

IMPROVING HUMAN FACE RECOGNITION USING
DEEP LEARNING BASED IMAGE REGISTRATION
AND MULTI-CLASSIFIER APPROACHES

Mohannad Abuzneid

Under the Supervision of Dr. Ausif Mahmood

DISSERTATION

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE

AND ENGINEERING

THE SCHOOL OF ENGINEERING

UNIVERSITY OF BRIDGEPORT

CONNECTICUT

December, 2018


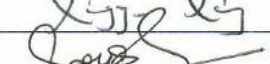
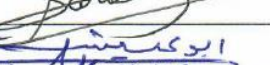

IMPROVING HUMAN FACE RECOGNITION USING DEEP LEARNING BASED IMAGE REGISTRATION AND MULTI- CLASSIFIER APPROACHES

Mohannad Abuzneid

Under the Supervision of Dr. Ausif Mahmood

Approvals

Committee Members

Name	Signature	Date
Dr. Ausif Mahmood		11-30-2018
Dr. Miad Faezipour		11, 30, 2018
Dr. Xingguo Xiong		11/30/2018
Dr. Sarosh Patel		30 NOV 2018
Dr. Akram Abu-aisheh		11/29/18

Ph.D. Program Coordinator

Dr. Khaled M. Elleithy		11/30/18
------------------------	--	----------

Chairman, Computer Science and Engineering Department

Dr. Ausif Mahmood		11-30-2018
-------------------	--	------------

Dean, School of Engineering

Dr. Tarek M. Sobh		12/3/2018
-------------------	--	-----------

IMPROVING HUMAN FACE RECOGNITION USING DEEP LEARNING BASED IMAGE REGISTRATION AND MULTI- CLASSIFIER APPROACHES

© Copyright by Mohannad Abuzneid 2018

IMPROVING HUMAN FACE RECOGNITION USING DEEP LEARNING BASED IMAGE REGISTRATION AND MULTI- CLASSIFIER APPROACHES

ABSTRACT

Face detection, registration, and recognition have become a fascinating field for researchers. The motivation behind the enormous interest in the topic is the need to improve the accuracy of many real-time applications. Countless methodologies have been acknowledged and presented in the past years. The complexity of the human face visual and the significant changes based on different effects make it more challenging to design as well as implementing a powerful computational system for object recognition in addition to human face recognition. Using supervised learning often requires extensive training for the computer which results in high execution times. It is an essential step in the face recognition to apply strong preprocessing approaches such as face registration to achieve a high recognition accuracy rate. Although there are exist approaches do both detection and recognition, we believe the absence of a complete end-to-end system capable of performing recognition from an arbitrary scene is in large part due to the difficulty in alignment. Often, the face registration is ignored, with the assumption that the detector will perform a rough alignment, leading to suboptimal recognition

performance.

In this research, we presented an enhanced approach to improve human face recognition using a back-propagation neural network (BPNN) and features extraction based on the correlation between the training images. A key contribution of this paper is the generation of a new set called the T-Dataset from the original training data set, which is used to train the BPNN. We generated the T-Dataset using the correlation between the training images without using a common technique of image density. The correlated T-Dataset provides a high distinction layer between the training images, which helps the BPNN to converge faster and achieve better accuracy. Data and features reduction is essential in the face recognition process, and researchers have recently focused on the modern neural network. Therefore, we used using a classical conventional Principal Component Analysis (PCA) and Local Binary Patterns (LBP) to prove that there is a potential improvement even using traditional methods. We applied five distance measurement algorithms and then combined them to obtain the T-Dataset, which we fed into the BPNN. We achieved higher face recognition accuracy with less computational cost compared with the current approach by using reduced image features. We test the proposed framework on two small data sets, the YALE and AT&T data sets, as the ground truth. We achieved tremendous accuracy. Furthermore, we evaluate our method on one of the state-of-the-art benchmark data sets, Labeled Faces in the Wild (LFW), where we produce a competitive face recognition performance.

In addition, we presented an enhanced framework to improve the face registration using deep learning model. We used deep architectures such as VGG16 and VGG19 to train our method. We trained our model to learn the transformation parameters (Rotation,

scaling, and shifting). By learning the transformation parameters, we will be able to transfer the image back to the frontal domain. We used the LFW dataset to evaluate our method, and we achieve high accuracy.

ACKNOWLEDGEMENTS

My thanks are wholly devoted to God, who has helped me complete this work successfully. I owe a debt of gratitude to my family, my parents, my lovely wife, and my kids Fahad and Feras for their understanding and encouragement. I am very grateful to my father for raising me and encouraging me to achieve my goal. I could have never achieved this without their support.

I would like to express special thanks to my supervisor Dr. Ausif Mahmood for his constant guidance, comments, and valuable time. The door to Prof. Mahmood office was always open whenever I ran into a trouble spot or had a question about my research or writing. My appreciation also goes my committee members Dr. Miad Faezipou, Dr. Xingguo Xiong, Dr. Sarosh Patel and Dr. Akram Abu-aisheh for their valuable time and suggestions. I also would like to thank my brother Abdel-shakour Abuzneid for his support.

TABLE OF CONTENTS

ABSTRACT.....	iv
ACKNOWLEDGEMENTS.....	vii
TABLE OF CONTENTS.....	viii
ABBREVIATIONS	xi
LIST OF TABLES	xiii
LIST OF FIGURES	xiv
CHAPTER 1: INTRODUCTION	1
1.1 Research Problem and Scope	1
1.2 Motivation behind the Research	3
1.3 Potential Contributions of the Proposed Research	4
CHAPTER 2: LITERATURE SURVEY.....	8
2.1 Preprocessing and Image Registration	9
2.2 Feature Extraction for Face Recognition	11
2.2.1 Appearance-Based Feature Extraction Approach	12
2.2.2 Feature-Based Feature Extraction Approach	15
2.3 Classification	17
2.4 Neural Network.....	18
2.5 Deep Learning Background	22
2.6 Convolutional Neural Network	23
2.6.1 Convolutional Layer	24

2.6.2 Pooling Layer.....	26
2.6.3 Fully Connected Layer.....	28
2.7 Literature Review for Face Registration	28
2.7.1 Speed-Up Robust Features (SURF)	28
2.7.2 Minimized Cost Function	30
2.7.3 Random Sample Consensus (RANSAC)	30
CHAPTER 3: RESEARCH PLAN AND SYSTEM ARCHITECTURE	32
3.1 Human Face Datasets	32
3.1.1 ORL Dataset.....	32
3.1.2 Yale Dataset	33
3.1.3 Labeled Faces in the Wild.....	34
3.2 Classical Face Recognition System.....	34
3.2.1 Principle Component Analysis.....	38
3.2.2 Local Binary Patterns Histogram (LBPH)	40
3.2.3 Similarity Measurements Methods	43
3.3 Face Recognition using PCA and NN Proposed System	45
3.3.1 Proposed Method	45
3.4 Face Recognition using LBPH and NN Proposed System	50
3.4.1 Proposed Method	50
3.5 Face Registration Based on a Minimalized Cost Function	54
3.5.1 Proposed Method	54
3.6 Deep Learning Face Registration	56
3.6.1 Obtaining the Training Dataset.....	57
3.6.2 VGGNet Model.....	58
3.6.3 Residual Neural Network (ResNet) Model	59

3.6.4 Proposed Method Configurations	61
CHAPTER 4: RESULTS	63
4.1 Classical Face Recognition System Result	63
4.1.1 Classical Face Recognition Using PCA and KNN Result	63
4.1.2 Classical Face Recognition Using LBPH and KNN Result.....	65
4.2 Proposed Face Recognition System Result	67
4.2.1 Proposed Face Recognition Result Using the PCA and NN.....	67
4.2.2 Proposed Face Recognition Result Using the LBPH and NN	68
4.3 Face Registration Based on a Minimalized Cost Function Result	70
4.4 Deep Face Registration Result	71
CONCLUSIONS	74
REFERENCES	75
APPENDIX.....	86

ABBREVIATIONS

BPNN	Back-propagation Neural Network
2DPCA	2- Dimensional Principal Component Analysis
BGD	Batch Gradient Descent
BRIEF	Binary Robust Independent Elementary Feature
CNN	Convolutional Neural Networks
DBN	Deep Belief Network
DCT	Discrete Cosine Transform
DWT	Discrete Wavelet Transform
FFT	Fast Fourier Transform
ICA	Independent component analysis
KNN	K-Nearest-Neighbor
LBP	Local Binary Patterns
LDA	Linear discriminant analysis
LFW	Labeled Faces in the Wild
MLP	Multi-Layer Perceptron
NN	Neural Network
ORL	Olivetti Research Laboratory Dataset
PCA	Principal Component Analysis
RANSAC	Random Sample Consensus

ReLU	Rectified Linear Unit
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
RR	Recognition Rate
SA	Stacked Autoencoder
SGD	Stochastic Gradient Descent
SIFT	Scale-Invariant Feature Transform
SRS	Square-Root of the Sum
SSE	Sum-Squared-Error
SURF	Speeded-Up Robust Features
SVM	Support Vector Machines

LIST OF TABLES

Table 1.1	The optimal match scenario	5
Table 1.2	The partially match scenario	6
Table 1.3	The complete mismatch scenario	6
Table 3.1	An example of how to obtain the new training data (column 6) and the expected out (column 7) for one of the training images (image X)	48
Table 4.1	Experiment results using PCA + KNN	63
Table 4.2	Experiment results using LBPH + KNN	65
Table 4.3	Experiment results using PCA + NN with 50% training set and 50% testing set	67
Table 4.4	Experiment results using LBPH + NN with 50% training set and 50% testing set	69
Table 4.5	Comparison between the proposed framework and the other existing methods	70
Table 4.6	Registration error rate for variant matching point number	71
Table 4.7	Deep face registration result.	72

LIST OF FIGURES

Figure 1.1	Same person with variant facial expression	2
Figure 1.2	Under variant lighting environments, the face can look different for the same person based on the light source	2
Figure 1.3	Transformation Image	2
Figure 1.4	Face Recognition System Process	3
Figure 2.1	Classical image registration	10
Figure 2.2	Feature Extraction categories	11
Figure 2.3	Three layers neural network	18
Figure 2.4	The typical structure of a CNN	24
Figure 2.5	Convolution operation	26
Figure 2.6	Max pooling in CNN	27
Figure 3.1	Sample of the ORL dataset	33
Figure 3.2	Sample images of Yale dataset	34
Figure 3.3	Sample images of LFW dataset.	35
Figure 3.4	Classical face recognition system using The PCA and KNN classifier methods	36
Figure 3.5	Classical face recognition system using the LBPH and KNN classifier method	36
Figure 3.6	Example of Prepressing Methods including cropping, resizing and Histogram	37

Figure 3.7	Example of the matching case and mismatching case image using KNN with Mahalanobis distance	38
Figure 3.8	An example of PCA (a) Original data. (b) Correlated data	39
Figure 3.9	(a) Original faces (b) Corresponding Eigen-faces	41
Figure 3.10	Original LBP Operator	42
Figure 3.11	Face description with local binary patterns	43
Figure 3.12	Proposed recognition system using PCA and NN classifier methods	49
Figure 3.13	Proposed recognition system using LPBH and NN classifier method	53
Figure 3.14	The proposed image registration method	54
Figure 3.15	Key-points for Lena with variant threshold. (a) Threshold =100. (b) Threshold =50. (3)Threshold =20	55
Figure 3.16	Matching points with false matching points	55
Figure 3.17	Matching points after eliminating the false points	56
Figure 3.18	Example of image registration. (a) Source and target images with two matching points. (b) Registered image	56
Figure 3.19	Reference Image.	57
Figure 3.20	The 6 facial landmarks associated with the eye.	58
Figure 3.21	VGGNet model architecture.	59
Figure 3.22	A residual block.	60
Figure 3.23	ResNet50 model architecture.	60
Figure 3.24	Proposed method using a simple CNN.	61
Figure 3.25	Proposed method using VGG16 and VGG19.	62
Figure 3.26	Proposed method using ResNet50.	62
Figure 4.1	Experiment results using PCA + KNN for the Yale data-set.	64

Figure 4.2	Experiment results using PCA +KNN for ORL data-set	64
Figure 4.3	Experiment results using LBPH + NN for the Yale data-set	66
Figure 4.4	Experiment results using LBPH + NN for the ORL data-set	66
Figure 4.5	Experiment results using PCA + NN with 50% training set and 50% testing set	68
Figure 4.6	Experiment results using LBPH + NN with 50% training set and 50% testing set	69
Figure 4.7	Registration error rate for variant matching point number on Lena image	71
Figure 4.8	Models loss curve.	72
Figure 4.9	Example for registered faces: (a) Original face (b) Predicted registered face (c) Expected registered face.	73

CHAPTER 1: INTRODUCTION

1.1 Research Problem and Scope

Human face recognition is a challenging task because of the variability of the facial expression, personal appearance, variant poses, and the various illumination as shown in Figure 1.1 [1-4]. Also, due to the variability in lighting intensity, the number of sources, direction, and the camera orientation as illustrated in Figure 1.2, it is a challenging task to design a face recognition system in the real-time with high accuracy recognition rate. Image transformation by rotation, scaling, and a translation is one of the most challenging tasks to solve and has a significant influence on the image recognition as shown in Figure 1.3. The changes in the human face personality should have less effect compared to the pose variation and illumination [5]. Reducing the image dimension is necessary to improve the classification processing time since the object recognition system requires an enormous volume for the computing process. PCA and LBP are one of the popular conventional approaches; both used for robust data representation, as well as histograms, for features reduction [6-13]. Higher accuracy can be achieved by finding a strong representation of the human face by retaining the most dissimilarities in the image data after reducing the dimensionality of the image.



Figure 1.1. Same person with variant facial expression.



Figure 1.2. Under variant lighting environments, the face can look different for the same person based on the light source.



Figure 1.3. Transformation Image.

Classical human face recognition systems are divided into three phases as shown in Figure 1.4: The first step is preprocessing, which consists of many types of operations,

such as image registration, scaling, face normalization, reducing the effect of background noise, detection and resizing, all of which affect the face recognition accuracy. Feature extraction is the second phase, which can be achieved by using powerful transformation approaches. The image dimension can be reduced to a smaller dimension by retaining significant features [12-13]. The final phase is the classification which is using powerful classifiers such as deep neural networks and the fully connected neural networks [14-16]. In this research, we focused on the preprocessing phase and the feature extraction phase which they have a potential improvement.

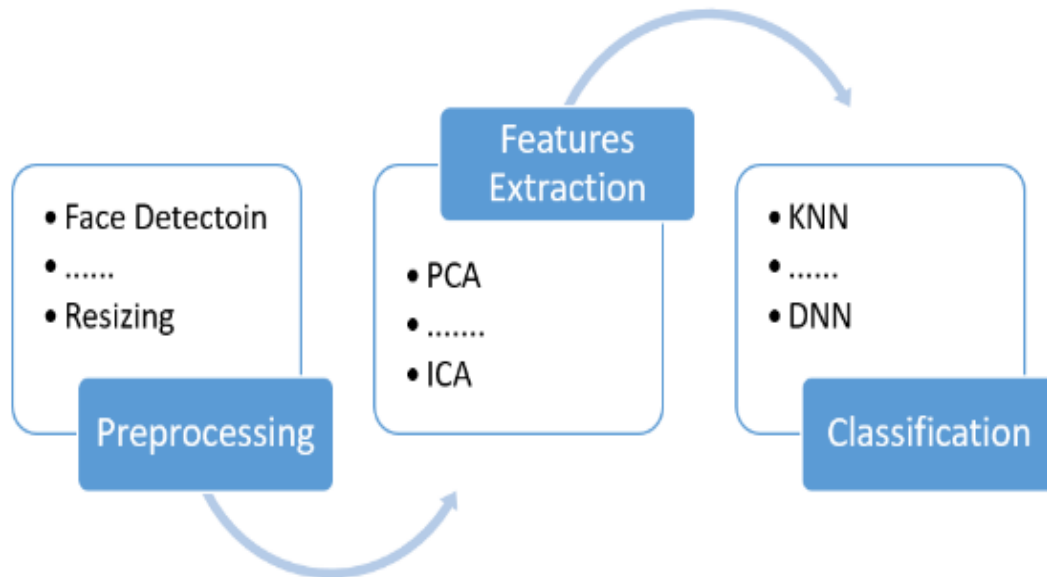


Figure 1.4. Face Recognition System Process.

1.2 Motivation behind the Research

There are many existing algorithms handle the human face recognition. The purpose of these algorithms to achieve an optimal recognition rate or near optimal

recognition rate in real-time processing time. However, none of the algorithms achieved a 100% recognition rate. Therefore, human face recognition is an incredibly exciting field for researchers. The neural network classifier is relying on the strong features extraction methods. Neural network performs well with low error rate by feeding strong distinction patterns and features.

This research is motivated by the drawbacks and limitations of existing systems. Existing methods relies on simple preprocessing methods such as face detection and simple image registration. Therefore, we felt that there is a potential improvement can be achieved by implementing a strong image registration based on deep learning approach and a strong features extraction approach which can provide strong distinction patterns and features based on the existing feature extraction methods such as PCA and LBP.

1.3 Potential Contributions of the Proposed Research

In this dissertation, we introduce an enhanced human face recognition framework with a high recognition rate. This improvement based on improving the features extraction approach and improving the image registration based on active shape model deep learning. The proposed framework is suitable for different features extraction methods such as PCA and LBP which we used in this research, and it can be extended to other algorithms such as a deep neural network.

The contribution of the features extraction approach emanated from the first experiment. After we had obtained the training dataset based on the five distance methods, we noticed three scenarios. The first scenario is the optimal matching as shown in Table

1.1. All the distance methods were able to find the right match image of the person 1 which is the image 3 of the person 1. The second scenario is partially matching as shown in Table 1.2. Euclidean, Manhattan, and Mahalanobis methods been able to find the right match for person 1 image which was image 2. However, Correlation and Canberra found the wrong match which was the image belong to person 28. The last scenario is the complete mismatching as shown in Table 1.3. All the distance methods failed to find the right image for person 1. Combining the five distance methods using equation (1.1) makes the system more robust and the recognition rate higher since the training dataset will be more significant for the neural network.

$$\sqrt{\sum_{i=1}^5 DIS_i^2} \quad (1.1)$$

Table 1.1. The optimal match scenario.

		Correlation	Euclidean	Canberra	Manhattan	Mahalanobis
Person 1	Picture1	0.22	0.24	0.71	0.32	0.56
	Picture2	0.12	0.19	0.67	0.25	0.47
	Picture3	0.06	0.14	0.58	0.21	0.47
	Picture4	0.12	0.19	0.64	0.27	0.48
Person 2	Picture1	1	0.7	0.83	0.69	0.73
	Picture4	0.97	0.74	0.82	0.56	0.77
					
Person 40	Picture1	0.38	0.43	0.75	0.47	0.54
	Picture4	0.59	0.67	0.9	0.7	0.66

Table 1.2. The partially match scenario.

		Correlation	Euclidean	Canberra	Manhattan	Mahalanobis
Person 1	Picture1	0.39	0.24	0.71	0.34	0.56
	Picture2	0.31	0.14	0.67	0.21	0.36
Person 28	Picture1	1	0.7	0.83	0.69	0.66
	Picture2	0.24	0.51	0.71	0.44	0.64
	Picture3	0.72	0.54	0.85	0.48	0.79
	Picture4	0.4	0.77	0.61	0.56	0.74
					
Person 40	Picture1	0.41	0.73	0.78	0.41	0.71
	Picture4	0.39	0.54	0.92	0.69	0.69

Table 1.3. The complete mismatch scenario.

		Correlation	Euclidean	Canberra	Manhattan	Mahalanobis
Person 1	Picture1	0.39	0.34	0.71	0.34	0.56
	Picture2	0.31	0.32	0.67	0.42	0.49
	Picture3	0.41	0.28	0.77	0.29	0.39
	Picture4	0.29	0.29	0.64	0.27	0.48
Person 5	Picture3	0.72	0.54	0.85	0.48	0.79
	Picture4	0.19	0.77	0.61	0.56	0.32
					
Person 21	Picture2	0.34	0.41	0.57	0.32	0.74
Person 27	Picture1	0.51	0.57	0.74	0.21	0.64
Person 40	Picture1	0.41	0.27	0.78	0.41	0.71
	Picture4	0.39	0.54	0.92	0.69	0.69

On the other hand, we are trying to achieve higher accuracy by improving the face registration approach which will lead to a robust end-to-end face recognition system. Since the classical face registration is outdated, we are working on the deep learning based face registration, and we decided to build our deep learning system based on the deep learning concept. We used VGG16, VGG19 and ResNet50 architectures to build our model then we applied the model on one of the State-of-Art datasets such as LFW.

CHAPTER 2: LITERATURE SURVEY

The humans can easily and successfully perform face recognition task using their eyes. However, the automatic human face recognition still far from optimal and the researchers with a variant background such as pattern recognition, computer vision, and neural network consider it an area which can be improved. Therefore, the literature survey on human face recognition is diverse. In this survey, a detailed view of the human face recognition methods is presented. Researchers introduced variant algorithms with different accuracy and sometimes inconsistent results comparing to each other. The objective of this survey is to provide an overview of the face recognition process. We focused on the popular categories of feature extraction methods and face registration since the features characterize the whole image [17]. The main purpose of the features extraction is to reduce the image dimension by selecting the most significant features with retaining the relevant information and should be diverse enough among classes for good classification performance. However, the strength of the features extraction methods relies on strong preprocessing approaches like the face registration. The extracted features can be used to classify and to recognize patterns that are present in the source images. Therefore, the face registration and feature extraction process are the key point of the classification performance and thus, in the overall human face recognition.

2.1 Preprocessing and Image Registration

Image registration is an essential method used in the image processing systems such as face and object recognition [18-22], object detection, motion estimation [19], and medical application [20]. The information inherited from two related images for the same scene is different. Therefore, they need a proper registration to make the two images uniform and transfer them to the same coordinate system. The classical steps of the image registration are divided into four phases as shown in Figure 2.1 [21]. The first step is to extract the most significant features of the source image and the target image such as edges, corners, and intersections, etc. The key points can be found using methods such as Gaussians difference algorithm [23], segmentation methods [24], representations of general line segments or elongated anatomic structures [25], virtual circles [26], and local curvature discontinuities detected using the Gabor wavelets [27]. These algorithms are recommended if the image contains detectable objects.

