FACE RECOGNITION USING IMPROVED DEEP LEARNING NEURAL NETWORK

by

Zukisa Emerson NANTE

Dissertation submitted in fulfillment of the requirements for the degree

Master of Engineering

at the

University of South Africa

Supervisor: Professor Zenghui Wang

29 September 2022

Declaration

Name: Zukisa Emerson NANTE

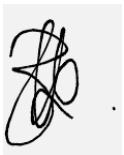
Student number: 3722-95-8

Degree: Magister Technologiae (MTech): Electrical Engineering

The exact wording of the title of the dissertation as appearing on the electronic copy submitted for examination:

Face Recognition Using Improved Deep Learning Neural Network

I affirm this dissertation to be unique and complied with the verifying tool and that its submission fulfills the requirements.



_____ 29 September 2022_____

SIGNATURE DATE

Acknowledgment

First and foremost, my profound gratitude goes to my supervisor, Professor Zenghui Wang from the Department of Electrical Engineering at the University of South Africa (UNISA), for his support, carriage, and exceptional supervision. He always availed himself to read and give feedback on my progress. The time he spent initiating Skype calls during the publication process and his attention to detail has helped me to develop a critical eye to improve my writing and work. His assistance and advice made it possible to complete and submit this dissertation.

With gratitude of gratification, I acknowledge the researchers who have managed to publish their work as a source of information and education.

Moreover, special gratitude to my family especially my loving partner, my parents, and my son, for their support and for giving me strength when I needed it most.

Lastly, gratitude to the Department of Electrical Engineering at the University of South Africa for allowing me to pursue and complete my master's degree.

Dedication

A dissertation devoted to South African Academics, especially My Mother, who defined education as a source of food against poverty and the only weapon to fight hunger, division amongst human beings due to the color of their skin, and to encourage a diverse society.

Abstract

In recent years the importance and need for computer vision systems increased due to security demands, self-driving cars, cell phone logins, forensic identification, banks, etc. In security, the idea is to distinguish individuals correctly by utilizing facial recognition, iris recognition, or other means suitable for identification. Cell phones use face recognition to unlock the screen and authorization. Face recognition systems perform tremendously well, however, they still face challenges of classification. Their major challenge is the ability to identify or recognize individuals in an image or images. The causes of this challenge could be lighting (illumination) conditions, the place or environment where the image is taken and this can be associated with the background environment of the image, posing, and facial gestures or expressions. This study investigates a possible method to bring a solution. The method proposes a combination of the Principal Component Analysis (PCA), K-Means clustering, and Convolutional Neural Network (CNN) for a face recognition system. Firstly, apply PCA to reduce dataset dimensions, enable smaller network usage and training, remove redundancy, maintain quality, and produce Eigenfaces. Secondly, apply PCA output to K-Means clustering to select centres with better characteristics, and produce initial input data for CNN. Lastly, take K-Means clustering output as the input of the CNN and train the network. It is trained and evaluated using the ORL dataset. This dataset comprises 400 different faces with 40 classes of 10 face images per class. The performance of this technique was tested against (PCA), Support Vector Machine (SVM), and K-Nearest Neighbour (KNN). This method's accuracy after 90 epochs achieved 99% F1-Score, 99% precision, and 99% recall in 463.934 seconds. It outperformed the PCA that obtained 97% F1-Score and KNN with 84% F1-Score during the experiments. Therefore, this method proved to be efficient in identifying faces in the images.

Keywords: Face Recognition (FR), Principal Component Analysis (PCA), Deep Convolutional Neural Network (DCNN), Feature Extraction (FE).

LIST OF CONTENTS

Preliminaries

Declaration……i

Acknowledgment……...ii

Dedication……iii

Abstract…..……..iv

Table of contents.…….v

List of figures……viii

List of tables. ….……..x

List of acronyms…...……..xi

1. Introduction……1

1.1 Theoretical background……1

1.2 Motivation……...7

1.3 Research aim and objectives……....9

1.4 Research design and methodology..……..……10

1.5 Discussion……10

1.6 Dissertation outline……11

2. Preliminaries and theoretical background……13

2.1 Chapter overview……13

2.2 Overview of CNN architecture……...13

2.2.1 Early developments based on neuroscience……15

2.2.2 Input layer……17

2.2.3 Convolutional layers ……17

2.2.4 Pooling layers……18
2.2.5 Fully connected layers…..……..……..….18
2.2.6 Image processing using CNN…..……19
2.2.6.1 Image categorization…...…..……..……..……19
2.2.6.2 Image brief description.……..……22
2.2.6.3 Image processing……..……..…22
2.2.7 Related Work……..……25
2.2.8 CNN advantages..……..……...26
2.3 Principal Component Analysis…….……...27
2.3.1 Feature Extraction (FE) using PCA…..……..28
2.4 K-Means Clustering……..…….……30
2.4.1 Data clustering……..……30
2.4.2 Algorithm description.……30
2.5 Conclusion…..……31

3. Proposed Methodology…..……...32
3.1 Chapter Overview……...32
3.2 CNN model selection using PCA and K-Means as a driving force….32
3.2.1 Introduction.……..32
3.2.2 Methodology……32
3.2.3 Using SVM……33
3.2.4 PCA For Data Compression and Classification……..34
3.2.5 Classification using SVC……38
3.2.6 Convolutional Neural Networks (CNN)……38
3.3 Training and simulation of CNN……41
3.4 Conclusion……...44
4. Empirical Findings (Experimentation)…...……45
4.1 Overview ……..……..……..45
4.2 Experimental Setup ……..……45
4.2.1 Development environment……45
4.2.2 Data..……45
4.2.3 Implementation ……46

4.2.3.1 PCA Algorithm Steps……46
4.2.3.2 K-Means Clustering Algorithm Steps……..46
4.2.3.3 CNN Algorithm Steps……47
4.3 Conclusion ……..……51

5. Conclusions And Future Work……..……..52
5.1 Summary and Conclusions..…..……52
5.1.1 Contribution……..52
5.2 Limitations. ……..53

References ……55

Appendix A: Python source code sample.……64

Appendix B: CNN Model Sequel Representation.……....74

Appendix C: Approved Ethical Clearance……..75

Appendix D: List of Publications……..78

List of figures

Figure 1.1: Face Recognition system ……...9

Figure 2.1: Artificial Neurons (Neural Pathway Diagram, 2016)…..……...14

Figure 2.2: CNN schematic architecture diagram (Phung and Rhee, 2019). ….....14

Figure 2.3: Neural Processing of Visual Information and V1 location …….16

Figure 2.4: Subsampling the feature maps (Vahid Mirjalili & Sebastian Raschka, 2017) ……...18

Figure 2.5 (a): Sparse connections due to small kernel. Creating *s,* through convolving 3 span input/output similarity functions, influenced by *x*, (Goodfellow, Bengio, and Courville, 2016) ….....……..……...20

Figure 2.5 (b): Dense connections. Creating *s* by multiplying the matrix, removing sparse connection, thus, every output is impacted by x_3, (Goodfellow, Bengio, and Courville, 2016) ……..……...20

Figure 2.6: Sharing of parameters. Dark arrows specify the networks that utilize a certain parameter in two distinct models. (a) Convolution shares the same parameters across all spatial locations, the dark arrow shows a 3-element kernel in CNN model. (b) Traditional matrix multiplication does not share any parameters, single arrow showing the most important feature of the weight matrix in FC replica …….21

Figure 2.7: Different padding modes (Vahid Mirjalili & Sebastian Raschka, 2017) …….24

Figure 2.8: Padded matrix X_{5x5} (Vahid Mirjalili & Sebastian Raschka, 2017) ……24

Figure 2.9: PCA studies the conversion line to be certain that the first principal component specifies the direction of the ultimate current variance coordinates. *(left)* ***x*** initial data patterns. There is a possibility that the obtained variance is not parallel towards centre line orientation. *(right)*The converted $x = x^T\boldsymbol{W}$ is aligned along $\boldsymbol{z_1}$ axis. $\boldsymbol{z_2}$ reduced variance orientation ……..…….29

Figure 3.1: ORL faces Dataset…….34

Figure 3.2: 24 Eigenfaces with Highest Eigenvalues ……37

Figure 3.3: This study's suggested convolutional neural network architecture. (a) shows a complete architecture, (b) shows the layers in detail…..……39

Figure 3.4: PCA + K-Means Clustering + CNN flow diagram used in this dissertation……...42

Figure 3.5: Cumulative explained variance…….43

Figure 3.6: Convolutional Neural Network training phase ……43

Figure 3.7: Convolutional Neural Network Classification Simulations ……44

Figure 4.1: Model Accuracy of 99% ……48

Figure 4.2: Model Loss of 1% ……...48

Figure 4.3: Filter of the second layer ……50

Figure 4.4: First convolutional layer feature maps ……50

List of tables

Table 4.1 Confusion Matrix ……49

Table 4.2: F1-Score of 99%.……49

Table 4.3: F1-Score Performance Analysis for the Proposed Methodology ……...51

List of acronyms

AD	Anisotropic Diffusion
AFR	Automated Facial Recognition
BPSO	Binary Particle Swarm Optimization
CNN	Convolutional Neural Network
CNNs	Convolutional Neural Networks
Conv	Convolution
FC	Fully Connected
FDA	Fisher Discriminant Analysis
FR	Face Recognition
fSVD	flustered Singular Value Decomposition
GDA	Gaussian Discriminant Analysis
GF	Gabor Filter
GPU	Graphics Processing Unit
HOGs	Histograms of Oriented Gradients
ICA	Independent Component Analysis
IDLNN	Improved Deep Learning Neural Network
KNN	K-Nearest Neighbors
LBPs	Local Binary Patterns
MLP	Multilayer Perceptron

NB	Naive Bayes
NN	Neural Network
P	Pooling
PCA	Principal Component Analysis
ReLU	Rectified Linear Unit layer
RBF	Radial Basis Function
RR	Recognition Rate
SIFT	Scale Invariant Feature Transform
SVC	Support Vector Classification
SVD	Singular Value Decomposition
SVM	Support Vector Machines

CHAPTER 1

Introduction

1.1 Theoretical background

The face is an important part of the human body. Face images can be used in distinguishing people from one another, assessing people's feelings, transmitting various kinds of information, communicating with other persons, etc. The information obtained from face images contains abundant meanings, such as race, gender, age, health, emotion, psychology/mentation, profession, etc.

Woodrow W. Bledsoe (1960) created the earliest semi-automated FR system. The system had to be able to find facial attributes namely, eyes, nose, ears, and mouth from images. During the 1970s FR took a step forward (Goldstein, Harmon, and Lesk, 1971) and utilized 21 indicators like hair color and the upper and the lower edges of the opening of the mouth width to computerize FR, however, quantification and positions were by hand computed. In 1988 an initial semi-automated FR system was implemented and immediately after, the same year, the Eigenface (Sirovich and Kirby, 1987) technique was developed. This technique showed possibilities to minimize the quantity of values required for the approximation of a suitably aligned and normalized face image. During this time a lot of research was conducted and (Turk and Pentland, 1991) discovered that the residual error resulting from the Eigenfaces technique could be reused to detect faces from images and make real-time face recognition possible. This discovery arose the limitations due to environmental factors and again a lot of research was instigated. To date, this technology still faces challenges of light, environment, pose, and expression.

FR unique physical characteristics can distinctively distinguish or validate individuals by associating and scrutinizing the samples established on one's facial outline. Usually, it is utilized in guarding systems, however, the demand for other applications is high. To be exact, a lot of business and legal institutions begin to realize their significance and capability to enforce the law.

Facial recognition is sometimes recognized as face recognition (Bowles N., The Guardian, Apr. 8, 2016). Several methods exist whereby FR applications are implemented; however, their application is to compare chosen face features from other faces in the database. It is defined as a Biometric Artificial Intelligence application that uniquely identifies a person by analysing patterns based on the person's facial textures and shape (Zavyalova V., 2018, Science & Tech Nov 27, 2018).

Biometrics (Jain, Hong, and Pankanti, 2000) technically refers to unique physical characteristics. Therefore, standards of measurements are associated with personal features (Arya, Pratap, and Bhatia, 2015). It has identifiers to distinguish the individuals. This concept is belonging to an earlier time as 500 B.C. when fingerprints were used ("Babylonian business transactions are recorded in clay tablets that include fingerprints.").

Stan and Anil (2011) define FR as an unconscious activity for individuals as it happens automatically, fast, and regularly. Compared to fingerprints and iris biometric modalities face recognition has more than a few advantages. In addition, being natural and non-intrusive, a significant advantage is a face capturing at a distance and covert manner. FR is becoming more important due to speedy developments in appliances that take images such as surveillance cameras, cameras in cellular phone phones, Web demands for face images, and greater security. Kanade T. (1973) developed the first automated face recognition system in his Ph.D. thesis work. But then it became latent over the years till (Sirovich and Kirby, 1987) work on lower dimensions of the ambient space on the face description, originated from Karhunen–Loeve transform or PCA. The ground-breaking work of (Turk and Pentland, 1991) revived the FR research after their ground-breaking on Eigenface. The Fisherface method applied Linear Discriminant Analysis (LDA) after a PCA step to accomplish higher precision. Gabor jets confined filters for efficient characteristics of the face together with the AdaBoost learning built on cascade classifier architecture at the actual time to detect a face. However, under unconstrained environments, Automated Facial Recognition (AFR) systems still face many challenges when face images are acquired.

According to Shamshad Ansari (2020) FR represents the detection and identification problem of how computers can gain a high level of understanding to identify a face of an individual from a video or image. FR follows certain steps to perform recognition and the first step is identifying and locating the face's whereabouts from the input image and this step falls under object detection. After face identification, its attributes are created from numerous key points. Typically, a face of a human being consists of 80 landmarks, but FaceIT system can obtain better recognition accuracy just by utilizing 14 to 22 landmarks (nodal points). They are more intrigued by the profound area of the face[1]. To establish identification these landmarks are compared to the database faces.

The problem regarding FR systems is the inaccuracy and uncertainty in identifying faces. These systems may emerge false results that may cause individuals to be accused of offenses they know nothing about. In America, the FBI system has failed to address the identification inaccuracy problem and their incapability and that resulted at least 15% of the time (Zafeiriou et al., 2014).

Research conducted by the FBI also found out that it misidentifies people of color. Its disproportionality towards people of color was due to how it was programmed. This also affects society as some could not find jobs due to criminal activities they never committed.

Russian only current FindFace Security FR system and the one that was developed by NtechLab claims 99% of accuracy during the 2018 World Cup. But the question or concern on whether FR is a good way of identifying criminals is still open. Also, systems designed by the tech giants like Microsoft and Amazon are still contentious as they strive to identify individuals of various people other than whites or just execute wrong results or mistakes. MIT's Media Lab tested different FR systems from different companies like Microsoft, IBM, and China's Megvii and found that up to 35% of darker-skinned women had their gender misidentified by the systems (Baskar B, Anushree PS, Divya

[1] FaceIt: https://patents.google.com/patent/US7634662B2/

Shree S & KV Mahendra Prashanth, 2015). Back in 2015, Google identified a software engineer's black friends in a photo as "gorillas," and had to apologize for the error. This information does not only highlight the error of programming but also the issue of light as this system was only programmed to follow white people only and to stop when other races were identified. Today the current challenges facing the FR systems are lighting circumstances, environmental settings, or space, pose (of the head), and expression. This also includes the challenges of the hydration of the skin, aesthetic products such as facial lotion and make-up, imaging sensor and camera, and the distance of the subject from the camera (Abhishree et al., 2015).

Bhaskar et al. (2015) proposed a FR system based on Hybrid Gaborlet and Fisher Analysis. It was suggested to overcome the problem of pose, light, age, occlusion, and expressions where the results were exceptional using Dr. Libor Spacek (800 face images of 40 classes with 20 views/class) and Caltech (360 face images with 18 classes with 20 views/class) segmented databases. The importance of a FR system is the accuracy that it can produce by developing a set of features that increase system performance. This system discourages the direct employment of pixel values as traits of face image because of its huge dimension (Golbon-Haghighi et al., 2018). Therefore, PCA, LDA or FDA and ICA techniques were suggested for dimensionality reduction. However, these methods agonize from high computational overload and therefore, the wavelet transform form of multi-resolution analysis manipulates wavelet basis vectors so that it can decompose an image at different scales and orientations. Wavelet transform can be discretely sampled and one of the customaries is Daubechies. Gabor wavelet linear filter named after Dennis Gabor is used for image edge detection (Shrivakshan G., 2012). It is a good adaption ability for different spatial frequencies and orientations. It fragments images into sub-bands and performs convolution to execute the extraction of features. It is a Gaussian Kernel function, and its features improve recognition performance compared to grayscale features. The proposed method FDA extends two-class kernel Fisher methods by subjugating multiclass pattern classification snag. Consequently, it provides unique

solutions that cannot be utilized using GDA. By deducing the matrix's singular values, a new image can be obtained from the original. Hence, fSVD is used to develop illumination invariant image.

