1, y_train1, y_test1=train_test_split(data_reshape, target, test_size=0.3, str
print(X_train1.shape, y_train1.shape, X_test1.shape, y_test1.shape)
```

A.3 PCA for dimension Reduction on the face mask data

```
from sklearn.decomposition import PCA

for n in range(1, 616, 10):

pca = PCA(n_components=n, whiten=True)
pca.fit(X_train1)

X_train1_pca = pca.transform(X_train1)
X_test1_pca = pca.transform(X_test1)
```

Appendix B

Classification process

```
#### Decision treesimport plotly.express as px

dtree = DecisionTreeClassifier()
dtree.fit(X_train1_pca, y_train1)
#z_dt = dtree.predict(X_train_pca)
ac_dt = dtree.score(X_test1_pca, y_test1)

list_dt1.append(ac_dt)
#print(X_train_pca.shape)
#print(y_train.shape)

#np.save("list_dt", list_dt)
#### KNeighbors classifier

knn = KNeighborsClassifier(n_neighbors = 3, metric = 'minkowski', p = 2)
knn.fit(X_train1_pca, y_train1)
#z_KNN = KNN.predict(X_train_pca)
ac_knn = knn.score(X_test1_pca, y_test1)

list_knn1.append(ac_knn)
#np.save("list_dt", list_knn)
#### Random forest

from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(n_estimators=200)
rf.fit(X_train1_pca, y_train1)
ac_rf = rf.score(X_test1_pca, y_test1)

list_rf1.append(ac_rf)
#np.save("list_dt", list_rf)
#### svm
```

```
from sklearn.ensemble import RandomForestClassifier

svm = SVC()
svm.fit(X_train1_pca, y_train1)
ac_svm = svm.score(X_test1_pca, y_test1)

list_svm1.append(ac_svm)
#np.save("list_dt", list_svm)
#### XBoot


xboost = XGBClassifier()
xboost.fit(X_train1_pca, y_train1)
ac_xboost = xboost.score(X_test1_pca, y_test1)
list_xboost1.append(ac_xboost)
#np.save("list_dt", list_xboost)
#### lda


lda = LinearDiscriminantAnalysis()
lda.fit(X_train1_pca, y_train1)
ac_lda = lda.score(X_test1_pca, y_test1)
list_lda1.append(ac_lda)
#np.save("list_dt", list_lda)
#### lgr


lgr = LogisticRegression()
lgr.fit(X_train1_pca, y_train1)
ac_lgr = lgr.score(X_test1_pca, y_test1)
list_lgr1.append(ac_lgr)
#np.save("list_dt", list_lgr)
#### nvb


nvb = GaussianNB()
nvb.fit(X_train1_pca, y_train1)
ac_nvb = nvb.score(X_test1_pca, y_test1)
list_nvb1.append(ac_nvb)
#np.save("list_dt", list_nvb)
#### mlp


mlp = MLPClassifier()
```

```
mlp.fit(X_train1_pca, y_train1)
ac_mlp = mlp.score(X_test1_pca, y_test1)
 list_mlp1.append(ac_mlp)
#np.save("list_dt", list_mlp)

#This code use PCA with the olivetti dataset
list_dt_olivetti = []
list_knn_olivetti = []
list_rf_olivetti = []
list_svm_olivetti = []
list_xboost_olivetti = []
list_lda_olivetti = []
list_lgr_olivetti = []
list_nvb_olivetti = []
list_mlp_olivetti = []

# PCA decomposition and shifting

from sklearn.decomposition import PCA

for n in range(1, 120, 10):

pca = PCA(n_components=n, whiten=True)
pca.fit(X_train1)

X_train1_pca = pca.transform(X_train1)
X_test1_pca = pca.transform(X_test1)

#### Decision treesimport plotly.express as px





knn = KNeighborsClassifier(n_neighbors = 3, metric = 'minkowski', p = 5)

ac_knn = knn.score(X_test1_pca, y_test1)

list_knn_olivetti.append(ac_knn)
#np.save("list_dt", list_knn)
#### Random forest

from sklearn.ensemble import RandomForestClassifier

rf)
```

```
#### svm

from sklearn.ensemble import RandomForestClassifier

svm = SVC()
svm.fit(X_train1_pca, y_train1)
ac_svm = svm.score(X_test1_pca, y_test1)




lda = LinearDiscriminantAnalysis()

list_lda_olivetti.append(ac_lda)
#np.save("list_dt", list_lda)
#### lgr


ti.append(ac_lgr)
#np.save("list_dt", list_lgr)
#### nvb



ac_nvb = nvb.score(X_test1_pca, y_test1)