On the other hand, the medical images usually have one object and considered as a lake of details. Fast Fourier transform (FFT) used to extract the features in the frequency domain and obtain the parameters based on cross-correlation [28]. The discrete wavelet transform is another method used for feature extraction with root mean square error (RMSE) method [29]. The second step is to find the matching features between the two images from the features which we extracted in step one. Obtaining the corresponding points between the images has been a motivation of many invariant algorithms such as the Scale-Invariant Feature Transform (SIFT) [23], Speeded-Up Robust Features (SURF) [30-32], and Binary Robust Independent Elementary Feature (BRIEF) [33]. Even with these

methods, it is still a challenge to obtain the appropriate matches. The RANSAC method is used to eliminate all the mismatching points by finding the best fitting on random subsets of the matches then select the best fitting subset. RANSAC [34] is robust to mismatches but finds a sub-optimal estimation, where LMedS [35] is a more accurate estimation, however, requires at least 50% correct matches. The third step is to find the affine transformation parameters such as translation, scaling, reflection, rotation, and shearing using some of the methods such as the minimized cost function. The last step is transforming the target image to the source image coordinate system using the affine transformation parameters which obtained from the third step.

The researchers moved toward a deep learning based image registration approaches because of the classical image registration limitation. P. Gadde et al. [87] proposed an Image registration with artificial neural networks using spatial and frequency features. In their study, the registration of images is investigated with two novel neural network based approaches, namely, SIFT-DCT and SIFT-DWT. Scale-invariant Feature Transform



Figure 2.1. Classical image registration.

(SIFT), Discrete Cosine Transform (DCT), and Discrete Wavelet Transform (DWT) are employed in these approaches. Both new approaches combine features in the spatial domain (SIFT) and frequency domain (DCT or DWT) to provide more robust feature

extraction methods for image registration. The learning ability and nonlinear mapping ability of artificial neural network provide a flexible and intelligent tool for data fusion on feature matching and transform model parameter estimation. However, this proposed method obtain the training data using original methods not based on deep learning.

2.2 Feature Extraction for Face Recognition

Feature extraction can be accomplished using numerous mathematical models, image processing techniques, and intelligent computational tools such as neural networks or fuzzy logic. The approaches divided into four categories feature-based, appearance-based, and template-based and part-based approaches as shown in Figure 2.2. In our research, we focused on the feature-based and appearance-based.

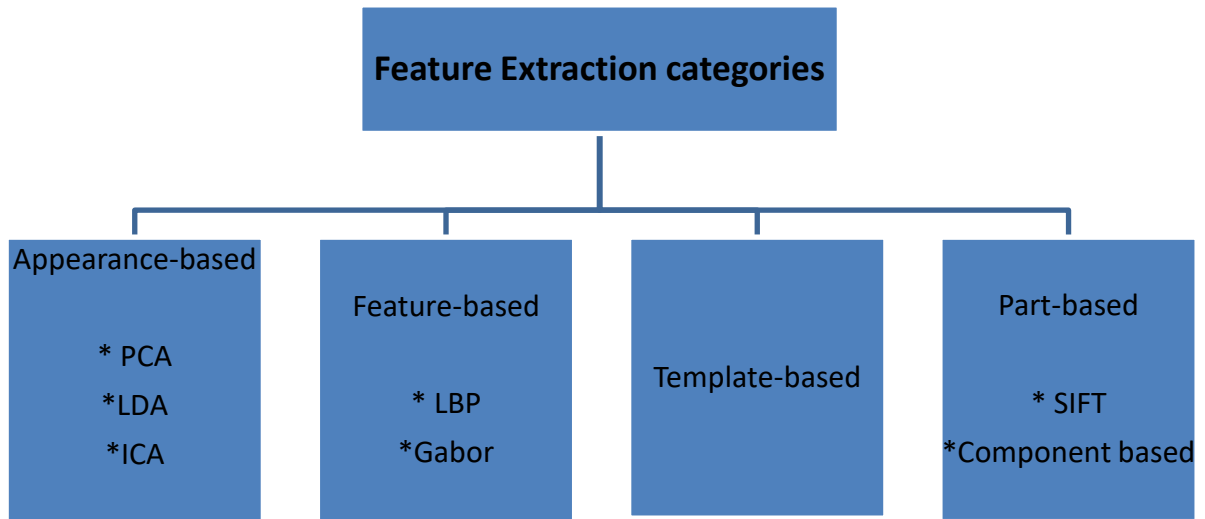


Figure 2.2. Feature Extraction categories

2.2.1 Appearance-Based Feature Extraction Approach

Appearance-based approaches also known as holistic-based methods identify faces using global features based on the whole image instead of local features of the human face. The new reduced dimension representation of the face obtain by applying some transformation on the entire image. However, the feature-based method obtains the information from some detected fiducial points like eyes, noses, and lips, etc. The fiducial points are usually determined from domain knowledge and discard other information. However, the feature obtained from statistics in the appearance-based methods by performing transformations on the entire face. Holistic-based methods took the most attention against other approaches for the past 3 decades. In this section, we will present an overview of Eigen-face [12] based on the PCA, fisher face based on the LDA, and independent component analysis (ICA). More methods can be found in [36] and [37].

A) Karhunen-Loeve expansion, also known as PCA is one of the popular common approaches, which is wildly used for data representation and features reduction [38]. A solid representation of the human face is achieved by retaining the most variations in the image data after reducing the dimensionality of the image. The concept of the PCA is to translate the human face into a smaller set of features data and keep the variations in the image data-characteristic, which is called Eigen-Faces and they are the principal components of the initial training set of the human face images. The unknown face image in the recognition testing process is projected into a reduced-dimension human face space obtained by the Eigen Faces then classified by distance classifiers or statistical method. Sirovich and Kirby

efficiently represent pictures of human faces in 1978 using the PCA. In 1991 Turk and Pentland [12], proposed the popular Eigen-faces method for face recognition. PCA uses the Eigen-Faces to represents human face images as a subset of their Eigen Vectors. Many methods proposed in the computer vision field based on the PCA such as Diagonal PCA [38], and Curvelet-based PCA [39]. Yang et al. [40] proposed Kernel PCA and Kernel FLD for human face recognition, which they called Kernel Fisher-face and Kernel Eigen-face methods. The modular PCA [41] approach has achieved the high accuracy of PCA in cases of extreme change of pose variations, illumination, and expressions. The 2- Dimensional principal component analysis (2DPCA) was introduced as a new approach for feature extraction and representation by Yang et al. [42]. The 2DPCA has numerous advantages over conventional PCA, and it is more straightforward than the PCA to use for face image extraction because 2DPCA regarding the image matrix. Based on Yang method, the 2DPCA is better than conventional PCA in terms of recognition accuracy and is computationally more efficient than conventional PCA therefore; it can improve the process time of image feature extraction significantly. On the other hand, the conventional PCA based image representation is more efficient than the 2DPCA-based image representation regarding storage space because 2DPCA needs more coefficients for face representation.

B) Lu et al. [43] in 2003 and Martinez et al. [44] in 2001 proposed the Fisher's linear discriminant analysis (LDA) as a better alternative to the PCA. LDA successfully applied to face recognition area in the past few years. LDA explicitly provides discernment among the classes, while the PCA deals with the input image as entire

without paying any attention to the principal structure. The main objective of the LDA is to find a base vector which is providing the best discrimination between the classes to help to maximize the differences between the classes and minimizing the differences within the same classes. The classes are represented by the corresponding scatter matrices \mathbf{S}_b and \mathbf{S}_w while the ratio is the derivative of $|\mathbf{S}_b|/|\mathbf{S}_w|$ has to be maximized. LDA outperform the PCA and provide robust classification performances only when a wide training set is an available base on some results discussed by Martinez and Kak which is confirm this thesis and it called the SSS (Small Sample Size) problem. Belhumeur et al., 1997 considered the PCA as an initial step in order to reduce the dimensionality of the input space then LDA is applied to the resulting space in order to perform the real classification. However, Chen et al., 2000; Yu and Yang, 2001 applied LDA directly on the input space and claimed that combining the PCA and LDA, discriminant information together with redundant one is discarded. Lu et al. (2003) proposed a hybrid between the Direct LDA and the Fractional LDA, a variant of the LDA, in which weighed functions are used to avoid that output classes, which are too close, can induce a misclassification of the input.

C) Generalization of PCA called Independent Component Analysis (ICA) was introduced by Bartlett et al. [45] and Draper et al. [46]. They assumed a better basis of the human face images might be found by methods which are sensitive to these high-order statistics. Moghaddam [47] claimed that the ICA-based approach does not provide a significant advantage over the PCA-based method. Yang [48] showed that the Kernel PCA method outperforms the classical PCA method by

applying the Kernel PCA for human face feature extraction and recognition. However, Kernel PCA and ICA are both computationally more expensive than PCA.

2.2.2 Feature-Based Feature Extraction Approach

Feature-based methods exploit more ideas from image processing, computer vision, and domain knowledge from a human face. However, appearance-based methods rely more on statistical learning and analysis. We compared the differences between holistic based methods and feature-based methods and in this section, we discuss two outstanding features for face recognition, the Gabor wavelet feature and the local binary pattern.

A) Local Binary Pattern (LBP) is one of the feature descriptor widely used in face recognition systems. The original LBP operator was introduced by Ojala et al. [49] and was proved a powerful means of texture description. The most important properties of LBP features are the tolerance against illumination changes. LBP is one of the best accomplishment descriptors as it contains the microstructure as well as macro-structure of the face image. Despite its popularity, the LBP approach has some shortcomings, including sensitivity to noise, scale changes, and rotation in the image. The LBP assigns an 18 label to every pixel of an image by thresholding the 3x3-neighborhood of each pixel with the center pixel value, resulting in a binary number [50]. Ahonen et al. [6] applied the LBP on the FERET dataset show good robust performance using one sample per person for training. Besides LBP, other features that widely used in computer vision field can also be used in face recognition, such as fractal

features. For example, Komleh et al. [51] presented a method based on fractal features for expression invariant face recognition. Their method is tested on the MIT face database with 100 subjects. One image per subject was used for training while 10 images per subject with different expressions for testing. Experimental results show that the fractal features are robust against expression variation.

- B) The Gabor filters represent a powerful tool in image processing and image coding based on the capability of capturing important visual features, such as spatial localization, spatial frequency, and orientation selectivity. In most cases, the Gabor filters are used to extract the main features from the face images. The application of Gabor wavelet for face recognition is pioneered by Lades et al.'s work [52]. They used an elastic graph matching framework to find feature points, build the face model and to perform distance measurement, while the Gabor wavelets are used for extracting local features at these feature points, and a set of complex Gabor wavelet coefficients for each point is called a jet. Lades et al. used a simple rectangular graph to model faces in the database while each vertex is without the direct object meaning on faces. In the database building stage, the deformation process mentioned above is not included, and the rectangular graph is manually placed on each face, and the features are extracted at individual vertices. When a new face I comes in, the distance between it and all the faces in the database are required to calculate, that means if there are totally N face samples are present in the database, we have to construct N graphs

for I based on each face sample. This matching process is very computationally expensive, especially for a large database.

2.3 Classification

The simplest method for matching feature vectors is using the nearest neighborhood classifier. It calculates the distance between the source image vector to be classified and the dataset of images vectors, and then assigns the probe the class label of its nearest neighbor in the dataset. If the distance is zero, then the image matched are exactly the same. The distance measure can be converted to a similarity measure simply by negating it, such that the chosen match is the one with the maximum similarity value. The choice of distance metric depends on the type of task, such as Euclidean distance, cosine distance, and chi-square similarity. For an analysis of nearest neighbor pattern classification, see the article by Cover et al. [53]. The advantage of the NN-classifier is that it does not require any training stage and that it naturally extends to multi-class classification. Training other classifiers such as support vector machines (SVM's) [54] and neural networks [14-16] is often computationally demanding for high-dimensional data with many examples. Even though they may increase matching performance by accounting for non-linearity in the data, training would have to be done every time the gallery is altered.

2.4 Neural Network

Computer vision needs powerful classification methods to achieve a high recognition system rate with low computing time and resource. Neural network classification is widely used for training the neural network since NN is simple, efficient to compute the gradient descent, and straightforward to implement. Determine the size of the neural network, the number of samples and the weights is a challenging task, and it is important to fit the neural network output. The NN is divided into three layers which are training input layer, hidden layer (one or more), and the expected output layer as shown in Figure 2.3.

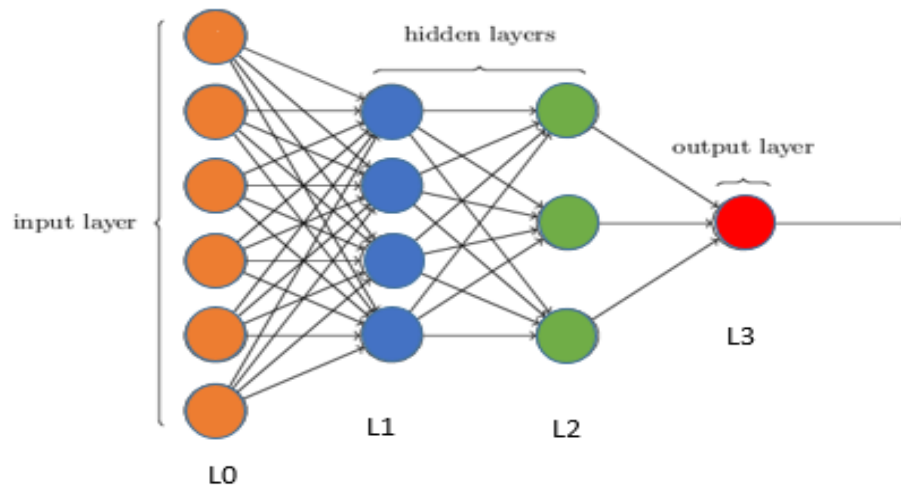


Figure 2.3. Three layers neural network.

One popular training method is the backpropagation algorithm that uses a gradient descent algorithm [55] to update the parameters of deep learning. In order for gradient descent to map the arbitrary inputs to the target outputs in an accurate manner, gradient descent has to find parameters such as weights w and biases b that minimize loss function.

The input data forward through the network layers to calculate the outputs to compare them with the expected outputs and compute the error of the loss function. The gradient of the loss function computes during the back-forward to update the parameters that minimize the loss function.

The common backpropagation algorithm can be described as the following:

1. The weights $w_{ij}^{[l]}$ and the thresholds $\vartheta_j^{[l]}$, randomly initialize let $n=1$.
2. Calculate the output of all layers according to equation (2.1) after feeding the prepared training dataset I_p and the output dataset O_p to the NN.

$$y_{jp}^{[l+1]} = f(\sum_{i=1}^{N_1} w_{ij}^{[l+1]} y_{ip}^{[l]} + \vartheta_j^{[l+1]}) \quad (2.1)$$

3. In each layer, compute the square root error as follows:

Equation (2.2) used to calculate the square error at the output layer:

$$er_{jp}^{[L]} = f'(net_{jp}^{[L]})(d_p - y_{jp}^{[L]}) \quad (2.2)$$

In the i^{th} hidden layer ($i=L-1, L-2 \dots i$):

$$er_{jp}^{[l]} = f'(net_j^{[l]}) \sum_{k=l+1}^{N_{l+1}} er_{kp}^{[l+1]} w_{jk}^{[l+1]} \quad (2.3)$$

4. The change in the weights between the input and the output will be calculated based on equation (2.4) and (2.5).

$$\vartheta_{ij}^{[l]}(n+1) = \vartheta_i^{[l]}(n) + \eta \cdot e_{jp}^{[l]} \quad (2.4)$$

$$\mathbf{w}_{ij}^{[l]}(\mathbf{n} + \mathbf{1}) = \mathbf{w}_{ij}^{[l]}(\mathbf{n}) + \eta \cdot \mathbf{e}_{jp}^{[l]} \cdot \mathbf{y}_{ip}^{[l-1]} \quad (2.5)$$

5. Go back to step 2 if the mean-squared error more than the threshold otherwise stop and print the weight value.

There is many of neuron activation function used in the neural network, and the sigmoidal function is what we used in our proposal system which is shown in equation (2.6).

$$f(x) = \frac{1}{1+e^{-x}} \quad (2.6)$$

Sigmoidal function derivative is:

$$f'(x) = f(x)(1 - f(x)) \quad (2.7)$$

J. Toms in 1990 improved the backpropagation algorithm using the hybrid neuron because in the big size neural network system was difficult to reach to the minimum mean-squared-error using the sigmoidal activation function compared to the small size neural network which the patterns of the input images are normally classified.

$$f(x) = \lambda \cdot s(x) + (1 - \lambda) \cdot h(x) \quad (2.8)$$

Where $h(x)$ is the hard-limiting function which is defined in equation (2.9) and the derivatives of the hybrid neuron is defined in equation (2.10)

$$h(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases} \quad (2.9)$$

$$f'(x) = \lambda s(x)(1 - s(x)) \quad \lambda \neq 0 \quad (2.10)$$

PBN often trapped into the local minimum and the learning speed is updated according to equation (2.11) where ESS is the Sum-Squared-Error. To make the NN faster and reach to zero error by adding a coefficient α to the steepness of the sigmoidal function as defined in Equation (2.12).

$$\lambda(n) = e^{-1/SSE} \quad (2.11)$$

$$f(x) = \frac{1}{1+e^{-\alpha x}} \quad (2.12)$$

Its derivative is:

$$f'(x) = \alpha \cdot f(x)(1 - f(x)) \quad (2.13)$$

Algorithm (1.1) highlights the essential steps of SGD with mini-batch in iteration k .

Algorithm 1.1. Stochastic Gradient Descent with mini-batch at iteration k

- 1: **Input:** Learning rate ϵ , initial parameters w, b , mini-batch size (m')
 - 2: **While** stopping criterion not met **do**
 - 3: Pick a random mini-batch with size m' from the training set $(x^{(1)}, \dots, x^{(m)})$
 - 4: with corresponding outputs $y^{(i)}$
 - 5: Compute gradient for w : $\nabla_w = \frac{1}{r} \nabla_w \sum_{i=1}^{m'} \ell(x^{(i)}, y^{(i)}, w)$
 - 6: Compute gradient for b : $\nabla_b = \frac{1}{r} \nabla_b \sum_{i=1}^{m'} \ell(x^{(i)}, y^{(i)}, b)$
 - 7: Apply update for: $w = w - \epsilon \cdot \nabla_w$
 - 8: Apply update for: $b = b - \epsilon \cdot \nabla_b$
 - 9: **end while**
-

Gradient descent can be categorized into Stochastic Gradient Descent (SGD) and Batch Gradient Descent (BGD). The difference between the two algorithms is how to handle the input data. BGD updates the gradient based on the entire training dataset in each iteration which is considered as a disadvantage, and it could be slow and expensive. However, the convergence is smoother, and the termination is more easily detectable. On the other hand, SGD is less expensive because the gradient computed for each training example and suffers from noisy steps and its frequent updates can make the loss function heavily fluctuate [56].

2.5 Deep Learning Background

Deep learning in the machine learning field achieved numerous performance in the computer vision and the processing of human language applications [57-61]. Deep learning is driven by understanding how the human brain processes information. The brain is organized as a deep architecture with several layers that process the information among many levels of non-linear transformation and representation [62]. Deep learning learns the hierarchy, structure, and pattern of the features from the lower level features using multi-level of hidden layers of non-linear transformations [60]. Very complex functions can be learned with enough such transformations. The higher layers of representation increase aspects of the inputs that are important for discrimination and suppress irrelative variation for any object recognition. For human face recognition, higher layers of representation amplify features of the inputs that are significant for discrimination and subdue irrelative features [63]. The first layer learns the low-level features such as curves, edges, and point from the image pixels. The low-level features are combined in the following layers to

produce higher features; for example, points and combined into lines and curves then they combined into shapes and more complex shapes. Once this is done, the deep neural network delivers a probability that these high-level features contain a particular object or scene. The main goal of deep learning is to automatically learn the most discriminative features from the raw data without human involvement. Convolutional Neural Networks (CNN) Stacked Auto-encoder (SA), Recurrent Neural Network (RNN), and Deep Belief Network (DBN) are the popular models for deep learning. [64, 65].

2.6 Convolutional Neural Network

CNN is a class of deep, feed-forward artificial neural networks, most commonly applied to analyzing visual imagery that takes advantage of the spatial construction of the inputs. A CNN consists of an input and an output layer, as well as multiple hidden layers.

CNN consist of alternating convolutional layers followed sometime by pooling layers and dropout layers to avoid the overfitting issue. The network end with few of fully-connected layers followed by classifier layer such as soft-max classifier or regression classifier as shown in Figure 2.4.

The CNNs gain the advantage by learning features representation automatically without depending on human-crafted features using end-to-end system starting from raw pixels to classifier outputs [61, 66]. Since 2012, researchers focus on improving the performance of CNNs architecture and methods such as layer design, activation function, and regularization, and exploring the performance in different fields [67, 68]. Resnet50,

Inception v4 and FaceNet are some of the existing models which they can be used to train a dataset on different domain issue.

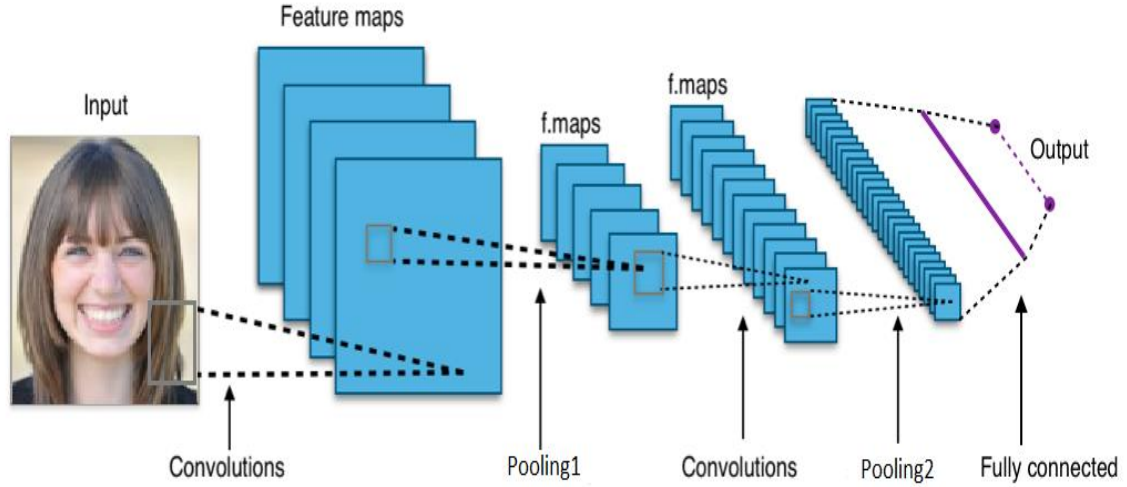


Figure 2.4. The typical structure of a CNN.