Due to the challenges of 2D dimension Abate et al., (2007) suggested a method named 2D-3D FR. The idea behind this technique was to collectively utilize various parameters of 2D and 3D graphic images and model FR based respectively. These parameters were input volume, the number of targeted chores, and recognition measure. This technique compared to other techniques used in FR it provides a future perspective on enabling new techniques for researchers. Therefore, Eigenfaces, and stereovision techniques are applied to enhance 2D FR system performance with 3D information known as the disparity of face. Also, matching a face from various positions with scan-lined-based NN's help. PCA utilised for the extraction of features and recognition. 2D-3D FR accuracy enhanced; however, 3D FR faces minimized the challenges of posture variants, obstruction, and various lighting situations. This is because they resemble a real image, numerous textures, and various frameworks which convolute in three dimensions. Image acquisition technology was applied on 3D face database in comparison to various situations. The proposition was to conduct a study on 3D FR based on local features. Local descriptor division into curves, pointers, and surface were employed. Feature extraction was regarded as one of the important modules in FR. Therefore, different studies were conducted concerning various kinds of face descriptors and attribute extractors for 3D. Bidirectional relighting was done to help normalize between probe and gallery. Also, face expression and occlusions challenges were considered, and correlation metrics were introduced regarding any similarity scores and the idea of pose and illumination normalized signatures for frequently applied confirmations. The courage of utilizing 3D FR technology was to try and conquer 2D FR systems downside. Various databases and different augmentation techniques were used for testing purposes. An enhancement was also made with the help of experienced sensors camera capturing for better 3D face image that can generate 3D face models. Noticed advantage of the 3D FR system is that it so

not affected by light intensity. Wen et al. (2018) proposed improvement with domain adaption and tried to evaluate FR by taking Labelled Faces in the Wild (LFW) dataset as a benchmark and achieved 99% accuracy. Though the performance is still not enough for real-world applications, and that is caused by the problem data bias. To their knowledge, it is the first time that domain adaption technique is applied in unconstraint FR problem with a million-scale dataset. They incorporated face verification threshold into FaceNet triplet loss function explicitly and achieved 99.33% on the LFW benchmark with only a single CNN model and similar performance even without face alignment. This technique combined with Viola methods shows room for improvement.

In the research conducted by Abhishree et al., (2015) the three stages of pre-processing, feature extraction (FE), and Feature Selection (FS) were examined for the entire process of FR system. They used a technique to process the images by flipping an image from left-right (from original to pre-processed images). This is called image pre-processing. This was accomplished using 2D Wiener low pass filter for de-noising. While a 2D Gaussian filter removes the noise it also blurs the image because of its limited bandwidth.

AD based pre-processing and GF based feature extraction were the techniques proposed to enhance FR system performance (Abhishree et al., 2015). AD focuses on enhancing and smoothing facial image edges while GF keeps aligned facial features at specific angles. AD improves the RR of the system by inhibiting the noise the conventional filters have on blurring the edges. 2D GF showed improvements in variance light, attitude, and expression. Accompanying these two techniques is a BPSO based feature selection algorithm to make sure that the space needed is utilized as required for optimal feature subset. These techniques were proposed to improve the challenges of pose, illumination, and expression in computer vision. Moreover, to overcome the problems of image degradation by de-noising an image from a stored database.

The image quality, head orientation, lighting conditions, partial occlusion, and facial expressions play an important role during feature extraction (Singh and Prasad, 2018). The extraction of meaningful features is a very important task, especially in FR, thus, a feature-based system speeds up the process more than a pixel-based system (Viola and Jones, 2004). FR techniques use features like mouth, eyes, chin, nose, and geometrically assess relationships amongst them. Zhao et al. (2020), proposed a FR system using a deep neural network with PCA, jointly with Bayesian framework, and achieved 98.52% performance from their own dataset, which is the CAS-PEAL dataset. Ren et al. (2015), proposed a Region Proposal Network (RPN) that reveals full-image conv features to the detection network.

Arya, Pratap, and Bhatia, (2015) have noted the challenges FR systems still facing i.e., light, posture deviation, expression changes, and facial disguises. According to their research, these are the results of systems based on traditional methods on Visible Spectrum (VS). Therefore, to overcome these limitations of identification and verification the Infrared Spectrum (IRS) was proposed (Baskar B, Anushree PS, Divya Shree S & KV Mahendra Prashanth 2015). Also, emphasized the use of Multi/ Hyperspectral Imagery Data in FR as this system can minimize limitations from existing and conventional FR systems. This system is the future of FR as it provides valuable discriminants for individual appearance (Arya, Pratap, and Bhatia, 2015). However, this improvement still faces eyeglass and physiological problem challenges. This technique's accuracy has not been proven by any method experimentally, and therefore, other techniques were proposed i.e., Persistent Physiological Features and Multi-Model Fusion (VS and IRS) on developing robust Identity Descriptors (ID). The images captured by IRS have shown a significant improvement in IR images as the environment does not pose limitations of light or dark.

1.2 Motivation

The human brain can instantly and automatically recognize familiar and non-familiar faces effortlessly, but the way processes signals from the eyes is still

not fully known, and this is an interesting problem. This raises some questions on how to manipulate computers to interpret images as humans do, what important attributes to consider, and by what means these attributes can be processed. Face recognition systems can be found in security systems at the airport, by the police, etc. For criminal identification and verification systems and many more. This study only focuses on the general improvement of FR systems utilizing CNNs regardless of a specific application. FR is a very interesting biometric modality as it is the natural mode of identification amongst humans and is very unobtrusive. As the name suggests the process of face recognition happens when a face of a person is recognized. A FR system comprises four stages:

1. Face Detection - detects the localization of the image, verify if a face or faces exist(s) in an image and if it does it draws a bounding box on the face, see Figure 1.1.
2. Face Alignment – normalization of a face to be exact and comparable with the database format such as photometric and geometry.
3. Feature Extraction – carefully extracting usable face features to help during the recognition assignment.
4. Face Recognition - compares these features from the faces in the database, verifies if a match exists and if it does it recognizes that distinct person by assigning a label trained on it.

Figure 1 depicts a face recognition system. Face detection and recognition differ in the sense that in detection the interest is only to know if a face exists from an image or static picture or video, but the recognition task is a procedure of recognizing an already detected face or identifying who the person is (Singh and Prasad, 2018). Nair and Hinton (2010) describe object recognition as the way to keep or maintain the same input properties in the output invariance. FR systems work by comparing selected facial features from a given image with faces within a database. However, these systems still face problems of illumination, pose variation, expression changes, and facial disguises. Different lighting environments affect detection and recognition accuracy. In recent years CNNs have demonstrated to perform well in FR and they can accommodate a big dataset (Song et al., 2020), however, that may

increase the training time. This dissertation aims to develop a new lightweight method to train faster and increase accuracy.

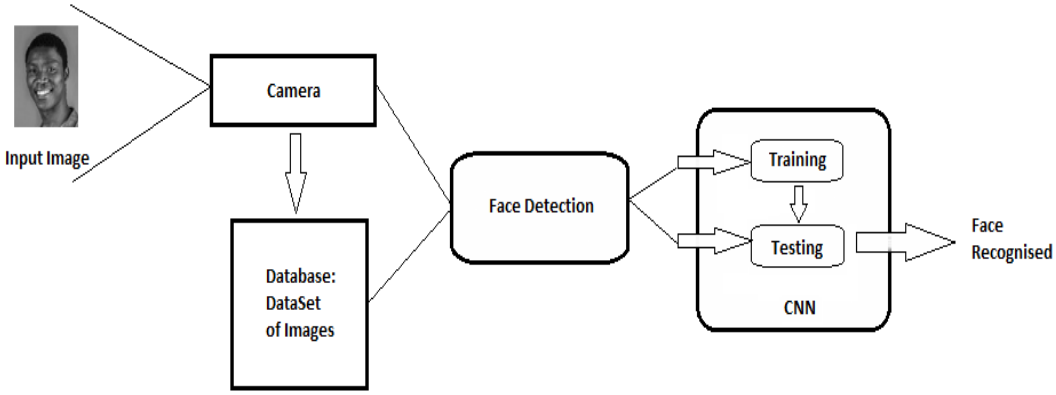


Figure 1.1: Face Recognition system.

1.3 Research aim and objectives

In this dissertation, the aim conveys the intention of achieving the desired outcomes, however, the objectives mainly focus on defining resources to fulfil the aim.

Aim: *This study aims to improve (DNN) by investigating deep CNN on facial recognition systems. To investigate theories on feature extraction, illumination, and noise that affect the process of facial recognition in 2D FR systems. To improve and add technical knowledge in the field of FR systems and enhance accuracy and performance on identifying faces.*

Objectives: The main objectives of this dissertation are to investigate the fundamental concepts of face recognition using DCNN, establish and define image classification concepts regarding face recognition, analyze the different architectures of DCNN and the motivations behind it, propose a methodology for FR that improves the efficiency and accuracy of DCNN results, and

evaluate obtained results from these methods in order to measure accuracy and effectiveness in an attempt for improvement.

1.4 Research design and methodology

This dissertation adopts a positivism research philosophy, it adheres to factual knowledge obtained through empirical observations (Collins H., 2010) in an objective manner. Therefore, it is based on data collection, interpretation, and depended on quantifiable observations. It involves already well-known algorithms, however, these were never combined before in FR systems. It is then followed by an experimental substation of the presented methods, which are analyzed in detail. Experimental results and comparisons to other techniques are obtained using traditional computer vision benchmarks.

1.5 Discussion

In this dissertation, to address the defined challenges mentioned above in Section 1.1, an attempt to address the face feature extraction problem is addressed, so that good features are preserved to obtain better results by combining PCA, K-Means clustering, and the CNN architecture. The aim is to analyze the existing methods, extract important features, observe the accuracy of each, and combine the methods that promised to improve the performance and accuracy. To deal with this problem, to reach a solution, the following FR methods are discussed:

1. Neural Network – CNNs easily recognize normalized and aligned faces and increase the efficiency of the model.
2. PCA – it linearly modifies the original inputs into new uncorrelated features. This technique can help in dimensionality reduction; however, it agonizes from high computational overload.
3. Geometric Based – this method analyzes local facial features and their relationship. Can be constructed with tools like PCA and Support Vector Machines (SVM).
4. K-Means Clustering – groups samples based on their feature similarities. While this algorithm is exceptional at recognizing clusters

with a spherical space, it has a drawback, and that is it must be given the number of clusters, k, a priori.

DCNNs have been chosen as they leverage local patch relations, local connectivity, and translational equivariance that can help improve the model's accuracy and performance. They can work with inputs of variable sizes. Sparse interactions are derived by letting the kernel smaller than the input. This plays an important role in image processing. For instance, the picture elements from an image as an input may entail a vast number of pixels, however, tiny, significant characteristics namely edges in conjunction with input/output similarity function to dominate only plenty of picture elements (Goodfellow, Bengio, and Courville, 2016). In conclusion, this means, only fewer parameters are stored to limit memory required and enhance model architecture. Unlike in traditional neural networks, sharing parameters, the same weights are utilized to various functions, and therefore, for different patches of the input image. Thus, it learns a specific distinct list of elements instead of studying whole separate distinct lists of elements.

1.6 Dissertation outline

The sections below cover or support the research objectives of this dissertation mentioned in Section 1.3 and are organized thusly:

Chapter 2: This chapter contends the fundamental concepts and theoretical background of applying the deep convolutional neural networks in face recognition. It starts with a general overview of CNNs, then proceeds to the neuroscience early developments and elaborates on the properties of convolutional neural networks (CNNs) related to or taken from this literature. After it discusses image processing in detail and explains the origins of an image. Further looks at the facial features extraction, PCA motivations, and K-Means clustering theoretical background. In conclusion, the motivation and related work of CNNs in FR systems is reviewed.

Chapter 3: This chapter focuses on giving the background of the suggested methodology of this dissertation. It starts by elaborating on the methodology, then discusses SVM, and PCA data reduction advantages towards this technique. It then goes further to discuss SVC classification, subsequently, displays the CNN flowchart proposed and its training phase. In conclusion, CNN classification and simulations are reviewed.

Chapter 4: The chapter portrays the empirical results of the methodology suggested in this study. It commences by stipulating the system requirements pertaining to the training of this algorithm. Next, discusses the data used to get the results, and finally discusses the steps taken per algorithm to demonstrate the origins of the concatenation idea.

Chapter 5: In this chapter, represents the summary of this study. Next, the conclusions, limitations, contributions, and future direction are discussed.

CHAPTER 2

Preliminaries and theoretical background

2.1 Chapter overview

In this chapter, all three algorithms proposed in this dissertation are looked at in detail. It starts by discussing an overview and theoretical details regarding convolutional neural networks. Next, discusses the layers of CNN and finally motivates the usage of CNNs in FR. Thereafter, it delves into the PCA algorithm and discusses how it extracts features. Then, the K-Means algorithm, and finally the chapter conclusion.

2.2 Overview of CNN architecture

CNN models were motivated by the fundamental working cortex of the human brain (McCullock and Pitts, 1943) when recognizing objects, see Section 2.2.1. Figure 2.1 depicts nerve cells that are connected to perform signal or wave handling and transmission of mechanical and magnetic waves, and they are called neurons. The CNN evolution originated around the 1990s after LeCun et al. (1989) suggested a new different NN model classification for figures written by hand from images. They are neural networks (NN) that are composed of a mathematical operation called convolution, hence the name CNN. Their ability to spontaneously learn features from unprocessed data makes pre-processing easy in comparison to other image classification algorithms mentioned in Section 1.1. They construct a feature arrangement by merging minor details of the image like edges in the form of layers to form recognizable features. They compute feature maps from the input data and in this case from the face image, in which, every single essential part originates from a neighboring patch of pixels. This local patch is called the local receptive field (Vahid Mirjalili & Sebastian Raschka, 2017). Convolutional neural networks result from neurons containing weights and biases. Their goal is to

go from the unprocessed input data in the first layer to the exact class in the last layer. They differ from normal neural networks due to the type of layers used in them and the way they treat the input data. They assume input data as images and that allows them to extract properties particular to images (Prateek J., 2017). Figure 2.2 illustrates the multilayer CNN architecture.

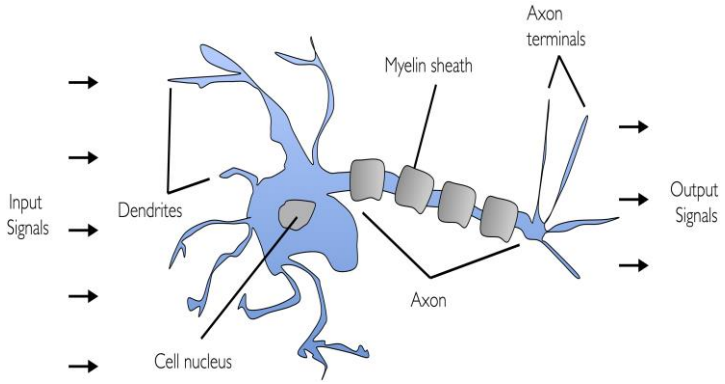


Figure 2.1: Artificial Neurons (Neural Pathway Diagram, 2016) [2]

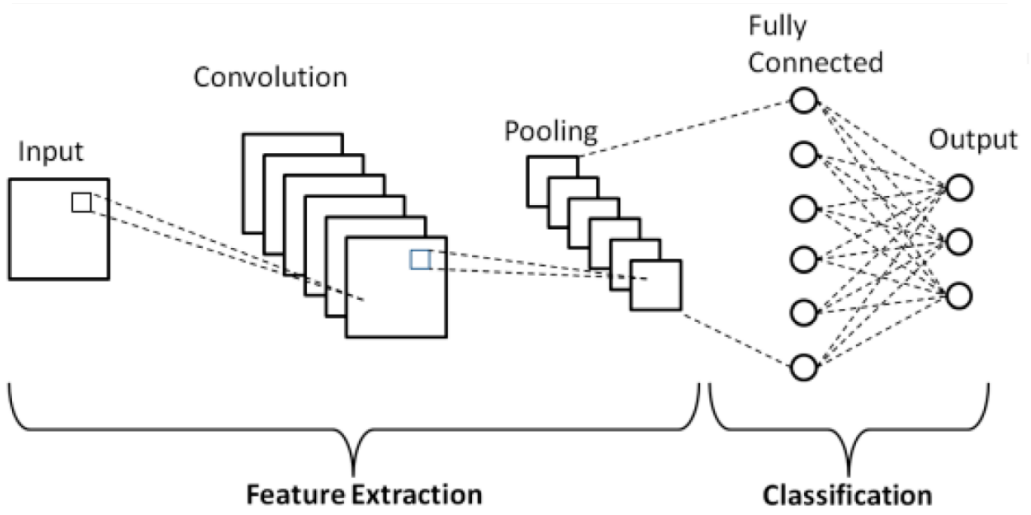


Figure 2.2: CNN schematic architecture diagram (Phung and Rhee, 2019).

[2] Neural pathway: Neural pathway diagram - Visual cortex - Wikipedia

2.2.1 Early developments based on neuroscience

CNNs keep on demonstrating to be the great biologically inspired artificial intelligence (AI) in deep learning for image processing. CNN concepts were taken from neuroscience. For many years neurophysiologists (Hubel and Wiesel, 1962) investigated the functionality of the mammalian visual system. From their observations of the cat's brain responses to images, they discovered that the visual behaviour of the neurons reacted very intensely towards certain lighting environments. Their work helped to formulate the deep learning simplified focal point of view of brain function.

Consequently, the simplified view of the brain that is of interest is the primary visual cortex (V1). V1, the brain's initial part, is found within and across the calcarine fissure in the occipital lobe. It substantially executes advanced visual input processing. It creates images through the optical occurrence delivered in the eye and stimulates the optic nerve. The optic nerve (retina) is a tissue sensitive to light from an eye background. Thus, the optical nerve neurons manage the image without changing its features or layout. Then it goes past the optical nerve and lateral geniculate nucleus. To conclude, their role is mainly to coordinate communication between the eye and V1.

CNNs mimic some properties of the V1; because the V1 arrangement is in the spatial map, CNNs make use of this trait on defining its features in terms of 2D maps. V1 comprises a lot of simple cells. A simple cell is characterized by a linear function of the image in a small, spatially localized receptive field. Hence, CNNs detector units imitate simple cells properties in their designs. V1 also encompasses numerous complex cells, and these respond to similar features as the ones detectable in similar cells, however, these cells are immutable to tiny changes of the feature location. The inspiration behind pooling units of the CNNs originates from this property of the complex cells. These second-class cells are likewise immutable to alterations in illumination that are impossible to capture merely by pooling over spatial locations. These immutable or never-changing cells encouraged cross-channel pooling strategies of the CNNs like maxout units (Goodfellow, Bengio,

and Courville 2016, p. 354). Figure 2.3 depicts the neural processing of visual information and V1 location in the brain.

