2013

Application of Neural Networks with CSD Coefficients for Human Face Recognition

Ayesa Parvin

Application of Neural Networks with CSD Coefficients for Human Face Recognition

by

Ayesa Parvin

A Thesis
Submitted to the Faculty of Graduate Studies
through Department of Electrical and Computer Engineering
in Partial Fulfillment of the Requirements for
the Degree of Master of Applied Science at the
University of Windsor

Windsor, Ontario, Canada

Application of Neural Networks with CSD Coefficients for Human Face Recognition

by

Ayesa Parvin

APPROVED BY:

_____

Dr. David Ting, External Reader
Department of Mechanical, Automotive, and Materials Engineering

_____

Dr. Huapeng Wu, Departmental Reader
Department of Electrical and Computer Engineering

_____

Dr. Roberto Muscedere, Co-Advisor
Department of Electrical and Computer Engineering

_____

Dr. Majid Ahmadi, Co-Advisor
Department of Electrical and Computer Engineering

_____

Dr. Rashid Rashidzadeh, Chair of Defense

December 20, 2012

DECLARATION OF ORIGINALITY

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

Face recognition is one of the most popular, reliable and widely used applications in real world. It is the main biometric used by humans in many security, law enforcement and commercial systems and high demand of this application attracts researchers from various fields such as image processing, pattern recognition, neural network and computer vision etc. In a Human Face Recognition Systems, we start with pre-processing of the data followed by feature extraction for dimensionality reduction and then classification. In this thesis, neural network classifier with CSD coefficients is used to make the area required for implementation of recognition system more efficient. The FPGA implementation of the proposed technique indicates almost 50% saving in the area required for face recognition application by using neural network classifier with CSD coefficients while the processing speed is improved in comparison to its binary counterpart. Extensive experimental results were conducted to show the utility of the proposed technique.

DEDICATION

I would like to dedicate this thesis to my parents, grand-parents and Shifat.

ACKNOWLEDGEMENTS

First of all, I want to give thank to the Almighty Allah for helping me through the research work. Next, I am very thankful my honourable and knowledgeable advisors, Dr. Majid Ahmadi and Dr. Roberto Muscedere for their constant support, guidance and constructive feedback. I would like to give thanks my honourable committee members for their thoughtful modifications and inputs regarding the thesis work during my first and second seminars. I want to thank departmental stuff Andria Ballo for supporting me in many administrative ideas. Finally I want to thank my parents, grandfather and Shifat for their constant support and help.

TABLE OF CONTENTS

IV. RESULTS AND DISCUSSION

V. CONCLUSIONS AND FUTURE WORKS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AF	Activation Function
ANN	Artificial Neural Networks
ASIC	Application Specific Integrated Chip
CSD	Canonical Signed Digit
DSP	Digital Signal Processor
FFNN	Feed Forward Neural Networks
FPGA	Field Programmable Gate Array
LDA	Linear Discriminant Analysis
LM	Lavenberg-Marquardt
LUT	Look Up Table
PCA	Principle Component Analysis
RBF	Radial Basis Function
ROM	Read Only Memory

CHAPTER I

INTRODUCTION

Human face recognition is one of the active area of research and it is a preliminary step to a wide range of practical applications which includes image processing, pattern recognition, personal identity verification, video-surveillance, facial expression extraction, gender classification, advanced human and computer interaction, computer vision [54]. Most of these methods are using Artificial Neural Networks (ANN) classifiers approaches coupled with appropriate feature extraction for recognition of human faces. ANN can learn from real time examples and it has the ability to adapt with the changes in environment. It is widely used classifier for its fault tolerance and robustness to noise which have opened the application of ANN in various fields of engineering, science, economics, etc [4]-[6].

1.1 Motivation

Nowadays, face recognition is a hottest topic in the area of pattern recognition and image processing since 1990 due to its vast demand in many real life applications and availability of feasible technologies. All practical applications of face recognition require being very small in size with high-speed recognition for controlling and accessing authorized systems. Therefore it is necessary to use specialized hardware system for face recognition in order to meet real time performance. ANN is a widely used and well- known classifier for human face recognition for its robustness and good learning capability. It can be implemented in two ways: software implementation and hardware implementation. The processing speed for face recognition in software system is very slow which makes it quite impractical to implement the recognition system in software. Moreover, the face database contains very large amount of data which

requires large memory space in software system. Therefore, to realize the full benefits of face recognition application with neural network classifier, it is important to implement the system in dedicated hardware system such as FPGA implementation which offers high space management with good speed and recognition rate for this real time application by taking the full advantages of inherent parallelism of ANN architecture [2]. Thus hardware based ANN for face recognition application offers some levels of flexibility over software based system because it requires less area with high- speed recognition rate in hardware platform than comparing to software system.

1.2 Proposed Work

This thesis presents specialized hardware implementation of human face recognition by using ANN classifier to perform face recognition from large face database. This dedicated hardware system requires very low-area which is more suitable for large face database in practical system. The proposed architectures presented in this thesis can easily be applied to other applications.

The preprocessed face images are applied to feature extractor for dimensionality reduction and to get the important features for ANN classifier for recognition purpose. In hardware platform, the area and speed are two important parameters and efficient design of ANN is mostly depends on how to efficiently design a single neuron where the weights of the neuron are saved. When feature vector of an face image are applied to the network, the feature elements and corresponding synaptic weights are multiplied to get the weighted inputs and the summation of all weighted inputs are added with bias before passing through the Activation Function (AF) to get the final output of a neuron. AF is used to limit or squash the amplitude of a signal to some finite value. The computation process in hardware platform is usually done by using binary

system. When binary coefficients are saved in the network, it consumes more area and the binary multiplication process to get the weighted inputs consumes most of the processing time in a single neuron which is a major problem of hardware based ANN. Therefore if it is possible to reduce the size of single neurons with much not compromising the computation time for multiplication process of neurons, then the size of whole neural network system will be small and compact in comparing to binary network. As a result the network can also accommodate more information about large face database.

In the proposed method, Canonical Signed Digit (CSD) coefficient is used instead of binary coefficient in hardware platform to get weighted input by multiplying CSD coefficient with corresponding binary input of neurons in ANN system. The advantage of CSD coefficient is that the CSD weight contains smaller number of nonzero elements among all signed digit number system and the number of maximum possible nonzero elements can never be greater than the half of total number of bits in a CSD system. Thus the CSD coefficients requires less area in a network and CSD multiplier also produces less partial products in comparison to binary multiplier which means less area requirement with almost same level of accuracy and speed in hardware based recognition system. Look up table (LUT) based Tangent-Sigmoid (TanSig) is used to realize the activation function. After implementing the whole NN system, feature vectors from two popular feature extraction method for face images were applied separately to test the performance of the hardware implemented ANN system.