#np.save("list_dt", list_nvb)
#### mlp
mlp = MLPClassifier()
mlp.fit(X_train1_pca, y_train1)
```

Bibliography

[1] A.H. Phan C. Caiafa G. Zhou Q. Zhao A. Cichocki, D. Mandic and L. de Lathauwers. Tensor decomposition for signal processing applications. *IEEE Signal Processing Magazine*, 21(4), 2000.

[2] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Abubakar Malah Umar, Okafor Uchenwa Linus, Humaira Arshad, Abdullahi Aminu Kazaure, Usman Gana, and Muhammad Ubale Kiru. Comprehensive review of artificial neural network applications to pattern recognition. *IEEE access*, 7:158820–158846, 2019.

[3] Insaf Adjabi, Abdeldjalil Ouahabi, Amir Benzaoui, and Abdelmalik Taleb-Ahmed. Past, present, and future of face recognition: A review. *Electronics*, 9(8):1188, 2020.

[4] Salman Ahmadi-Asl, Cesar F Caiafa, Andrzej Cichocki, Anh Huy Phan, Toshihisa Tanaka, Ivan Oseledets, and Jun Wang. Cross tensor approximation methods for compression and dimensionality reduction. *IEEE Access*, 9:150809–150838, 2021.

[5] M.M. Bronstein A.M. Bronstein and R. Kimmel. Facial feature detection and face recognition from 2d and 3d images. *Pattern Recognition Letters*, 23(10), 2005.

[6] A Anwar and A Raychowdhury. Masked face recognition for secure authentication. arxiv 2020. *arXiv preprint arXiv:2008.11104*.

[7] Elnaz Barshan, Ali Ghodsi, Zohreh Azimifar, and Mansoor Zolghadri Jahromi. Supervised principal component analysis: Visualization, classification and regression on subspaces and submanifolds. *Pattern Recognition*, 44(7):1357–1371, 2011.

[8] Mohammad Bataineh and Timothy Marler. Neural network for regression problems with reduced training sets. *Neural networks*, 95:1–9, 2017.

[9] Casey Battaglino, Grey Ballard, and Tamara G Kolda. A practical randomized cp tensor decomposition. *SIAM Journal on Matrix Analysis and Applications*, 39(2):876–901, 2018.

[10] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[11] Christopher M Bishop. Neural networks: a pattern recognition perspective. In *Handbook of neural computation*, pages B6_1–B6. CRC Press, 2020.

[12] Jerome Brachat, Pierre Comon, Bernard Mourrain, and Elias Tsigaridas. Symmetric tensor decomposition. *Linear Algebra and its Applications*, 433(11-12):1851–1872, 2010.

[13] A.M. Bronstein, M.M. Bronstein, and R. Kimmel. Three dimentiona face recognition. *International Journal of Computer Vision*, 61(1), 2005.

[14] TTT Bui, NH Phan, and VG Spitsyn. Face and hand gesture recognition algorithm based on wavelet transforms and principal component analysis. In *2012 7th International Forum on Strategic Technology (IFOST)*, pages 1–4. IEEE, 2012.

[15] Şener Büyüköztürk and Ömay Çokluk-Bökeoğlu. Discriminant function analysis: Concept and application. *Eurasian Journal of Educational Research (EJER)*, (33), 2008.

[16] Guoqing Chao, Yuan Luo, and Weiping Ding. Recent advances in supervised dimension reduction: A survey. *Machine learning and knowledge extraction*, 1(1):341–358, 2019.

[17] Andrzej Cichocki, Danilo Mandic, Lieven De Lathauwer, Guoxu Zhou, Qibin Zhao, Cesar Caiafa, and Huy Anh Phan. Tensor decompositions for signal processing applications: From two-way to multiway component analysis. *IEEE signal processing magazine*, 32(2):145–163, 2015.

[18] Jacob Cohen, Patricia Cohen, Stephen G West, and Leona S Aiken. *Applied multiple regression/correlation analysis for the behavioral sciences.* Routledge, 2013.

[19] Fengyu Cong, Qiu-Hua Lin, Li-Dan Kuang, Xiao-Feng Gong, Piia Astikainen, and Tapani Ristaniemi. Tensor decomposition of eeg signals: a brief review. *Journal of neuroscience methods*, 248:59–69, 2015.

[20] Subrat Kumar Dash, Sanjay Rawat, and Arun K Pujari. Use of dimensionality reduction for intrusion detection. In *International Conference on Information Systems Security*, pages 306–320. Springer, 2007.