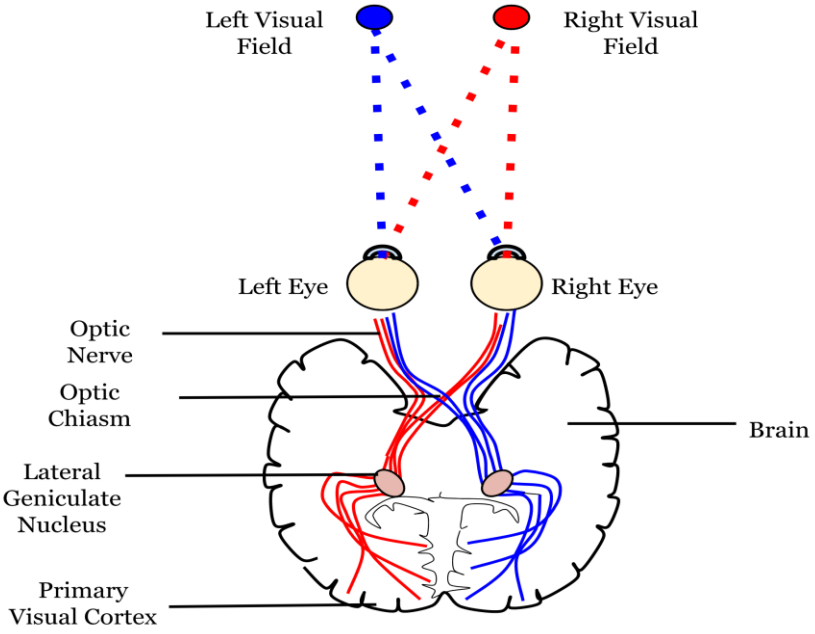


Figure 2.3: Neural Processing of Visual Information and V1 location[3]

The brain consists of multiple anatomical layers and moving deeper into the brain, cells responding to some concept and that are immutable to a vast number of input conversions are found and called grandmother cells. The idea of the grandmother cells is that the neuron of a person activates when that person sees his or her grandmother irrespective of the location view angle, shot zoomed out image of the entire body, close-up face or either is illuminated or in the shade. In the brain, they are in the area called the medial temporal lobe (Quiroga et al., 2005). Their neurons are slightly typical in comparison with current CNNs that do not mechanically generalize in distinguishing an individual. An analogy to CNNs attributes of the final layer can be the brain part namely, the inferotemporal cortex (IT).

[3] Neural pathway diagram - Visual cortex - Wikipedia

2.2.2 Input layer

The input layer is the first layer of the CNN, and it takes the raw image data as it is. It is important to note that this layer is the entry-level of the whole neural network image processing, it signifies the pixel matrix of the image. Input images are encoded into channels namely, three channels (C_3) and one channel (C_1). C_3 represents Red, Green, and Blue (RGB) colors and C_1 black and white colors. Therefore, the contained unprocessed image information intensifies each channel color into a dimension vector of batch size width x height x C_n.

2.2.3 Convolutional layers

The convolution layer calculates the convolutions between the neurons and various patches. It is a feature extractor layer as it extracts the image features from the previous layer. Thus, its output is a product of weights and a small patch of the preceding layer. It takes the image input height, width, and channels and convolves the image into a feature map. After this abstraction, the image has input number, feature map height, feature map width, and feature map channels and this is called an activation map. Neurons in convolutional layers make sure that the processed data is unique and has its receptive field. This means that neurons, only receive input from a restricted area. In 2D input such as X_{n1xn2} and the filter matrix W_{m1xm2} where m1 <= n1 and m2 <=n2, then the matrix Y = X * W exists because of low-dimensional conv of a signal as well as filter. Mathematically it is presented like below:

$$y = \mathrm{X} * \mathrm{W} \rightarrow \mathrm{Y}[\mathrm{i},\mathrm{j}] = \sum_{\mathrm{k1}=-\infty}^{+\infty} . \sum_{k_2=-\infty}^{+\infty} \mathrm{X}[\mathrm{i} - k_1, \quad \mathrm{j} - k_2]\mathrm{W}[k_1, k_2] \tag{2.1}$$

This layer encompasses a Rectified Linear Unit layer (ReLU) to assign a zero to a negative value. ReLU works by activating the output of the preceding layer. The ReLU function is defined as f(x) = max(0, x), and it generalises well to any function type by adding non-linearity to the network.

2.2.4 Pooling layers

Pooling layers immediately take the output of the convolution layer as their input, sample, and modify it further using a pooling function. This replaces this output at a particular location with a statistic summary of the nearby outputs. Its vital role is to reduce the number of computations (downsample the output of a convolution layer) and increase spatial invariance of the height and width (Vahid Mirjalili and Sebastian Raschka, 2017). They are usually denoted by $P_{n1xn2,}$ and this subscript defines the pooling size of the neighborhood of pixels. Using these layers help to keep only the important parts of the net and operate on each feature map independently. Though they consist of max pooling and min pooling, max pooling is frequently used. Max pooling (Zhou and Chellappa, 1988, Cireşan et al., 2012) is used to downsample the 2D volume. It works like a convolution kernel, together with a particular stride, taking out max values rather than computing dot products. A stride of 1 would preserve the volume size while a stride of 2 would divide the width and height dimension by two (Habrman, 2016). Figure 2.4 depicts the pooling layer also known as a sampling layer:

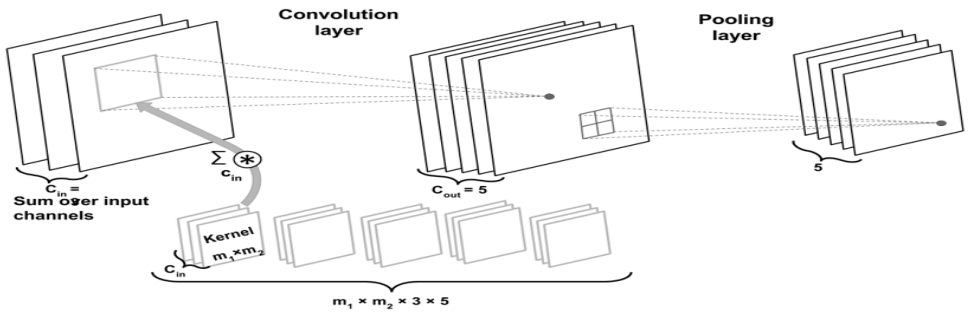


Figure 2.4: Subsampling the feature maps (Vahid Mirjalili & Sebastian Raschka, 2017).

2.2.5 Fully connected layers

FC layers represent a multilayer perceptron by which each vector (initial data signal) *i* brings together each production element *j* and filter w_{ij} . These layers

are at the end (last layer) of the CNN, and they calculate the output scores. Thus, the resulting output is of size 1x1xL, where L represents the number of classes in the training dataset. They produce the final feature map classification task. FC layers only accept 1D data, meaning, conversion from 3D or 2D data is needed and this can be performed with the help of Python flatten functions.

2.2.6 Image processing using CNN

2.2.6.1 Image Categorization

The image categorization process arranges data into a fixed number of categories so that it can be used effectively and efficiently, therefore categorizing images into one or more classes. Classifying images with deep CNNs requires the use of TensorFlow. The CNN FC layers are fundamentally related to several layer perceptron whereby each initial data *i* brings together each production element *j* and filter *wij*. Multilayer neural networks in CNNs construct a system of features by merging blobs and edges attributes in layers formulating events and objects attributes.

CNNs aid to develop the systems by (1) Sparse interactions, (2) equalizing a set of parameters, and (3) equivariant representations. sparse weights are obtained by ensuring that the input similarity is lesser than the output similarity. Thus, only one component within the activation plan is linked against the lesser blotch picture elements. For example, an input image with lots of picture elements, however, only a few eloquent features like edges containing input/output similarities that inhabit a lesser detected pixels quantity. Therefore, it reduces model memory requirements by storing fewer parameters and improving its statistical efficiency. For instance, m inputs and *n* outputs, require a multiplication matrix (*m x n*) parameters and practically this would be *O(m x n)* runtime. Hence, limiting the number of connections in each output to *k*, enables the sparsely connected approach to need only *k x n* parameters and *O(k x n)* runtime. Figure 2.5 (a & b) illustrates sparse

connections, seen from below, single highlighted input element, x_3, plus *s,* highlighted for affected output units by this unit.

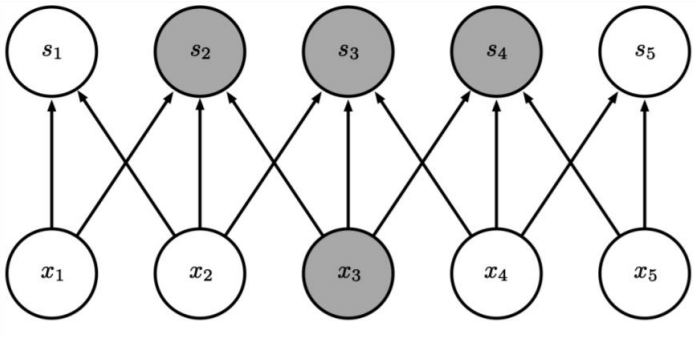


Figure 2.5 (a): Sparse connections due to small kernel. Creating *s,* by convolving 3 span input/output similarity functions, influenced by *x*, (Goodfellow, Bengio, and Courville, 2016).

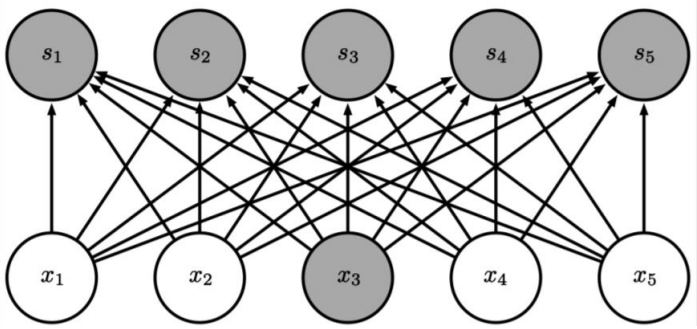
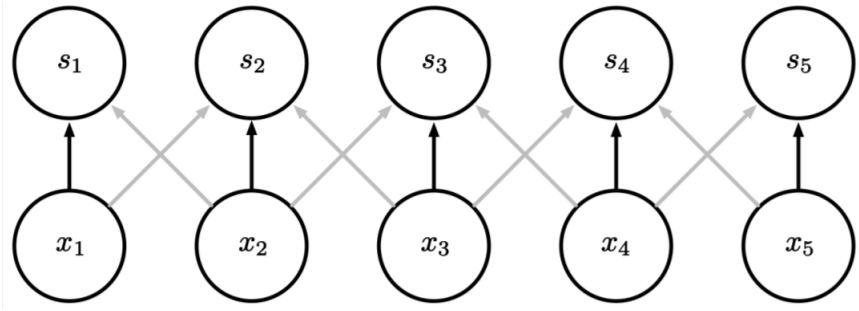
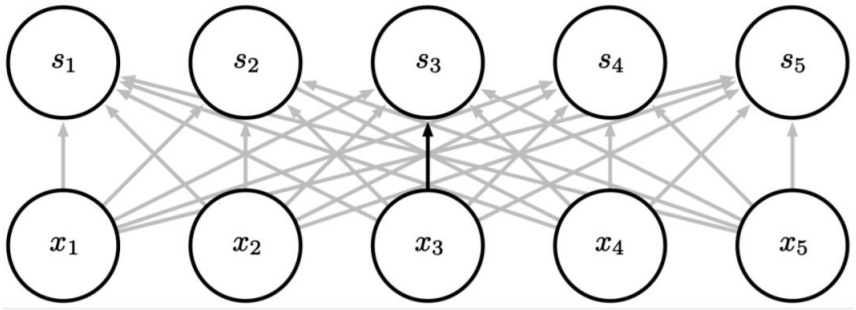


Figure 2.5 (b): Dense connections. Creating *s* by multiplying the matrix, removing sparse connection, thus, every output is impacted by x_3, (Goodfellow, Bengio, and Courville, 2016).

Parameter sharing uses identical weights for various patches of the input image. Unlike traditional neural networks, CNNs use tied weights, meaning, the value of the weight applied to one input is tied to the value of a weight applied somewhere else. Thus, instead of learning a distinct set of parameters present in each location, it learns only one set. It has no influence on the compilation time of moving from the input to the output O*(m x n)* and reduces repository space needed to *k* numerical factors. It adds to the translation invariance of the CNN architecture (Mouton, Myburgh, and Davel, 2021). Figure 2.6 depicts the difference in CNNs parameter sharing and traditional nets (Goodfellow et al., 2016).



(a)



(b)

Figure 2.6: Sharing of parameters. Dark arrows specify the networks that utilize a certain parameter in two distinct models. (a) Convolution shares the same parameters across all spatial locations, the dark arrow shows a 3-element kernel in the CNN model. (b) Traditional matrix multiplication does not share any parameters, a single arrow shows the most important feature of the weight matrix in the FC replica.

Finally, equivariant representations, this is a specific form of parameter sharing that causes the layer to have a property called equivariance to translations. Equivariance means if there are changes in the input, these changes also occur in the output in an identical manner. Let's say for an example that a function *f(x)* is equivariant to a function *g* if *f(g(x)) = g(f(x))* and convolution is used, then if *g* is any function that transforms the input, meaning, shifts it, then the conv function is equivariant to *g*. To illustrate this, let function *I* give brightness to the image at integer coordinate. Secondly, let function *g* map a single task of the image to one more task of the image, so, *I´= g(I)* is

the image unit of code defined with $I'(x,y) = I(x - 1,y)$ to move each picture element of I a single element to the right. Thus, whenever such conversion is applied towards I, and followed by convolution, the results are identical as if convolution is applied to I' and the applied g transformation to the output. When dealing with images, convolution generates a 2D map where a variety of features are seen in the input. In CNNs, when processing images it is convenient to detect edges in the first layer. This helps when processing cropped images that ought to be designated in the centre of a person's face. Therefore, different features at different locations are extracted. Features processed at the top of the face are eyebrows and at the bottom a chin.

2.2.6.2 Image brief description

From the beginning of this dissertation, images were not defined, and therefore, it is better to give the fundamental knowledge before delving into the whole process of image processing. Images are an important part of humans on daily basis. Images help to identify, interpret, illustrate, represent, memorize, educate, communicate, evaluate, navigate, survey, entertain, etc (Awcock and Thomas, 1995). Humans do all this without conscious effort as it comes naturally to them, but machines do not. For machines to perform all the properties of the image mentioned above, an image needs to be processed and this is called applied image processing, but this is discussed in the following section.

2.2.6.3 Image processing

Applied image processing attempts to provide practical, reliable, and affordable ways to let machines deal with images. Therefore, by contrast, the term image processing has been associated with modifying images. Meaning, correcting errors introduced in images during re-creation, or visual system enhancement. Its main purpose is to process an input image and generate a

modified image output. For pattern classification, this helps take a feature vector input and generate a class number output.

Image pre-processing is a vital step to make sure that raw data is of the required form and shape, however, according to the study of Paul and Acharya (2020) pre-processing is not required in CNNs. Pre-processing enhances image data by suppressing unwanted or undesired distortions and increasing relevance in image features for further processing. In this dissertation, the sequence of face images is targeted to take out significant attributes such as the contour of the eye sockets, nose, mouth, and chin. Features transformation to gamut [0, 1] or a traditional regular allocation of a naught average with unit variation (Vahid Mirjalili and Sebastian Raschka, 2017).

CNNs perform image analysis by segmenting the image and performing object detection and recognition. In Section 2.2.3 (convolutional layers), eq. (2.1), represents a 2D output vector y computation with input vector x and filter or kernel w where i runs through all elements. Eq. (2.2) represents convolution in one dimensional (1D), as it applies the same principles as in 2D:

$$y = x * w \rightarrow y[i] = \sum_{k=-\infty}^{+\infty} x[i-k]w[k] \tag{2.2}$$

The advantage of CCNs is to correctly compute the summation of the above formula by assuming that x and y are filled with zeros, and this is called padding. The common three modes of padding in CCNs are full mode, same mode, and valid mode. Full mode is rarely used in CNN architectures due to its increase in output dimensions. Same padding ensures output and input dimensions match. In valid mode, there is no padding at all and that is a disadvantage in comparison to the other padding modes. This decreases the volume of the tensors substantially and may hold back the performance of the model. However, practically, the occupying space dimensions utilizing the same padding for convolution layers are preserved and its dimensions can be decreased via P layers instead. The interest of this dissertation is the same

padding as the expectations are to retrieve the same features of the input and classify them accordingly to get the same output. Figure 2.7 further illustrates these modes:

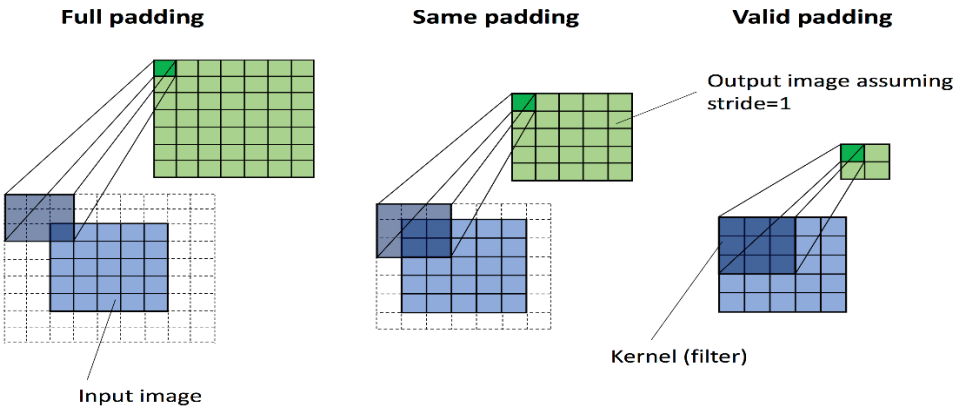


Figure 2.7: Different padding modes (Vahid Mirjalili and Sebastian Raschka, 2017).