1.3 Thesis Organization

The rest of the thesis is organized as follows:

Chapter II contains a brief description on face recognition, its importance in today's world and application in real world, necessary steps and algorithms to complete the procedure of face recognition for example PCA feature extraction, ANN for classification etc. Chapter III explains the software and hardware implementation procedure of ANN classifier which is used to recognize face images. The results found from these two types of implementation and their comparison are describes in Chapter IV and the last chapter contains the conclusion and future work.

CHAPTER II

BACKGROUND STUDY AND REVIEW OF LITERATURE

2.1 Face Recognition

An automatic face recognition system is used to recognize or identify a person automatically in still or video images of a scene from a stored database for security, authentication etc. It is one of most popular topic in the area of image processing and pattern recognition since 1990 due to its increasing importance and application in many commercial opportunities and security systems. Moreover, availability of feasible technologies and high demand of this reliable method for law enforcement applications draw the attention of many researchers from various fields such as pattern recognition, image processing, neural networks, computer vision etc. to conduct research in this area.

But face recognition system is unique among other recognition systems such as fingerprint and iris recognition system, where the sensor needs to touch the object directly or laser beams require scanning a person's eye directly which requires a great deal of human participation. However, human face recognition is a non-invasive biometric method for identification and recognition of a face image from the faces in a stored database of several images [54]. This allows the processing for recognition without a person's possible awareness.

There are some standard test databases for testing the performance of face recognition system. There are many universities and institutions which have their own face databases; for example, the Yale face database, AT&T "The database of faces" (also formerly known as "The ORL database of faces"), FERET database, SCface - Surveillance Cameras face database, MIT-CBCL face recognition database, NIST mugshot identification database, The AR face database

(of Purdue University) etc [7], [8]. Each database has its own characteristics, so it is necessary to select appropriate one based on the application at hand. These databases deal with a set of data containing, different illumination, facial expression, aging, occlusion, etc. In this research, ORL face database [9] were selected for its some distinct advantages for example well organized database, small size of the images, mono color etc. This database contains 10 images for each person and all of these 10 images are identical to each other. There are total 40 classes. A dark homogeneous background with an upright, frontal position was used for all these images and they vary in rotation, expression, gray scale, position etc. Some of the images were taken at different times with variation in lightings, facial details with facial expressions (like with or without glasses, smiling or without smiling, closed or opened eyes) etc. The size of each image is 92x112 and the gray level per pixel varies from 0 to 255. Some sample images from this database are shown in Figure 2.1.

Figure 2.1 Samples of facial images from ORL database

2.2 An Automatic Face Recognition System

There are some major steps need to be considered for performong an automatic face recognition sytem namely preprocessing, feature extraction and classification. These steps are shown by using a block diagram in figure 2.2. At first, the face images are collected together which is known as face databese and then the next step is preprocessing step which allows to enhance the image quality because the images may taken at different situations. Images may be degraded with noise and poor illumination. So it is necessary remove the noise and normalize the color of images. Among many other methods, histogram equalization method is a common method to enhance the histogram of pixel intensities of images [48], thus the image quality improves. The third and fourth steps are used to reduce the dimension and extract important features from face images and save those features for classificatio purpose. The last step consists of classification method which allows to recognize an unknown face image depending on the extracted features of the database in previous step.

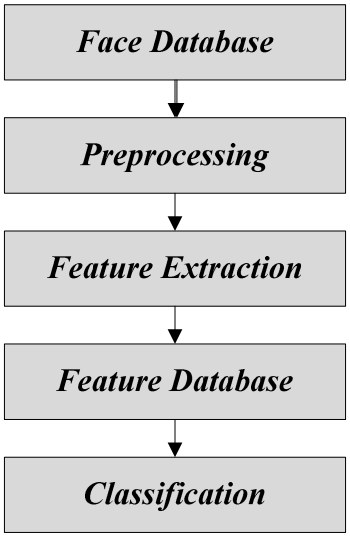

Figure 2.2 Block diagram for NN based face recognition

2.3 Feature Extraction Method

Face recognition is a very high-dimensional and complex pattern recognition problem. Therefore it is necessary to reduce the dimension and extract the necessary features of face images before applying it to a classifier so that the recognition rate improves. Principle Component Analysis (PCA) [23]-[27] and Linear Discriminant Analysis (LDA) [20] are two of the widely used method for feature extraction and they are useful for large face databases [19], [21], [22] .

2.3.1 Principle Component Analysis (PCA)

Principle Component Analysis (PCA) is an effective feature extraction method based on Eigenfaces [24], [26], [27] . It's function is to extract less quantity of feature without missing most important information and reduce the dimention of original face pattern which spans over high dimension [25], [28]. Principal components or eigenfaces are known as a small set of characteristic feature information of face images and these features are built from the variance between training samples [24]. For M number of images with pixel size of $(Nx * Ny)$, the pixel elements of each images of training set, $\gamma = [\gamma_1\ \gamma_2\ \ldots\ \gamma_M]$ are alphabetically ordered. Then the mean, δ is taken for the vectors as shown in equation (2.1) and subtracted from γ_i to get the mean subtracted images, Φ .

$$\delta = \frac{1}{M_t}\sum_{i=1}^{M_t}\gamma_i \tag{2.1}$$

$$\phi_i = \gamma_i - \delta \tag{2.2}$$

Then the covariance matrix, C is computed for the mean subtracted images:

$$C = \frac{1}{M_t}\sum_{i=1}^{M_t}\phi_i\ \phi_i^t \tag{2.3}$$

Here C is a high dimensional matrix for M images. Next eigenvectors of covariance matrix is calculated from by solving (2.4) where λ is the eigenvalue.

$$Cv = v\lambda \tag{2.4}$$

Eigenvectors are a set of nonzero orthonormal vectors and when it operates on another vector function, generates a scalar multiple result known as eigenvalues which describes best about the distribution of that vector function [24]. Therefore, principal component, p can be obtained by multiplying eigenvectors, v with matrix, C:

$$p_i = C \,.\, v_i \tag{2.5}$$

Now PCA projection, p is used for both training and testing samples projection which results corresponding set of weights, ω_k for each images as in (2.6). The arrangement of M set of sample weights are expressed by Ω in (2.7).

$$\omega_k = p_k^T . \phi = p_k^T \,.(\gamma - \delta) \tag{2.6}$$

$$\Omega = [\omega_1 \ \omega_2 \ \dots \ \omega_M] \tag{2.7}$$

2.3.2 Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) is another popular and widely used feature extraction method [19], [21]. PCA has a lack of discrimination ability and it may hold some unwanted features from face images because of considering all variations across training samples for example lighting variation, facial expressions [31] etc. PCA extracts features which are important to represent a class [30], [31] and it might perform better for small number of samples per class [30]. LDA extracts the most effective features for class separability [21] and for more number of samples per class; LDA outperforms PCA [30]. But due to high computational cost of LDA [32], it is not efficient to use LDA as feature extractor for high dimensional image. Therefore, to get effective result, both PCA and LDA is used where PCA is used for dimension

reduction and LDA is used to extract features for class seperability from the reduced dimensional images found from PCA [21].