2.6.1 Convolutional Layer

The convolutional layer is the core building block of a CNN. The convolutional layer's parameters comprised of a set of learnable kernels (or filters). The convolutional layer extracts local features from the input by sliding a filter over the input and computing the dot product and producing a 2-dimensional activation map of that filter as shown in Figure 2.5. The feature maps connected to a small region of the input called receptive field, and the new feature map is generated by convolution operation and followed by a non-linear activation function as shown in Equation (2.14) to introduce non-linearity into the model.

$$x_f^{(l)} = f\left(\sum_{s,s} x^{(l-1)} * w_f^{(l)} + b_f^{(l)}\right) \quad (2.14)$$

Where the f is non-linear activation function, $b_f^{(l)}$ is shared bias of the feature map, $x^{(l-1)}$ is the output of the previous layer, $*$ is convolution operation, and $w_f^{(l)}$ is convolution filter with size $S \times S$.

CNN compute the gradient of the loss function with respect to the weights (w) and biases (b) of the respective layer in the backward phase as follows:

$$\nabla w_f^{(l)} l = \sum_{s,s} \left(\nabla x_f^{(l+1)} l \right)_{s,s} (x_{s,s}^{(l)} * w_f^{(l)}) \quad (2.15)$$

$$\nabla b_f^{(l)} l = \sum_{s,s} \left(\nabla x_f^{(l+1)} l \right)_{s,s} (x_{s,s}^{(l)} * b_f^{(l)}) \quad (2.16)$$

All units share the same weights (filters) among each feature map. The advantage of sharing weights is the reduced number of parameters and the ability to detect the same feature, regardless of its location in the inputs [69]. The hyper-parameters of each convolutional layer must be chosen carefully in order to generate desired outputs such as filter size, the number of learnable filters, and stride.

Numerous nonlinear activation functions are existing, such as sigmoid, tanh, and ReLU. However, ReLU is preferable because it makes training faster relative to the others [70, 71]. The filter size and stride decides the output features map size ($W \times W$) based on the input image with a size of ($H \times H$) over a filter with a size of ($F \times F$) and a stride of (S) as:

$$W = \left\lfloor \frac{H-F}{S} \right\rfloor + 1 \quad (2.17)$$

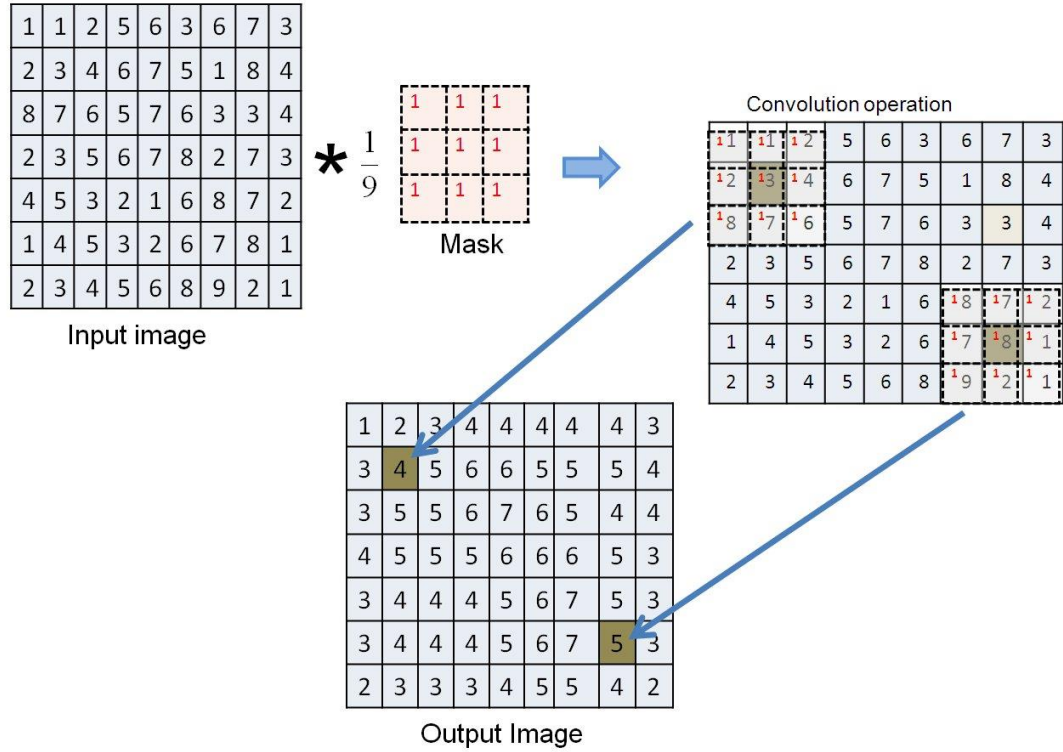


Figure 2.5. Convolution operation

2.6.2 Pooling Layer

Typically the pooling layers (down-sampling layer) applied after the convolutional layers to reduce the resolution of the previous feature maps and preserve most relevant feature. Pooling provides a fixed size output, which is important for classification. Pooling produces invariance to a small transformation and/or distortion. Pooling splits the inputs into disjoint regions with a size of $(R \times R)$ to produce one output from each region [72]. Two pooling are exist: Max-pooling and Average-pooling and the output size will be obtained from the input with a size of $(W \times W)$ by:

$$Pooling = \left\lfloor \frac{W}{R} \right\rfloor \quad (2.18)$$

We are losing global information about locality, and where in image something happened by applying a max-pooling in CNN [55, 73]. However, we are keeping information about whether the most important feature appeared in the image or not.

The maximum value of non-overlapping blocks from the previous feature map ($l-1$) is calculated during the forward phase as follows:

$$x^{(l)} = \max_{R,R} (x^{(l-1)})_{R,R} \quad (2.19)$$

There are no any learnable parameters in the Max pooling. Therefore, the gradient from the next layer is passed back only to the neuron that achieved the max value and all of the other neurons receive zero gradient. Figure 2.6 shows how the max-pooling works.

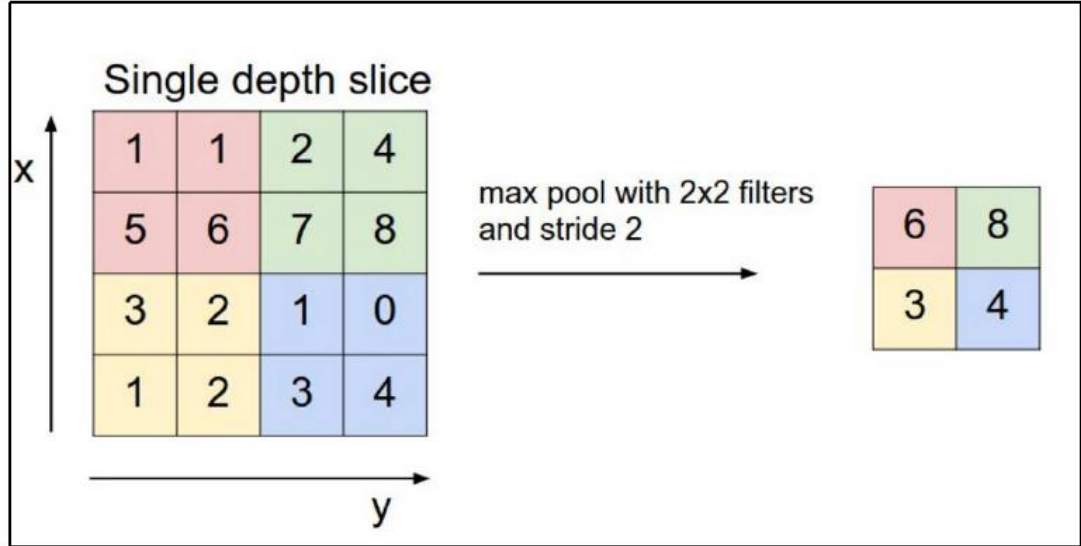


Figure 2.6. Max pooling in CNN.

2.6.3 Fully Connected Layer

The CNN ends with one or more of the fully connected layers that connect every neuron in one layer to every neuron in another layer and has the only number of neurons hyper-parameter. It is in principle the same as the traditional multi-layer perceptron neural network (MLP). Fully connected layer purpose is to extract the global features of the inputs, and the output is computed by Equation (2.20):

$$x^{(l)} = f((w^{(l)})^T \cdot x^{(l-1)} + b^{(l)}) \quad (2.20)$$

Where $x^{(l)}$ is the input, $w^{(l)}$ is the weights, and $b^{(l)}$ are the biases of the current layer. $x^{(l-1)}$ is the output of the previous layer, \cdot is a dot product, and f is the non-linear activation function. The last layer is the classifier layer such as soft-max classifier and regression classifier.

2.7 Literature Review for Face Registration

2.7.1 Speed-Up Robust Features (SURF)

The Speeded-up Robust Features (SURF) is a method to obtain the local features which will be used to align two related images which are taken at a different time or a different position. SURF process includes feature detection, feature description, and feature matching. The SURF steps are:

- 1) Find the integral image ($I\Sigma$) based on the input image I to achieve fast computation of the convolution filters. The value for each point $P = (x, y)$ is represented by the sum of all pixels in the input image I . Then we calculate the

sum of the intensities within a rectangular region formed by the origin and P using equation (2.21).

$$\Sigma = I\Sigma(C) - \Sigma(B) - \Sigma(D) + \Sigma(A) \quad (2.21)$$

- 2) Find the interesting points using hessian matrix by finding the maximum hessian matrix corresponding to a point $P=(x, y)$.
- 3) Subtract the adjacent Gaussian images to obtain the difference of Gaussian images by repeatedly convolved with Gaussians.
- 4) Optimize the key-points after obtaining image gradients using three methods to obtain the descriptor.
- 5) Finding the orientation Assignment: using the pixel differences, we compute orientation $\theta(x, y)$ and gradient magnitude $M(x, y)$ for each image sample $L(x, y)$.

$$M(x, y) = \sqrt{A^2 + B^2} \quad (2.22)$$

- 6) Obtaining Key-point descriptors: 64 element vector obtain by combining all the orientation histogram entries.
- 7) Perform the k-nearest-neighbor (KNN) on the feature values to find the distance between the features using four steps:
 - i. Compute the Euclidean distance on the obtained features.
 - ii. Descending sort of the labeled example.
 - iii. Based on root mean square deviation (RMSE), find the optimal K of the KNN.
 - iv. Represent the image based on these KNN.

2.7.2 Minimized Cost Function

Matrix equation (2.23) is used to minimize the cost of the image registration (without shear) when the related points in two images X and Y are identified. Summation indicates the sum over all points in an image. We used in our implementation the sigmoidal function as the neuron activation functions.

$$Cost = \sum (I1 - T(I2))^2 = \sum \left(\begin{pmatrix} x1 \\ y1 \end{pmatrix} - \left(\begin{pmatrix} a & b \\ -a & a \end{pmatrix} \begin{pmatrix} x2 \\ y2 \end{pmatrix} + \begin{pmatrix} t1 \\ t2 \end{pmatrix} \right) \right)^2 \quad (2.23)$$

To find the optimal transformation that will align image 2 to image 1, take the partial derivatives of the above cost with respect to a, b, t1 and t2 and set these to 0 ($\partial C/\partial a = 0$, $\partial C/\partial b = 0$, $\partial C/\partial t1 = 0$, $\partial C/\partial t2 = 0$). We express the four resulting equations in matrix form as is shown in equation (2.24).

$$\sum \begin{pmatrix} 2x_2^2 + 2y_2^2 & 0 & 2x_2 & 2y_2 \\ 0 & 2y_2^2 + 2x_2^2 & 2y_2 & -2x_2 \\ 2x_2 & 2y_2 & 2 & 0 \\ 2y_2 & -2x_2 & 0 & 2 \end{pmatrix} \begin{pmatrix} a \\ b \\ t1 \\ t2 \end{pmatrix} = \sum \begin{pmatrix} 2x_1x_2 + 2y_1y_2 \\ 2x_1y_2 - 2x_2y_1 \\ 2x_1 \\ 2y_1 \end{pmatrix} \quad (2.24)$$

After calculating the sum over all points for the 4x4 matrix and the right-hand 4x1 vector in equation (2.24), we can compute the required transformation by:

Matrix Ainv = A.Inverse and Matrix Res = Ainv * B.

2.7.3 Random Sample Consensus (RANSAC)

Fischer et al. introduced the RANSAC algorithm in 1981. RANSAC is one of the most suitable algorithms to eliminate the false matched points in the source and target

images in the presence of noise [17]. Some of the disadvantages of RANSAC are computing time, correct matches count, and the dependency of mismatches removal on the amount of threshold value. RANSAC algorithm is divided into four steps. The first step is to select a suitable model based on the transformation model. Equation (2.25) is used to calculate the number of related points which are required to calculate the transformation parameters. q is the minimum number of the related points and p is the number of the parameters need to calculate.

$$q = \frac{p}{2} \quad (2.25)$$

The second step is selecting the best model in a specified iterations. In each iteration, the minimum number of the related points randomly selected to estimate the transformation parameters based on equation (2.26) for 6 parameters a , b , c , d , e , and f .

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} x_1' \\ y_1' \\ 1 \end{bmatrix} \Rightarrow \begin{cases} x_1' = ax_1 + by_1 + c \\ y_1' = dx_1 + ey_1 + f \end{cases} \quad (2.26)$$

The third step is to calculate the distance between the source image and the transformed image. We consider a point is a right match if the distance is less than the threshold. Otherwise, we eliminate the point since it is not a true match. In the last step, for each iteration, we count the numbers of the true match, and if they are more than the desired value, or reaches the predetermined maximum number of iterations, then the algorithm stops. Transformation model selected which has the highest matching count.

CHAPTER 3: RESEARCH PLAN AND SYSTEM ARCHITECTURE

This research aims to achieve a higher human face recognition with less computation processing time. Image registration and feature extraction are the key points of improving face recognition. Therefore, we implemented our system based on multiple classification methods. To compare our results with the existing face recognition systems, we applied our frameworks on two of the well-known human face image databases, Yale and Olivetti Research Laboratory Human Face Datasets. We implemented some of the existing face recognition systems to compare our testing results to their results. We randomly obtain the training set and the testing set with different scenarios. Different size of dataset applied to like 90% of the picture as training and 10% as a testing set or 70% to 30%. Our goal is to achieve higher accuracy of the 50% to 50% scenario. Finally, recognition rate (RR) is calculated using equation (3.1) after the system finishes training the NN then test the testing set.

$$RR(\%) = \frac{\text{Number of correct match}}{\text{Number of training set}} * 100 \quad (3.1)$$

3.1 Human Face Datasets

3.1.1 ORL Dataset

Olivetti Research Laboratory Dataset (ORL) [74]. ORL dataset represents images of 40 different persons with ten different pictures for each person. A total of 400 face

images used for training and testing the system. The 400 images are in grayscale, and the size is 92X112 pixels with variant expressions, timing, pose, and gender. Figure 3.1 shows a sample of the ORL dataset.



Figure 3.1. Sample of the ORL dataset.

3.1.2 Yale Dataset

The Yale face dataset has a total of 165 face images which represent 15 different persons with 11 images per person [75]. Different facial expression, gender, light configuration, and with or without wearing eyeglasses. The 165 images are in a grayscale

domain and images resized to 92x112 pixels after we cropped the face only. Figure 3.2 shows Yale sample images.



Figure 3.2. Sample images of Yale dataset.

3.1.3 Labeled Faces in the Wild

The data set contains more than 13,000 images of faces collected from the web. Each face has been labeled with the name of the person pictured. 1680 of the people pictured have two or more distinct photos in the data set. The only constraint on these faces is that they were detected by the Viola-Jones face detector. The images are in color scale, and the size is 250x250 pixels with variant expressions, timing, pose, and gender. Figure 3.3 shows a sample of the LFW dataset.

3.2 Classical Face Recognition System

We implemented the classical face recognition system as a baseline for our research. First, we used the PCA and the KNN classifier [1] only as shown in Figure 3.4.

The goal of the PCA is to reduce the face image dimension by using only the highest K Eigen-values and their corresponding Eigen-vectors with losing minimal information, which helps to reduce the computation process. Second, we used LBPH and the KNN classifier as shown in Figure 3.5. The goal of the LBPH is to reduce the face image dimension by dividing the image into small regions called cell and represent each cell by 59 dimensions with losing minimal information. We used five different distance classifiers to measure the similarity between the source image and the test image to show how accuracy is different between the methods, which lead us to combine them in the proposed framework to achieve higher accuracy.



Figure 3.3. Sample images of LFW dataset.

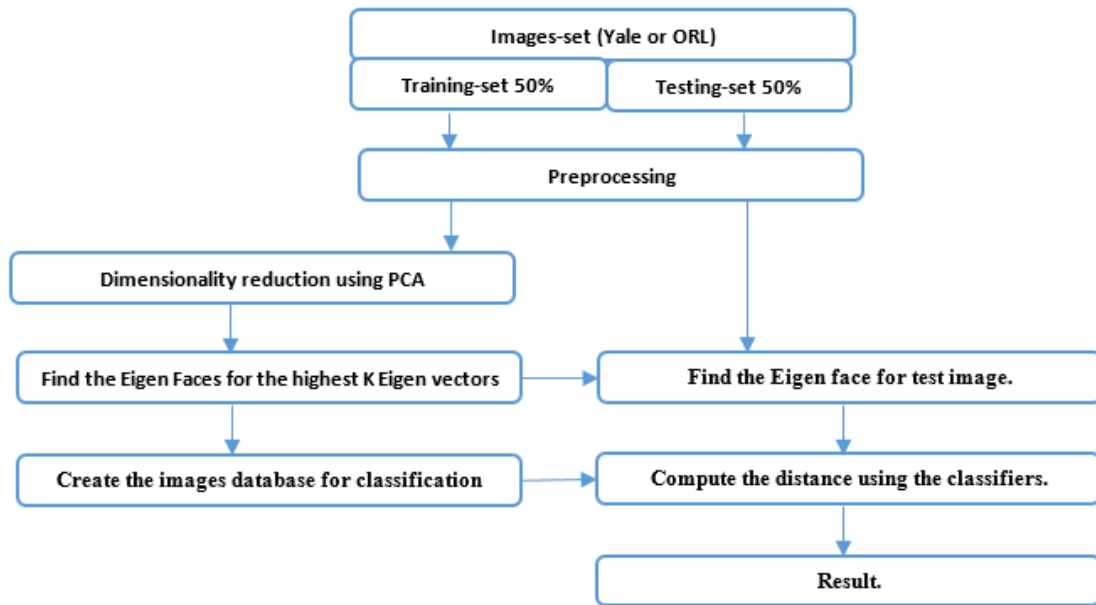


Figure 3.4. Classical face recognition system using The PCA and KNN classifier methods.

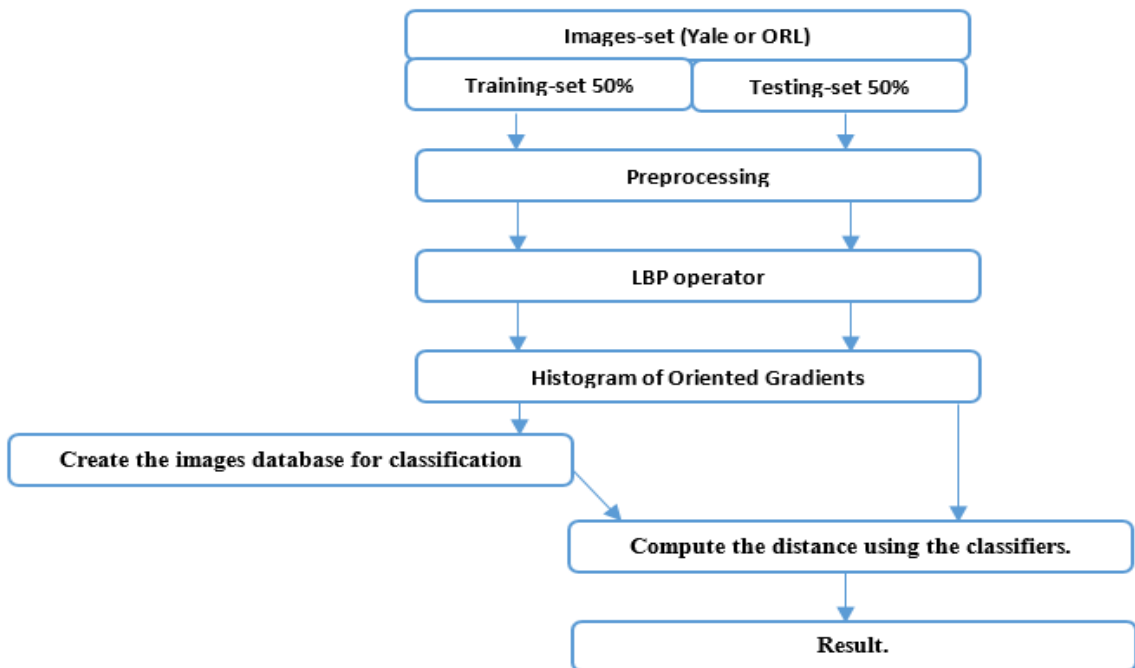


Figure 3.5. Classical face recognition system using the LBPH and KNN classifier methods.

We reduced the computation time in both frameworks by preprocessing the face images using different methods as needed. The preprocessing starts with resizing the image to a reasonable size followed by cropping the face only to eliminate the face background effect. Reducing the noise and the illumination by converting the face images to grayscale images and histogram equalization to build a robust face recognition system, Figure 3.6 shows some of the preprocessing methods. The new face representation is obtained from the PCA or the LBPH which present the face image in a smaller dimension. Finally, using one of the classifier measurements we calculate the similarity between the source image and the target image and we consider the matching occurs only if the lowest KNN neighbor matches the source as shown in Figure 3.7(a) otherwise it will be considered as a mismatch as shown in Figure 3.7(b). Then, we calculate the recognition rate for the whole system using equation (3.1).