The output quantity can be defined by how many times filter w shifts beside input vector x. Eq. (2.3) shows this as follows:

$$o = \left[\frac{\mathrm{n} + 2\mathrm{p} - \mathrm{m}}{\mathrm{s}} \right] \tag{2.3}$$

Padding in 2D is possible when both dimensions are extended independently. Figure 2.8 gives a glance at this.

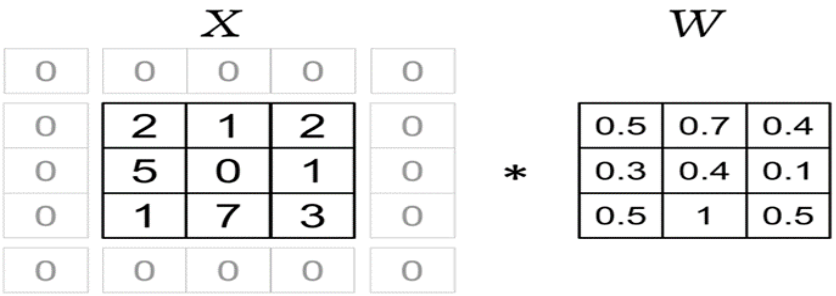


Figure 2.8: Padded matrix X_{5x5} (Vahid Mirjalili & Sebastian Raschka, 2017).

2.2.7 Related Work

This dissertation focuses on FR in images and videos, a problem that has received substantial attention in the recent past. The literature proposed many methods and the distinction between these is shallow (do not use deep learning) and deep (use deep learning). The first step of the shallow methods is to extract the representation of the face image with the help of handcrafted local image descriptors, i.e., Scale Invariant Feature Transform (SIFT), Local Binary Patterns (LBPs), and Histograms of Oriented Gradients (HOGs) (Parkhi et al., 2015). They used the Fisher Vector as a pooling mechanism to form local descriptors into a general face descriptor (Parkhi et al., 2014).

In this dissertation the focus is mainly on deep learning architectures for FR, such methods use CNN feature extractor, a learnable function acquired by combining several linear and non-linear operators. In the beginning, CNNs specifically AlexNet (Krizhevsky et al., 2012a) and VGGNET (Krizhevsky et al., 2012a) restricted the quantity of input images in the account of their architecture design. They were composed of mainly two parts, the conv layer, and FC layer. The convolution layer does not need a determined or limited image size and can produce a feature map of any size (Qin et al., 2020). Recent CNNs, namely, Inception (Szegedy et al, 2016), ResNet (He et al., 2016), and DenseNet (Huang et al., 2017), stopped using the FC layer and switch to Global Average Pooling layer (Lin et al., 2013). To their discovery, it became clear that not only does the Global Average layer solve many parameters problems in the FC layer but can accommodate any image size. Though this advantage of increasing the size of images improves accuracy (Zheng et al., 2016), it also has a disadvantage, and that is, the amount of computation required becomes costly to obtain accuracy.

Mishra et al. (2021) proposed a multiscale parallel deep CNN (mpdCNN) feature fusion architecture for FR for real low-resolution images taken from long distances with different resolutions, illumination, and pose. They discovered that their architecture performed better in SCface challenging database for both low-resolution (86% accuracy) and high-resolution (99% accuracy) images. Due to these findings, they concluded that this method is

suitable for FR systems in real-world applications namely criminal investigation procedures.

Zhang, et al. (2015) suggested a flexible FR CNN system, without performance comparison. Basically, it extends CNN architecture by analysing errors and the identification accuracy of the data trained. Therefore, it extends the network globally up until it meets the average error and recognition rate and it obtained 91.67% accuracy in face identification using the ORL face database.

Due to the popularity of the ORL database in FR, Kamencay, et al. (2017) also proposed a CNN that comprised of only two convolution layers, FC, ReLU, and two pooling layers. It obtained 98.3% accuracy with 80% and 20% training and testing respectively. Research by S. Almabdy et al. (2019) came up with another method to enhance the effectiveness of the FR system by taking out the features learned for the image from the CNN (AlexNet type) as well as RasNet-50 and after, a SVM classifier. In 2019, a CNN combination of VGG16, ResNet50, and MobileFaceNet obtained promising results in children's database (Oo and Oo, 2019). It focused on classifiers and feature extraction. A year later, Sharma et al. (2020) proposed a PCA face feature extractor and blended it with multilayer perceptron (MLP), Naïve Bayes, and SVM to training and testing ratio of 80:20 using ORL database.

2.2.8 CNN advantages

In Section 2.2.6.3 (image processing) segmentation was mentioned, however, this is not a requirement for CNN architectures (Yamashita et al., 2018). CNNs robustness eliminate the need for hand-crafted feature extraction and contains loads of learnable parameters for estimation. The downfall of this, it is computationally expensive and demands more system software enhancements for processing, i.e., graphical processing unit (GPU) to model the training.

2.3 Principal Component Analysis

PCA is an unsupervised linear method applied in many disciplines and mostly in favour of feature extraction and dimensionality reduction. PCA can also be used in empirical data analysis, signal de-nosing, and in the field of bioinformatics for genome data and gene expression levels analysis (Vahid Mirjalili & Sebastian Raschka, 2017). It has two criteria, learning representation(s) of the lower dimensionality than the raw input, and learning representation(s) with elements without a straight-line connection amongst themselves (Goodfellow, Bengio and Courville, 2016). It learns linear transformation of the data of two vectors that are perpendicular to each other and casts input ***x*** to a ***z*** depiction. Figure 2.9 elucidates this graphically.

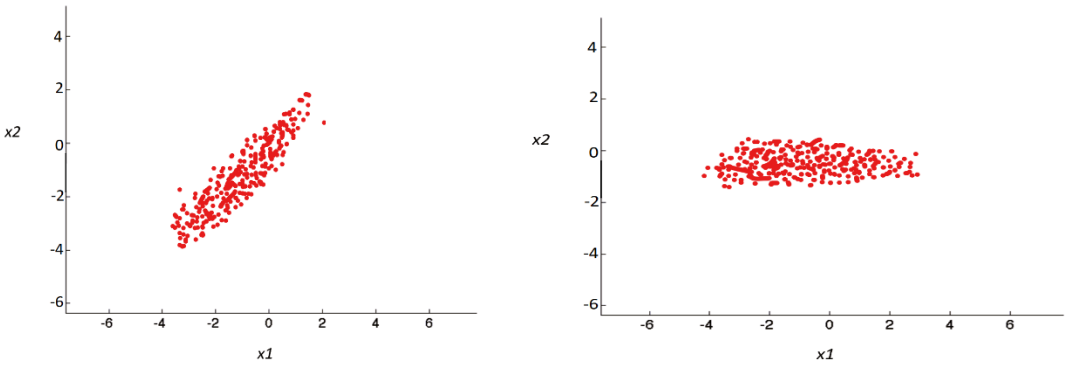


Figure 2.9: PCA studies the conversion line to be certain that the first principal component specifies the direction of the ultimate current variance coordinates. *(left)* ***x*** initial data patterns. There is a possibility that the obtained variance is not parallel towards centreline orientation. *(right)*The converted $x = x^T\boldsymbol{W}$ is aligned along $\boldsymbol{z_1}$ axis. $\boldsymbol{z_2}$ reduced variance orientation.

PCA decorrelates the raw data representation ***X***, for example, considering the matrix design ***X*** to be $m \times n$, and supposedly the data is average naught, $\mathbb{E}[x] = \mathbf{0}$. Otherwise, it centres the data by deducting the mean from instances in the processing phase. Thus, ***X***, is linked with the impartial sample covariance matrix as can be seen in eq. (2.4).

$$\mathrm{Var}[\mathrm{x}] = \frac{1}{m-1}\boldsymbol{X}^T\boldsymbol{X} \tag{2.4}$$

where, $\boldsymbol{X}^T\boldsymbol{X}$ are the eigenvectors. From eq. (4.1) it derives a $\boldsymbol{z} = \boldsymbol{W}^T x$ representation, where Var[z] is diagonal. It also provides possibility to obtain principal components via a Single Vector Decomposition (SVD), and to be exact they are the right singular vectors (the eigenvectors) of ***X***. To demonstrate this, let ***W*** be the right vectors of the decomposition $\boldsymbol{X} = \boldsymbol{U}\sum\boldsymbol{W}^{\boldsymbol{T}}$. Therefore, from this, the original eigenvectors equation with ***W*** is derived as shown in eq. (2.5).

$$\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X} = (\boldsymbol{U}\sum\boldsymbol{W}^{\boldsymbol{T}})^{\boldsymbol{T}}\,\boldsymbol{U}\sum\boldsymbol{W}^{\boldsymbol{T}} \;=\; \boldsymbol{W}\sum^{\boldsymbol{2}}\boldsymbol{W}^{\boldsymbol{T}} \tag{2.5}$$

Then, SVD can be applied to prove the existence of diagonality in PCA results in Var[z], and thus, from SVD of ***X*** the variance of ***X*** can be represented with eq. (4.1) and by substituting $\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}$, with the results obtained in eq. (2.4). Then, eq. (2.6) is obtained,

$$\mathrm{Var}[\mathrm{x}] = \frac{1}{m-1}\,\boldsymbol{W}\sum^{\boldsymbol{2}}\boldsymbol{W}^{\boldsymbol{T}} \tag{2.6}$$

thus, since the ***U*** matrix of SVD is orthogonal, the covariance of ***z*** is diagonal, and eq. (2.7) represents this.

$$\mathrm{Var}[\mathrm{z}] = \frac{1}{m-1}\,\sum^{\boldsymbol{2}} \tag{2.7}$$

So, the above analysis represents an important PCA property, which is, the capability to transform data into representations of mutually uncorrelated elements. When projecting the feature ***x*** to ***z***, via a straight-line conversion ***W***, the output is diagonal to covariant of matrix $\sum^{\boldsymbol{2}}$.

2.3.1 Feature Extraction (FE) using PCA

FE is used to convert or launch information onto a new characterized collection of features. However, when used in the context of dimensionality reduction its purpose is to compress data and highly maintain the relevant features.

Therefore, the predictive performance is enhanced by reducing curse dimensionality. Bellman (1957) defines curse dimensionality as an exponential in volume associated with adding extra dimensions to Euclidean space. This essentially means an error increases with the increase in the number of features. Here, PCA for FE and dimensionality reduction is used as it helps in distinguishing the shapes of the input signal connected to facial attributes. Its purpose is to locate the utmost divergence path in the high dimensionality input signal and project it to a set of lesser vectors compared to the initial data. Hence, PCA dimension reduction constructs the *d* x *k*-dimensional conversion matrix W which permits mapping of test value *x* against a recent *k*-dimension attribute vector space with less size compared to initial *d*-dimension attribute space such that:

$$x = [\mathbf{x_1}, \mathbf{x_2}, \dots, \mathbf{x_d}], x \in \mathbb{R}^d \quad (2.8)$$

$$z = [\mathbf{z_1}, \mathbf{z_2}, \dots, \mathbf{z_k}], x \in \mathbb{R}^k \quad (2.9)$$

Xiang et al. (2015) expresses PCA to be an ancient practice of extracting attributes and presenting data in pattern recognition and computer vision. Sharma and Patterh (2015) proposed a new FR system that utilized the adaptive filter median filter for preprocessing to denoise the face images and then PCA features to extract Eigenvectors. FE uses covariance matrix to generate matrix and eigenvectors of the face and rebuild images using PCA, (Zhou et al., 2013). Gumus et al. (2010) evaluated the efficiency of PCA in FR and used the Eigenfaces technique for data reduction and feature extraction. However, though PCA proves to work best in FR linear extraction, it becomes insufficient regarding nonlinearities.

2.4 K-Means Clustering

2.4.1 Data clustering

Data clustering method groups together enormous sets of data into clusters of smaller sets of similar data (Faraoun and Boukelif, 2007). Therefore, an algorithm to utilize this technique investigates pure groups of data for similarities and its centroid. It distinguishes the distance between a point and the cluster centroids. This algorithm should preserve the qualities of producing all potential subsets for a distance function (Kleinberg, 2002). Wang et al. (2021) defines it as unsupervised learning that strives to group together similar clusters.

2.4.2 Algorithm description

K-Means clustering is a representation learning algorithm that separates the data to be trained into x distinct clusters of instances near to one another It, therefore, supplies a k-dimensions with a single high bit value h indicating an input x. For instance, say x is from cluster i, so $h_1 = 1$, then other representations of h are nil. One-hot a sparse representation, meaning, a vast number of its submissions are nil to all input entry. The advantage of k-means is that it groups samples based on their feature similarities. While this algorithm is exceptional at recognizing clusters with a spherical space, it has a drawback, and that is to be given a quantity of clusters, k, a priori.

This algorithm initializes k various geometric centers $\{\mu^{(1)}, \dots, \mu^{(k)}\}$ of various features, however, alternates amid two distinct strides in anticipation of converging. From a single stride, every single training incident goes to group i, whilst i represents the position of the closest geometric center $\mu^{(1)}$. Whilst other stride $\mu^{(1)}$ geometric center or centroid is modernized to median training values of $\boldsymbol{x}^{(j)}$ given to group i. Though this algorithm works well, it not well-constituted, meaning, in practice, no basic principle can determine or ensure the clustering performance. One of its characteristics namely, Euclidean, it can be measured in the cluster geometric center to cluster representatives. That clarifies the possibilities available to rebuild training data from the assignments

of the cluster, but the output does not tell what similarities are, and that is where the one-hot representation comes in to give the ability to measure the similarities. To conclude, the objective of this algorithm is to deduce the objective function, and that is the error function. Eq. (2.10) mathematically represents this:

$$\mathrm{J} = \sum_{j=1}^{k} . \sum_{i=1}^{n} \| x_i{}^j - c_j \|^2 \qquad (2.10)$$

where $\| x_i{}^j - c_j \|^2$ represents a gap amongst a feature position and the center c_j of similar groups and signifies the gap of *n* feature position and corresponding similar groups centers.

2.5 Conclusion

This chapter discussed deep learning and CNNs. The CNN architecture was included, and its layers namely, convolutional, pooling, dropdown/softmax layers were discussed. Image processing is explained by firstly elaborating what an image is and why it is important. Differences in 1D and 2D computations in CNN. Padding role in CNN image classification and advantages of CNNs and some challenges pertaining this technique in contrast along with other algorithms in FR systems. The motivation of CNNs from the neuroscience perspective and outlined the fundamentals behind their development. Following this, was the discussion of image classification using CNNs. Finally, FR CNN related work, referenced advances on recent FR systems to tackle the challenges of illumination, feature extraction, pose, etc, aiming to identify methods that will be discussed or presented in the remaining chapters of this study. K-Means clustering and CNNs for feature extraction were reviewed. In chapter 3, the proposed method of this dissertation is discussed in detail.

CHAPTER 3

Proposed Methodology

3.1 Chapter overview

In this chapter the new proposed methodology of face recognition called, Improved Deep Learning Neural Network (IDLNN) is presented. It incorporates the application of PCA, K-Means clustering and CNN algorithms. Firstly, it begins by describing and formalising the methodology based on the three selected algorithms. Secondly, it presents the steps taken to execute these three algorithms.

3.2 CNN model selection using PCA and K-Means as a driving force

3.2.1 Introduction

In this section, PCA data compression and classification are discussed, SVM algorithm to minimize misclassification, SVC classification, CNN architecture, and results are explained. Therefore, basic concepts behind FR systems are elaborated. First, these are discussed to obtain CNN architecture. Next, the algorithm is introduced. The remainder of the work discusses future works.

3.2.2 Methodology

The proposed technique namely IDLNN comprises DCNN model which is tested for FR as they allow extraction of wide range of features from images. It utilizes PCA first for dimensionality reduction and pre-processing and k-means clustering before it enters the DCNN. This is discussed in detail in the following subsections, but let's first discuss the problem and fundamental concepts behind it.

The primary Problem Definition of a FR system is the ability to recognize a person's face from an image in a set of testing images from a dataset in the database. Illumination, pose, and facial expression affect the process of Feature Extraction and Selection. Hence, de-noising the image with a filter

helps preserve appropriate features namely edges to enhance accuracy during training process. Thus, finding the best CNN architecture and hyperparameters for image classification unleashed ways on how to manipulate the raw data. CNN input data undergone two methods, (1) application of PCA for data dimensionality reduction and classification to find eigenvectors (reshaping face images to vectors), see Section 2.3; (2) application of K-Means clustering to the PCA output. Whence, CNN takes the output of the K-Means and use it to optimise the model. The optimisation of the IDLNN method is determined by the manipulation of the architectural parameters, such as, defined number of convolutional layers, convolutional filters size per layer, FC layers filter size, number of classes, and the softmax layer. Hyperparameters are held fixed during the training process, but the learning parameters such as, optimiser, learning rate, batch size, weight initialisation method were constantly manipulated to enhance the results.

It is crucial to fully understand the essential concepts behind FR to effectively comprehend the proposed technique. Pre-processing was already discussed in Section 2.2.6.3, Feature Extraction (FE) using PCA in Section 2.3.1 and Support Vector Machine (SVM) classification is considered for its effectiveness in high dimensional spaces, and this is discussed in the following Section, 3.2.3.