LDA is applied on the set of feature vectors found from principle component projections of training samples in (2.7) which is used to find another subspace for second projection [22]. Therefore LDA requires two training samples to calculate scatter matrices. If i^{th} class has q_i training samples and M is the total number of classes, then mean image per class, μ_i and total mean, μ_o can be calculated by:

$$\mu_i = \frac{1}{q_i}\sum_{k=1}^{q_t} \Omega_k \tag{2.8}$$

$$\mu_o = \frac{1}{M_t}\sum_{k=1}^{M_t} \Omega_k \tag{2.9}$$

Above two equations are used to calculate within-class scatter matrix, S_w and between-class scatter matrix, S_b where $P(C_i)$ is the prior class probability:

$$S_w = \sum_{i=1}^{c} P(C_i)\,(\Omega - \mu_i).(\Omega - \mu_i)^T \tag{2.10}$$

$$S_b = \sum_{i=1}^{c} P(C_i)\,(\mu_i - \mu_o).(\mu_i - \mu_o)^T \tag{2.11}$$

The advantage of applying LDA over PCA features is that it diminishes the complexity of singular problem of S_w in (2.10) by creating another subspace to optimal project the data based on Fisher Linear Discriminant criterion [31] as expressed in (2.12).

$$W = \frac{|W^T.S_b.W|}{|W^T.S_w.W|} \tag{2.12}$$

$$W_{opt} = [W_1.W_2 \ldots W_{M'}] \tag{2.13}$$

In (2.13), W_{opt} is the final set of eigenvectors of $S_b * S_w^{-1}$ matrix for M largest eigenvalues.

2.4 Classification

When the feature extraction method is applied and feature vectors for training and testing images are ready, then these vectors are sent to a classifier for recognization. There are different types of classifier used such as *k*-Nearest Neighbors, Support Vector Machines (SVM) and Artificial Neural Networks (ANN) etc.

k-Nearest Neighbors is a simple algorithm used as a classifier where usually Euclidian distance is used to calculate the distance between a test sample and training samples and the object is assigned to the class which most frequently responds among the *k* nearest training samples. It provides good performance for the optimal values of *k* [50].

A Support Vector Machine (SVM) finds the hyperplane in the possible feature space and tries to maximize the distance between hyperplane and data points by constructing two parallel hyperplanes. The classification accuracy of a test sample increases with the larger distance between these two parallel hyperplanes [29].

Artificial Neural Networks (ANN) is one of the most popular and widely used classifier to face recognition and other pattern recognition problem. Its massive architecture, potential to fault tolerance and learning capability make it widely acceptable among other classifiers. In [50], it has shown that ANN performs better than *k*-Nearest Neighbors. Moreover, ANN outer performs SVM for larger training set size [51]. Following section contains the brief description about ANN.

2.4.1 Artificial Neural Networks (ANN)

The biological operation of neural system has long been facinated by humans [1]. So it is highly desirable to understand the operation performed by this biological neural system in order to

realize the closely related or same features and apply these features in real life practical applications. Many research effort has been put to derive ideas from biological paradigms in order to make efficient intelligent systems. The main objective of modern neural network research is to realize ANN from understanding different aspects of the biological counterpart. An ANN is a massive parallel distributed structure with a large number of nodes and interconnections to store experimental knowledge through learning process and make it available to use. So there are two key properties for information processing capabilities which make ANN to solve complex face recognition and other pattern recognition problem by using its computational power. The first one is the ANN potential to fault tolerance due to its massive parallel interconnected architecture so that ANN continues to work even when a neuron ot its connecting links is damaged or disconnected and secondly, the learning capability from real life examples and generalize it where generalization refers to provide reasonable outputs for inputs while testing even in degraded conditions [10].

Supervised learining is one of the popular learning paradigm. Here input-output mapping is done which involves the modification of synaptic weights of NN by applying a whole set of training examples in batch mode. Each example contains a unique set of inputs and desired response which is represented to the network and the synaptic weights of NN are modified by learning in an iterative manner to reduce the difference between desired and actual output. The network reaches to a steady position when the inputs of whole training examples are presented to the network and it is repeated for many times therefore no significant modification is required in the synaptic weights after certain iterations [10]. The learning procedure of ANN is time consuming. It requires many training samples for learning to get the network's outputs similar to

desired responses. But after the network finishes learning, it requires small time for testing a sample.

ANN is used as target classifier for face recognition system in this thesis because it provides greater degree of fault tolerance or robustness and it has the capability to adapt with the changes in image data. Depending on the implementation type of ANN, it can be divided in to three categories: analog implementation, digital implementation and hybrid implementation. Analog implementation is difficult to design, but it offers higher density and efficiency in circuit level, area, power consumption and its resolution is better and requires fewer components than digital implementation. However, analog implementation suffers from inaccurate results due to offsets and mismatch. It is more sensitive to noise and it gains errors due to fabrication difficulties. Another main problem of analog implementation is storage of weights which requires initialization and periodic refreshment. But most of the real world applications of ANN are embedded in digital system. Digital implementation is more perfect for the complexity of larger ANN due to less sensitivity in noise and fabrication difficulties. Moreover, digital implementation offers more flexibility over analog counterparts in terms of fabrication technology, simulation because for any given time, digital system is always more desirable than analog technology [52]. However, digital implementation suffers from slow computation and requirement of large area because of the requirement of multiplication and accumulation processes to get weighted inputs. Hybrid implementation takes the advantages of both analog and digital system where most of the circuitry performs analog computation to gain more speed and power efficiency but the synaptic weights are stored by using digital circuit for long term storage in digital registers and better noise immunity. However, the objective of this thesis is to reduce

the area requirement of digital neural network as a classifier for face recognition application by using CSD coefficients which will be described later.

2.4.2 Model of a Neuron

ANN has a fundamental and basic information processing unit which is known as neuron. The operation of NN mainly depends on the functionality of neuron. A neuron model may contain three basic elements, namely a set of synapses or connecting links, an adder and activation function [10]. The first element is also known as strength or weight of a neuron. Adder is used for summing the weighted inputs. Here weighted inputs refer to the inputs which are weighted or multiplied by the corresponding synaptic weights. The structure of each neuron in the network includes a nonlinear function at the output end. The relation between input and output could be reduced if there is an absence of nonlinearities [10] and an activation function (AF) is used to express this nonlinear properties. An AF limits or squashes the amplitude of the output of a neuron to some finite value. A typical normalized range for the output of a neuron can be varied from 0 to1 or -1 to 1 depending on the functionality of AF. Figure 2.3 represents the basic components for the nonlinear model of a single neuron.