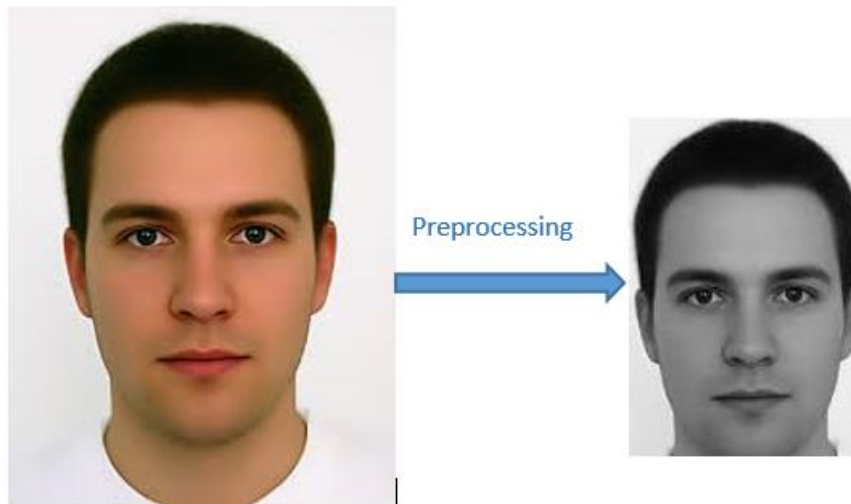


Figure 3.6. Example of Preprocessing Methods including cropping, resizing and Histogram.

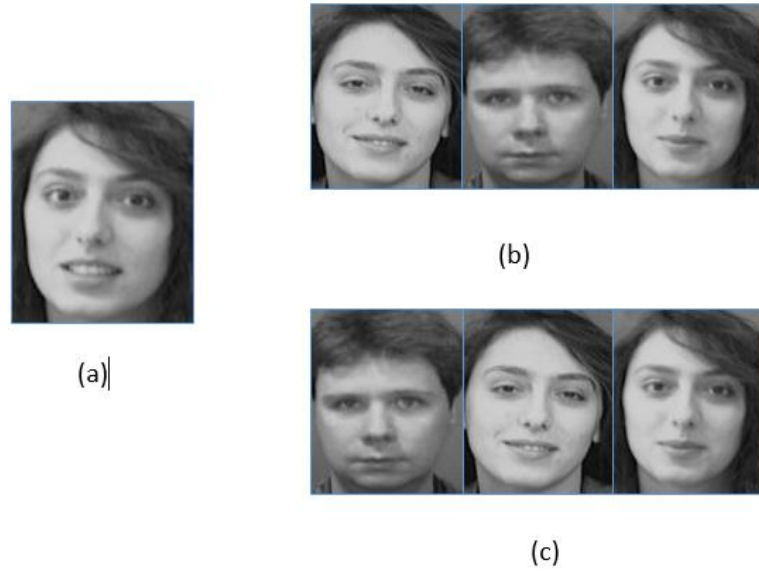


Figure 3.7. Example of the matching case and mismatching case image using KNN with Mahalanobis distance. (a) Test image (b) Matching case. (c) Mismatching case.

3.2.1 Principle Component Analysis

Image recognition and detection need a massive resource of storage and powerful system to reduce the computation processing time and cost. Therefore, the dimension reduction and image re-representation are needed as the first step in any face recognition systems. PCA is one of the popular statistical transform methods. PCA reduces the image dimension by analyzing the image and identifying distinction patterns which can be used as a new representation of the image without losing an enormous information content from the image. Face dimension reduction is obtained by applying the PCA and finding the highest K Eigen-Values and their corresponding Eigen-Vectors. The face image can be re-represented using only 15% of the Eigen-Values with a minimal losing of information [76]. PCA is relying on the variance-covariance matrix. Therefore, the images are not a significant comparison to the number of face images in the training dataset. The advantages

of the PCA are low noise sensitivity, eliminating the data redundancy by providing the orthogonal components and reducing the image complexity. The disadvantages of the PCA methods are evaluating the variance-covariance matrix and capturing the invariance. Evaluating the variance-covariance matrix in an accurate manner and capturing the invariance is difficult unless the training dataset explicitly provides the information. Figure 3.7 shows an example of the PCA methodology. Figure 3.8(a), the face data are randomly distributed, and Figure 3.8(b) shows the correlated data grouped in the same coordinate face space.

The PCA process starts with standardizing the image scale by obtaining the same face size vector Γi for all of the training images $I_1, I_2 \dots I_M$. The training data-set of M faces written as $I = (I_1, I_2, I_N)$ and the average image Ψ is obtained by equation (3.2).

$$\Psi = \frac{1}{M} \sum_{i=1}^M \Gamma i \quad (3.2)$$

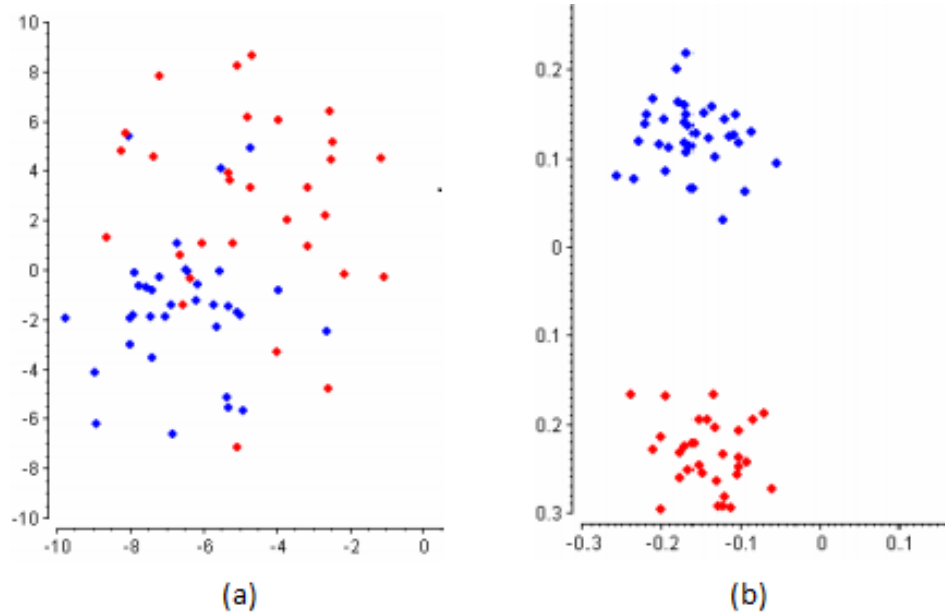


Figure 3.8. An example of PCA (a) Original data. (b) Correlated data

Then centralize each training image by subtracting the mean, which is the average across all dimensions from each image and finds the vector $\Phi = \Gamma i - \Psi i$. PCA uses equation (3.3) to calculate the covariance-matrix C, which is used to find the Eigen-Value.

$$C = \frac{1}{M} \sum_{M=1}^M \Phi m \Phi m^T = A A^T \quad (M^2 \times M^2 \text{ matrix}) \quad (3.3)$$

Where $A = [\Phi_1 \Phi_2 \dots \dots \Phi_M]$ ($N^2 \times M$ matrix).

We can calculate the Eigen-Values μ_i and the Eigen-Vectors v_i from the covariance matrix since the matrix is square using equation (3.4).

$$A^T A v_i = \mu_i v_i \quad (3.4)$$

PCA significantly orders the Eigen-Values from highest to lowest. We then ignore low significance Eigen-Values to reduce the face domain, and we obtain the corresponding Eigen-Vectors. PCA loses some of the information from the image. However, this will not affect the recognition since most of the data discrepancy exists in the first 15% of the face dimension. We obtain the Eigen-Faces by transpose the Eigen-Vectors then multiply them by the original faces dataset. Eigen-Faces appears as ghostly faces in Figure 3.9.

$$\text{Eigen-Faces} = (\text{Original data-set}) \times (\text{Eigen-Vectors})$$

3.2.2 Local Binary Patterns Histogram (LBPH)

Correlation methods require substantial computation time and enormous amounts of storage. Therefore, features reduction and face representation are needed in the face recognition system. LBPH is usually a preferred method in computer vision, image processing, and pattern recognition; it is appropriate for the feature because it describes the texture and structure of an image. We represent the face image and reduce the image

dimension by applying the LBPH method, extracting the features texture of the image by dividing the image into local regions and extracting the binary pattern for each local region. The original LBP operator, which works on eight neighbors of a pixel, was introduced by Ojala et al. [49]. The image is divided into small regions called the cell. Each pixel in the cell is compared with each of its eight neighbors. The center pixel value will be used as the threshold value [6-11]. The eight-neighbors-pixel will be set to one if its value is equal to or greater than the center pixel; otherwise, the value is set to zero. Accordingly, the LBP code for the center pixel is generated by concatenating the eight neighbor pixel values (ones or zeroes) into a binary code, which is converted to a 256-dimensional decimal for convenience as a texture descriptor of the center pixel. The original LBP operator is shown in Figure 3.10.

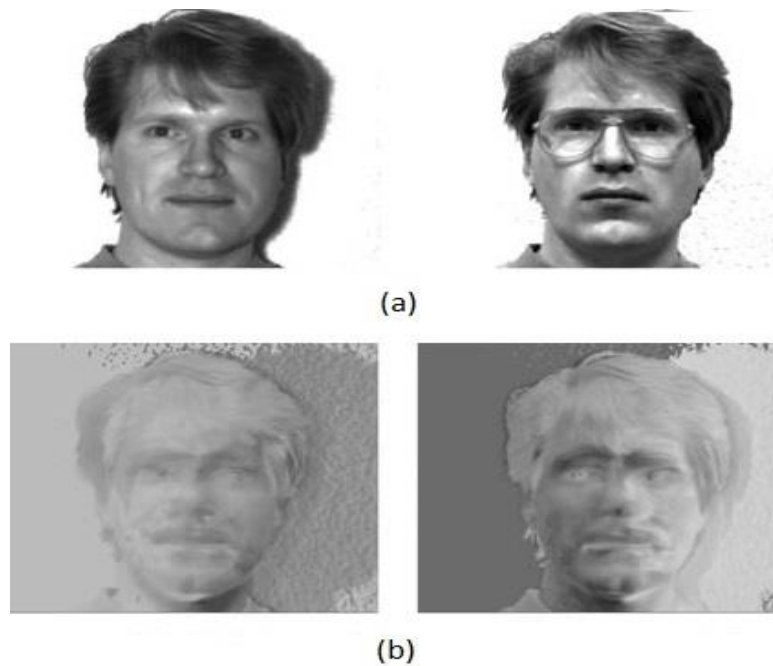


Figure 3.9. (a) Original faces (b) Corresponding Eigen-faces.

The mathematical formulation of LBP operator is given by:

$$LBP(x) = \sum_{i=1}^8 s(G(x^i) - G(x))2^{i-1} \quad (3.5)$$

$$s(t) = \begin{cases} 1 & t \geq 0 \\ 0 & t < 0 \end{cases} \quad (3.6)$$

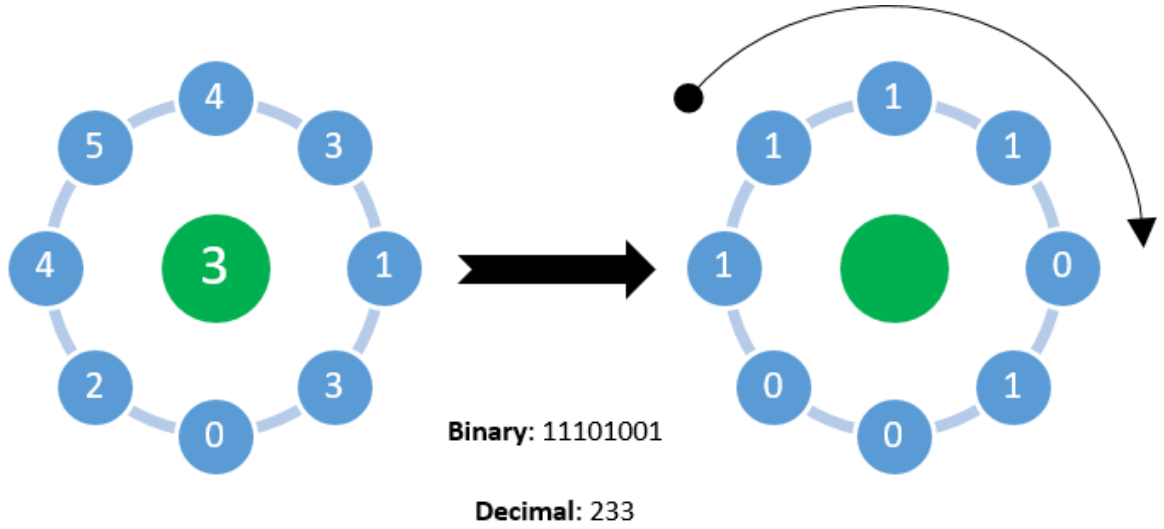


Figure 3.10. Original LBP Operator.

We used a modified LBP operator called a uniform pattern. The pattern is the number of bitwise transitions from 1 to 0 or vice versa. The LBP is called uniform if its uniformity measure is at most 2. For example, the patterns 11111111 (0 transitions), 01111100 (2 transitions) and 11000111 (2 transitions) are uniform, while the patterns 10001000 (3 transitions) and 11010011 (4 transitions) are not. For dimension reduction, we used the histogram to reduce the image features from a 256-dimensional decimal to a 59- dimensional histogram, which contains information about the local patterns. The histogram uses a separate bin for each uniform pattern, and one separate bin for all non-uniform patterns. In the 8-bit binary number, we have 58 uniform patterns; therefore, we

used 58 bins for them and one bin for all non-uniform patterns. The global description of the face image is obtained by concatenating all regional histograms. The overall value of LBPH can be represented in a histogram as (3.7):

$$H(k) = \sum_{i=0}^n \sum_{j=1}^m f(LBP_{P,R}(i,j), k), k \in [0, k] \quad (3.7)$$

Where P is the sampling points, and R is the radius.

Figure 3.11 shows the process of getting the feature vector for each image, which will be fed to the classifier.

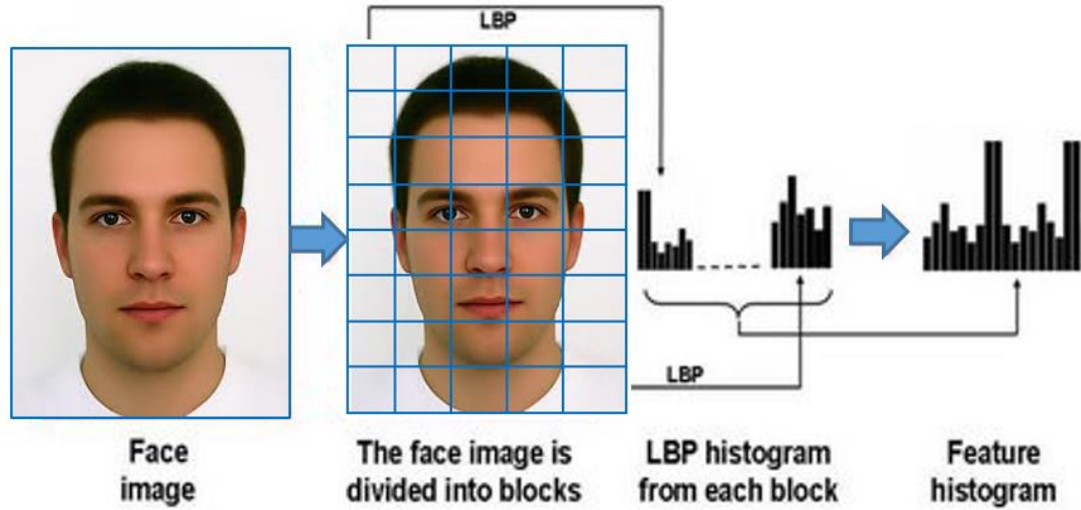


Figure 3.11. Face description with local binary patterns.

3.2.3 Similarity Measurements Methods

The K-Nearest-Neighbors (KNN) is one of the methods used in the computer vision. Most of the KNN use the Euclidean distances. However, it produces less accurate results than the other methods. Each distance method provides different levels of accuracy based on the problem domain. Therefore, the first contribution is to combine some of them

to improve the face recognition accuracy. Mahalanobis distance measurement provides more accurate result than Minimum Distance depending on the covariance matrix between the two vectors in the (3.8) [77].

$$\text{Mahalanobis}(x, y) = \sqrt{(x_i - y_i)^T S^{-1} (x_i - y_i)} \quad (3.8)$$

Where S^{-1} is the covariance matrix inverse.

Correlation distance classifier was introduced by Székely, Rizzo, and Bakirov in 2007 [78]. A valuable property is the measure of dependence equal zero and is sensitive to a linear relationship between two vectors.

$$\text{Correlation}(x, y) = \frac{\text{Cov}(x, y)}{\sigma_x \sigma_y} \quad (3.9)$$

Where Cov is the covariance and σ_x and σ_y are the standard deviations of x and y .

Euclidean distance method is considered the basis of many measures of similarity and dissimilarity. We use (3.10) to calculate the Euclidean distance between corresponding elements of the two vectors space.

$$\text{Euclidean}(x, y) = \sqrt{\sum_{i=1}^M (x_i - y_i)^2} \quad (3.10)$$

The Canberra distance classifier is a numerical measure of the distance between two points in a vector space, which is presented in (3.11):

$$\text{Canberra}(x, y) = \sum_{i=1}^M \frac{|x_i - y_i|}{|x_i| + |y_i|} \quad (3.11)$$

The Manhattan distance classifier is another method to measure the distance between two vectors and is introduced in (3.12):

$$\text{Manhattan}(x,y) = \sum_{i=1}^M |x_i - y_i| \quad (3.12)$$

We used different distance classifier methods to provide a variant dataset to improve the training of the neural network.

3.3 Face Recognition using PCA and NN Proposed System

3.3.1 Proposed Method

We proposed in this paper an enhanced Face Recognition Framework Based on Correlated Images and Back- Propagation Neural Network. The main contributions of our work are:

- Using five distance methods and combining them will provide a clear pattern which helps the NN to converge faster and more accurate.
- Obtaining the T-Set based on the correlation between the training dataset will provide robust data which we used as an input of the NN.
- Each distance method performs well in a different direction. Therefore, adding a strength factor helps to improve the accuracy rate.

The proposed framework is divided into five steps as shown in Figure 3.12

- 1) Preprocessing step: Face recognition needs huge storage and CPU resources. Therefore, we applied a few of the preprocessing operations to reduce the computing time as shown in Figure 3.5. Haar-cascade

detection is used to detect the face then we cropped the face to reduce the background effect. We converted the images to a gray-scale image then we applied a histogram equalizer to reduce the noise effect. Finally, we resized the images to the size we preferred.

- 2) Features extraction: We used the PCA algorithm to reduce the dimensionality of the images by eliminating the redundant data between the training images while retaining the variation between them. The PCA is transforming the dataset into a new set of variables which called the principal components (PCs). The first PC retains the maximum variation in the dataset. The PCA sorts the Eigen-Vectors and selects to top K values to reduce the dimensions. The training dataset must be scaled, and the complexity of calculating the covariance matrix are some of the drawbacks of the PCA. However, we used the PCA to prove that there is a potential accuracy improvement using the traditional methods by adding an extra step (step 3) to obtain the T-Set based on the correlated training dataset images.
- 3) Obtaining the T-Set: We added this step to obtain a correlated dataset which we called the T-Set and use it as an input of our NN. The T-Set has strong distinction patterns which improved the overall accuracy rate of the face recognition system. The next steps are applied to each of the training images to obtain the T-Set:
 - a. Based on the reduced dimension of each training image from step 2 and using Mahalanobis, Manhattan, Correlation, Canberra

and Euclidean distance methods; we separately computed the distance between each training image and all other images.

- b. First, we trained our NN using each method individually, and we achieved different accuracy rates as shown in Table 4.1. Therefore, we decided to combine the five distance methods using equation (3.13).

$$DSS = \sqrt{\sum_{i=1}^5 DIS_i^2} \quad (3.13)$$

Where DIS_i is one of the distance methods.

- c. However, based on the classical face recognition experiment, each distance algorithm has an advantage over the other algorithms in different dimensions. Therefore, we modified (3.13) to (3.14) by adding a strength factor α to improve the accuracy result in the final scenario. Table 4.1 shows that the Mahalanobis and Manhattan distances have an advantage over the other distance methods. Therefore, we assign the strength factors as: Mahalanobis and Manhattan = 0.3, Canberra = 0.2, Correlation and Euclidean = 0.1.

$$DSS_{\alpha} = \sqrt{\sum_{i=1}^5 \alpha_i DIS_i^2} \quad (3.14)$$

Where $\sum_{i=1}^5 \alpha_i = 1$

- The KNN method is used to find the expected output for each training image, and we selected K=1 to avoid majority voting, which leads to incorrect votes since the dataset has identical or nearly identical images. Our decision is based on the nearest neighbor, and we considered a match to have occurred if the nearest neighbor matches the source image.

Table 3.1 shows an example of how to obtain the T-Dataset (column 6) and the expected output (column 7) for one of the training images (image X). We assume the training dataset has 200 images that represent 40 persons.

Table 3.1. An example of how to obtain the new training data (column6) and the expected output (column 7) for one of the training images (imageX).

		Column1	Column2	Column3	Column4	Column5	Column6	Column7
	Distance Between image X and	Using Correlation	Using Euclidean	Using Canberra	Using Manhattan	Using Mahalanobis	Combine the result using: $RSS_{\alpha} = \sqrt{\sum_{i=1}^5 \alpha_i DIS_i^2}$	expected Output Based on the KNN (K=1)
person1	Image1	0.39	0.34	0.71	0.34	0.56	1.091	1 (best match)
	Image2	0.31	0.32	0.67	0.42	0.49	1.033	
	Image3	0.41	0.28	0.77	0.29	0.39	1.037	
	Image4	0.29	0.29	0.64	0.27	0.48	1.533	
	Image5	0.72	0.54	0.85	0.48	0.79	1.544	
Person2	Image6	0.19	0.77	0.61	0.56	0.32	1.190	0
.
.
.
	Image198	0.51	0.57	0.74	0.21	0.64	1.259	
	Image199	0.41	0.27	0.78	0.41	0.71	1.233	
	Image200	0.39	0.54	0.92	0.69	0.69	1.497	

- 4) Set up and train the NN: We start the training after we set up the NN parameters such as the number of the hidden layers, the number of the neurons of each layer, number of the iteration, threshold value, setup the input matrix and finally, setup the output matrix.
- 5) Testing the system: We found the reduced data of the testing image using the Eigen-Vectors which we obtained from step 2. Then, we fed the testing image the trained NN to calculate the predicted output label. We computed the accuracy based on comparing the predicted label and the expected label. Finally, we calculated the overall accuracy rate of the framework.