3.2.3 Using SVM

SVM perceptron algorithm is used to minimise misclassification errors. However, we use it to maximise the image margins. Here, the margin represents a gap pulling apart dimensions a bit lesser than their ambient space, and trainable samples nearest against the bound's conclusions. Hyperplanes of bigger edges possess less chances of overfitting and underfitting while those of tinier edges are prone to over or underfitting. The basic equation behind the margin maximisation is given by eq. (3.1):

$$\frac{w^T\left(x_{pos} - x_{neg}\right)}{||w||} = \frac{2}{||w||} \tag{3.1}$$

where w is the vector length, x_{pos} denotes a positive hyperplane and x_{neg} the negative hyperplane. The left side of eq. (3.1) signifies the margin to maximize which is the distance between the positive and negative hyperplanes. The SVM function becomes the maximisation of this margin by maximizing the righten side of the equation on constraints that the classification of samples is correct in the dataset. The following subsections as mentioned at the beginning of this section are to elaborate and illustrate the proposed contributions of our proposed methodology.

3.2.4 PCA For Data Compression and Classification

Using PCA as an unsupervised dimensionality reduction, the feature extraction algorithm is utilised to convert data to a novel attribute space with the goal of maintaining most of the relevant information by using proper values and vectors. Thus, this improves the required memory calculation of the training algorithm and prognostic execution by decreasing the dimensionality curse. To understand the training algorithm of Eigenfaces let's consider a face of $I(x, y)$ to be a 2D $N \times N$ array. Figure. 3.1 displays the original face images to demonstrate the calculation of Eigenfaces.

Figure 3.1: ORL faces Dataset.

These images are converted into vectors of size N^2 so that an image of 64×64 turns into a vector of 4096 dimension or equivalent to a point in a 4096 space dimension. The main purpose here is to get vectors that at a higher level consider the dissemination of face images in the entire image space. Since these vectors are the eigenvectors of the variance-covariance matrix equivalent to the original images and that they are face-like in appearance, they are referred to as "Eigenfaces" (Eleyan and Demirel, 2005). Assuming the training set of images to be $x_1, x_2, x_3 \ldots x_m$ the median computation of all these face vectors is represented by

$$\Psi = \frac{1}{M}\sum_{n=1}^{M} x_1 \tag{3.2}$$

and because each face *c* varies from the average face Ψ it can be subtracted by

$$\Psi = c - \Psi \tag{3.3}$$

By considering all face vectors a matrix size of $N^2 * M$, *A* = [a_1 a_2 a_3 … a_m] can be obtain. This allows to find the variance-covariance matrix by performing *A* by A^T multiplication and because *A* has $N^2 * M$ dimensions, thus A^T has $M * N^2$ dimensions. Therefore, multiplying $N^2 * N^2$ results in N^2 of N^2 range that is computationally not useful to calculate. Hence, computing covariance matrix multiplying *A* by A^T results in $M * M$ matrix with *M* (presuming *M* << N^2) eigenvectors of size *M*. This results to:

$$Covariance = A^T A \tag{3.4}$$

where A is formed by the different vectors, i.e., $A = [\Phi_1, \Phi_2 \ldots \Phi_M]$. The above covariance (*C*) formula helps to calculate eigenvalues and eigenvectors. In

practice, the dimension of C is $N * M$. Considering the eigenvectors v_i such that

$$A^T A v_i = \lambda_i c \tag{3.5}$$

Thus, remultiplying both sides of the above formula by *A*, it becomes

$$A^T A v_i = \lambda_i c \tag{3.6}$$

where, it can be noted that Av_i are the eigenvectors and λ_i equivalent to μ_i that are the eigenvalues of $C = AA^T$ and $\mu_i = Av_i$. Therefore, this analysis concludes that C' and C eigenvalues are the same and the relation on their eigenvectors is $\mu_i = Av_i$. Therefore, constructing $M * M$ the M eigenvectors, v_i, of variance-covariance matrix provides M the biggest eigenvalues of C'. Now taking the normalised training faces x_i and characterize every single face vector in a direct combination of the most exceptional K eigenvectors (where K < M) results to

$$X_i - \Psi = \sum_{j=1}^{K} w_j \mu_j \tag{3.7}$$

where, μ_j are called Eigenfaces. Figure 3.2 depicts eigenfaces. Given a new face (Γ), preprocessing is performed to make sure the face is positioned in the center of the image and that it has identical dimensions as the face being trained. Therefore, it is transformed into its eigenface components by subtracting the face from the average face **Ψ**, as in

$$\Phi = \Gamma - \Psi \tag{3.8}$$

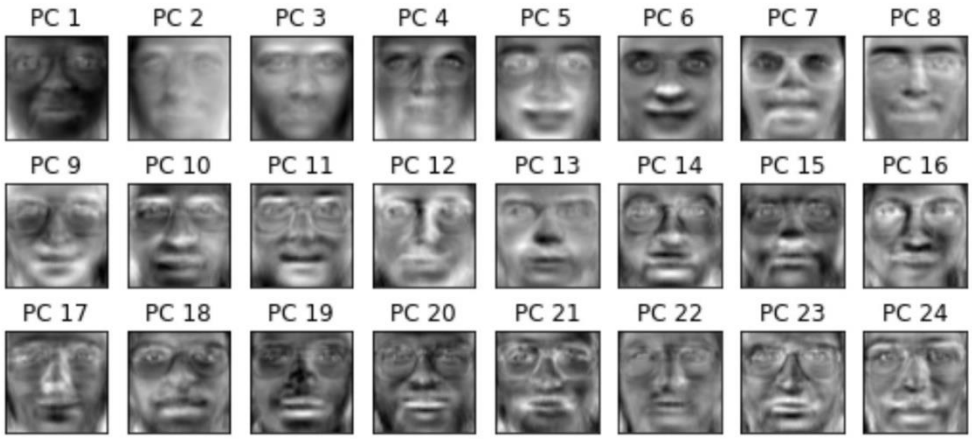


Figure 3.2: 24 Eigenfaces with Highest Eigenvalues.

Thus, the normalised vector is projected onto eigenspace to acquire the direct combination of eigenfaces simply by

$$\Phi = \sum_{j=1}^{K} w_j \mu_j \tag{3.9}$$

From this projection, the vector of the coefficient generates weights to form a feature vector,

$$\Omega^T = [\, w_1 \;\; w_2 \; w_3 \;\; \ldots \;\; w_m \,] \tag{3.10}$$

This feature vector recounts the involvement of each eigenface in representing the input image, treating the eigenfaces as a foundation set for face images. Then, a pattern of basic recognition algorithm is used to find which value of already defined face classes, if any, best defines the face. The class of a face Ω_k is computed by assessing the region of the results of the eigenface representation over a small number of face images of each individual. Classification is achieved by subtracting the feature vector from the training face image to get the minimum distance between the training and testing vectors. This is basically the Euclidean Distance in the middle of an input face

image and faces classes. The aim here is to get the class *k* of the face that minimizes the Euclidean Distance, as in:

$$\varepsilon_k = \min_k \| (\Omega - \Omega_k) \| \quad (3.11)$$

where Ω_k is a vector describing the k^{th} face's class. If ε_k is lower than the forbearance level T_k, then it is recognised with *k* face from the training face image, otherwise, the face is not matched with any faces in the training set. Though this algorithm is computationally inexpensive it is sensitive to illumination, and it requires frontal view of the face to work effectively.

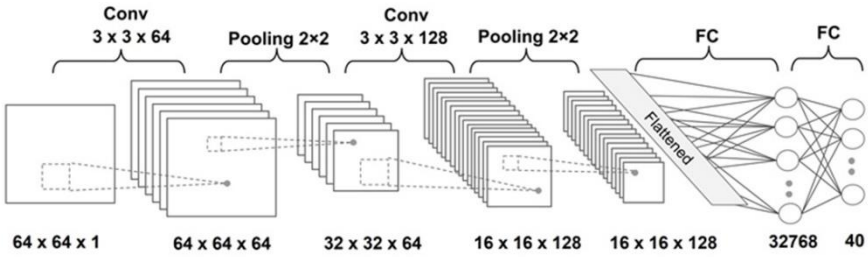
3.2.5 Classification using SVC

Support Vector Classification (SVC) class can perform binary and multi-class classification on a dataset. In this dissertation, it is preferred with Radial Basis Function (RBF) kernel, thanks to its effectiveness in high dimensional spaces (Hsu, Chang, and Lin, 2016). When training a Support Vector Machine (SVM) with this kernel *C* and gamma ought to be thought of. The kernel *C* trades off all misclassifications of training face images against maximisation of the decision surface. The aim is to get a high *C* to ensure all training faces are classified correctly. Chosen $C = 1000$ as this value showed a significant improvement. However, gamma (γ) verifies how much influence a single training face image has and the larger γ is, the closer other face images to be affected. The value of this parameter was given as gamma=0.001 as it also improved classification.

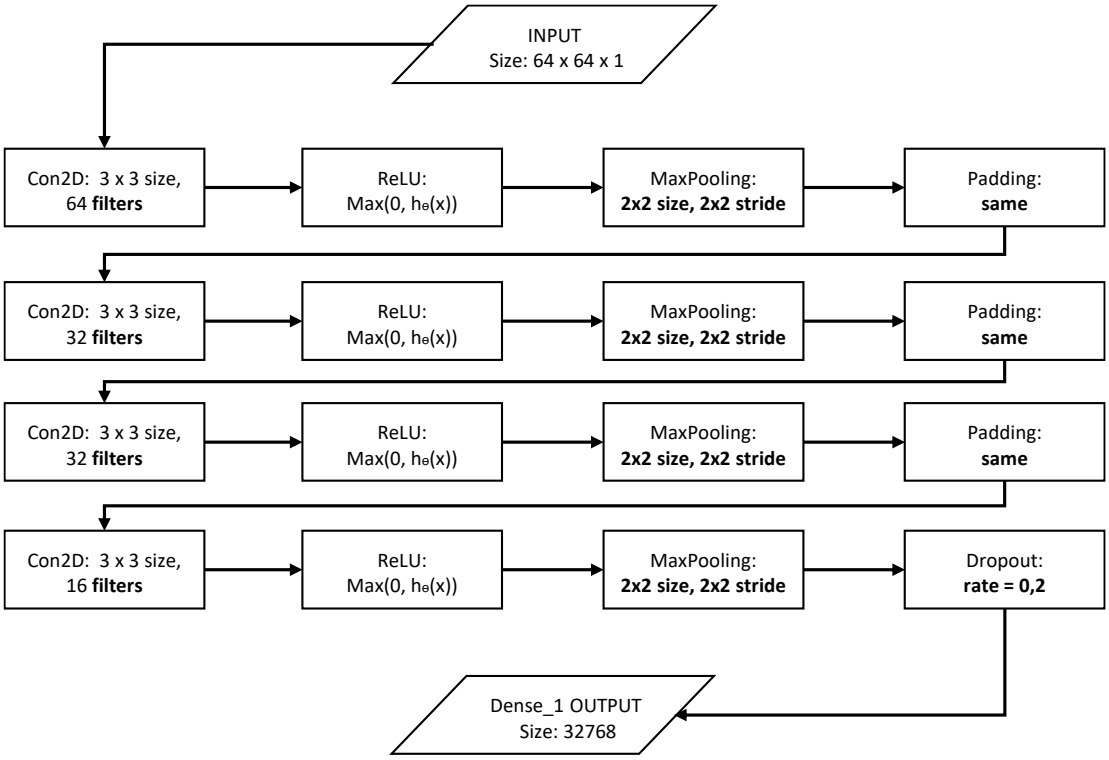
3.2.6 Convolutional Neural Networks (CNN)

In chapter two, Section 2.2, the properties, and motivations behind CNNs in FR are explained. In this section, the CNN architecture used in this dissertation

is presented. Figure 3.3 illustrates the proposed multilayer CNN architecture for this dissertation.



(a)



(b)

Figure 3.3: This study's suggested convolutional neural network architecture. (a) shows a complete architecture, (b) shows the layers in detail.

Figure 3.4 depicts the flowchart of the proposed method. Convolutional neural networks consist of numerous layers as mentioned in Section 2.2 of chapter two, namely, convolutional, Pooling (P), and Fully Connected (FC) layers.

Each layer transforms its volume of Rectified Linear Unit (ReLU) activations to another layer through different functions. They are also comprised of a dropout layer for overfitting (Srivastava et al., 2014). In this dissertation, three main types of layers are incorporated, which are, Convolution (Conv) layer, Pooling layer, and Fully Connected layer. The layers of this architecture are illustrated below as follows:

- Input layer consists of raw image data with a dimension vector of batch size 64 x 64 x 1. This means batch size x height x width x grayscale images (1 channel images).

- Convolutional layer (Conv_1) entails a batch size x 64 x 64 x 64 with a kernel of 3 x 3 and 64 output feature maps. This layer computes the convolutions between the neurons and the various patches in the input. Detailed in Section 2.2.2.

- Pooling layer (Pooling_1) with batch size x 32 x 32 x 64 with 64 representing output feature maps. This layer samples the previous layer and results in reduced dimensions. This layer assists only in keeping important elements and the Max pooling is used to keep maximum value in each *K x K* window, see details in Section 2.2.3. Max pooling presents some sort of local invariance, consequently, it helps to produce vigorous features against noise in the input data. Here, the same padding is used to get the same size output as the same as that of the input, vector *x*. By doing so, it computes the padding parameter, p, corresponding to the filter size to make sure that the input size is the same as the output size as required (Vahid Mirjalili and Sebastian Raschka, 2017).

- ReLU is an activation function that is applied to the matrix to make it linear (Jarrett et al., 2009.). It is linear in the positive dimension but zero in the negative dimension. Its linearity in the positive dimension has an alluring attribute that inhibits non-saturation of gradients,

though one-half of the actual line its gradient is zero. It takes the summed weighted input from the node and transform it into the activation of the node or output for that input.

- Convolutional layer (Conv_2) with batchsize x 16 x 16 x 128 with kernel of 3 x 3 and 128 output feature maps.

- Pooling layer (Pooling_2) with batchsize x 16 x 16 x 128 with 128 representing output feature maps.

- Fully Connected (FC_1) layer with batchsize x 32768.

- Dropout layer or output shape 16 x 16 x 128.

- Fully Connected (FC_2) and softmax layer compute the output scores resulting in size of 1 x 1 x 40, where 40 defines the value of classes in the training dataset. According to (Long, Shelhamer and Darrell, 2015), this layer can be trained end-to-end, pixels-to-pixels on semantic image segmentation (classification and localisation) rather than only to predict the dense outputs.

3.3 Training and simulation of CNN

From the ORL database of 40 classes a CNN was implemented. First, eigenfaces and feature prognosis vectors are computed for the faces in the database, but this is detailed in Section 3.4.2.3.1. Figure 3.2, Section 3.2.4, showcases Eigenfaces obtained during training. Figure 3.5 depicts the cumulative explained variance ratio.

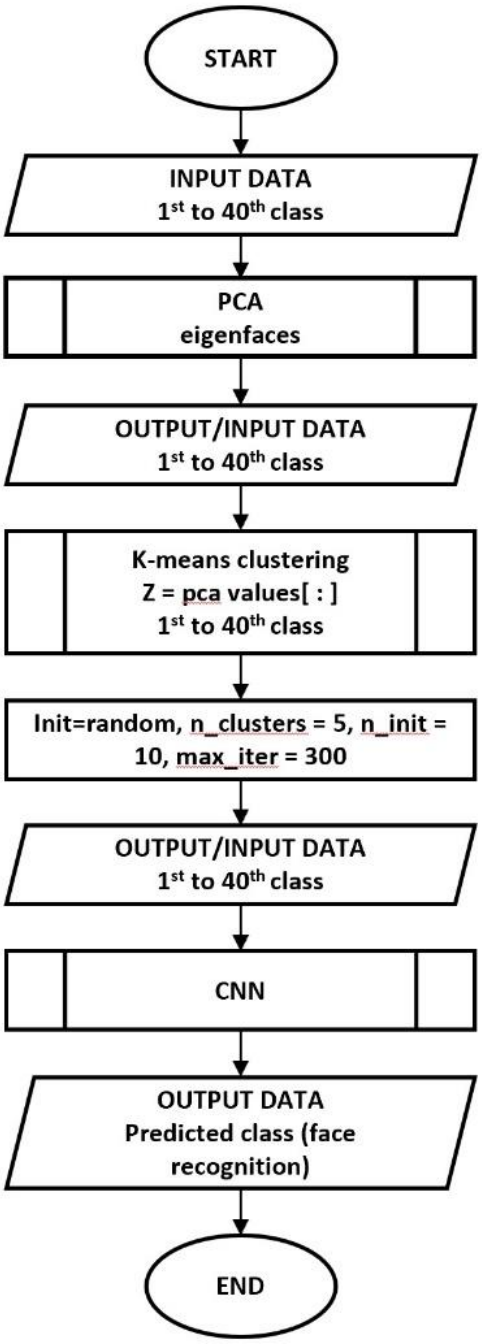


Figure 3.4: PCA + K-Means Clustering + CNN flow diagram used in this dissertation.