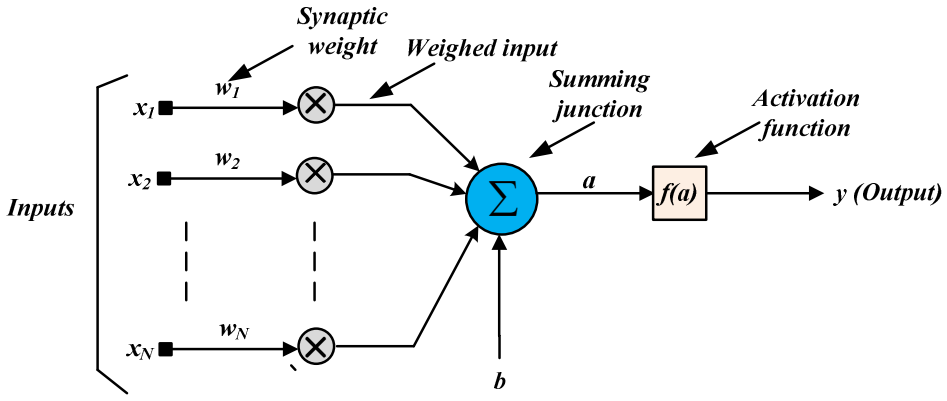

Figure 2.3 Basic components of an artificial neuron

Mathematically the input-output relationship of a neuron can be described by the following equation (2.14) and (2.15) where $X = x_1, x_2, \dots, x_N$ are the inputs to the neuron and $W = w_1, w_2, \dots, w_N$ are the corresponding synaptic weights of the neuron.

$$a = (\sum_{j=1}^{N} w_j x_j) + b \tag{2.14}$$

$$y = f(a) \tag{2.15}$$

The bias, b is added with the summed result of weighted inputs as shown in (2.14) and passed through activation function to get the final output of the neuron which is denoted by *y* in (2.15). The vector notation for (2.14) can be expressed by:

$$a = W^T x + b \tag{2.16}$$

One of the most important parts of a neuron is the activation function. It has the capability of fault tolerance and it is used to force the final output of neuron according of activity level at the input of that neuron [10]. This nonlinear excitation function can approximate almost any complex function. Sigmoid, hyperbolic tangent activation functions are the most commonly nonlinear activation functions [10], [11] and they are mathematically expressed by (2.17) and (2.18) respectively and graphically shown in figure 2.4. For this thesis the hyperbolic tangent activation function is used both for all layers in NN for better accuracy [12].

$$Sigmoid(a) = \frac{1}{1+e^{-a}} \tag{2.17}$$

$$Tanh(a) = \frac{e^{a}-e^{-a}}{e^{a}+e^{-a}} \tag{2.18}$$

The hyperbolic tangent AF is differentiable and it ranges from -1 to +1which means an asymmetric form can be assumed by this function with respect to the origin. So this activation function has some analytical benefits as it has the opportunity to assume both positive and negative values [10] which makes the ANN learning faster. The nonlinear characteristic of hyperbolic tangent AF makes the learning of ANN more powerful.

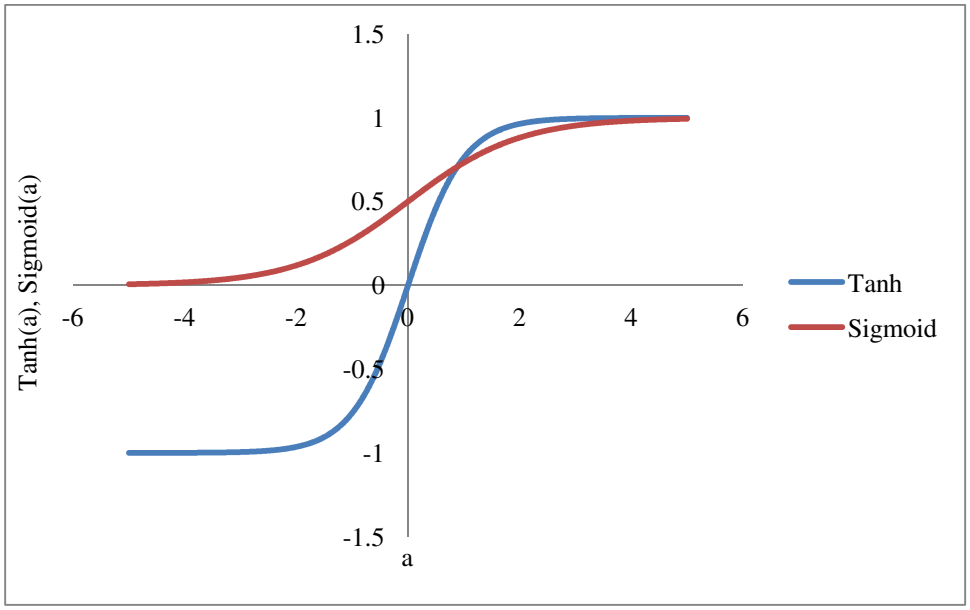

Figure 2.4 Hyperbolic Tangent and Sigmoid Activation Function

2.4.2.1 CSD Coefficient System and Multiplier

Recalling figure 2.3, in each neuron, inputs are multiplied with the corresponding synaptic weights and these weighted inputs are added with bias before passing through AF. Different types of multiplier are used in literature for input and coefficient multiplication for example binary multiplier, booth multiplier etc. In this thesis, Canonical Signed Digit is used as coefficient for multiplication.

Canonical Signed Digit (CSD) number system is a special type of redundant representation of radix-2 Signed Digit (SD) number system with ternary coefficient set {-1, 0, 1} to minimize the number of nonzero digits [33], [37], [57]-[60] . A two's complement number can be converted in to canonic vector by using Reitwiesner's algorithm [56] where it has shown that the CSD representation for a number is unique (canonic) if the binary expansion of that number is padded with an initial zero. The conversion algorithm works from LSB to MSB means from right to left. Table 1 represents Reitwiesner's right to left conversion algorithm for 2's complement number to CSD number.

b_{i+1}	b_i	c_i	y_i	c_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	0	0
1	0	1	$\bar{1}$	1
1	1	0	$\bar{1}$	1
1	1	1	0	1

Table 1: Binary to CSD conversion table based on Reitwiesner's right to left conversion algorithm

In table 1, for any two's complement number, b, the resulting CSD representation is y. It is assumed that the initial auxiliary carry variable, $c_0 = 0$ and for other steps, c_i is the carry generated in previous step $i - 1$ and c_{i+1} is the carry out at step i. For example, if $b = 478_{10} = 0111011110_{2,}$ then after using table 1, converted CSD number, $y = 1000\bar{1}000\bar{1}0$. From this example, we can see that binary number, b contains seven nonzero elements but after converting

in to canonical form, the number contains three nonzero elements which mean nonzero elements are reduced by four. Moreover, if we consider the original Booth algorithm [34], the Booth's representation for b is $100\bar{1}100\bar{1}0$ which means the number of nonzero elements is four. So, from the above example, it is obvious that, CSD representation of almost any number contains fewest numbers of nonzero elements among all signed digit representations [35], [59].