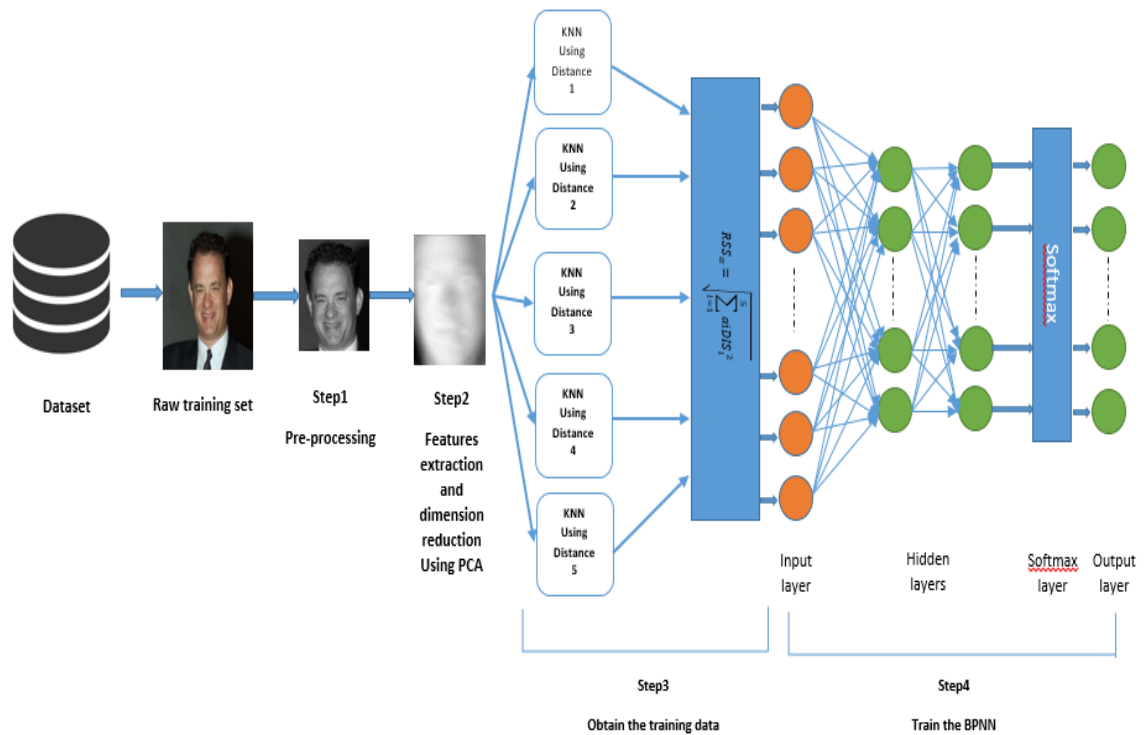


Figure 3.12. Proposed recognition system using PCA and NN classifier methods.

3.4 Face Recognition using LBPH and NN Proposed System

3.4.1 Proposed Method

We proposed in this work an enhanced human face recognition using LBPH descriptors, multi-KNN, and BPNN neural network. Figure 3.13 shows the proposed framework in detail. Our main contribution is based on the fact that obtaining a robust T-Dataset will help the BPNN to converge quickly with improved accuracy. We gathered a robust T-Dataset relying on the correlation between the training images, not the density of images. Our method is divided into five steps. In step one, we applied some of the preprocessing methods on the raw training images, including resizing and cropping using Haar-cascade detection, to eliminate the face background effect. Noise and illumination were reduced by converting the images to grayscale images and using histogram equalization to build a robust face recognition system; Figure 3.6 shows some of the preprocessing methods.

In Step 2, we extracted the most significant local features from each image using the $LBP_{8,2}^{U2}$ descriptor and combined them into a global description using the histogram method.

Here is how it is done:

- We divided the images into 25 small cells after we tried different grid sizes. We found that the 5x5 grid gives us better performance with a reasonable time. Smaller grid sizes such as 4x4 provide fewer features ($4 \times 4 \times 59 = 944$) compared to ($5 \times 5 \times 59 = 1475$ features), which leads to less accuracy

and perhaps to an under-fitting problem with the neural network training. A larger grid size provides more features; however, it increases the computing time with a slight improvement in accuracy.

- We applied the LBP method on image pixels by thresholding the 3 x 3 neighborhood of each pixel with the center value and considering the result as a binary number.
- Finally, we applied the histogram method to concatenate the new cells description and obtain a new representation (25 cell * 59 dimensions = 1475) for each training image, which helps to reduce the computation time.

Step 3 was added as an extra step to obtain a robust T-Dataset, which we used as an input to our BPNN instead of using the LBPH descriptor of each training image. As mentioned earlier, the T-Dataset is gathered based on the correlation between the new representations of all training images.

- Based on the LBPH presentation of each image, which we obtained from step 2, we calculated the distance between each training image with all other training images using five distance methods (Correlation, Euclidean, Canberra, Manhattan, and Mahalanobis).
- We tried different scenarios to achieve higher accuracy. First, we trained the BPNN using each distance method separately, and we achieved variant accuracy as shown in table 4.2. In another scenario, we combined the five

distances using the square-root of the sum of the squares (SRS) (23) to provide a robust distinction T-Dataset in a reduced dimension.

- However, based on the classical face recognition experiment, each distance algorithm has an advantage over the other algorithms in different dimensions. Therefore, we modified (3.13) to (3.14) by adding a strength factor α to improve the accuracy result in the final scenario. Table 4.1 shows that the Mahalanobis and Manhattan distances have an advantage over the other distance methods. Therefore, we assign the strength factors as Mahalanobis and Manhattan = 0.3, Canberra = 0.2, Correlation and Euclidean D 0.1.
- The KNN method is used to find the expected output for each training image, and we selected K=1 to avoid majority voting, which leads to incorrect votes since the dataset has identical or nearly identical images. Our decision is based on the nearest neighbor, and we considered a match to have occurred if the nearest neighbor matches the source image.

Table 3.1 shows an example of how to obtain the T-Dataset (column 6) and the expected output (column 7) for one of the training images (image X). We assume the training dataset has 200 images that represent 40 persons.

In Step 4, the BPNN parameters are set up then the training begins. Our BPNN architecture contains an input layer followed by two fully connected hidden layers, followed by a soft-max classification layer.

- Set the number of layers and neurons.

- Set the number of iterations and set the threshold value.
- Set the input matrix and the expected output from the previous step.
- Randomly initialize the weights and bias then start the training.

Finally, we test the accuracy of the system by Applying steps 1 to 3 for each testing image.

- Feeding the testing image data to the trained BPNN and obtaining the predicted output.
- Based on the image label, we know whether the prediction is correct.
- Finally, the overall system accuracy is calculated.

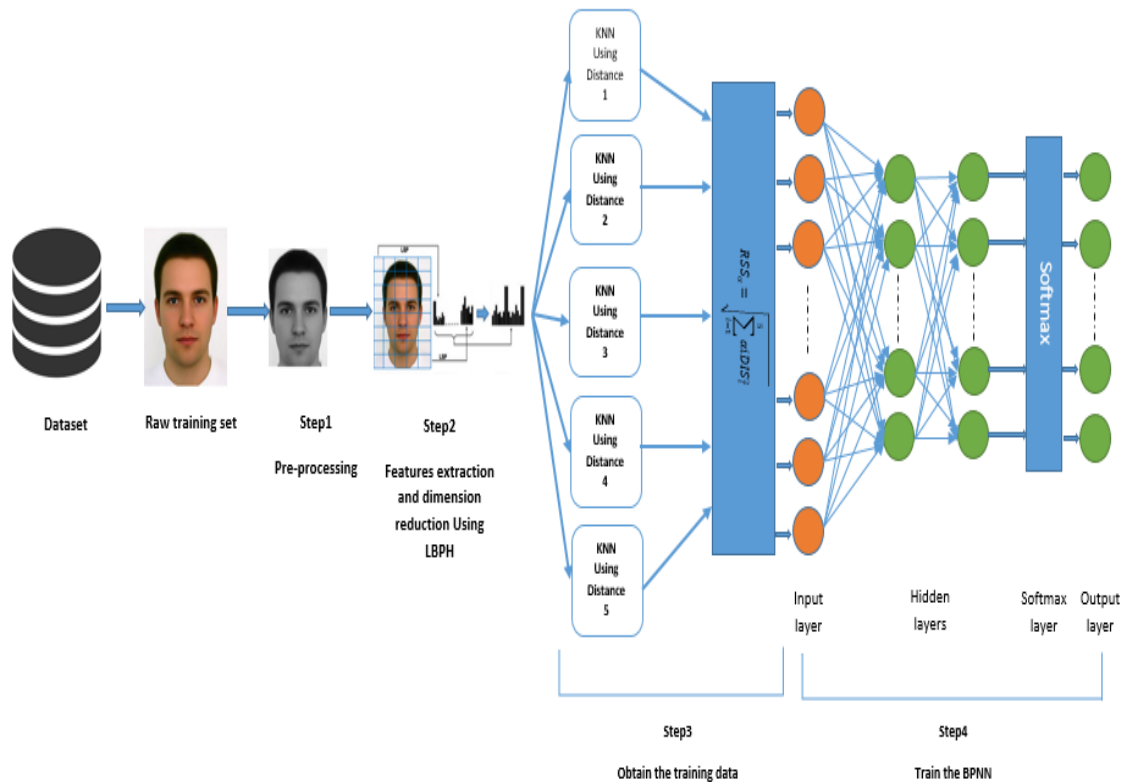


Figure 3.13. Proposed recognition system using LBPH and NN classifier method.

3.5 Face Registration Based on a Minimalized Cost Function

3.5.1 Proposed Method

In this experiment, we applied our proposed method to a 256x256 pixels Lena image. Image registration objective to align two images of the same object to the same coordinator. Suppose we obtain two images for the same parson I1 and I2 which are acquired at a different angle or distance. In image I1 (x, y) is a point and (x', y') is a related point in the image and the transformation model between the two images include scale, rotation, and translation which can be expressed in equation (3.15).

$$\begin{bmatrix} x_1' \\ y_1' \\ 1 \end{bmatrix} = s \begin{bmatrix} \cos(\theta) & -\sin(\theta) & tx \\ \sin(\theta) & \cos(\theta) & ty \\ 0 & 0 & 01 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \quad (3.15)$$

Where tx and ty are horizontal, vertical translation parameters, s is the scale parameter, θ is the rotation angle parameter. Our purpose is to find these key-points then the matching points, calculate the affine transformation parameters, then align two images as is shown in Figure 3.14.

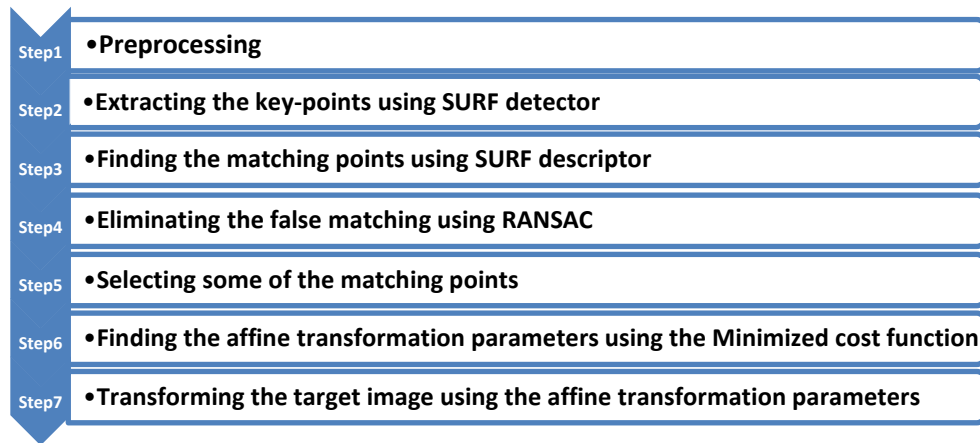


Figure 3.14. The proposed image registration method.

We used the SURF algorithm in our proposed method to find the key-points for both source and target images. We choose a high threshold to reduce the number of the key-points since we are looking for a few of the matching points to register the image using the minimized cost function. Figure 3.15 shows the key-points for an image with a different threshold (100, 50 and 20).

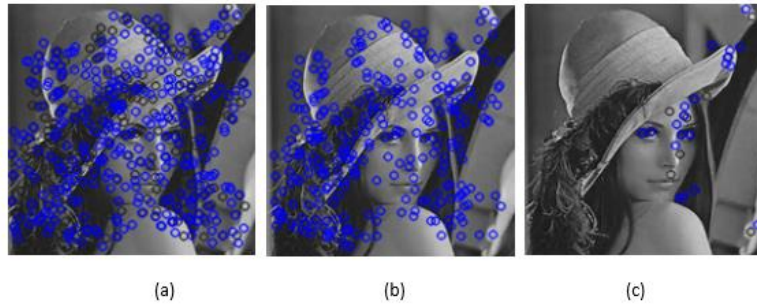


Figure 3.15. Key-points for Lena with variant threshold. (a) Threshold =100. (b) Threshold =50. (3)Threshold =20.

After we obtain the key-points, we used the SURF descriptor to find the matching points. With a large number of key-points, we found some of the false matching points which we have to eliminate to achieve higher registration accuracy. Figure 3.16 shows the matching points between two images with some of the false matching points. Then, we used the RANSAC algorithm to eliminate all the false matching points as shown in Figure 3.17.

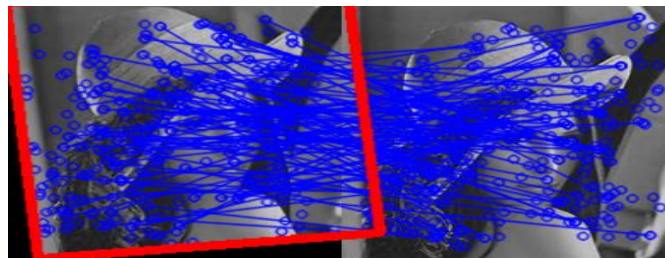


Figure 3.16. Matching points with false matching points.



Figure 3.17. Matching points after eliminating the false points.

The next step is to find the affine transformation parameters using the minimized cost function with few matching points. We have been able to achieve the lowest error with only two matching points. Figure 3.18 shows the registration example with only two matching points.



Figure 3.18. Example of image registration. (a) Source and target images with two matching points.
(b) Registered image.

3.6 Deep Learning Face Registration

The software engineering is nowadays moving in the direction of machine intelligence. Machine Learning has become essential in every segment as a way of making machines intelligent. More simply, Machine Learning is a set of algorithms that analyze data, learn from them, and then apply what they've learned to make intelligent decisions. In this proposed method, we are trying to take advantage of the deep learning to build a robust face registration system based on the transfer learning from pre-trained models.

3.6.1 Obtaining the Training Dataset

Based on our survey, we could not find a dataset used to the face registration problem. Therefore, we had to build our face registration dataset based on one of the existing datasets. We decided to go with the labeled faces in the wild dataset which we described in section 3.1.3.

We selected one of the LFW images as a reference image for all other images in the dataset. The reference image is centered and has a frontal face with the assumption that, no any transformation (rotation, scaling and shifting) applied to the reference image. Figure 3.19 shows our reference image which we used to find the transformation parameters.



Figure 3.19. Refrenace Image.

Next, we applied the haar-cascade face detection algorithm on the reference image to find the face boundary. Then, we detected the 6 facial landmarks associated with each eye based on the haar-cascade algorithm as it is shown in Figure 3.20. In addition to the reference image, we apply the same method to all images in the dataset. Each eye is represented by 6 (x, y)-coordinates, starting at the left-corner of the eye (as if you were

looking at the person), and then working clockwise around the remainder of the region. The total of points we will use to calculate the transformation parameter is 12 points, and we will eliminate any image has less than that from the training dataset.

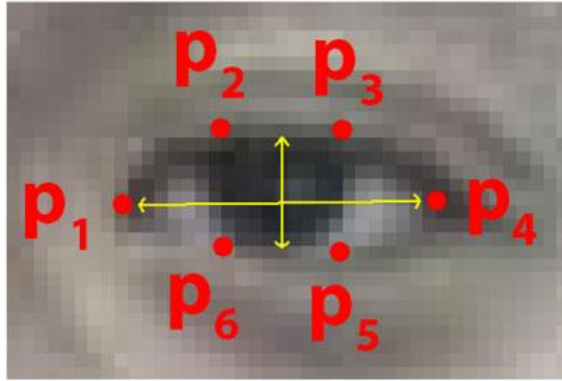


Figure 3.20. The 6 facial landmarks associated with the eye.

To find the transformation parameters for an image, we passed the 12 points of the reference image and the 12 points of the target image to the minimized cost function method which we described in section 3.5. Minimized cost function will return the transformation parameters (Rotation, Scaling shifting). We obtained our training dataset by applying the same method on all images.

3.6.2 VGGNet Model

VGG is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition” [57]. The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. VGG shows that a significant improvement on the prior-art configurations can be achieved by

increasing the depth to 16-19 weight layers, which is substantially deeper than what has been used in the prior art. To reduce the number of parameters in such very deep networks, VGG use very small 3×3 filters in all convolutional layers (the convolution stride is set to 1). However, VGGNet consists of 138 million parameters, which can be a bit challenging to handle. Figure 3.21 shows the general VGG model architecture.

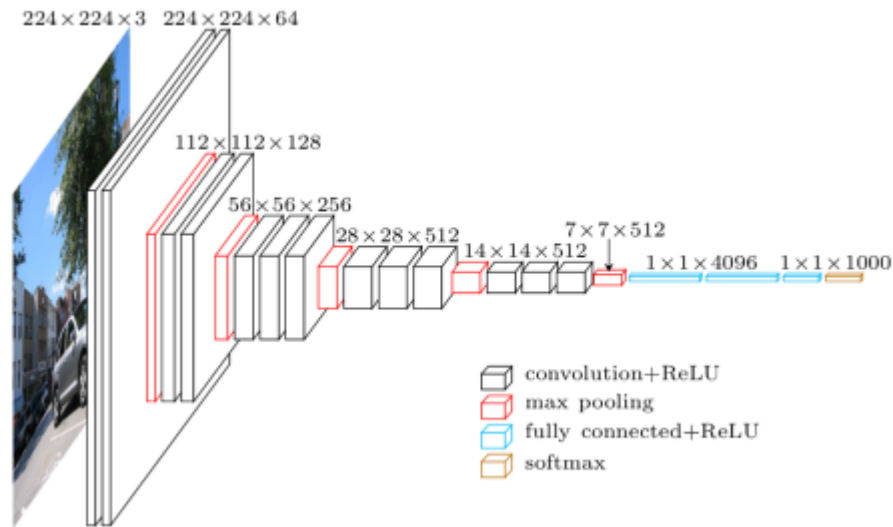


Figure 3.21. VGGNet model architecture.

3.6.3 Residual Neural Network (ResNet) Model

Residual Network was possibly the most groundbreaking work in the computer vision/deep learning community in the last few years. ResNet makes it possible to train up to hundreds or even thousands of layers and still achieves engrossing performance. Increasing network depth does not work by simply stacking layers together. Deep networks are hard to train because of the infamous vanishing gradient problem—as the gradient is back-propagated to earlier layers, repeated multiplication may make the gradient

infinitively small. As a result, as the network goes deeper, its performance gets saturated or even starts degrading rapidly.

The main idea of ResNet is introducing a so-called “identity shortcut connection” that skips one or more layers, as shown in Figure 3.22. The authors of the ResNet argue that stacking layers shouldn’t degrade the network performance, because we could simply stack identity mappings (layer that doesn’t do anything) upon the current network, and the resulting architecture would perform the same. This indicates that the deeper model should not produce a training error higher than its shallower counterparts. They hypothesize that letting the stacked layers fit a residual mapping is easier than letting them directly fit the desired under-laying mapping. And the residual block above explicitly allows it to do precisely that. Figure 3.23 shows the ResNet model architecture.

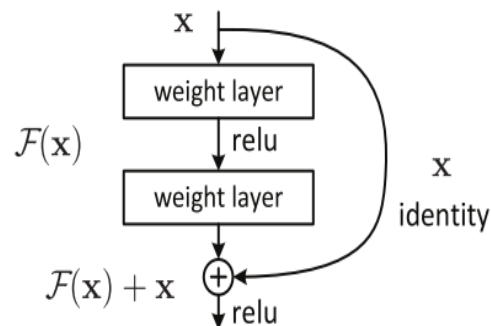


Figure 3.22. A residual block.

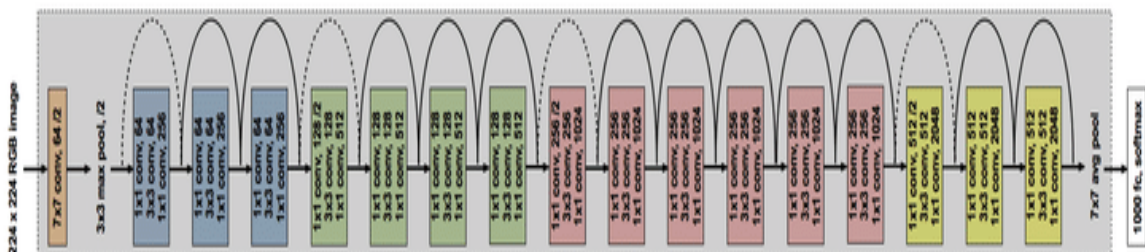


Figure 3.23. ResNet50 model architecture.

3.6.4 Proposed Method Configurations

Our implementation for deep face registration evaluated few models architectures. We started with a simple CNN network with few convolution layers followed by MaxPooling layer and batch normalization (BN) right after each convolution and before activation, and we did not use dropout. In the classification layers, we used two fully-connected layers one with 128 neurons and one with 64 neurons followed with the. We randomly initialize the weights, and we use the Relu activation function. We used Adam optimizer with a mini-batch size of 32. We used the default learning rate which is 0.001, and the model was trained up to 80 epochs. Figure 3.24 shows the model architecture.



Figure 3.24. Proposed method using a simple CNN.