[21] Inger Teixeira de Campos Tuñas, Eduarda Teodoro Da Silva, SB Santoro Santiago, Katlin Darlen Maia, and Geraldo Oliveira Silva-Júnior. Coronavirus disease 2019 (covid-19): A preventive approach to dentistry. *Rev Bras Odontol*, 77:E1766, 2020.

[22] H. F. de Groote. *Lectures on the complexity of bilinear problems.* Springer-verlag, 1985.

[23] Nizam Ud Din, Kamran Javed, Seho Bae, and Juneho Yi. A novel gan-based network for unmasking of masked face. *IEEE Access*, 8:44276–44287, 2020.

[24] Stephan Dreiseitl and Lucila Ohno-Machado. Logistic regression and artificial neural network classification models: a methodology review. *Journal of biomedical informatics*, 35(5-6):352–359, 2002.

[25] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL: `http://archive.ics.uci.edu/ml`.

[26] Md Sabbir Ejaz, Md Rabiul Islam, Md Sifatullah, and Ananya Sarker. Implementation of principal component analysis on masked and non-masked face recognition. In *2019 1st international conference on advances in science, engineering and robotics technology (ICASERT)*, pages 1–5. IEEE, 2019.

[27] Raphael Féraud and Fabrice Clérot. A methodology to explain neural network classification. *Neural networks*, 15(2):237–246, 2002.

[28] Vladimir Filipović. Optimization, classification and dimensionality reduction in biomedicine and bioinformatics. *Biologia Serbica*, 39(1), 2017.

[29] Shiming Ge, Jia Li, Qiting Ye, and Zhao Luo. Detecting masked faces in the wild with lle-cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2682–2690, 2017.

[30] Rucha Golwalkar and Ninad Mehendale. Masked-face recognition using deep metric learning and facemasknet-21. *Applied Intelligence*, pages 1–12, 2022.

[31] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[32] Guy Mathias Gouaya. *Algebraic and multilinear-algebraic techniques for fast matrix multiplication*. PhD thesis, 2015.

[33] Mahesh M Goyani and Narendra Manorbhai Patel. Multi-level haar wavelet based facial expression recognition using logistic regression. *International Journal of Next-Generation Computing*, pages 131–151, 2018.

[34] Guodong Guo and Na Zhang. A survey on deep learning based face recognition. *Computer vision and image understanding*, 189:102805, 2019.

[35] Yorick Hardy and Willi-hans Steeb. *Problems and Solutions in Introductory and Advanced Matrix Calculus*. World Scientific Publishing Company, 2016.

[36] W Hariri. Efficient masked face recognition method during the covid-19 pandemic (2020). *PREPRINT (Version 1) available at Research Square, DOI: https://doi. org/10.21203/rs*, 3.

[37] Susith Hemathilaka and Achala Aponso. A comprehensive study on occlusion invariant face recognition under face mask occlusion. *arXiv preprint arXiv:2201.09089*, 2022.

[38] David Hong, Tamara G Kolda, and Jed A Duersch. Generalized canonical polyadic tensor decomposition. *SIAM Review*, 62(1):133–163, 2020.

[39] Baojin Huang, Zhongyuan Wang, Guangcheng Wang, Kui Jiang, Zheng He, Hua Zou, and Qin Zou. Masked face recognition datasets and validation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1487–1491, 2021.

[40] Shiping Huang, M.O. Ward, and E.A. Rundensteiner. Exploration of dimensionality reduction for text visualization. In *Coordinated and Multiple Views in Exploratory Visualization (CMV'05)*, pages 63–74, 2005. `doi:10.1109/CMV.2005.8`.

[41] R. Jafri and R.A. Hamid. A survey of face recognition techniques. *Journal of Information Processing Systems*, 5(2), 2009.

[42] Eric Kernfeld, Misha Kilmer, and Shuchin Aeron. Tensor–tensor products with invertible linear transforms. *Linear Algebra and its Applications*, 485:545–570, 2015.