CSD representations have some unique properties which are given below:

a) Canonic representation is a non-redundant system which and there cannot be any adjacent nonzero elements ($y_i * y_{i-1} = 0$), which implies that there can be maximum of [(y+1)/2] nonzero elements for any y bit CSD number [57].
b) Each number has a unique CSD representation.
c) For a y bit CSD number, the minimum number of nonzero digits is $y/3+1/9+O(2^{-y})$ which means on average, a CSD representation of a two's complement number consist of almost 33% fewer non-zero elements than its binary counterpart [39], [58].

A neural network may have few thousands of weights. As canonic representation contains minimum possible nonzero elements, thus neural network requires less space to save the weights in hardware system in comparing to binary representation. Moreover, CSD coefficient multiplier reduces the number of partial products hence allowing minimum number of addition, subtraction and shift operations to generate the product in a hardware system [38]. It has shown that for any n bit multiplication with CSD coefficient, the total number of addition, subtraction and shift operations never exceed by n/2 [36]. There are different types of CSD multiplier is designed in literature such as constant coefficient CSD multiplier [59], [63], low error fixed

width multiplier [61], [62] etc. In constant coefficient multiplier, the coefficient is constant which provides the opportunity to design a system with low space and high speed. In fixed width multiplier, a 2n bit product found from n bit multiplication is truncated to n bit product by eliminating (n-1) least significant bits [61], [62]. Usually a NN system has large number of weights. Therefore, it requires designing large number of constant coefficient multiplier for the system. Moreover, the number of inputs of each image depends on the size of face database and the size of face database is always changing in real time system for improving the recognition rate which means the number of feature elements of each image also changes. As a result, the network's weights also changes. Therefore, constant coefficient multiplier is not considered for the design of NN system. In fixed width array multiplier, an error compensation method is required to use for reducing the truncation error produced by the multiplier. But in NN, the AF has the greater degree of fault tolerance hence low error fixed width multiplier is not selected for the design of NN. Therefore, a CSD multiplier which performs shift and addition or subtraction operation is used in this project.

2.4.3 Network Architecture

The architecture of ANN depends on the learning algorithm because the structure of a neuron in the network is closely related to the learning algorithm used for training the network. Feed forward neural networks (FFNN), radial basis function (RBF) networks, recurrent neural networks are mostly used network architectures. Feedforward neural networks can be devided in to two sub catagories: single-layer feedforward networks and multilayer feedforward neural networks [10].

Single-layer feedforward networks can not map curve like problem due to its limited mapping ability and it can learn linearly separable patterns only. So it can not solve nonlinear problems where the inputs are not linearly seperable [13]. But multilayer feedforward neural network has single or more hidden layer with hidden neurons which helps NN to get an extra set of synaptic connections and adds extra dimention to the network. Multilayer feedforward neural network gives NN the ability to extract higher order statistics which is important when the input layer's size is large [10]. So, multiplayer feedforward neural network is used in this thesis so that it will be helpful to classify large and complex face data. Figure 2.5 illustrates the architecture of fully connected feedforward neural network with single hidden layer and output layer. The input feature vector, $X = [x_1\ x_2\ \ldots\ x_N]$ propagates through the network on a basis of forward direction from input layer to hidden layer(s) and next to output layer. The final output, $Y = [y_1\ y_2\ \ldots\ y_K]$ is produced by the neurons in output layer.

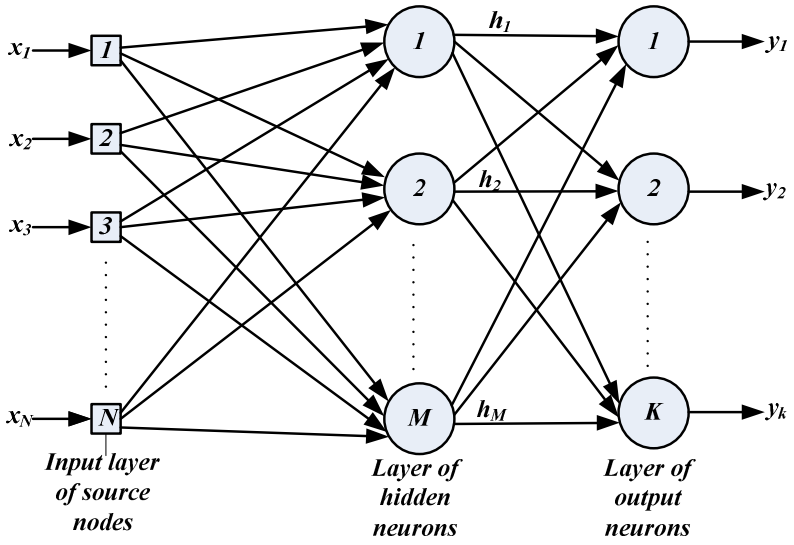

Figure 2.5 Fully connected feedforward neural network with single hidden layer and output layer

2.4.4 Training of Neural Networks

ANN has the ability to learn from surrounding environment. Thus the network can improve the performance through learning which develops over time by following some prescribed measure. Basically a network starts gaining more knowleadge about its own environment after each iteration of the learning process by adjusting the synaptic weights and biases. Figure 2.6 illustrates the a taxonomy of the learning process [10].