In the second model, we went deeper and used the VGGNet. Ones with 16 layers (VGG16) and the second one with 19 layers (VGG19) as Figure 3.25 shows. We added two fully-connected layers one with 128 neurons and one with 64 neurons followed with the output layers with four neurons at the end to fit our problem. We used Adam optimizer

with a mini-batch size of 32. However, we changed to the learning rate to 0.0001, and the model was trained up to 100 epochs.



Figure 3.25. Proposed method using VGG16 and VGG19.

The last model, we used the ResNet50 model. We added a drop out layer before the fully-connected layers to make the model generalized. We used the same fully-connected and the output layers as the VGGNet model. We used Adam optimizer with a mini-batch size of 32. We used the default learning rate which is 0.001, and the model was trained up to 80 epochs. Figure 3.26 shows the ResNet50 model.

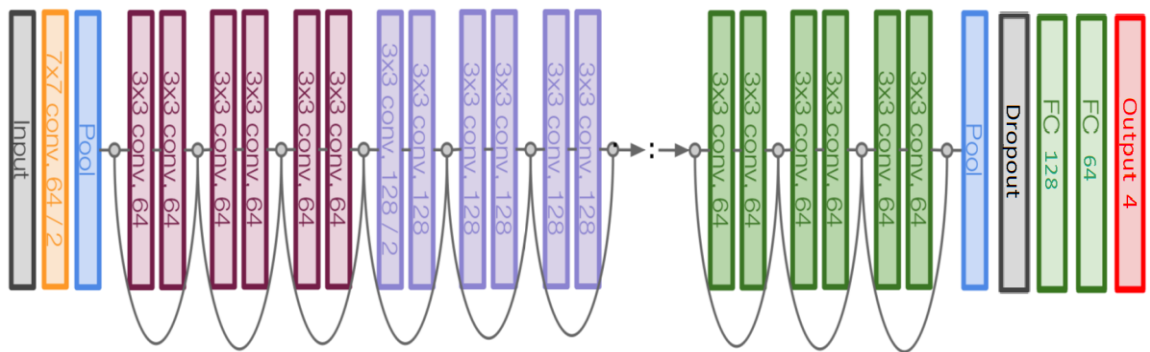


Figure 3.26. Proposed method using ResNet50.

CHAPTER 4: RESULTS

4.1 Classical Face Recognition System Result

4.1.1 Classical Face Recognition Using PCA and KNN Result

In the PCA and KNN experiment, the variant recognition rate is achieved based on different distance methods. Table 4.1 show the results for three scenarios, 50% to 50%, 70% to 30% and 90% to 10%. In the 50% to 50% scenario, Manhattan distance outperforms the other distance methods by mismatching only 5 images with 94.4% accuracy in the YALE dataset and Mahalanobis distance mismatch 9 images and achieved 95.5% accuracy for the ORL dataset. In the second scenario, Mahalanobis and Canberra methods achieved 100% accuracy in the YALE dataset, and Manhattan achieved 95.8% accuracy for the ORL dataset. Finally, we achieved a higher accuracy in the last scenario since we left only one image for testing. The comparison between the three scenarios is shown in Figure 4.1 and Figure 4.2.

Table 4.1. Experiment results using PCA + KNN

Method	Recognition Rate (%)					
	50% 50%		70% 30%		90% 10%	
PCA +KNN Using:	Yale	ORL	Yale	ORL	Yale	ORL
Euclidean Distance	87.8	89	95.6	93.3	100	90
Correlation Distance	88.8	91	97.8	91.7	100	95
Canberra Distance	90	92.5	100	93.3	93.3	97.5
Manhattan Distance	92.2	93.5	95.6	95.8	100	97.5
Mahalanobis Distance	93.3	94	100	95.8	100	97.5

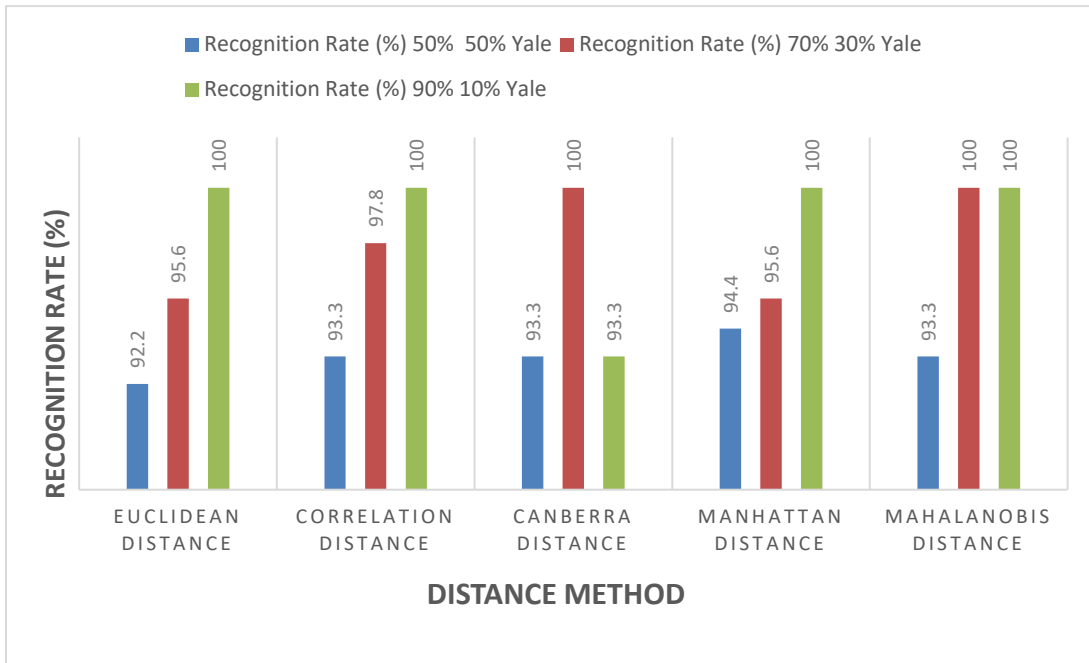


Figure 4.1. Experiment results using PCA + KNN for the Yale data-set.

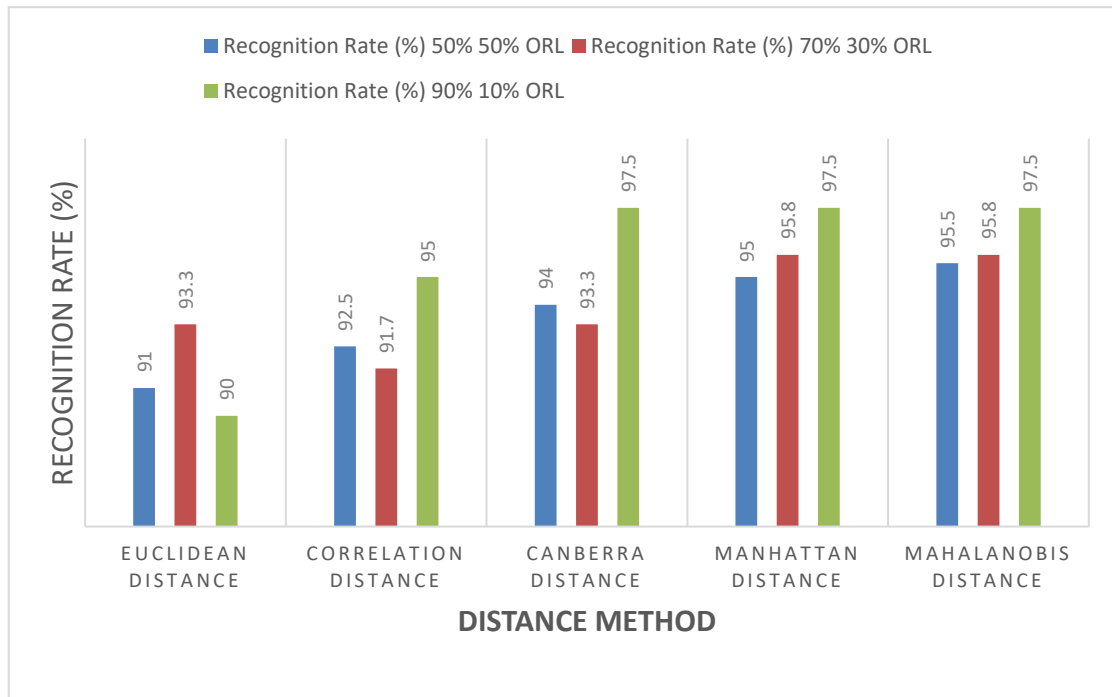


Figure 4.2. Experiment results using PCA + KNN for ORL data-set.

4.1.2 Classical Face Recognition Using LBPH and KNN Result

In the LBPH and KNN experiment, the variant recognition rate is achieved based on different distance methods. Table 4.2 show the results for three scenarios, 50% to 50%, 70% to 30%, and 90% to 10%. In the 50% to %50 scenario, Manhattan distance outperforms the other distance methods by mismatching only 5 images with 94.4% accuracy in the YALE dataset and Mahalanobis distance mismatch 6 images and achieved 97% accuracy for the ORL dataset. In the second scenario, Mahalanobis and Canberra methods achieved 100% accuracy in the YALE dataset and Mahalanobis achieved 97.5% accuracy for the ORL dataset. Finally, we achieved a higher accuracy in the last scenario since we left only one image for testing. The comparison between the three scenarios is shown in Figure 4.3 and Figure 4.4.

Table 4.2. Experiment results using LBPH + KNN.

	Recognition Rate (%)					
Method	50% 50%		70% 30%		90% 10%	
LBP +KNN Using:	Yale	ORL	Yale	ORL	Yale	ORL
Euclidean Distance	88.8	90	95.6	93.3	100	94
Correlation Distance	90	92.5	97.8	94.5	100	95
Canberra Distance	91.1	93.5	100	96	93.3	98
Manhattan Distance	93.3	94	95.6	97.3	100	99
Mahalanobis Distance	94.4	95	100	97.5	100	98

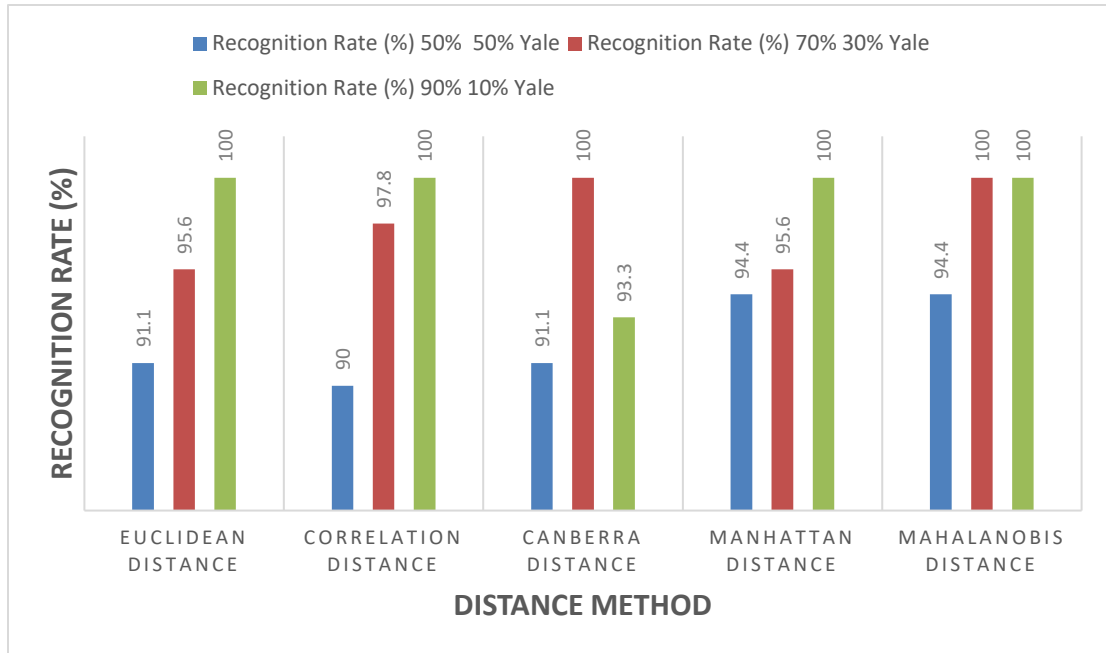


Figure 4.3. Experiment results using LBP-HOG + NN for the Yale data-set.

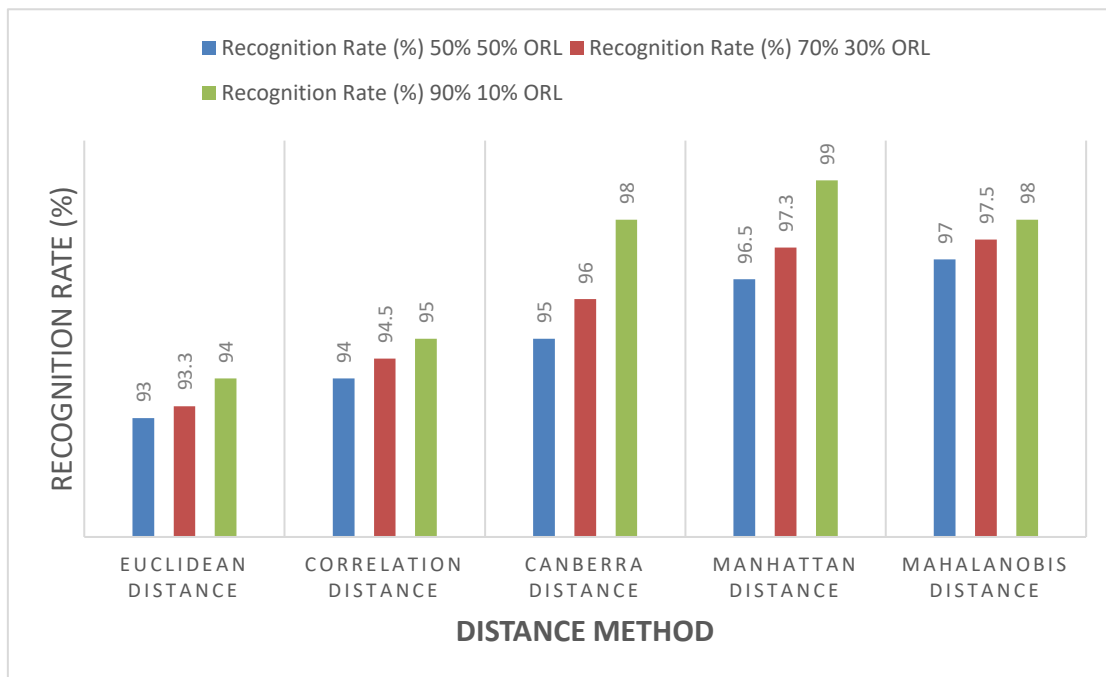


Figure 4.4. Experiment results using LBPH + NN for the ORL data-set.

4.2 Proposed Face Recognition System Result

4.2.1 Proposed Face Recognition Result Using the PCA and NN

In the second experiment, based on the fact which we concluded from the first experiment, combining the distance methods will allow us to achieve a higher recognition rate. In the proposed framework using multi-classifier PCA and NN, we achieved higher accuracy using equation (3.13) then equation (3.14). We achieved 97.7% accuracy in the Yale database with only 2 mismatching of 165 images and 97.5% with 5 mismatching of 200 testing images in 50% training set and 50% testing scenario. Table 4.3 and Figure 4.5 shows the result in details.

Table 4.3. Experiment results using PCA + NN with 50% training set and 50% testing set.

Method PCA + NN using:	Recognition Rate (%)	
	Yale	ORL
Canberra Distance	88.9	80
Euclidean Distance	75.6	79.5
Correlation Distance	89.9	85.5
Manhattan Distance	87.8	89.5
Mahalanobis Distance	94.4	95.5
Proposed: $\sqrt{\sum_{i=1}^5 DIS_i^2}$	96.6	96.5
Proposed: $\sqrt{\alpha \sum_{i=1}^5 DIS_i^2}$	97.7	97.5

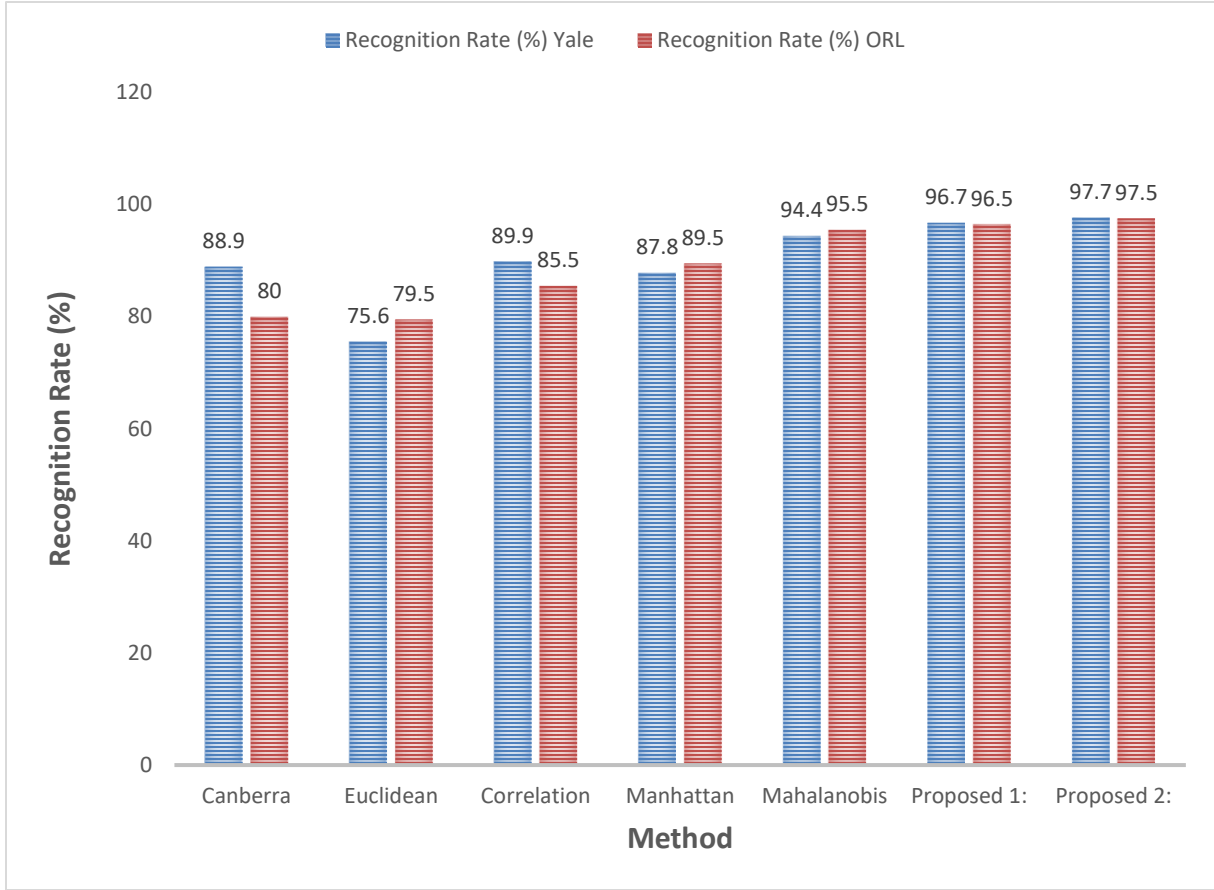


Figure 4.5. Experiment results using PCA + NN with 50% training set and 50% testing set.

4.2.2 Proposed Face Recognition Result Using the LBPH and NN

In the proposed framework using multi-classifier LBPH and NN, we achieved higher accuracy using equation (3.13) then equation (3.14). We achieved 97.7% accuracy in the Yale dataset with only 2 mismatching of 165 images and 98% with 4 mismatching of 200 testing images in 50% training set and 50% testing scenario. Table 4.4 and Figure 4.6 shows the result in details. Table 4.5 shows the comparison between the proposed framework and the other existing methods on the ORL.

Table 4.4. Experiment results using LBPH + NN with 50% training set and 50% testing set.

Method PCA + NN using:	Recognition Rate (%)	
	Yale	ORL
Canberra Distance	91.1	93.5
Euclidean Distance	91.1	95
Correlation Distance	92.2	95.5
Manhattan Distance	94.4	96
Mahalanobis Distance	95.4	96.5
Proposed: $\sqrt{\sum_{i=1}^5 DIS_i^2}$	96.6	97.5
Proposed: $\sqrt{\alpha \sum_{i=1}^5 DIS_i^2}$	97.7	98

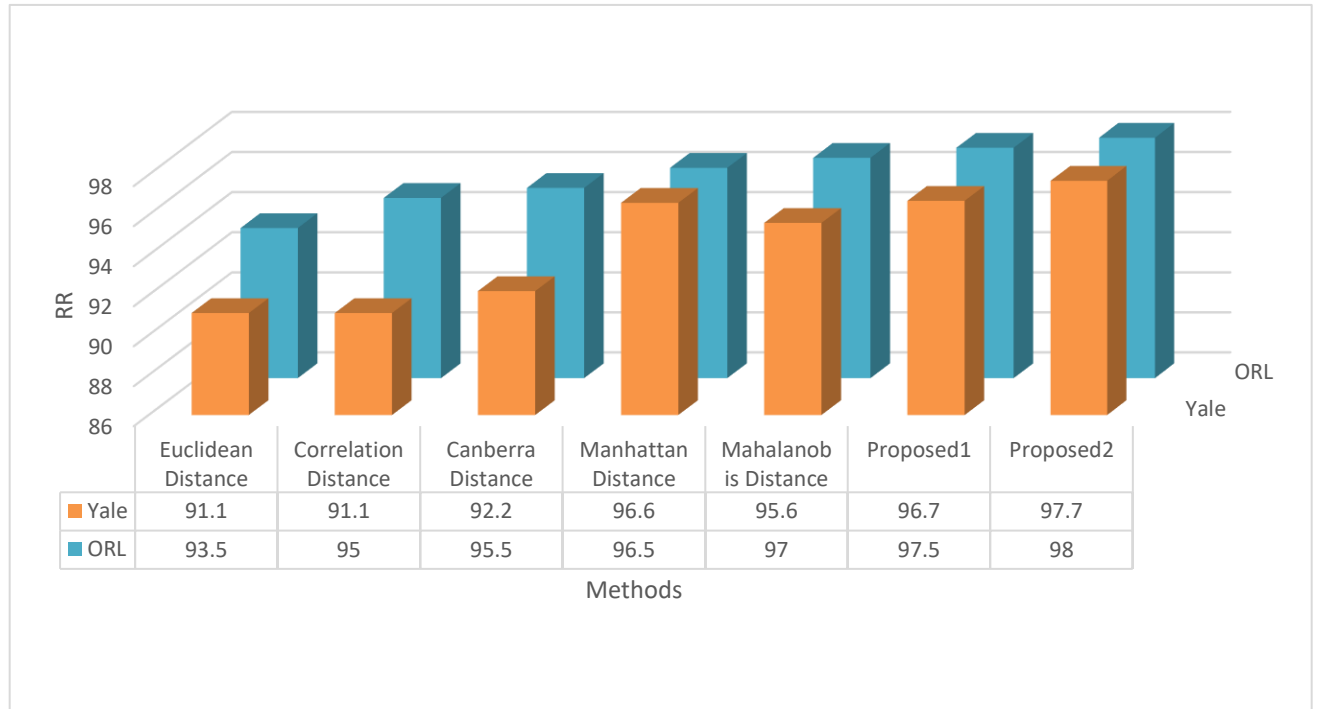


Figure 4.6. Experiment results using LBPH + NN with 50% training set and 50% testing set.