[43] Eric Kernfeld, Misha Kilmera, and Shuchin Aeronb. Tensor-tensor products with invertible linear transforms.

[44] Insook Kim and Dongwoo Kim. Optimal design of extended slot allocation for multiuser relay networks. In *2012 1st IEEE International Conference on Communications in China (ICCC)*, pages 475–480. IEEE, 2012.

[45] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[46] Sotiris B Kotsiantis, Ioannis Zaharakis, P Pintelas, et al. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160(1):3–24, 2007.

[47] Liwei Kuang, Fei Hao, Laurence T Yang, Man Lin, Changqing Luo, and Geyong Min. A tensor-based approach for big data representation and dimensionality reduction. *IEEE transactions on emerging topics in computing*, 2(3):280–291, 2014.

[48] S. Kumar and H. Kaur. Face recognition techniques: Classification and comparisons. *International Journal of Information Technology and Knowledge Management*, 5(2), 2012.

[49] Sirovich L and Kirby M. Low-dimensional procedure for the characterization of human faces. *Journal of the optimal Society of America*, 4:519 – 524, 1987.

[50] l. Lita and E. Pelican. A low-rank tensor-based algorithm for face recognition.

[51] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000. URL: `http://dx.doi.org/10.1137/S0895479896305696`, `arXiv:http://dx.doi.org/10.1137/S0895479896305696`, `doi:10.1137/S0895479896305696`.

[52] S. Lemm, G. Curio, Y. Hlushchuk, and K.-R. Muller. Enhancing the signal-to-noise ratio of ica-based extracted erps. *IEEE Transactions on Biomedical Engineering*, 53(4):601–607, 2006. `doi:10.1109/TBME.2006.870258`.

[53] M.A.O.Vasilescu and D. Terzopoulos. Multilinear analysis of image ensembles: Tensorfaces. *IEEE conference on computer Vision and Pattern Recognition (CVPR'03)*, 2003.

[54] Aleix Martinez and Robert Benavente. The ar face database: Cvc technical report, 24. 1998.

[55] Turk Matthew and Pentland Alex. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71 – 86, 1991.

[56] Munir, Erna Piantari, and Fauzi Nur Firman. Dimensional reduction in behavioral biometrics authentication system. In *Proceedings of ICETIT 2019: Emerging Trends in Information Technology*, pages 984–992. Springer, 2020.

[57] Salifu Nanga, Ahmed Tijani Bawah, Benjamin Ansah Acquaye, Mac-Issaka Billa, Francis Delali Baeta, Nii Afotey Odai, Samuel Kwaku Obeng, and Ampem Darko Nsiah. Review of dimension reduction methods. *Journal of Data Analysis and Information Processing*, 9(3):189–231, 2021.

[58] James O' Neill. An overview of neural network compression. *arXiv preprint arXiv:2006.03669*, 2020.

[59] Mei L Ngan, Patrick J Grother, Kayee K Hanaoka, et al. Ongoing face recognition vendor test (frvt) part 6b: Face recognition accuracy with face masks using post-covid-19 algorithms. 2020.

[60] P.J. Michaels D.M. Blackburn M. Tabassi P.J. Phillips, R. Grother and M. Bone. A survey of face recognition techniques. *Face recognition vendor test. Analysis and Modeling of Faces and Gestures*, 2003.

[61] Nadine Renard and Salah Bourennane. Dimensionality reduction based on tensor modeling for classification methods. *IEEE Transactions on Geoscience and Remote Sensing*, 47(4):1123–1131, 2009.

[62] Miguel Rocha, Paulo Cortez, and José Neves. Evolution of neural networks for classification and regression. *Neurocomputing*, 70(16-18):2809–2816, 2007.

[63] Irene Rodriguez-Lujan, Gonzalo Bailador, Carmen Sanchez-Avila, Ana Herrero, and Guillermo Vidal-de Miguel. Analysis of pattern recognition and dimensionality reduction techniques for odor biometrics. *Knowledge-Based Systems*, 52:279–289, 2013.

[64] Alla Safonova, Jessica K Hodgins, and Nancy S Pollard. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Transactions on Graphics (ToG)*, 23(3):514–521, 2004.