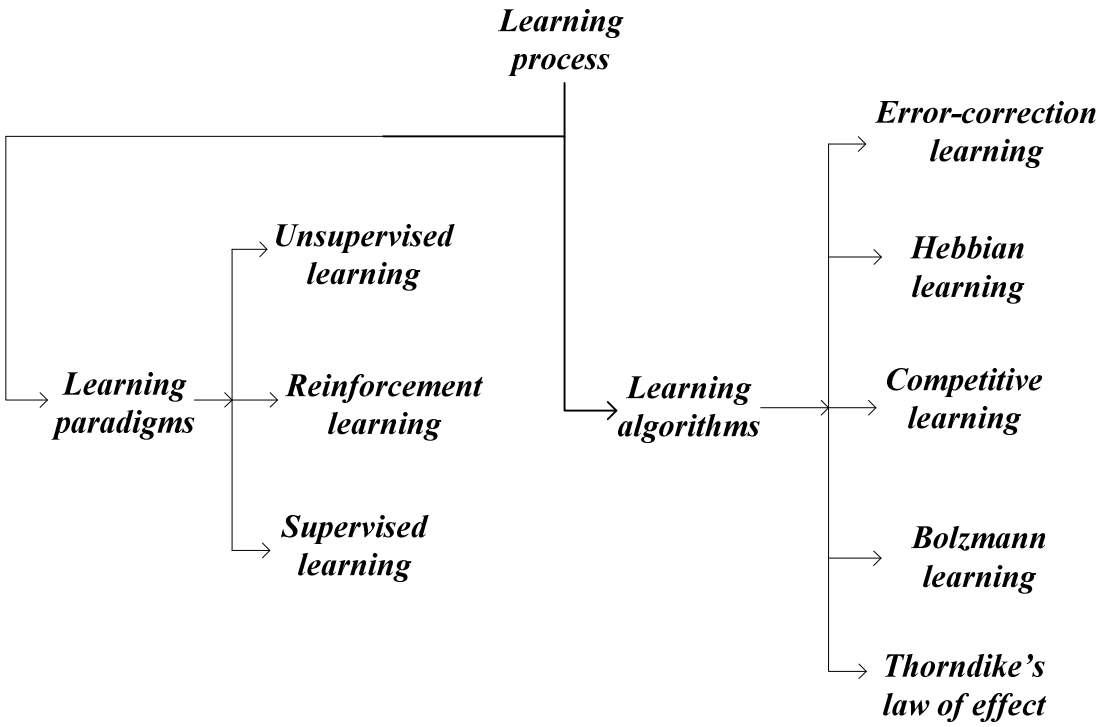

Figure 2.6 A taxonomy for learning process of neural network

The learning paradigm of NN can be divided in to three classes, namely supervised learning, reinforcement learning and unsupervised learning or self-organized learning (Figure 2.6). The training process for FFNN is performed under supervised error correction learning. This learning assumes that there is an availability of a set of training sample with *N* number of

input-output examples as expressed in (2.19) where $\boldsymbol{x}_i$ is the input vector and d_i is the desired response of i^{th} example respectively in training set.

$$T = \{(\boldsymbol{x}_i . d_i)\}_{i=1}^{N} \tag{2.19}$$

This learning is performed under the supervision of an external "teacher" where the teacher has its own built-in knowledge about the surroundings. The environment of interest is represented by a set of input-target examples which unknown to NN. The teacher knows about the precise correctness required for the outputs which are assigned to the network during error correction learning. Figure 2.7 illustrates the schematic diagram for supervised training process.

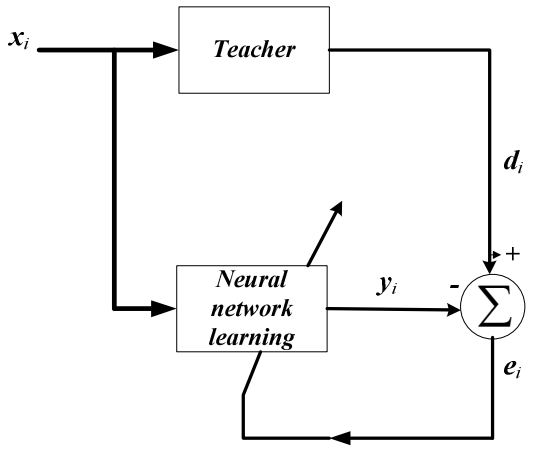

Figure 2.7 Supervised training for NN

When both the teacher or modeled system and the NN are exposed to a set of training sample, the teacher helps providing the instruction or guideline to the network to get closer to desired response for a specific training vector by adapting the parameters of the network in a step-by-step repetitive manner under the combined influence of the training vector and error signal. Here error signal, $e_i = d_i - y_i$ refers to the difference between the desired response,

d_i and actual response, y_i generated by the network in response to the training input, x_i. In this way, the teacher transfers own knowledge to the network as fully as possible and when the network is fully learned, then it can deal with the environment by itself [10].

2.4.5 Learning Algorithm

Learning algorithms are used to update the weights of the network. There are various algorithms for learning among which Lavenberg-Marquardt (LM) [16], [17] is a simple, robust and efficient learning algorithm (by realizing accuracy) for training feed forward neural networks [15]. The main objective of LM is to minimize the difference between actual output, y_i and desired response, d_i of the network [15] hence improving the performance index. So the network's performance parameters such as weights and biases are required to be adjusted through the leaning procedure in order to reduce the index or error function. The sum of squares function or error function of the network can be expressed by (2.20) where *N* is the number of elements of $e(w)$.

$$\begin{aligned} f\,(w) &= \sum_{i=1}^{N} (d_i - y_i)^2 \\ &= \sum_{i=1}^{N} e_i^2\,(w) \\ &= \sum_{i=1}^{N} e\,(w)\, e^T\,(w) \end{aligned} \tag{2.20}$$

Now the weight update matrix equation from Newton's method is

$$w_{k+1} = w_k - [\nabla^2 f\,(w_k)]^{-1}\, \nabla f\,(w_k) \tag{2.21}$$

Where $\nabla^2 f\,(w_k)$ = Hessian matrix and $\nabla f\,(w_k)$ = Gradient of error function.

Now we have to calculate Hessian and gradient of error function. If we differentiate (2.21) with respect to any weight, w_j to find the j^{th} element of gradient $\Delta f(w)$, we get

$$[\nabla f(w)]_J = \frac{\partial f(w)}{\partial w_J} = 2 \sum_{i=1}^{N} e_i(w) \frac{\partial\, e_i(w)}{\partial w_J} \tag{2.22}$$

Equation (2.22) can be written in matrix form:

$$\nabla f(w) = 2\, j^T(w) f(w) \tag{2.23}$$

Where $$J(w) = \frac{\partial e(w)}{\partial w^T} = \begin{bmatrix} \frac{\partial e_1(w)}{\partial w_1} & \frac{\partial e_1(w)}{\partial w_2} & \dots\dots\dots\frac{\partial e_1(w)}{\partial w_n} \\ \frac{\partial e_2(w)}{\partial w_2} & \frac{\partial e_2(w)}{\partial w_2} & \dots\dots\dots\frac{\partial e_2(w)}{\partial w_n} \\ \frac{\partial e_N(w)}{\partial w_2} & \frac{\partial e_N(w)}{\partial w_2} & \dots\dots\dots\frac{\partial e_N(w)}{\partial w_n} \end{bmatrix} \tag{2.24}$$

Here $J(w)$ is the Jacobian matrix [15]. To calculate $J(w)$, we need to find Marquardt sensitivity using back propagation which is shown in [14].