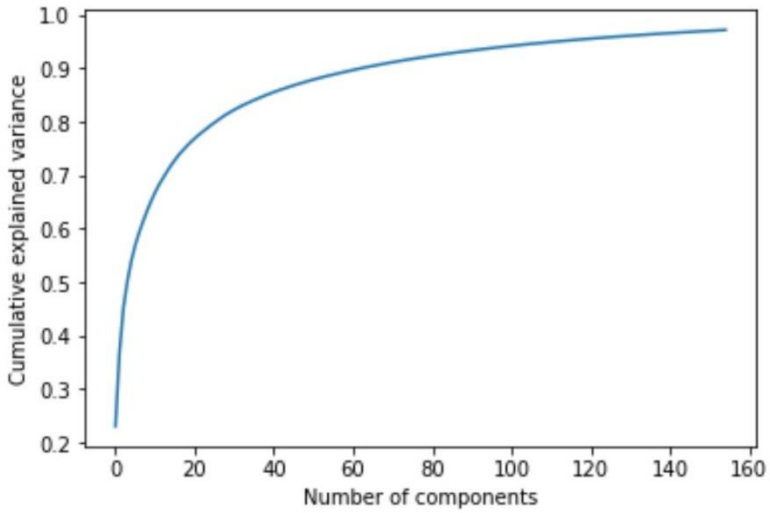


Figure 3.5: Cumulative explained variance.

Then, cluster these feature projection vectors using k-means clustering and feed them as inputs in the CNN. Figure 3.6 and 3.7 depicts the schematic diagram of the proposed CNN training phase and classification simulations.

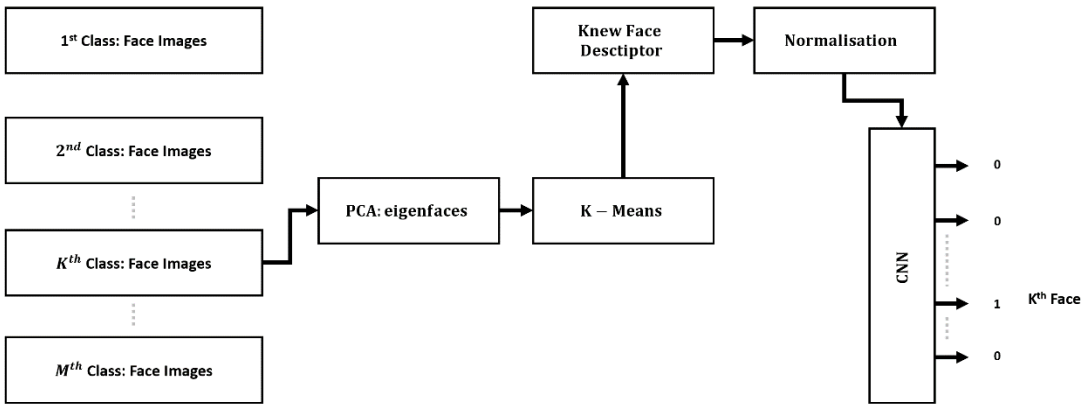


Figure 3.6: Convolutional Neural Network training phase.

When FR is considered for a new face image, the feature projection vector of this new face is calculated from the eigenfaces, classified with SVC, clustered using the KMeans algorithm to separate samples in *n* groups of equal variance and choosing centroids that minimise a criterion known as inertia and then this face image gets new face descriptors. These new face descriptors are then supplied to convolutional neural networks and this network is modelled with

these descriptors, where the network outputs are compared. Decision-making is based on the minimum and maximum outputs. At the highest output, this new face is chosen to be the part of the class of a person with this highest output.

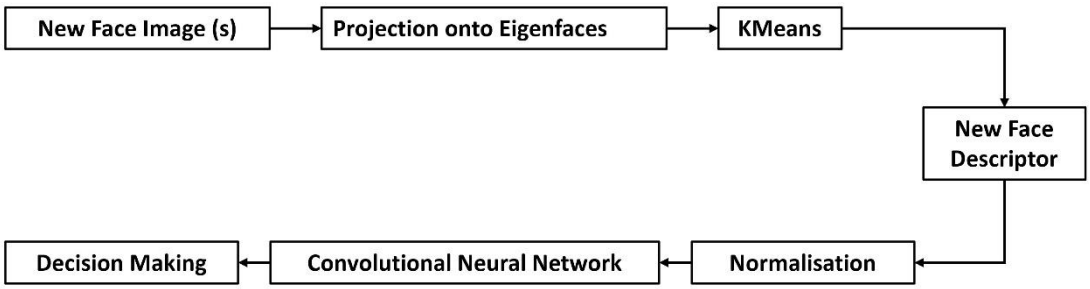


Figure 3.7: Convolutional Neural Network Classification Simulations.

3.4 Conclusion

This chapter discussed the suggested method in detail commencing by the usage of SVM and followed by SVC classification. Subsequently, discussed data compression and classification in PCA. Finally, discussed the CNN algorithm, training, and simulation, and illustrated these in Section 3.3, Figures 3.6 and 3.7. Chapter four, next, will be illustrating and discussing empirical findings or experimental results of the proposed method in details.

CHAPTER 4

Empirical Findings (Experimentation)

4.1 Overview

The upcoming sections introduce the experimental setup and implementation of this pilot study.

4.2 Experimental Setup

4.2.1 Development environment

All conducted simulations were done on Windows 10.1 with Intel® Core™ i7-9750H 9th Generation at 2.60GHz, 32.0 GB memory, GPU 0 (Intel® UHD Graphics 630), GPU 1 (NVDIA GeForce GTX 1650 with Max-Q Design), 2667 MHz RAM speed and A5.2 GB cashed. Software coding experiments were executed in Python using Keras (Haghighat and Juanes, 2021) application programming interface (API) with TensorFlow as a backend. PCA, K-Means clustering, and CNN models were created with this high-level API computed in the background with the help of TensorFlow. Python libraries and dependencies such as NumPy, Pandas, and matplotlib to generate figures were used. Jupyter Notebook interactive computational environment was used.

4.2.2 Data

To examine the proposed algorithm, the ORL_faces dataset was used. This dataset was first introduced between April 1992 and 1994 (Samaria and Harter, 1994). It comprises 10 different face images per class for every 40 distinct persons. These images are in greyscale format and were captured in different lighting conditions and occasions. Per class, one person is represented with different facial expressions such as an open or close mouth, with or without glasses, and with different head orientations (Abbas, Safi and Rijab, 2017). These images are used for training and testing. Figure 3.1 in

Section 3.2.4 shows among the views some of these faces. The shape (dimensions) of the images is 64 * 64 pixels, and each with a sample vector of 64 * 64 = 4096 dimensions.

4.2.3 Implementation

4.2.3.1 PCA Algorithm Steps

As stated already from the beginning of this study in the abstract section, PCA is the first algorithm implemented. The dataset is split already from this algorithm to 20 % test size and 80% training and its results are observed. This split was adjusted accordingly to assist on delivering high accuracy. PCA algorithm extracts the following steps (Vahid Mirjalili & Sebastian Raschka, 2017): (1) data standardization, (2) covariance construction, getting covariance matrix eigenvalues, and eigenvectors, (4) sort proper values in reducing order sequence on positioning proper vectors. During standardizing of data, the centroid is at the origin, all the variables consist of the same variance, making all variables to get a naught average and the square root of the variance. Parameters such as the number of components (n_components) were assigned a value of 155, and random state (random_state) of 42 to initialize the internal random number. Thereafter, SVC classification with RBF kernel, penalty parameter C of 1000 to add penalty to each misclassified data, and gamma parameter of 0.001 to monitor the gap that influences the single point of training was utilized to enhance the accuracy (Liang et al., 2022). Figure 3.5 Section 3.3 represents the cumulative explained variance obtained.

4.2.3.2 K-Means Clustering Algorithm Steps

Again, this algorithm was announced at the beginning of this study at the abstract section and explained in depth in Section 2.4. This algorithm takes the output data from the PCA, in this case, eigenfaces, and assigned 40 face classes and used hyperparameters such as, initialise (init) set to random,

number of clusters (n_clusters) set to 5, number of initialisations (n_init) set to 10, maximum iterations (max_iter) set to 300, and random_sate = 42. K-Means algorithm consists of phases namely: (a) arbitrarily choose k geometric centers form sample points like the preliminary group centres, (b) designate every sample of a closest geometric centre $\mu^{(j)}$, $j\epsilon\{1,\dots,k\}$, (c) change a position of the geometric centres towards the middle of the samples that were designated to it, (d) recite steps 2 and 3 until the cluster tasks do not alter or number of iterations is attained. Thereafter, fit this into the new dataFrame, and define the lowest sum of Euclidean distance. This new dataFrame becomes the CNN input data.

4.2.3.3 CNN Algorithm Steps

In Section 3.2.6, the model architecture with its layers is detailed but the hyperparameters and parameters were left out for this section. Therefore, the first step was to normalise the new dataFrame of images to prepare for CNN training. The training data was fitted in the standard scaler and both the test and train data were transformed and reshaped to 64 * 64 pixels and to channel (C_n) 1 for greyscale images. Subsequently, the classes of 40 face images were converted to matrix of binary values using to categorical[4] function provided by Keras to represent different categories of the data. This function yields the output binary values of 1 or 0. Training the model, certain parameters were considered to enhance the accuracy such as, the crossentropy loss parameter for computation between the labels and predictions, Adam optimisation parameter of 0.0001 learning rate, batchsize of 256, verbose of 2 and callbacks. Splitting the dataset to a test size of 20 % and 80 % training improved the accuracy and this proposed methodology obtained 99% in 90

[4] To categorical function is available from: Python Keras | keras.utils.to_categorical() - GeeksforGeeks

epochs. Figures 4.1 and 4.2 display the graphical representations of the model accuracy and loss after training. Table 4.1 depicts confusion matrix.

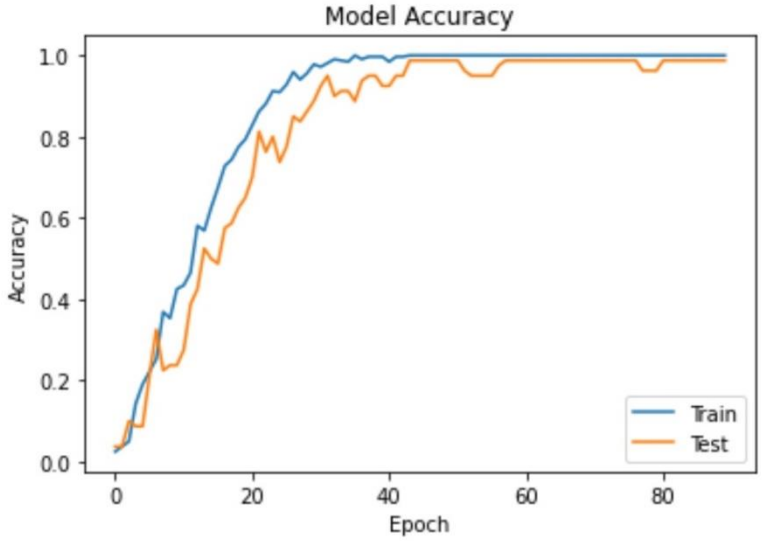


Figure 4.1: Model Accuracy of 99%.

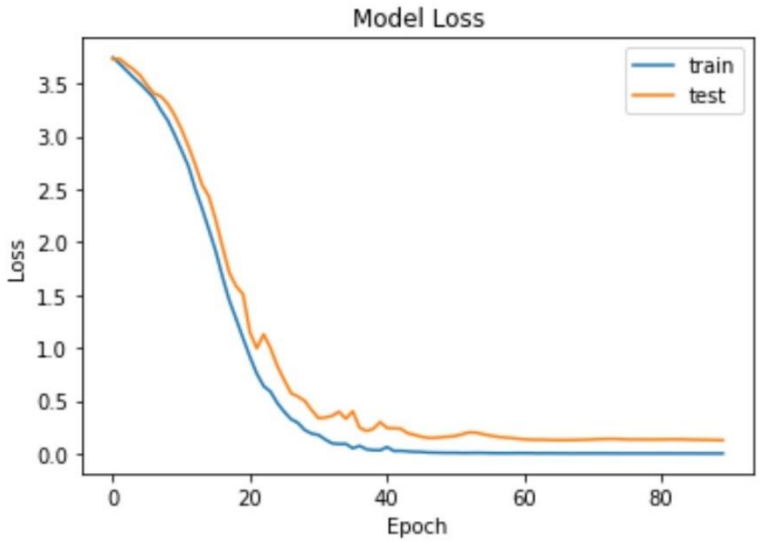


Figure 4.2: Model Loss of 1%.

Table 4.1 Confusion Matrix

Predicted / Real	Numerical Form					
	Class_1	***Class_2***	***Class_3***	***Class_4***	***Class_5***	***Class_6***
Class_1	3	0	0	0	0	0
Class_2	0	1	0	0	0	0
Class_3	0	0	2	0	0	0
Class_4	0	0	0	2	0	0
Class_5	0	0	0	0	4	0
Class_6	0	0	0	0	0	4

Table 4.2 Shows classification report where the f1-score accuracy is 99%.

Table 4.2: F1-Score of 99%

Scoring Metrics	Classification Report			
	Precision	***Recall***	***F1-Score***	***Support***
Accuracy			0.99	80
Macro avg	0.99	1.00	0.99	80
Weighted avg	0.99	0.99	0.99	80

F1-score is a weighted average of the precision and recall whereby at its best it reaches a value of 1 and at worst score a 0. Thus, our model's accuracy at its best is 0.99 which is close to 1.

Figure 4.3 depicts the first 6 filters out of 64 filters from layer 2 of this proposed paradigm with 18 face images. Each filter with one row and each channel with one column. Inhibitory weights are the dark squares and excitatory weights the

light squares. Excitatory inputs are often positively weighted and valued while inhibitory inputs are negatively weighted and valued. Each neuron has a fixed threshold for firing, and this can be achieved by an excitatory input.

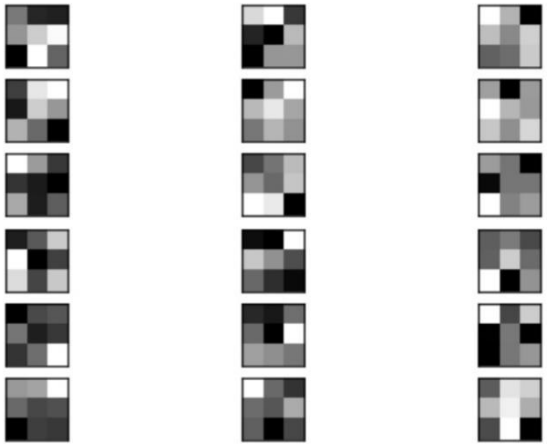


Figure 4.3: Filter of the second layer.

To view the feature maps of the input face image, a visualisation is performed. This visualisation entails input features that are identified and maintained from the feature maps. Figure 4.4 depicts this in the first conv layer by showing different face image sorts of various emphasized features. A few of these highlights can be seen as a focus on lines, and background of the foreground.

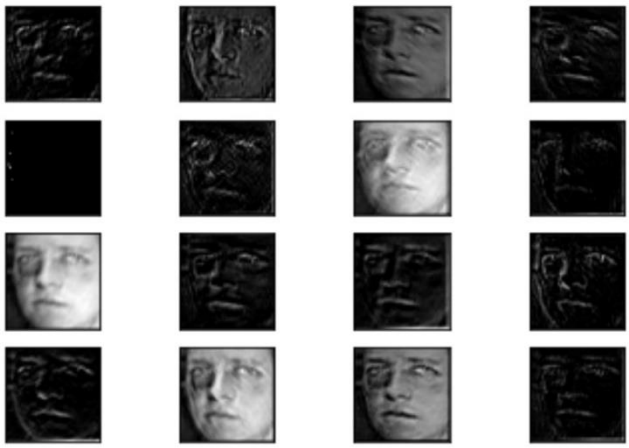


Figure 4.4: First convolutional layer feature maps.

Table 4.3 represents the F1-Score results obtained in contrast to other face algorithms such as PCA+SVC, SVM, Naive Bayes (NB), kNN and the proposed CNN algorithm.

TABLE 4.3: F1-Score Performance Analysis for the Proposed Methodology

Test Results							
Database	***Training Images %***	***Testing Images %***	***SVM***	***NB***	***KNN F1-Score Performance %***	***PCA + SVC F1-Score Performance %***	***CNN F1-Score Performance %***
ORL							
	50	50	94	48	61	93	91.5
	60	40	94	63	74	94	94
	70	30			76	96	91
	80	20	96	82	84	97	99

4.3 Conclusion

This chapter covered the basic implementation applied in this study. It started by describing the system software setup. Next, defined the ORL dataset with face images used to evaluate, test, and train the CNN model. Thereafter, it explained the steps taken to process the original data using PCA algorithm. Afterward, cited the steps taken using the K-Means clustering algorithm to manipulate the data to be ready as a CCN input data. Finally, discussed the CNN algorithm hyperparameters used to train the CNN model with the simulations results presented graphically.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1 Summary and Conclusions

This study aimed to find alternatives to improve the FR systems by investigating and exploring the fundamental concepts in the current literature, to enhance the accuracy and training time of the CNNs. Chapter 2 provided the theoretical background literature of the FR systems. In chapter 3 the methodology used in this study was presented and discussed with the well-known algorithms in the field of machine learning and face recognition such as PCA, K-Means clustering and CNNs. In chapter 4 the implementation process was explained, beginning with the development environment, to the dataset used in this study, and finally portrayed all the steps taken into training and testing the three identified algorithms of this study. Chapter 4 also presented the results obtained during the training of the proposed CNN model.

In this study, a face recognition system, based on improved deep learning neural network is proposed and implemented on Python programming language. It is based on PCA preprocessing, k-means clustering followed by convolutional neural networks (CNNs). The feature prognosis vectors acquired from the PCA technique are utilized as input vectors in k-means clustering and then normalised in preparation for the training and testing of the CNN architecture. This method performed better than the PCA+SVC, SVM, KNN and NB and obtained the best accuracy of 99% at 90 epochs. The results suggests that CNN surpasses the abovementioned algorithms for the ORL database and that there is a room for improvement.