Table 4.5. Comparison between the proposed framework and the other existing methods.

Reference	Year	Method	Number of Training Images per person	%Accuracy
[79]	2015	PCA+BPNN	N/A	88
[80]	2015	LDA	5	89.5
[81]	2015	Gabor + NMF	5	95
[82]	2014	PCA,LDA,DCT,ICA	5	85.5,88.5,91.5,87.5
[83]	2013	FKNN	5	87
[84]	2012	WT+PCA	8	95
[85]	2012	CASNN,FFNN	5	86.5,80
[86]	2011	PCA-DCT-Corr-PIFS	N/A	86.8
Proposed	2017	LBPH, multi-KNN, and BPNN	5	98

4.3 Face Registration Based on a Minimalized Cost Function

Result

In this experiment, we applied our proposed method to a 256x256 pixels Lena image. After converting Lena image to gray-scale image and adding some noise, we applied some transformation to Lena image such as 0.2 scaling, 0.2 rotation, -15 translation tx, and -5 translation ty. We achieved a high registration accuracy with the lowest error by using few matching points. We tried variant matching point scenarios, and we achieved a steady error rate. The main contribution in this paper is registering the images with only two sufficient matching points. Table 4.6 shows the error rate for affine parameters with variant matching point number. This table shows the proposed algorithm achieved the lowest passable error rate with few matching points. The proposed methods can be applied to the human face images as shown in Figure 4.7.

Table 4.6. Registration error rate for variant matching point number.

Number of Matching Points	Scaling Error	Rotation Error	tx Error	ty Error
30	0.007	0.066	0.34	2.21
20	0.01	0.067	0.4	2.5
10	0.012	0.07	0.45	2.43
5	0.015	0.064	1.37	0.9
3	0.017	0.07	1.4	1.7
2	0.019	0.065	0.86	1.09

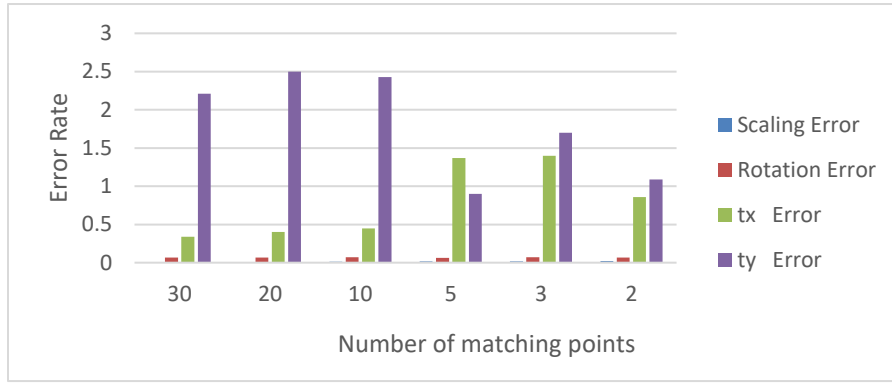


Figure 4.7. Registration error rate for variant matching point number on Lena image.

4.4 Deep Face Registration Result

In this experiment, we applied our proposed method on the LFW dataset. All the images resized to 224x224 and no preprocessing methods applied to them. All images normalized between $[0, 1]$. The dataset randomly split to 67% as a training dataset and 33% as a testing dataset. We evaluated 4 of the deep network models and we achieved a high accuracy. In the simple CNN model, we achieved %98.18 accuracy. However, when we use a deeper model we achieved a higher accuracy. The accuracy using the ResNet50 is

98.55% and 98.42% for VGG19 and VGG16 provided the highest accuracy which is 98.40%. Deeper networks takes more training time which is not critical since the network will be trained off line. The networks predicts the output in a real time miner which is with 0.009 second. Table 4.7 shows the accuracy, training time and the prediction time of the output. Figure 4.8 shows all the models converged smoothly.

Table 4.7. Deep face registration result.

Model	Accuracy	Training time	Prediction time	Total parameters
Simple CNN	98.18	2.6 Hours	0.004 Second	3 million
VGG16	98.42	6.6 Hours	0.009 Second	134 million
VGG19	98.40	7.6 Hours	0.01 Second	140 million
ResNet50	98.55	5.5 Hours	0.007 Second	23 million

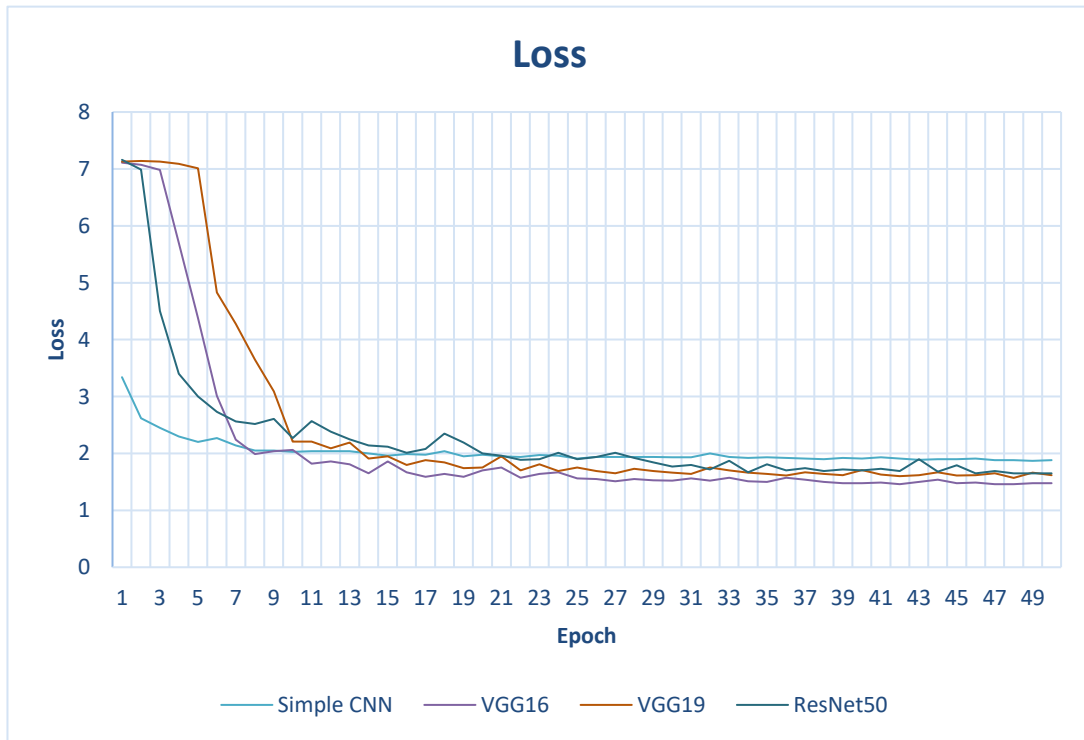


Figure 4.8. Models loss curve.

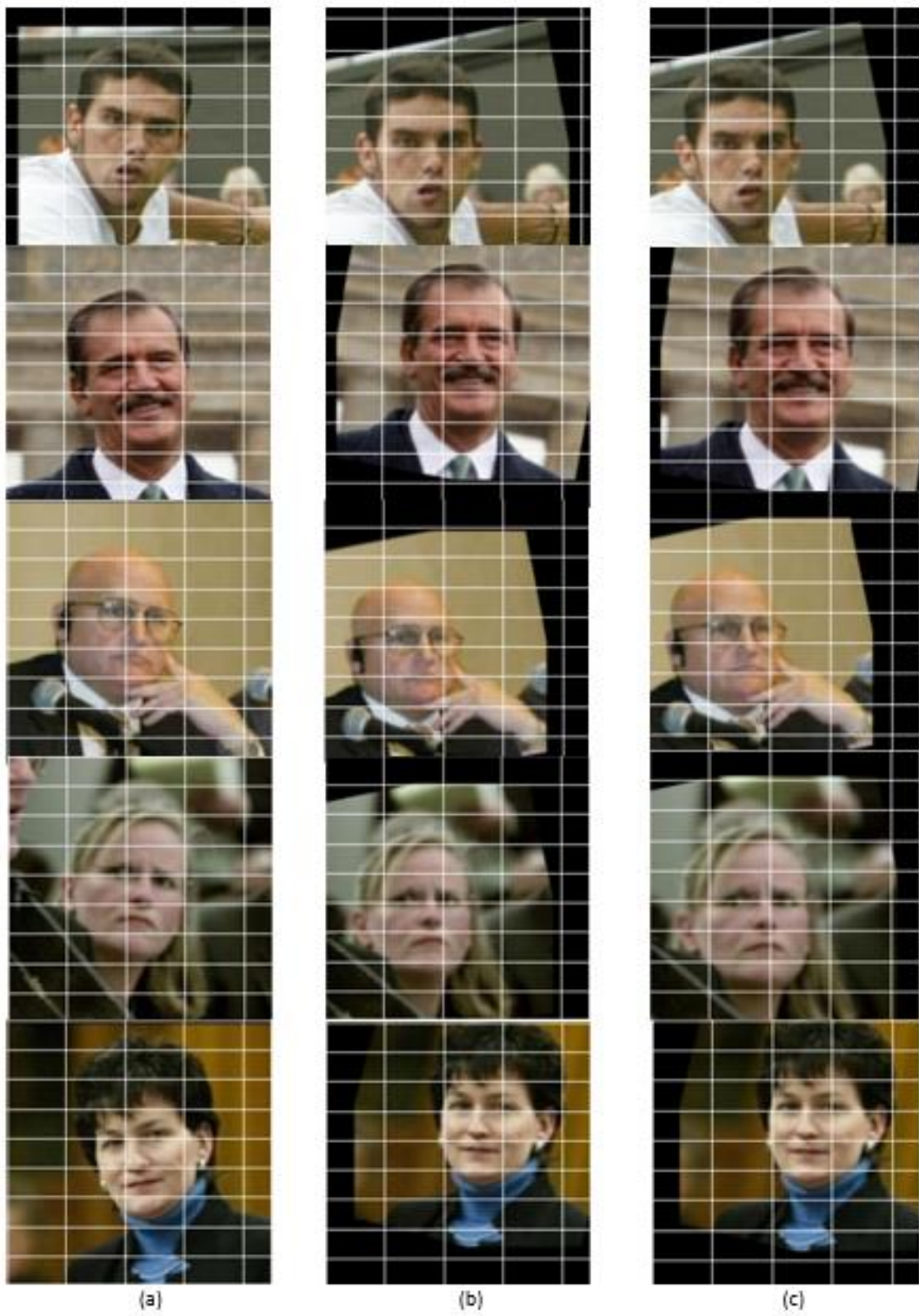


Figure 4.9. Example for registered faces: (a) Original face (b) Predicted registered face (c) Expected registered face.

CONCLUSIONS

In this thesis, we have proposed an improving human face recognition using deep learning based image registration and multi-classifier approaches. In our dissertation, we started with improving the recognition face system by improving the features extraction approach. The framework can work with different types of feature extraction methods. We used the Principal component analysis (PCA) for the first proposed system then the Local binary patterns (LBP) for the second proposed system. The main contribution is to provide training data with distinction patterns which will help the NN to converge faster and more accurate. We achieved this contribution by combining five distance methods since each distance methods has an advantage over the other methods and by combining them we added strength to the whole system. Experimental results showed that we achieved a higher accuracy, and we reduced the computation time. Also, we outperformed the existing frameworks. We can use the proposed framework with a robust feature extraction algorithms such as Support Vector Machine (SVM) and deep neural network which have some advantage over the LPB and PCA. Also, we proposed a deep face registration which can lead to a robust overall face recognition system. We achieved high accuracy using deep models such as VGGNet and ResNet models. Our deep face registration dataset and the deep models will be available for the public after we publish our result.

REFERENCES

- [1] R. Chellappa, C. Wilson, and S. Sirohey, “Human and machine recognition of faces: A survey,” *Proceedings of the IEEE*, vol. 83, pp. 705–740, 1995.
- [2] H. Fang, N. Parthalain, A. J. Aubrey, G. K. L. Tam, R. Borgo, P. L. Rosin, P. W. Grant, D. Marshall, and M. Chen, “Facial expression recognition in dynamic sequences: An integrated approach,” *Pattern Recognition*, pp. 740–748, 2014.
- [3] Z. Zeng, M. Pantic, G. Roisman, and T. Huang, “A survey of affect recognition methods: Audio, visual, and spontaneous expressions,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(1):39–58, 2009.
- [4] M. Chihaoui, A. Elkefi, W. Bellil, and C. B. Amar, “A Survey of 2D Face Recognition Techniques,” *MDPI Computers*, vol. 5, 2016.
- [5] P. Belhumeur, J. Hespanha, and D. Kriegman, “Eigenfaces versus fisherfaces: Recognition using class specific linear projection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, pp. 711–720, 1997.
- [6] T. Ahonen, A. Hadid, and M. Pietikäinen, “Face recognition with local binary patterns,” *European Conference on Computer Vision. Lecture Notes in Computer Science*, Springer, Berlin, vol. 3021, pp. 469–481, 2004.
- [7] T. Chen, X. Zhao, L. Dai, L. Zhang, and J. A. Wang, “Novel Texture Feature Description Method Based on the Generalized Gabor Direction Pattern and Weighted Discrepancy Measurement Model,” *MDPI Symmetry*, vol. 8, 109, 2016.

- [8] Y. Xiao, J. Wu, J. Yuan, "CENTRIST: A multi-channel feature generation mechanism for scene categorization," *IEEE Transaction on Image Processing*, vol. 23, no. 2, pp. 823–836, 2014.
- [9] C. K. Heng, S. Yokomitsu, Y. Matsumoto and H. Tamura, "Shrink boost for selecting multi-LBP histogram features in object detection," *IEEE Conference on Computer Vision and Pattern Recognition*, Providence, RI, pp. 3250-3257, 2012.
- [10] X. Zhao and S. Zhang, "Facial Expression Recognition Based on Local Binary Patterns and Kernel Discriminant Isomap," *Sensors*, vol. 11, no. 10, pp. 9573-9588, 2011.
- [11] L. Huang, C. Chen, W. Li, and Q. Du, "Remote sensing image scene classification using multi-scale completed local binary patterns and Fisher vectors", *Remote Sens.*, vol. 8, no. 6, pp. 483, 2016.
- [12] M. Turk, and M. Pentland, "Eigenfaces for Recognition," *J. Cogn. Neuroscience*, vol. 3, pp. 71–6, 1991.
- [13] S. Wang, and P. Liu, "A New Feature Extraction Method Based on the Information Fusion of Entropy Matrix and Covariance Matrix and Its Application in Face Recognition," *MDPI Entropy*, vol. 17, pp. 4664-4683, 2015.
- [14] H. Boughrara, M. Chtourou, C. Amar, and L. Chen, "Face recognition based on perceived facial images and multilayer perceptron neural network using constructive training algorithm," *IET Computer Visio*, vol. 8, no. 6, pp. 729-739, 2014.
- [15] M. J. Er, W. Chen, and S. Wu, "High speed face recognition based on discrete cosine transform and RBF neural network," *IEEE Transaction on Neural Network*, vol. 16, no.3, pp. 679-691, 2005.

- [16] Y. Xu, F. Liang, G. Zhang, and H. Xu, "Image Intelligent Detection Based on the Gabor Wavelet and the Neural Network," *MDPI Symmetry*, vol. 8, 130, 2016.
- [17] D. Hemanth, C. Kezi, and J. Anitha, "Application of Neuro-Fuzzy Model for MR Brain Tumor Image classification," *International Journal of Biomedical Soft Computing and Human Sciences*, vol. 16, no., 2010.
- [18] Z. Yang and F. S. Cohen, "Image registration and object recognition using affine invariants and convex hulls," *IEEE Transaction on Image Processing*, vol. 8, no. 7, pp. 934-946, July 1999.
- [19] F. Dufaux and J. Konrad, "Efficient, robust, and fast global motion estimation for video coding," *IEEE Transaction on Image Processing*, vol. 9, no. 3, pp. 497-501, March 2000.
- [20] J. B. A. Maintz and M. A. Viergever, "A survey of medical image registration," *Medical Image Analysis*, vol. 2, no. 1, pp. 1-36, 1998.
- [21] H. Dou, and Y. Lu, "Medical Image Registration based on Edge Inflection point," *Life Science Instrument*, vol. 10, no. 4, pp. 15-18, 2006.
- [22] R. Hartley and A. Zisserman, "Multiple view geometry in computer vision" *Cambridge university press*, 2003.
- [23] D. G. Lowe, "Distinctive image features from scale invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.
- [24] B. Zitova, and J. Flusser, "Image registration methods: a survey," *Image and Vision Computing*, vol. pp. 977-1000, 2003.

- [25] N. Vujovic, and D. Brzakovic, "Establishing the correspondence between control points in pairs of mammographic image," *IEEE Transactions on Image Processing* vol. 6, pp. 1388–1399, 1997.
- [26] H. Alhichri, and M. Kamel, "Virtual Circles: a new set of features for fast image registration," *Pattern Recognition Letters*, vol. 24, pp. 1181-1190, 2003.
- [27] B. Manjunath, C. Shekhar, and R. Chellapa, "A new approach to image feature detection with applications," *Pattern Recognition*, vol. 29, pp. 627–640, 1996.
- [28] S. Wisetphanichkij, and K. Dejhan, "Fast Fourier Transform Technique and Affine Transform Estimation-Based High Precision Image Registration Method," *GESTS Int'l Transition on Computer Science and Engineering*, vol. 20, no. 1, pp. 179-191, 2005.
- [29] R. Karani, and T. Sarode, "Image Registration using Discrete Cosine Transform and Normalized Cross Correlation," *IJCA Proceedings on International Conference and Workshop on Emerging Trends in Technology*, pp. 28-34, 2012.
- [30] E. Paul and A. Beegom, "Mining images for image annotation using SURF detection technique," *International Conference on Control Communication & Computing*, pp. 724-728, 2015.
- [31] W. Chen S. Ding, and Z. Chai, "FPGA-Based Parallel Implementation of SURF Algorithm," *IEEE International Conference on Parallel and Distributed Systems*, pp. 308-315, 2016.
- [32] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speedup-Up Robust Features(SURF)", *IEEE transactions*, vol. 14, no. 1, 2008.

- [33] M. Calonder, V. Lepetit, C. Strecha, and P. F. Brief, “Binary robust independent elementary features,” *European Conference on Computer Vision*, pp. 778–792, 2010.
- [34] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.
- [35] P. J. Rousseeuw, “Least median of squares regression,” *Journal of the American Statistical Association*, vol. 79, pp. 871–880, December 1984.
- [36] M. H. Yang, “Kernel Eigenfaces vs. Kernel Fisherfaces: Face recognition using kernel methods,” *Automatic Face and Gesture Recognition*, pp. 205–211, 2002.
- [37] C. Liu and H. Wechsler, “Evolutionary pursuit and its application to face recognition,” *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 22, no. 6, pp. 570–582, 2000.
- [38] D. Zhang, Z. Zhou, and S. Chen, “Diagonal Principal Component Analysis for Face Recognition,” *Pattern Recognition*, vol. 39, pp. 140–142, 2006.
- [39] T. Mandal, Q. M. J. Wu, and Y. Yuan, “Curvelet based face recognition via dimension reduction,” *Elsevier Signal Processing*, vol. 89, no. 3, pp. 2345–2353, 2009.
- [40] M. H. Yang, “Kernel Eigenfaces vs. Kernel Fisherfaces: Face Recognition Using Kernel Methods,” *Automatic Face and Gesture Recognition*, pp. 215–220, May 2002.
- [41] R. Gottumukkal and V. K. Ansri, “An improved face recognition technique based on Modular PCA approach,” *Pattern Recognition letters*, vol. 25, pp. 429–436, 2004.
- [42] J. Yang, D. Zhang, A. F. Frangi, and J. Y. Yang, “Two-dimensional PCA: a new approach to appearance-based face representation and recognition,” *IEEE*

- Transaction on Pattern Analysis and Machine Intelligence*, vol. 26, no. 1, pp. 131-137, Jan. 2004.
- [43] J. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos, "Face Recognition using LDA based Algorithms," *IEEE Transaction on Neural Networks*, vol. 14, no. 1, pp 195-200, 2003.
 - [44] A. Martez, and A. Kak, "PCA versus LDA," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 23, no. 2, pp. 228-233, 2001.
 - [45] M. S. Bartlett, J. R. Movellan, and T. J. Sejnowski, "Face Recognition by Independent Component Analysis," *IEEE Transaction on Neural Networks*, vol. 13, no. 6, pp. 1450-1464, 2002.
 - [46] B. A. Draper, K. Baek, M. S. Bartlett, and J. R. Beveridge, "Recognizing Faces with PCA and ICA," *Computer Vision and Image Understanding: special issue on face recognition*, pp. 115-137, 2003.
 - [47] B. Moghaddam, "Principal manifolds and bayesian subspaces for visual recognition," *IEEE International Conference on Computer Vision*, pp. 1131-1136, 1999.
 - [48] P. C. Yuen and J. H. Lai, "Face representation using Independent Component Analysis," *Pattern Recognition*, vol. 35, no. 6, pp. 1247-1257, 2002.
 - [49] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study of texture measures with classification based on feature distributions," *Pattern Recognition*, vol. 29, no. 1, pp. 51–59, 1996.
 - [50] W. Zhang, S. Shan, W. Gao, and H. Zhang. "Local Gabor binary pattern histogram sequence (LGBPHS): a novel non-statistical model for face representation and recognition," *IEEE ICCV*, vol. 1, pp. 786–791, 2005.