Now the Hessain matrix can be found by differentiating (2.23) with respect to any weights w_k and w_j:

$$[\nabla^2 f(w)]_{kJ} = \frac{\partial^2 f(w)}{\partial w_k \partial w_J} = 2 \sum_{i=1}^{N} \left\{ \frac{\partial e_i(w)}{\partial w_k} \cdot \frac{\partial e_i(w)}{\partial w_J} + e_i(w) \frac{\partial^2 e_i(w)}{\partial_k \partial w_J} \right\} \tag{2.25}$$

The above equation can be written in matrix form:

$$\nabla^2 f(w) = 2\, J^T J(w) + 2\, S(w) \tag{2.26}$$

Where $$S(w) = \sum_{i=1}^{N} e_i(w) \nabla^2 e_i(w) \tag{2.27}$$

Assuming $S(w)$ is small, so approximate Hessain matrix is

$$\nabla^2 f(w) \cong 2\, J^T(w) J(w) \tag{2.28}$$

Now substituting equation (2.23) and (2.28) in (2.29), we get:

$$w_{k+1} = w_k - [J^T(w_k)\, J(w_k)]^{-1} J^T e(w_k) \tag{2.29}$$

In (2.29), $H = J^T J$ may not be invertible which is a problem with Gauss-Newton and it can be overcome by using LM modification [18] to Hessain matrix:

$$G = J^T J + \mu I$$

$$= H + \mu I \tag{2.30}$$

Here I is the identity matrix. Therefore, equation (2.29) becomes

$$w_{k+1} = w_k - \left[J^T(w_k)J(w_k) + \mu_k I\right]^{-1} J^T e(w_k) \tag{2.31}$$

$$\Delta w_k = -\left[J^T(w_k)J(w_k) + \mu_k I\right]^{-1} J^T e(w_k) \tag{2.32}$$

Now if μ increases, for larger μ, equation (2.31) becomes

$$w_{k+1} \cong w_{k-\frac{1}{\mu_k} J^T(w_k)\, e(w_k)}$$

$$= w_k - \frac{1}{2\mu_k} \nabla f(w) \tag{2.33}$$

Now μ is multiplied by an adjacent factor, β when $f(w)$ increases in a step. But when $f(w)$ decreases in a step, μ is divided by β. Usually $\mu = 0.01$ at starting point with $\beta = 10$. Now when μ is large, LM becomes steepest decent algorithm with step $1/\mu$ as shown in (2.32) but for smaller μ, LM becomes Gauss-Newton thus providing nice comparison between these two algorithms.

In general, the target of LM is to improve the performance by reducing the sum of squared error as shown in (2.20) by solving from equation (2.21). So, each learning iteration (epoch) of neural network with LM training is accomplished by following some basic steps [16], [55] which are described below:

1. For initial step, setting $k = 0$ and presenting the training set to the network. Initializing $\mu_0 = 0.001$, $\mu_{max} = 10^{10,}$ $\beta = 10$ and w_0.
2. Computing Jacobian matrix, $J(w_k)$ and $f(w_k)$ by using (2.24) and (2.20) respectively. Terminating the process if $J(w_k)$ or $\nabla f(w_k)$ in (2.23) is less than predefined threshold or if μ_k is equal or greater than $\mu_{max.}$
3. If $\mu_k < \mu_{max.}$, computing $J(w_{k+1})$ and w_{k+1} from (2.24) and (2.31) respectively. If $J(w_{k+1}) \geq J(w_k)$, setting $\mu_k = \beta\mu_k$ and computing this step again. Otherwise we have to compute step 4. If $J(w_{k+1}) < J(w_k)$,setting $\mu_{k+1} = \mu_k / \beta,$ updating the weight, w_{k+1} and computing step 2.
4. If $\mu_k \geq \mu_{max.}$, completing the learning by terminating the process.

When the learning of NN is finished, the weights of the network are fixed and then the network can be simulated with testing samples and the network can be retrained for unsatisfactory performance.

CHAPTER III

IMPLEMENTATION

3.1 System Level Implementation

To evaluate the performance of ANN with CSD coefficient for human face recognition, experimental studies were carried out on AT&T Laboratories Cambridge's Face Database [9]. To realize the performance evaluation in system level, neural network toolbox of Matlab was used. The image database is equally divided in to training set and test set where each set contains 200 images. From each class, images are selected randomly for both sets. The training set of image database was applied to PCA and LDA separately to extract important features from images as discussed in section 2.3.1 and 2.3.2 respectively. After extracting features from PCA and LDA, the feature vector produced by both methods contains 39 elements for each image. So the total size of feature matrix was 39 x 200 both for training and test set. The test feature inputs from PCA and LDA were applied separately to Matlab NN toolbox and FFNN with single hidden and output layer was selected for classification purpose. The input layer contained 39 inputs and the output layer contained 6 output neurons for 200 samples. The optimum number of neurons in hidden layer was selected 28 and 29 for PCA and LDA respectively. The number of hidden layers can be varied but usually single hidden layer with non -linear neuron is sufficient enough to provide good balance between complexity and accuracy for most of the applications [40]. FFNN with LM training [15] algorithm (refer section 2.4.5) was used for training the neural network. The training samples were applied in batch mode [14]. The number of neurons in hidden layer was varied both for PCA-NN and LDA-NN to find out the optimum number of hidden neurons for which the recognition rates were maximum and it was found that the network

performs best in terms of accuracy with 28 hidden neurons for PCA-NN and 29 hidden neurons for LDA-NN. Table 3.1 represents the performance of the network in terms of accuracy (no CSD multiplier was used) where the accuracies were started increase both for PCA-NN and LDA-NN if the hidden neurons were increased up to certain number. But for 28 hidden neurons, the recognition rate was 92% which the performance began to deteriorate if the neurons in hidden layer are increased further more. The same characteristic was also observed for LDA-NN where the recognition accuracy saturated after 29 hidden neurons. Therefore, 28 and 29 hidden neurons were selected for PCA-NN and LDA-NN respectively.

Number of feature elements from each feature vector	Number of hidden neurons	Classification accuracy	
		PCA-NN	LDA-NN
39	23	86.0	88.5
	24	88.0	90.0
	25	90.5	90.0
	26	91.0	91.5
	27	91.0	92.0
	28	92.0	93.5
	29	92.0	94.0
	30	91.5	94.0
	31	92.0	93.5

Table 3.1: Variation in accuracy for different number of hidden neurons in Matlab

ANN training can be performed more efficiently if certain preprocessing steps are performed on the inputs and desired responses of the nerwork before stating the training process As a result, both inputs and targets are normalized and the actual outputs fall into a normalized range. Normally inputs vectors and target vectors both are normalized between [0,1] which helps to reduce possible noise and interference from the inputs and targets. Moreover, after using this

process, there is a transformation happens to the input images so that it becomes easier for NN to classify the datas [14].

If the weights of NN are converted in to 18 bit CSD representation and the nonzero elements of CSD weights are reduced by one, then these weights are not exactly same to the original weights produced by the network. Therefore, some error was added to the network by reducing the nonzero elements by different numbers from the coefficients and for each nonzero elements reduction, the performance of the network was observed. This reduction procedure was continued up to four nonzero elements both for PCA-NN and LDA-NN and for each nonzero element reduction, the recognition accuracy was counted for both network. Table 3.2 presents typical decimal weight examples produced by the network and how the weights are affected after reducing the nonzero elements. The first column presents the actual weights in decimal number. These actual weights were converted in to CSD format and after reducing nonzero elements, weights were converted back in decimal format. Column 2 to 5 of table 3.2 present decimal representations of nonzero element reduced CSD weights.