5.1.1 Contribution

Though CNNs (LeCun et al.,1989) remain the specialised sort of neural networks in manipulating data with identified grid-like topology and a group of

models motivated by the functionality of a visual cortex in the human brain whilst recognising objects, today they still face challenges of illumination, noise, and feature extraction. This dissertation aimed to contribute on addressing the FR system challenges by improving deep CNN accuracy and performance. This study will also be of value to current and future researchers interested in FR using deep CNN. Also, to highlight the benefits of NN in FR such as nonlinearity. Basically, a neuron is a nonlinear device, and its nonlinearity is a highly important property (Haykin, 1994). To improve the techniques of FR such as model-based and appearance-based by investigating the methodologies of PCA and LDA to construct and create 3D model of human face. PCA technique uses eigenfaces which are just merely 2D spectral facial images that are composed of grayscale features (Das, 2015). The deliberate contributions and initial assumptions from other work are, a study to motivate CNNs image classification for FR and their issues, more on the selection of parameters to train the model, combination of PCA, K-Means clustering, and CNNs to maintain valuable features and further the enhancement of CNNs training time, and finally, K-Means application to recreate the PCA output data for CNNs as input data.

5.2 Limitations and Future work

The size of the network was limited to reduce the training time, but also, due to time span of this dissertation. Thus, if time was not the issue different parameters would have been evaluated and tested in different network configurations. The image dataset would have been increased to large datasets such as Googles dataset. Instead of only using PCA and K-Means clustering for preprocessing purposes other preprocessing possibilities would have been investigated and applied in the first layer. Regardless of the encouraging findings the hyperparameters can be tuned more to even obtain better results, for example, convolutional layers, these can be manipulated by changing the parameters to observe the output results.

For future work, it would be suggested that different databases are tested using this approach and that a live face recognition system is used to observe how accurate the classification of this model with different ethnicities is. It is important to note that the algorithms applied in this dissertation can be utilized to any image classification problem regardless of the dataset used. PCA was chosen due to its capability to reduce data dimension allowing the usage of smaller networks to reduce the training time. As for the CNN, other forms or steps of preprocessing might be utilized to substitute the input layer.

Due to the time constraints concerning this dissertation, this led to limit the size of the network and the number of times the training is run. Therefore, for future work, the assessment of larger networks should be implemented and so as more parameter tuning such as the learning rate. Moreover, time evaluation highly depends on the system hardware to speed up the training time. Training larger datasets enhances the accuracy of the network (Schroff, Kalenichenko and Philbin, 2015). Finally, future work could consist of the system development of practical applications such as identity verifications at the airport or by the police.

REFERENCES

Cireşan, D., Meier, U., Masci, J. and Schmidhuber, J. (2012). Multi-column deep neural network for traffic sign classification. *Neural Networks*, 32, pp.333–338.

Prateek J.. (2017). *Artificial Intelligence with Python.* Second edition. Packt Publishing Ltd. UK Birmingham.

Goldstein, A.J., Harmon, L.D. and Lesk, A.B. (1971). Identification of human faces. *Proceedings of the IEEE*, 59(5), pp.748–760.

Jain, A., Hong, L. and Pankanti, S. (2000). Biometric identification. *Communications of the ACM*, 43(2), pp.90–98.

Bowles N.. (2016). *I Think My Blackness Is Interfering: Does Facial Recognition Show Racial Bias*. *The Guardian*.

Zavyalova, V. (2018). *In your face: New facial recognition system catches criminals in Russia*. Science & Tech, *Nov 27 2018*. Available at: https://www.rbth.com/science-and-tech/329587-facial-recognition-system-catches-criminals. .

Arya, S., Pratap, N. and Bhatia, K. (2015). Future of Face Recognition: A Review. *Procedia Computer Science*, 58, pp.578–585.

.

Zafeiriou, S., Kotsia, I., and Pantic, M. (2014). Unconstrained Face Recognition. *Middlesex University London & Imperial College London*, no.2, pp. 22.

Bhaskar, B., Anushree, P.S., Shree, S.D. and Prashanth, K.V.M. (2015). Quantitative Analysis of Kernel Principal Components and Kernel Fishers Based Face Recognition Algorithms Using Hybrid Gaborlets. *Procedia Computer Science*, 58, pp.342–347.

Abhishree, T., Latha, J., Manikantan, K. and Ramachandran, S. (2015). Face Recognition using Gabor Filter Extraction with Anisotropic Diffusion as a pre-processing technique. *Procedia Computer Science,* no. 45, pp. 312-321.

LeCun, Y., Boser, B.E., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W.E., and Jackel, L.D. (1989). Handwritten Digit Recognition with a Back-Propagation Network. *Neural Information Processing Systems (NIPS),* pp. 396-404.

Das, R. (2015). *Biometric technology: authentication, biocryptography, and cloud-based architecture*. CRC Press, UK.

Singh, S. and Prasad, S.V.A.V. (2018). Techniques and Challenges of Face Recognition: A Critical Review. *Procedia Computer Science*, no. 143, pp.536–543.

Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation. Macmillan Publishing Company, New York.*

Vahid Mirjalili and Sebastian Raschka, (2017). *Python Machine Learning,* Packt Publishing Ltd, UK Birmingham, UK.

Stan Z. Li and Anil K. Jain, 2011, *Handbook of Face Recognition*, Springer-Verlag, London Limited.

Shamshad Ansari (2020). *Building computer vision applications using artificial neural networks : with step-by-step examples in OpenCV and TensorFlow with Python*. Berkeley, California. Apress. Centreville, VA, USA.

Shrivakshan, G. (2012). A Comparison of various Edge Detection Techniques used in Image Processing. *IJCSI International Journal of Computer Science Issues*, 9(1).

McCulloch, W.S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4), pp.115–133. Doi: 10.1007/BF02478259.

Habrman, D. (2016). Face Recognition with Preprocessing and Neural Networks.

Hubel, D.H. and Wiesel, T.N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, [online] 160(1), pp.106–154.

Quiroga, R.Q., Reddy, L., Kreiman, G., Koch, C. and Fried, I. (2005). Invariant visual representation by single neurons in the human brain. *Nature*, [online] 435(7045), pp.1102–1107.

Mouton, C., Myburgh, J.C., Davel, M.H. (2020). Stride and Translation Invariance in CNNs. In: Gerber, A. (eds) Artificial Intelligence Research. SACAIR 2021. Communications in Computer and Information Science, vol 1342. Springer, Cham. https://doi.org/10.1007/978-3-030-66151-9_17.

Parkhi, O., Vedaldi, A. and Zisserman, A. (2015). ´Deep Face Recognition`, Visual Geometry Group, University of Oxford.

Parkhi, O.M., Simonyan, K., Vedaldi, A. and Zisserman, A. (2014). A Compact and Discriminative Face Track Descriptor. *2014 IEEE Conference on Computer Vision and Pattern Recognition*. pp. 1693-1700.

Krizhevsky A., Sutskever I. and Hinton G.E., 2012, ´Advances in Neural Information Processing Systems`, pp. 1097-1105

Qin, J., Pan, W., Xiang, X., Tan, Y., & Hou, G. (2020). A biological image classification method based on improved CNN. *Ecol. Informatics, 58*, 101093.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 2818-2826, doi: 10.1109/CVPR.2016.308.

He, K.M., Zhang, X.Y., Ren, S.Q. and Sun, J. (2016). *Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, 27-30 June 2016, 770-778*.

Huang, G., Liu, Z., Van Der Maaten, L. and Weinberger, K.Q. (2017). Densely Connected Convolutional Networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 2261-2269

Zheng L., Zhao Y.L., Wang S.J. and Tian Q., 2016, ´Good Practice in CNN Feature Transfer`, pp. 1604.00133.

Mishra, N.K., Dutta, M. and Singh, S.K. (2021). Multiscale parallel deep CNN (mpdCNN) architecture for the real low-resolution face recognition for surveillance. *Image and Vision Computing*, 115, p.104290.

Zhang, Y., Zhao, D., Sun, J., Zou, G. and Li, W. (2015). Adaptive Convolutional Neural Network and Its Application in Face Recognition. *Neural Processing Letters*, 43(2), pp.389–399.

Kamencay, P., Benco, M., Mizdos, T. and Radil, R. (2017). A New Method for Face Recognition Using Convolutional Neural Network. *Advances in Electrical and Electronic Engineering*, 15(4), pp. 663–672.

Oo, S.L., and Oo, A.N. (2019). Child Face Recognition with Deep Learning. *2019 International Conference on Advanced Information Technologies (ICAIT)*, 155-160.

Sharma, S., Bhatt, M.V., and Sharma, P. (2020). Face Recognition System Using Machine Learning Algorithm. *2020 5th International Conference on Communication and Electronics Systems (ICCES)*, pp. 1162-1168.

Goodfellow, I., Bengio, Y. and Courville, A. (2016). *Deep Learning*, MIT Press, Cambridge, MA.

Eleyan, A., Demirel, H. (2005). Face Recognition System Based on PCA and Feedforward Neural Networks. In: Cabestany, J., Prieto, A., Sandoval, F. (eds) Computational Intelligence and Bioinspired Systems. IWANN 2005. Lecture Notes in Computer Science, vol 3512. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11494669_115.

Bellman, R. (1957) *Dynamic Programming.* Princeton University Press, Princeton, NJ. - References - Scientific Research Publishing.

Xiang, X., Yang, J. and Chen, Q. (2015). Color face recognition by PCA-like approach. *Neurocomputing*, 152, pp.231–235.

Sharma, R. and Patterh, M.S. (2015). A new pose invariant face recognition system using PCA and ANFIS. *Optik*, 126(23), pp.3483–3487.

Zhou, C., Wang, L., Zhang, Q. and Wei, X. (2013). Face recognition based on PCA image reconstruction and LDA. *Optik*, vol. 124, pp. 5599-5603.

Gumus, E., Kiliç, N.Z., Sertbas, A., and Ucan, O.N. (2010). Evaluation of face recognition techniques using PCA, wavelets and SVM. *Expert Syst. Appl., 37*, 6404-6408.

Faraoun, K.M., and Boukelif, A. (2007). Neural Networks Learning Improvement using the K-Means Clustering Algorithm to Detect Network Intrusions. *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering, 1*, 3138-3145.

Wang, X., Wang, Z., Sheng, M., Li, Q. and Sheng, W. (2021). An adaptive and opposite K-means operation based memetic algorithm for data clustering. *Neurocomputing*, vol. 437, pp.131–142.

Awcock G. J. and Thomas R., (1995), *Applied Image Processing*, Macmillan Press LTD, Houndmills, Basingstoke, Hampshire RG21 2XS and London companies and representatives.

Kleinberg, J. (2002). An impossibility theorem for clustering. In Proc. NIPS, (Vol. 2002 pp. 446–453).

Yamashita, R., Nishio, M., Do, R.K.G. and Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, 9(4), pp.611–629.

Paul, S. and Acharya, S.K. (2020). A Comparative Study on Facial Recognition Algorithms. *SSRN Electronic Journal*.

Kanade, T. (1973). Picture Processing System by Computer Complex and Recognition of Human Faces. Department of Information Science, Kyoto University.

Sirovich, L. and Kirby, M. (1987). Low-Dimensional Procedure for the Characterization of Human Faces*. Journal of the Optical Society of America A,* 4, 519-524.

Turk, M. and Pentland, A. (1991). Eigenfaces for Recognition. *Journal of Cognitive Neuroscience*, 3(1), pp.71–86.

Nair, V., & Hinton, G.E. (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. *ICML*.

Viola, P. and Jones, M.J. (2004). Robust Real-Time Face Detection. *International Journal of Computer Vision*, 57(2), pp.137–154.

Zhao, F., Li, J., Zhang, L., Li, Z. and Na, S.-G. (2020). Multi-view face recognition using deep neural networks. *Future Generation Computer Systems*, 111, pp.375–380.

Ren, S., He, K., Girshick, R. and Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. pp. 91-99.

Collins, H. (2010). Creative Research: The Theory and Practice of Research for the Creative Industries. *Singapore AVA Publications*, p.38.

Hsu, C.-W., Chang, C.-C. and Lin, C.-J. (2016). *A Practical Guide to Support Vector Classification*. Department of Computer Science`, National Taiwan University, Taipei 106, Taiwan, pp. 4-5.

Phung and Rhee (2019). A High-Accuracy Model Average Ensemble of Convolutional Neural Networks for Classification of Cloud Image Patches on Small Datasets. *Applied Sciences*, 9(21), p.4500. doi: 10.3390/app9214500.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R. and Bengio, Y. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*,15 (56):1929-1958.

Jarrett, K, Kavukcuoglu, K, Ranzato, MA & LeCun, Y. (2009). What is the best multi-stage architecture for object recognition? in *2009 IEEE*

12th International Conference on Computer Vision, ICCV 2009., 5459469, Proceedings of the IEEE International Conference on Computer Vision, pp. 2146-2153, doi: 10.1109/ICCV.2009.5459469.

Long, J., Shelhamer, E. and Darrell, T. (2015). Fully Convolutional Networks for Semantic Segmentation*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* Boston, MA, 7-12 June 2015, 3431-3440. doi: 10.1109/CVPR.2015.7298965.

Haghighat, E. and Juanes, R. (2021). SciANN: A Keras/TensorFlow wrapper for scientific computations and physics-informed deep learning using artificial neural networks. *Computer Methods in Applied Mechanics and Engineering*, [online] 373, p.113552. arXiv preprint arXiv: 2005.08803.

Abbas, E.I., Safi, M.E. and Rijab, K.S. (2017). Face recognition rate using different classifier methods based on PCA. *International Conference on Current Research in Computer Science and Information Technology (ICCIT)*, pp. 37-40, doi: 10.1109/CRCSIT.2017.7965559.

Samaria, F.S. and Harter, A.C. (1994). Parameterisation of a Stochastic Model for Human Face Identification*. Proceedings of the Proceedings of 1994 IEEE Workshop on Applications of Computer Vision,* Sarasota, 5-7 December 1994, 138-142.

Liang, Y., Hu, S., Guo, W. & Tang, H., 2022, Abrasive tool wear prediction based on an improved hybrid difference grey wolf algorithm for optimizing SVM, Measurement, vol. 187, p.110247.

Golbon-Haghighi, M., Saeidi-Manesh, H., Zhang, G., & Zhang, Y. (2018). Pattern Synthesis for the Cylindrical Polarimetric Phased Array Radar (CPPAR). *Progress in Electromagnetics Research M, 66*, 87-98.

Abate, A.F., Nappi, M., Riccio, D. and Sabatino, G. (2007). 2D and 3D face recognition: A survey. *Pattern Recognition Letters*, 28(14), pp.1885–1906.

Wen, G., Chen, H., Cai, D. and He, X. (2018). Improving face recognition with domain adaptation. *Neurocomputing*, 287, pp.45–51.

Schroff, F., Kalenichenko, D. and Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 815-823.

Song, A.-P., Hu, Q., Ding, X.-H., Di, X.-Y. and Song, Z.-H. (2020). Similar Face Recognition Using the IE-CNN Model. IEEE Access, 8, pp.45244–45253. doi:10.1109/access.2020.2978938.