- [51] H. E. Komleh, V. Chandran, and S. Sridharan, "Robustness to expression variations in fractal-based face recognition," *Proc. IEEE Int. Symp. Signal Processing and its Applications*, vol. 1, Kuala Lumpur, Malaysia, pp. 359–362, 2001.
- [52] M. Lades, J. C. Vorbriiggen, J. Buhmann, J. Lange, C. von der Malsburg, R. P. Wiirtz, and W. Konen, "Distortion invariant object recognition in the dynamic link architecture," *IEEE Transaction on Computers*, vol. 42, no. 3, pp. 300-311, 1993.
- [53] T. Cover, and P. Hart, "Nearest Neighbor Pattern Classification," *IEEE Transaction on Information Theory*, Vol. 13, no. 1, pp. 21–27. 1967.
- [54] K. Lee, Y. Chung, and H. Byun, "SVM based face verification with feature set of small size," *Electronic letters*, vol. 38, no.15, pp. 787-789, 2002.
- [55] Y. LeCun, L. Bottou, G. B. Orr, and K. R. Muller, "Efficient BackProp," *Neural Networks: Tricks of the Trade*, pp. 9-50, 1998.
- [56] I. Chakroun, T. Haber, and T. J. Ashby, "SW-SGD: The Sliding Window Stochastic Gradient Descent Algorithm," *Procedia Computer Science*, vol. 108, pp. 2318-2322, 2017.
- [57] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *In Proc. International Conference on Learning Representations*, 2014.
- [58] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," *Association for Computational Linguistics*, pp. 655-665, 2014.
- Y. Kim, "Convolutional neural networks for sentence classification," *Empirical Methods in Natural Language Processing*, 2014.

- [59] A. Conneau, H. Schwenk, Y. LeCun, and L. Barrault, “Very deep convolutional networks for text classification,” *European Chapter of the Association for Computational Linguistics*, 2017.
- [60] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248-255, 2009.
- [61] V. Sze, Y. H. Chen, T. J. Yang, and J. S. Emer, “Efficient processing of deep neural networks: A tutorial and survey,” in *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295-2329, 2017.
- [62] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, pp. 436-444, 2015.
- [63] Y. Bengio, “Learning Deep Architectures for AI,” *Foundations and Trends® in Machine Learning*, vol. 2, no. 1, pp. 1-127, 2009.
- [64] L. Deng, “Three classes of deep learning architectures and their applications: a tutorial survey,” *APSIPA Transactions on Signal and Information Processing*, 2012.
- [65] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929-1958, 2014.
- [66] Y. Liu, E. Racah, P. J. Correa, A. Khosrowshahi, D. Lavers, K. Kunkel, M. Wehner, and W. D. Collins “Application of Deep Convolutional Neural Networks for Detecting Extreme Weather in Climate Datasets,” *arXiv preprint arXiv:1605.01156*, 2016.

- [67] A. Esteva, B. Kuprel, R.A. Novoa, Ko J., S.M. Swetter, H.M. Blau, S. Thrun, “Dermatologist-level classification of skin cancer with deep neural networks,” *Nature*, vol. 542, no. 7639, pp. 115-118, 2017.
- [68] T. Chen, R. Xu, Y. He, and X. Wang, “A gloss composition and context clustering based distributed word sense representation model,” *Entropy*, vol. 17, no. 9, pp. 6007-6024, 2015.
- [69] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ‘Imagenet classification with deep convolutional neural networks,” *Advances neural information processing systems*, pp. 1090-1098, 2012.
- [70] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” *International Conference on Machine Learning*, pp. 807-814, 2010.
- [71] A. Aldhaferi and J. Lee, “Event detection on large social media using temporal analysis,” *Computing and Communication Workshop and Conference*, pp. 1-6, 2017.
- [72] F. Schilling, “The Effect of Batch Normalization on Deep Convolutional Neural Networks,” *DiVA Publisher: Uppsala, Sweden*, 2016.
- [73] ORL dataset. Available online: (accessed on 12/11/2018)
<http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html>.
- [74] Yale Face Dataset. Available online: (accessed on 12/11/2018).
<http://vision.ucsd.edu/content/yale-face-database>.
- [75] M. Kirby and L. Sirvoich, “Application of the Karhunen-Loeve Procedure for the characterization of human faces,” *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 12, no. 1, pp. 103–108, 1990.

- [76] K. Perumal, and R. Bhaskaran, "Supervised classification Performance of multispectral images," *Journal of computing*, vol. 2, no. 2, pp. 124-129, 2010.
- [77] G. J. Szekély, M. L. Rizzo, and N. K. Bakirov, "Measuring and testing independence by correlation of distances," *The Annals of Statistics*, vol. 35, no. 6, pp. 2769-2794, 2007.
- [78] M. P. Rajath, R. Keerthi, and K. M. Aishwarya, "Artificial neural networks for face recognition using PCA and BPNN," *TENCON 2015 - 2015 IEEE Region 10 Conference*, pp. 1-6, 2015.
- [79] W. Ge, W. Quan, and C. Han, "Face description and identification using histogram sequence of local binary pattern". *International Conference on Advanced Computational Intelligence*, pp. 415-420, 2015.
- [80] F. Purnomo, D. Suhartono, M. Shodiq, A. Susanto, S. Raharja, and R. W. Kurniawan, "Face recognition using Gabor Wavelet and Non-negative Matrix Factorization," *SAI Intelligent Systems Conference (IntelliSys)*, pp. 788-792, 2015.
- [81] F. A. Bhat, and M. A. Wani, "Performance Comparison of Major Classical Face Recognition Techniques," *International Conference on Machine Learning and Applications*, pp. 521-528, 2015.
- [82] T. Ayyavoo, and J. S. Jayasudha, "Face recognition using enhanced energy of Discrete Wavelet Transform," *International Conference on Control Communication and Computing*, pp. 415-419, 2013.
- [83] C. I. Fan, X. T. Chen, and N. D. Jin, "Research of face recognition based on wavelet transform and principal component analysis," *International Conference on Natural Computation*, pp. 575-578, 2012.

- [84] A. J. Dhanaseely, S. Himavathi, and E. Srinivasan, "Performance comparison of cascade and feed forward neural network for face recognition system," *International Conference on Software Engineering and Mobile Application Modelling and Development*, pp. 1-6, 2012.
- [85] M. A. Lone, S. M. Zakariya, and R. Ali, "Automatic Face Recognition System by Combining Four Individual Algorithms," *International Conference on Computational Intelligence and Communication Networks*, pp. 222-226, 2011.
- [86] P. Gadde, and X. Yu, "Image Registration with Artificial Neural Networks Using Spatial and Frequency Features," *IEEE International Joint Conference on Neural Networks*, pp. 4643-4649, 2016.

APPENDIX

Simple CNN model:

Layer (type)	Output Shape	Param #
input_1 (Input Layer)	(None, 100, 100, 3)	0
conv2d_1 (Conv2D)	(None, 100, 100, 128)	3584
batch_normalization_1 (Batch Normalization)	(None, 100, 100, 128)	512
max_pooling2d_1 (MaxPooling2D)	(None, 50, 50, 128)	0
conv2d_2 (Conv2D)	(None, 50, 50, 128)	147584
batch_normalization_2 (Batch Normalization)	(None, 50, 50, 128)	512
max_pooling2d_2 (MaxPooling2D)	(None, 25, 25, 128)	0
conv2d_3 (Conv2D)	(None, 25, 25, 128)	147584
batch_normalization_3 (Batch Normalization)	(None, 25, 25, 128)	512
max_pooling2d_3 (MaxPooling2D)	(None, 13, 13, 128)	0
conv2d_4 (Conv2D)	(None, 13, 13, 128)	147584
batch_normalization_4 (Batch Normalization)	(None, 13, 13, 128)	512
flatten_1 (Flatten)	(None, 21632)	0
dense_1 (Dense)	(None, 128)	2769024
dense_2 (Dense)	(None, 64)	8256
dense_3 (Dense)	(None, 4)	260
Total params: 3,225,924		
Trainable params: 3,224,900		
Non-trainable params: 1,024		

VGG16 Model layers:

Layer (type)	Output Shape	Param #
input_4 (Input Layer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
Flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 4096)	102764544
fc2 (Dense)	(None, 4096)	16781312
fc3 (Dense)	(None, 128)	524416
fc4 (Dense)	(None, 64)	8256
Predictions (Dense)	(None, 4)	260
Total params: 134,793,476		
Trainable params: 134,793,476		
Non-trainable params: 0		

VGG19 Model layers:

Layer (type)	Output Shape	Param #
input_5 (Input Layer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv4 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv4 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv4 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
Flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 4096)	102764544
fc2 (Dense)	(None, 4096)	16781312
fc3 (Dense)	(None, 128)	524416
fc4 (Dense)	(None, 64)	8256
Predictions (Dense)	(None, 4)	260
Total params: 140,103,172		
Trainable params: 140,103,172		
Non-trainable params: 0		

ResNet50 Model layers:

Layer (type)	Output Shape	Param #
input_3 (InputLayer)	(None, 224, 224, 3)	0
conv1_pad (ZeroPadding2D)	(None, 230, 230, 3)	0
conv1 (Conv2D)	(None, 112, 112, 64)	9472
bn_conv1 (BatchNormalization)	(None, 112, 112, 64)	256
activation_99 (Activation)	(None, 112, 112, 64)	0
max_pooling2d_3 (MaxPooling2D)	(None, 55, 55, 64)	0
res2a_branch2a (Conv2D)	(None, 55, 55, 64)	4160
bn2a_branch2a (BatchNormalization)	(None, 55, 55, 64)	256
activation_100 (Activation)	(None, 55, 55, 64)	0
res2a_branch2b (Conv2D)	(None, 55, 55, 64)	36928
bn2a_branch2b (BatchNormalization)	(None, 55, 55, 64)	256
activation_101 (Activation)	(None, 55, 55, 64)	0
res2a_branch2c (Conv2D)	(None, 55, 55, 256)	16640
res2a_branch1 (Conv2D)	(None, 55, 55, 256)	16640
bn2a_branch2c (BatchNormalization)	(None, 55, 55, 256)	1024
bn2a_branch1 (BatchNormalization)	(None, 55, 55, 256)	1024
add_33 (Add)	(None, 55, 55, 256)	0
activation_102 (Activation)	(None, 55, 55, 256)	0
res2b_branch2a (Conv2D)	(None, 55, 55, 64)	16448
bn2b_branch2a (BatchNormalization)	(None, 55, 55, 64)	256
activation_103 (Activation)	(None, 55, 55, 64)	0
res2b_branch2b (Conv2D)	(None, 55, 55, 64)	36928
bn2b_branch2b (BatchNormalization)	(None, 55, 55, 64)	256
activation_104 (Activation)	(None, 55, 55, 64)	0
res2b_branch2c (Conv2D)	(None, 55, 55, 256)	16640
bn2b_branch2c (BatchNormalization)	(None, 55, 55, 256)	1024
add_34 (Add)	(None, 55, 55, 256)	0
activation_105 (Activation)	(None, 55, 55, 256)	0

res2c_branch2a	(Conv2D)	(None, 55, 55, 64)	16448
bn2c_branch2a	(BatchNormalizati	(None, 55, 55, 64)	256
activation_106	(Activation)	(None, 55, 55, 64)	0
res2c_branch2b	(Conv2D)	(None, 55, 55, 64)	36928
bn2c_branch2b	(BatchNormalizati	(None, 55, 55, 64)	256
activation_107	(Activation)	(None, 55, 55, 64)	0
res2c_branch2c	(Conv2D)	(None, 55, 55, 256)	16640
bn2c_branch2c	(BatchNormalizati	(None, 55, 55, 256)	1024
add_35	(Add)	(None, 55, 55, 256)	0
activation_108	(Activation)	(None, 55, 55, 256)	0
res3a_branch2a	(Conv2D)	(None, 28, 28, 128)	32896
bn3a_branch2a	(BatchNormalizati	(None, 28, 28, 128)	512
activation_109	(Activation)	(None, 28, 28, 128)	0
res3a_branch2b	(Conv2D)	(None, 28, 28, 128)	147584
bn3a_branch2b	(BatchNormalizati	(None, 28, 28, 128)	512
activation_110	(Activation)	(None, 28, 28, 128)	0
res3a_branch2c	(Conv2D)	(None, 28, 28, 512)	66048
res3a_branch1	(Conv2D)	(None, 28, 28, 512)	131584
bn3a_branch2c	(BatchNormalizati	(None, 28, 28, 512)	2048
bn3a_branch1	(BatchNormalizatio	(None, 28, 28, 512)	2048
add_36	(Add)	(None, 28, 28, 512)	0
activation_111	(Activation)	(None, 28, 28, 512)	0
res3b_branch2a	(Conv2D)	(None, 28, 28, 128)	65664
bn3b_branch2a	(BatchNormalizati	(None, 28, 28, 128)	512
activation_112	(Activation)	(None, 28, 28, 128)	0
res3b_branch2b	(Conv2D)	(None, 28, 28, 128)	147584
bn3b_branch2b	(BatchNormalizati	(None, 28, 28, 128)	512
activation_113	(Activation)	(None, 28, 28, 128)	0
res3b_branch2c	(Conv2D)	(None, 28, 28, 512)	66048
bn3b_branch2c	(BatchNormalizati	(None, 28, 28, 512)	2048
add_37	(Add)	(None, 28, 28, 512)	0

activation_114	(Activation)	(None, 28, 28, 512)	0
res3c_branch2a	(Conv2D)	(None, 28, 28, 128)	65664
bn3c_branch2a	(BatchNormalizati	(None, 28, 28, 128)	512
activation_115	(Activation)	(None, 28, 28, 128)	0
res3c_branch2b	(Conv2D)	(None, 28, 28, 128)	147584
bn3c_branch2b	(BatchNormalizati	(None, 28, 28, 128)	512
activation_116	(Activation)	(None, 28, 28, 128)	0
res3c_branch2c	(Conv2D)	(None, 28, 28, 512)	66048
bn3c_branch2c	(BatchNormalizati	(None, 28, 28, 512)	2048
add_38	(Add)	(None, 28, 28, 512)	0
activation_117	(Activation)	(None, 28, 28, 512)	0
res3d_branch2a	(Conv2D)	(None, 28, 28, 128)	65664
bn3d_branch2a	(BatchNormalizati	(None, 28, 28, 128)	512
activation_118	(Activation)	(None, 28, 28, 128)	0
res3d_branch2b	(Conv2D)	(None, 28, 28, 128)	147584
bn3d_branch2b	(BatchNormalizati	(None, 28, 28, 128)	512
activation_119	(Activation)	(None, 28, 28, 128)	0
res3d_branch2c	(Conv2D)	(None, 28, 28, 512)	66048
bn3d_branch2c	(BatchNormalizati	(None, 28, 28, 512)	2048
add_39	(Add)	(None, 28, 28, 512)	0
activation_120	(Activation)	(None, 28, 28, 512)	0
res4a_branch2a	(Conv2D)	(None, 14, 14, 256)	131328
bn4a_branch2a	(BatchNormalizati	(None, 14, 14, 256)	1024
activation_121	(Activation)	(None, 14, 14, 256)	0
res4a_branch2b	(Conv2D)	(None, 14, 14, 256)	590080
bn4a_branch2b	(BatchNormalizati	(None, 14, 14, 256)	1024
activation_122	(Activation)	(None, 14, 14, 256)	0
res4a_branch2c	(Conv2D)	(None, 14, 14, 1024)	263168
res4a_branch1	(Conv2D)	(None, 14, 14, 1024)	525312
bn4a_branch2c	(BatchNormalizati	(None, 14, 14, 1024)	4096

bn4a_branch1	(BatchNormalizatio	(None, 14, 14, 1024)	4096
add_40	(Add)	(None, 14, 14, 1024)	0
activation_123	(Activation)	(None, 14, 14, 1024)	0
res4b_branch2a	(Conv2D)	(None, 14, 14, 256)	262400
bn4b_branch2a	(BatchNormalizati	(None, 14, 14, 256)	1024
activation_124	(Activation)	(None, 14, 14, 256)	0
res4b_branch2b	(Conv2D)	(None, 14, 14, 256)	590080
bn4b_branch2b	(BatchNormalizati	(None, 14, 14, 256)	1024
activation_125	(Activation)	(None, 14, 14, 256)	0
res4b_branch2c	(Conv2D)	(None, 14, 14, 1024)	263168
bn4b_branch2c	(BatchNormalizati	(None, 14, 14, 1024)	4096
add_41	(Add)	(None, 14, 14, 1024)	0
activation_126	(Activation)	(None, 14, 14, 1024)	0
res4c_branch2a	(Conv2D)	(None, 14, 14, 256)	262400
bn4c_branch2a	(BatchNormalizati	(None, 14, 14, 256)	1024
activation_127	(Activation)	(None, 14, 14, 256)	0
res4c_branch2b	(Conv2D)	(None, 14, 14, 256)	590080
bn4c_branch2b	(BatchNormalizati	(None, 14, 14, 256)	1024
activation_128	(Activation)	(None, 14, 14, 256)	0
res4c_branch2c	(Conv2D)	(None, 14, 14, 1024)	263168
bn4c_branch2c	(BatchNormalizati	(None, 14, 14, 1024)	4096
add_42	(Add)	(None, 14, 14, 1024)	0
activation_129	(Activation)	(None, 14, 14, 1024)	0
res4d_branch2a	(Conv2D)	(None, 14, 14, 256)	262400
bn4d_branch2a	(BatchNormalizati	(None, 14, 14, 256)	1024
activation_130	(Activation)	(None, 14, 14, 256)	0
res4d_branch2b	(Conv2D)	(None, 14, 14, 256)	590080
bn4d_branch2b	(BatchNormalizati	(None, 14, 14, 256)	1024
activation_131	(Activation)	(None, 14, 14, 256)	0
res4d_branch2c	(Conv2D)	(None, 14, 14, 1024)	263168
bn4d_branch2c	(BatchNormalizati	(None, 14, 14, 1024)	4096

add_43 (Add)	(None, 14, 14, 1024)	0
activation_132 (Activation)	(None, 14, 14, 1024)	0
res4e_branch2a (Conv2D)	(None, 14, 14, 256)	262400
bn4e_branch2a (BatchNormalizati	(None, 14, 14, 256)	1024
activation_133 (Activation)	(None, 14, 14, 256)	0
res4e_branch2b (Conv2D)	(None, 14, 14, 256)	590080
bn4e_branch2b (BatchNormalizati	(None, 14, 14, 256)	1024
activation_134 (Activation)	(None, 14, 14, 256)	0
res4e_branch2c (Conv2D)	(None, 14, 14, 1024)	263168
bn4e_branch2c (BatchNormalizati	(None, 14, 14, 1024)	4096
add_44 (Add)	(None, 14, 14, 1024)	0
activation_135 (Activation)	(None, 14, 14, 1024)	0
res4f_branch2a (Conv2D)	(None, 14, 14, 256)	262400
bn4f_branch2a (BatchNormalizati	(None, 14, 14, 256)	1024
activation_136 (Activation)	(None, 14, 14, 256)	0
res4f_branch2b (Conv2D)	(None, 14, 14, 256)	590080
bn4f_branch2b (BatchNormalizati	(None, 14, 14, 256)	1024
activation_137 (Activation)	(None, 14, 14, 256)	0
bn4f_branch2c (BatchNormalizati	(None, 14, 14, 1024)	4096
add_45 (Add)	(None, 14, 14, 1024)	0
activation_138 (Activation)	(None, 14, 14, 1024)	0
res5a_branch2a (Conv2D)	(None, 7, 7, 512)	524800
bn5a_branch2a (BatchNormalizati	(None, 7, 7, 512)	2048
activation_139 (Activation)	(None, 7, 7, 512)	0
res5a_branch2b (Conv2D)	(None, 7, 7, 512)	2359808
bn5a_branch2b (BatchNormalizati	(None, 7, 7, 512)	2048
activation_140 (Activation)	(None, 7, 7, 512)	0
res5a_branch2c (Conv2D)	(None, 7, 7, 2048)	1050624
res5a_branch1 (Conv2D)	(None, 7, 7, 2048)	2099200
bn5a_branch2c (BatchNormalizati	(None, 7, 7, 2048)	8192

bn5a_branch1 (BatchNormalizatio	(None, 7, 7, 2048)	8192
add_46 (Add)	(None, 7, 7, 2048)	0
activation_141 (Activation)	(None, 7, 7, 2048)	0
res5b_branch2a (Conv2D)	(None, 7, 7, 512)	1049088
bn5b_branch2a (BatchNormalizati	(None, 7, 7, 512)	2048
activation_142 (Activation)	(None, 7, 7, 512)	0
res5b_branch2b (Conv2D)	(None, 7, 7, 512)	2359808
bn5b_branch2b (BatchNormalizati	(None, 7, 7, 512)	2048
activation_143 (Activation)	(None, 7, 7, 512)	0
res5b_branch2c (Conv2D)	(None, 7, 7, 2048)	1050624
bn5b_branch2c (BatchNormalizati	(None, 7, 7, 2048)	8192
add_47 (Add)	(None, 7, 7, 2048)	0
activation_144 (Activation)	(None, 7, 7, 2048)	0
res5c_branch2a (Conv2D)	(None, 7, 7, 512)	1049088
bn5c_branch2a (BatchNormalizati	(None, 7, 7, 512)	2048
activation_145 (Activation)	(None, 7, 7, 512)	0
res5c_branch2b (Conv2D)	(None, 7, 7, 512)	2359808
bn5c_branch2b (BatchNormalizati	(None, 7, 7, 512)	2048
activation_146 (Activation)	(None, 7, 7, 512)	0
res5c_branch2c (Conv2D)	(None, 7, 7, 2048)	1050624
bn5c_branch2c (BatchNormalizati	(None, 7, 7, 2048)	8192
add_48 (Add)	(None, 7, 7, 2048)	0
activation_147 (Activation)	(None, 7, 7, 2048)	0
avg_pool (GlobalAveragePooling2	(None, 2048)	0
dropout_3 (Dropout)	(None, 2048)	0
fc1 (Dense)	(None, 128)	262272
fc2 (Dense)	(None, 64)	8256
fc3 (Dense)	(None, 4)	260
=====		
Total params: 23,858,500		
Trainable params: 23,805,380		
Non-trainable params: 53,120		