[65] Nicholas D Sidiropoulos, Lieven De Lathauwer, Xiao Fu, Kejun Huang, Evangelos E Papalexakis, and Christos Faloutsos. Tensor decomposition for signal processing and machine learning. *IEEE Transactions on signal processing*, 65(13):3551–3582, 2017.

[66] Ksh Nareshkumar Singh, S Dickeeta Devi, H Mamata Devi, and Anjana Kakoti Mahanta. A novel approach for dimension reduction using word embedding: An enhanced text classification approach. *International Journal of Information Management Data Insights*, 2(1):100061, 2022.

[67] Narendra Singh, Pushpa Singh, and Mukul Gupta. An inclusive survey on machine learning for crm: a paradigm shift. *Decision*, 47(4):447–457, 2020.

[68] Lingxue Song, Dihong Gong, Zhifeng Li, Changsong Liu, and Wei Liu. Occlusion robust face recognition based on mask learning with pairwise differential siamese network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 773–782, 2019.

[69] Liang Tang, Silong Peng, Yiming Bi, Peng Shan, and Xiyuan Hu. A new method combining lda and pls for dimension reduction. *PloS one*, 9(5):e96944, 2014.

[70] Mahsa Tarantash, Hamed Nosrati, Hamidreza Kheiri Manjili, and Ali Baradar Khoshfetrat. Preparation, characterization and in vitro anticancer activity of paclitaxel conjugated magnetic nanoparticles. *Drug development and industrial pharmacy*, 44(11):1895–1903, 2018.

[71] Yumnam Kirani Singh Vanlalhruaia and N Debachandra Singh. Binary face image recognition using logistic regression and neural network. In *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, pages 3883–3888, 2017.

[72] M Alex O Vasilescu and Demetri Terzopoulos. Multilinear subspace analysis of image ensembles. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–93. IEEE, 2003.

[73] M.A.O. Vasilescu and D. Terzopoulos. Multilinear analysis of image ensembles: Tensorfaces. *European conference on computer Vision (ECCV)*, 2002.

[74] B Verma, M Blumenstein, and S Kulkarni. A neural network based technique for data compression. In *Proceedings of the IASTED International Conference on Modelling and Simulation, MSO97, Singapore*, pages 166–178, 1997.

[75] Chengrui Wang, Han Fang, Yaoyao Zhong, and Weihong Deng. Mlfw: A database for face recognition on masked faces. In *Chinese Conference on Biometric Recognition*, pages 180–188. Springer, 2022.

[76] Hongcheng Wang and Narendra Ahuja. A tensor approximation approach to dimensionality reduction. *International Journal of Computer Vision*, 76:217–229, 2008.

[77] Zhongyuan Wang, Guangcheng Wang, Baojin Huang, Zhangyang Xiong, Qi Hong, Hao Wu, Peng Yi, Kui Jiang, Nanxi Wang, Yingjiao Pei, et al. Masked face recognition dataset and application. *arXiv preprint arXiv:2003.09093*, 2020.

[78] Charles L Wilson, Gerald T Candela, and Craig I Watson. Neural network fingerprint classification. *Journal of Artificial Neural Networks*, 1(2):203–228, 1994.

[79] Peng Wu, BS Manjunath, and HD Shin. Dimensionality reduction for image retrieval. In *Proceedings 2000 International Conference on Image Processing (Cat. No. 00CH37101)*, volume 3, pages 726–729. IEEE, 2000.

[80] Yibo Yang, Stephan Mandt, Lucas Theis, et al. An introduction to neural data compression. *Foundations and Trends® in Computer Graphics and Vision*, 15(2):113–200, 2023.

[81] Guoqiang Peter Zhang. Neural networks for classification: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 30(4):451–462, 2000.

[82] Ligang Zhang, Brijesh Verma, Dian Tjondronegoro, and Vinod Chandran. Facial expression analysis under partial occlusion: A survey. *ACM Computing Surveys (CSUR)*, 51(2):1–49, 2018.

[83] Changjun Zhou, Lan Wang, Qiang Zhang, and Xiaopeng Wei. Face recognition based on pca and logistic regression analysis. *Optik*, 125(20):5916–5919, 2014.