Appendix A: Python source code sample[5]

IMPORT LIBRARIES

```
import keras
import numpy as np
import pandas as pd
import tensorflow as tf
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import matplotlib.image as mplib

from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from keras.utils import np_utils
from sklearn.model_selection import train_test_split
from keras.utils.np_utils import to_categorical
from tensorflow.keras.layers import Input,InputLayer, Dense, Activation
, ZeroPadding2D, BatchNormalization, Flatten, Conv2D
from tensorflow.keras.layers import AveragePooling2D, MaxPooling2D, Dro
pout
from tensorflow.keras.models import Sequential,Model
from tensorflow.keras.callbacks import ModelCheckpoint,LearningRateSche
duler, EarlyStopping
from tensorflow.keras import backend as K
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics

from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix
from time import time

# DATASET
faces = pd.read_csv('./face_data.csv')
faces.shape

# VIEW DATA
faces.head()

# VIEW SEABORN
sns.pairplot(faces, vars =['1', '7', '3', '4', '4095', 'target'])
```

[5] https://github.com/ZukisaNante/IMPROVED_DCNN_FACE_RECOGNITION/blob/main/SRC/PCA%20%2B%20CNN%20FACE%20RECOGNITION_FINAL%20(3).ipynb

```
plt.tight_layout()
plt.show()
# faces.hist(bins=1)
# plt.show()

# HELPER FUNCTIONS
def display_initial_faces(pixels):
    # SHOWING INITIAL IMAGES
    fig, axes = plt.subplots(6, 10, figsize=(11, 7),
                             subplot_kw={'xticks':[], 'yticks':[]})
    for i, ax in enumerate(axes.flat):
        ax.imshow(np.array(pixels)[i].reshape(64, 64), cmap='gray')
    plt.show()

def show_eigenfaces(p_c_a):
    # SHOW EIGENFACES
    fig, axes = plt.subplots(3, 8, figsize=(9, 4),
                             subplot_kw={'xticks':[], 'yticks':[]})
    for i, ax in enumerate(axes.flat):
        ax.imshow(p_c_a.components_[i].reshape(64, 64), cmap='gray')
        ax.set_title("PC " + str(i+1))
    plt.show()

X = faces.drop('target', axis = 1)
y = faces['target']

print (np.array(X).shape)


faces.corr()
display_initial_faces(X)
faces.describe()
# Divide dataset
(X_train_data_set, X_test_data_set, Y_train_data_labels, Y_test_data_se
t ) = train_test_split(X, y, test_size=0.2, random_state=42) #without
random state PCA is 97 %
```

Principal Component Analysis

```
p_c_a=PCA(n_components=155)
p_c_a_values= p_c_a.fit_transform(X_train_data_set)
var = p_c_a.explained_variance_ratio_
p_c_a.components_[0]
# Determine sets compression
var1 = np.cumsum(np.round(var,decimals = 4)*100)
var1
```

```
plt.plot(np.cumsum(p_c_a.explained_variance_ratio_))
plt.axis("tight")
plt.xlabel('Component Quantity')
plt.ylabel('Cumulative explained variance')
# Display eigenfaces only works with X_train in p_c_a instead of faces
show_eigenfaces(p_c_a)

# Directing the trained data to PCA
print("Projecting the input data on the eigenfaces orthonormal basis")
Xtrain_pca = p_c_a.transform(X_train_data_set)

# Classify initialization and fit training data

clf_init = SVC(kernel='rbf',C=1000,gamma=0.001)
clf_init = clf.init.fit(Xtrain_pca, Y_train_data_labels)
clf_init

# Perform testing and get classification report (WITHOUT random_state=4
2 IT GETS 97% AND THIS WAS TO IMPROVE CNN ACCURACY)
print("Predicting people's names on the test set")
t0 = time()
Xtest_pca = p_c_a.transform(X_test_data_set)
y_pred = clf_init.predict(Xtest_pca)
print("done in %0.3fs" % (time() - t0))
print(classification_report(Y_test_data_set, y_pred))
```

Testing Different Algorithms

```
# 1: Apply Decision Tree
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
dt = tree.DecisionTreeClassifier(criterion="entropy")
dt.fit(X_train_data_set, Y_train_data_labels)
ypred = dt.predict(X_test_data_set)

# 2: Confusion Matrix
confusion_matrix(Y_test_data_set, ypred)

# 3: Classification Report
print(classification_report(Y_test_data_set, ypred))

# 4: Implementing K-Nearest Neighbors
from sklearn.neighbors import KNeighborsClassifier
# Classify KNN
knn = KNeighborsClassifier(n_neighbors=5)
# Fitting training data in KNN
```

```
knn.fit(X_train_data_set, Y_train_data_labels )
# Predict
ypred = knn.predict(X_test_data_set)
ypred= pd.DataFrame(ypred)
print(classification_report(Y_test_data_set, ypred))

# 5: Apply Random Forest
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators = 300, criterion="entropy")
rf.fit(X_train_data_set, Y_train_data_labels )
y_pred_RF = rf.predict(X_test_data_set)
print(classification_report(Y_test_data_set, y_pred_RF))

# 6: Apply SVM
from sklearn.svm import LinearSVC
clf_init = LinearSVC(random_state=0, tol=1e-5)
clf_init.fit(X_train_data_set, Y_train_data_labels.ravel())
# Response prediction for testing data
ypred = clf_init.predict(X_test_data_set)
ytest = Y_test_data_set.to_numpy()
print(classification_report(ytest, ypred))
# Apply Naïve Bayes
nb = GaussianNB()
nb.fit(X_train_data_set, Y_train_data_labels )
y_pred_NB = nb.predict(X_test_data_set)
print(classification_report(Y_test_data_set, y_pred_NB))
```

K-Means Clustering

```
# Define 40 classes of z using PCA values
z =pca_values[: 40]
# Define new data space
new_df = pd.DataFrame(z, columns=["F"+str(i+1) for i in range(var1.shap
e[-1])])
new_df
from sklearn.cluster import KMeans
kmeans = KMeans(init="random", n_clusters=5, n_init=10, max_iter=300, r
andom_state=42)
kmeans.fit(new_df)
kmeans.labels_

The lowest Sum of Squared Error/Euclidean (SSE) value
print("SSE value: ", kmeans.inertia_)
# Last location
print("\n Last geometric center
location: \n", kmeans.cluster_centers_[:1])
# Merging Point
print("\n Iterations required to converge: ", kmeans.n_iter_)
```

```
# Storing cluster assignments in 1D Numpy array
print("\n Predicted labels: \n", kmeans.labels_)
# View the new reduced data
new_df.head()

faces = new_df
# Defining labels and assigning the target
faces.rename(columns={'F1':'label'}, inplace=True) # inplace=true is to
 delete it permanetly and has been depricated
faces.rename(columns={'F155':'target'}, inplace=True)
# View the new reduced data again with new labels
faces.head()


SEABORN
# View searborn of the few columns of the new data
sns.pairplot(faces, vars =['label', 'F3', 'F7', 'F154', 'target'])

sns.pairplot(faces, vars =['label', 'F3', 'F7', 'F154', 'F8'], hue='lab
el', kind='reg') # demands more RAM, takes time in a CPU, better GPU
```

Starting Normalisation to Prepare for CNN

```
from sklearn.preprocessing import MinMaxScaler
standard_scaler = MinMaxScaler()
standard_scaler.fit(X_train_data_set)
# Transform train and test data
X_train_data_set = standard_scaler.transform(X_train_data_set)
X_test_data_set = standard_scaler.transform(X_test_data_set)
X_train_data_set.shape, Y_train_data_labels.shape, X_test_data_set.shap
e
# Define data shape
X_train_data_set = X_train_data_set.reshape((X_train_data_set.shape[0],
 64, 64, 1)).astype('float32') #22, 7, 1 tried and results to 100%
score, the way it trains data is not convincing
X_test_data_set = X_test_data_set.reshape((X_test_data_set.shape[0], 64
, 64, 1)).astype('float32')
print(X_train_data_set.shape)
# Preparing data for CNN
Y_train_data_labels = to_categorical(Y_train_data_labels, 40)
Y_test_data_set = to_categorical(Y_test_data_set, 40)
Y_train_data_labels.shape, Y_test_data_set.shape
# Assign classes test shape
num_classes = Y_test_data_set.shape[1]
```

```
# View classes
num_classes
```

CNN

```
from tensorflow.python.keras import regularizers

# Stop training when no more improvement in the validation loss for 3 c
onsecutive epochs
callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=90
, min_delta=0, verbose=0, mode="auto", baseline=None,
    restore_best_weights=False,)

model = Sequential()

model.add(Conv2D(64, (3, 3), input_shape=(64, 64, 1), activation='relu'
, data_format="channels_last", kernel_initializer=tf.keras.initializers
.HeNormal(), padding="same"))
model.add(Conv2D(64, (3, 3), input_shape=(64, 64, 1), activation='relu'
, data_format="channels_last", kernel_initializer=tf.keras.initializers
.HeNormal(), padding="same"))
model.add(MaxPooling2D((2,2), strides=(2,2), padding='same'))

model.add(Conv2D(128, (3, 3), activation='relu', data_format="channels_
last", padding="same"))
model.add(Conv2D(128, (3, 3), activation='relu', data_format="channels_
last", padding="same"))
model.add(MaxPooling2D((2,2), strides=(2,2), padding='same'))

model.add(Dropout(0.2))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))
# Print model summary
model.summary()
```

TRAINING

```
from keras.optimizers import Adam
# View time taken to train the network
t0 = time()
```

```
model.compile(loss='categorical_crossentropy', optimizer=Adam(lr=0.0001
), metrics=['accuracy'])
history = model.fit(X_train_data_set, Y_train_data_labels, validation_d
ata=(X_test_data_set, Y_test_data_set), epochs=90, batch_size=256, call
backs=[callback], verbose=2)

# Last verification step
scores = model.evaluate(X_test_data_set, Y_test_data_set, verbose=0)
print("CNN Error: %.2f%%" % (100-scores[1]*100))


# SHOW MODEL PERFORMANCE OVER EPOCHS
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='lower right')
plt.savefig('Model_Accuracy.png')
plt.show()
# Summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['train', 'test'], loc='upper right')
plt.savefig('Model_loss.png')
plt.show()

model.save('weights.model')
model.save_weights("model.h5")
model_json = model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)

Y_test_data_set[1]

rounded_labels=np.argmax(Y_test_data_set, axis=1)
rounded_labels[1]

rounded_predictions = model.predict_classes(X_test_data_set, batch_size
=256, verbose=0)
rounded_predictions[1]

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(rounded_labels, rounded_predictions)
```

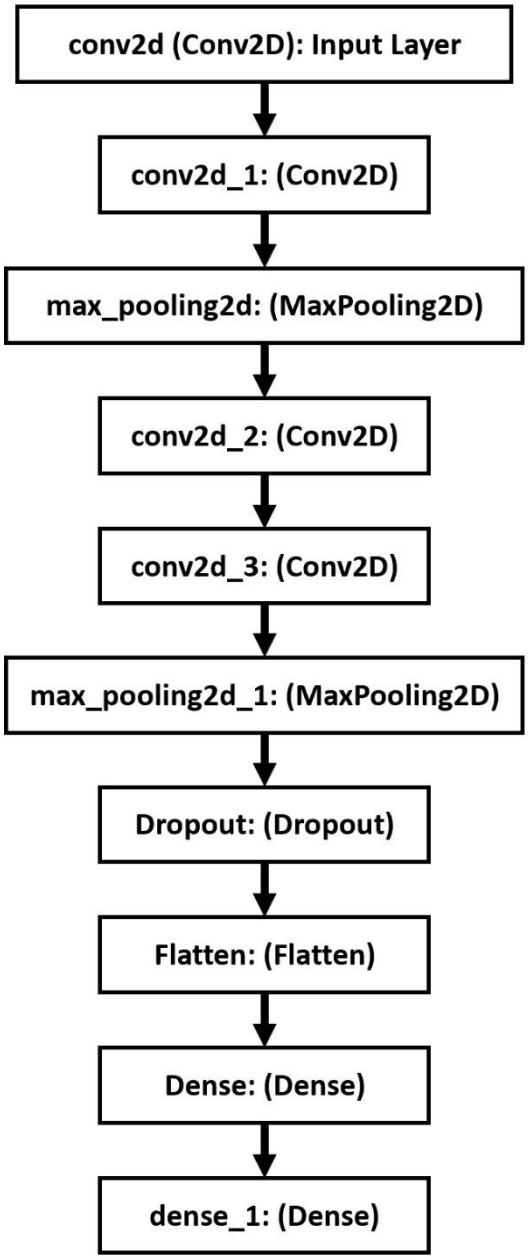
```
cm

X_test_data_set =np.argmax(X_test_data_set, axis=1)
# X_test_data_set[1]

from sklearn.metrics import f1_score, precision_score, recall_score, co
nfusion_matrix

print(precision_score(rounded_labels, rounded_predictions , average="ma
cro"))
print(recall_score(rounded_labels, rounded_predictions , average="macro
"))
print(f1_score(rounded_labels, rounded_predictions , average="macro"))


MODEL CLASSIFICATION REPORT

print("done in %0.3fs" % (time() - t0))
print(classification_report(rounded_labels, rounded_predictions))

DISPLAY EPOCHS NOT MORE THAN 90

len(history.history['loss'])

# Concatenated kernels summarized
for layer in model.layers:
    # conv check
    if 'conv' not in layer.name:
        continue

# Getting 2D Weights
filters, biases = layer.get_weights()
print(layer.name, filters.shape)

# Summary of 2D weights
from matplotlib import pyplot
# Concatenated kernels summarized
for layer in model.layers:
    # Test
    if 'conv' not in layer.name:
        continue
    # Getting 2D kernel
    filters, biases = layer.get_weights()
    print(layer.name, filters.shape)

# Get kernel
filters, biases = model.layers[1].get_weights()
```

```
# Enable visualization
f_min, f_max = filters.min(), filters.max()
filters = (filters - f_min) / (f_max - f_min)

# Showing a couple of low-level values in kernel matrix
n_filters, ix = 6, 1
for i in range(n_filters):
    # Identify concatenated kernels
    f = filters[:, :, :, i]
    # Display channels apart
    for j in range(3):
        # Specify subordinate kernel
        ax = pyplot.subplot(n_filters, 3, ix)
        ax.set_xticks([])
        ax.set_yticks([])
        # Display black and white channels
        pyplot.imshow(f[:, :, j], cmap='gray')
        ix += 1
# Plot
pyplot.show()

# Convolution size summary
from matplotlib import pyplot
# Summarize attributes
for i in range(len(model.layers)):
    layer = model.layers[i]
    # Conv layer check
    if 'conv' not in layer.name:
        continue
    # Summarize
    print(i, layer.name, layer.output.shape)

# Define model again
model = Model(inputs=model.inputs, outputs=model.layers[1].output)

# Attributes of the second layer after input layer
feature_maps = model.predict(X_train_data_set)

# Display four by four squares of the sixteen maps
square = 4
ix = 1
for _ in range(square):
    for _ in range(square):
        # Graphical presentation
        ax = pyplot.subplot(square, square, ix)
        ax.set_xticks([])
```

```
        ax.set_yticks([])
        # view black and white
        pyplot.imshow(feature_maps[0, :, :, ix-1], cmap='gray')
        ix += 1
# Plot
pyplot.show()

# Define again
ixs = [0]
outputs = [model.layers[i+1].output for i in ixs]
model = Model(inputs=model.inputs, outputs=outputs)
model.summary()

# Iput layer
model.layers[0].output

# Layer 1
model.layers[1].output
```

Appendix B: CNN Model Sequel Representation

CNN model utilized to all simulations – Plots utilizing Keras model representation



Appendix C: Approved Ethical Clearance

UNISA university of south africa

UNISA SCHOOL OF ENGINEERING (SOE) ETHICS REVIEW COMMITTEE

17/10/2022

ERC Reference #: 2021/CSET/SOE/073
Name: Mr Zukisa Emerson Nante
Student #: 37229958
Staff #: N/A

Dear Mr Zukisa Emerson Nante

Decision: Ethics Approval from 17/01/2022 to 17/01/2024 (No humans involved)

Researcher(s): Name: Mr Zukisa Emerson Nante
Email: 37229958@mylife.unisa.ac.za
Telephone #: +32 489 356 786

Supervisor (s): Name: Prof Zenghui Wang
E-mail address: wangz@unisa.ac.za
Telephone #: 011 471 3513

Working title of research:

Face Recognition Using Improved Deep Learning Neural Network

Qualification: MTech in Electrical Engineering

Thank you for the application for research ethics clearance by the Unisa School of Engineering (SOE) Research Ethics Review Committee for the above mentioned research. Ethics approval is granted for 3 years.

*The **negligible risk application** was expedited by the School of Engineering (SOE) Ethics Review Committee on 29 December 2021 in compliance with the Unisa Policy on Research Ethics and the Standard Operating Procedure on Research Ethics Risk Assessment. The decision will be ratified in the next SOE Ethics Review meeting.*

University of South Africa
Preller Street, Muckleneuk Ridge, City of Tshwane
PO Box 392 UNISA 0003 South Africa
Telephone: +27 12 429 3111 Facsimile: +27 12 429 4150
www.unisa.ac.za

The proposed research may now commence with the provisions that:

1. The researcher will ensure that the research project adheres to the relevant guidelines set out in the Unisa COVID-19 position statement on research ethics attached.
2. The researcher(s) will ensure that the research project adheres to the values and principles expressed in the UNISA Policy on Research Ethics.
3. Any adverse circumstance arising in the undertaking of the research project that is relevant to the ethicality of the study should be communicated in writing to the School of Engineering (SOE) Ethics Review Committee.
4. The researcher(s) will conduct the study according to the methods and procedures set out in the approved application.
5. Any changes that can affect the study-related risks for the research participants, particularly in terms of assurances made with regards to the protection of participants' privacy and the confidentiality of the data, should be reported to the Committee in writing, accompanied by a progress report.
6. The researcher will ensure that the research project adheres to any applicable national legislation, professional codes of conduct, institutional guidelines and scientific standards relevant to the specific field of study. Adherence to the following South African legislation is important, if applicable: Protection of Personal Information Act, no 4 of 2013; Children's act no 38 of 2005 and the National Health Act, no 61 of 2003.
7. Only de-identified research data may be used for secondary research purposes in future on condition that the research objectives are similar to those of the original research. Secondary use of identifiable human research data require additional ethics clearance.
8. No field work activities may continue after the expiry date 17 January 2024. Submission of a completed research ethics progress report will constitute an application for renewal of Ethics Research Committee approval.
9. Permission to conduct research involving UNISA employees, students and data should be obtained from the Research Permissions Subcommittee (RPSC) prior to commencing field work.
10. Permission to conduct this research should be obtained from the [company, CE organisation, DoE, etc name] prior to commencing field work.

Note

The reference number 2021/CSET/SOE/073 should be clearly indicated on all forms of communication with the intended research participants, as well as with the Committee.

University of South Africa
Preller Street, Muckleneuk Ridge, City of Tshwane
PO Box 392 UNISA 0003 South Africa
Telephone: +27 12 429 3111 Facsimile: +27 12 429 4150
www.unisa.ac.za

Yours sincerely,

Dr TY Leswifi
Chair of School of Engineering (SOE) Ethics Review Subcommittee
College of Science Engineering and Technology (CSET)
E-mail: leswity@unisa.ac.za
Tel: (011) 670 9288

Prof. F Nemavhola
Director: School of Engineering
College of Science Engineering and Technology (CSET)
E-mail: masitfj@unisa.ac.za
Tel: (011) 471 2354

Prof. B Mamba
Executive Dean
College of Science Engineering and Technology (CSET)
E-mail: mambabb@unisa.ac.za
Tel: (011) 670 9230

University of South Africa
Preller Street, Muckleneuk Ridge, City of Tshwane
PO Box 392 UNISA 0003 South Africa
Telephone: +27 12 429 3111 Facsimile: +27 12 429 4150
www.unisa.ac.za

Appendix D: List of Publications

Nante, Z., and Wang, Z. (2022). A new Face Recognition Using Principal Component Analysis, K-Means Clustering, and Convolutional Neural Network. International Conference on Artificial Intelligence Trends and Pattern Recognition (ICAITPR), Hyderabad, India, 10-12th March, 2022 (in press)