Original weights produced by network	Nonzero elements reduced by			
	1	2	3	4
22355	22356	22352	22336	22272
-7381	-7380	-7376	-7360	-7424
-3293	-3292	-3296	-3328	-3072
15164	15168	15104	15360	16384
15416	15424	15360	16384	0
2568	2560	2048	0	0
8064	8192	0	0	0

Table 3.2 Effects of nonzero element reduction from 18 bit CSD weight (CSD weights are represented in decimal format)

In the training period, weights are produced and updated by the network during learning iterations. These weights are multiplied by the corresponding input feature element to get the weighted inputs. To observe the performance of NN with CSD coefficients in system level, weights and inputs are converted in to 18 bit CSD and binary representation respectively. Then both the CSD coefficients and binary multiplicand are applied to 18 bit CSD multiplier function which was designed to provide the multiplication result in binary form in Matlab. Then all the multipliers' results and bias were summed together for each neuron and applied as input to tanh AF to get the final output of a neuron.

While training ANN, the nonzero bits of all weights were reduced by a number in all iterations and these weights are applied as coefficients to CSD multiplier. When the network completed learning for 200 face images, then the weights were used to test the performance of network. The testing procedure was quite simple where the weights are fixed because they were collected from training period. These weights and corresponding feature inputs from test sample of faces were applied to 18 bit CSD multiplier as CSD coefficient and multiplicand respectively. Then the weighted sums are added with bias (saved from training period) before applying to AF to get the final output of each neuron. The result found from the neurons in output layer is the final result for testing images. This above procedure was continued for different number of nonzero reduction from the weights and for each reduction, accuracy was counted for test images. Figure 3.1 represents a brief overview of the working procedure of NN for face recognition in both system level and hardware level.

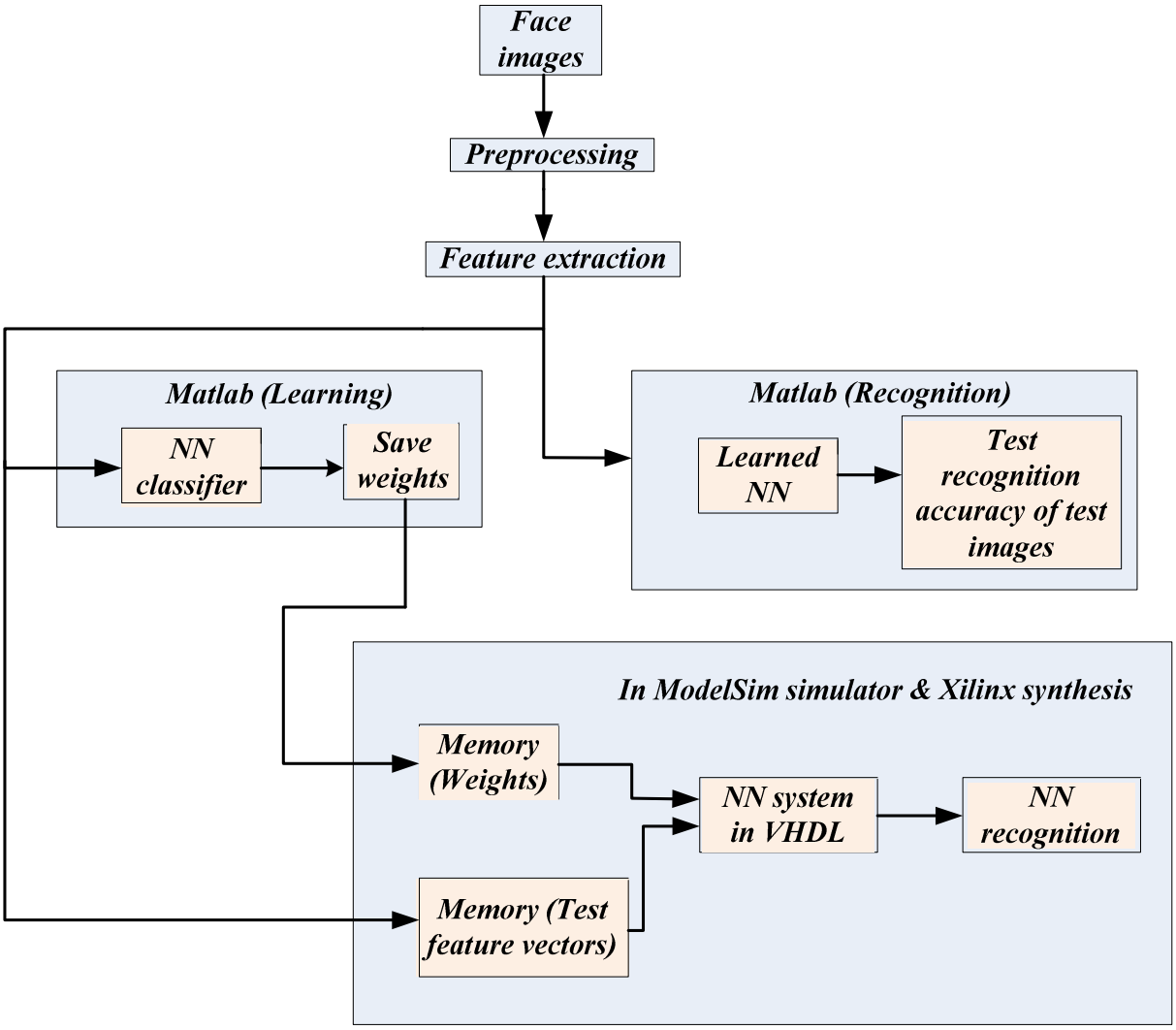

Figure 3.1 Working procedures in system level and hardware level

3.2 Hardware Implementation

To realize the full benefit of face or any pattern recognition application by using ANN classifier with CSD coefficients, it is important to implement ANN system in hardware level because of slower execution time in software based ANN system. Any real time application of face recognition requires consuming less area with high speed recognition rate. Therefore, NN system was designed in hardware platform to outperform the software based system.

Digital hardware based implementation of ANN can be divided in three sub categories: a) application specific integrated chip (ASIC) based implementations, b) field-programmable gate array (FPGA) based implementations, and c) digital signal processor (DSP) based implementations [43], [44]. The last type of implementation performs serial computation, thus it cannot provide the opportunity to preserve the inherent parallel architecture of *ANN*. The ASIC implementation suffers from non reconfigurability. Therefore the system cannot be reconfigured again after loading a design once. But FPGA can fulfill above conditions because it can be reconfigured and it is capable of performing parallel calculations. Thus FPGA implementation was selected for this project.

The hardware implementation of ANN was simulated by ModelSim simulator and then synthesized by using Xilinx ISE. Then the design was implemented in XC6VLX550T FPGA from Xilinx Vertex 6 family which contains 85,920 slices, 632 block RAM each with 36 Kbits in size and each slice contains four LUTs and eight flip-flops where only some slices can use their LUTs as distributed RAM [45]. The behavioral level design of ANN was done by using VHDL language.

In hardware implementation, the offline training of ANN was performed by using Matlab and the inputs, weights and bias data required for the hardware platform were transferred from Matlab as shown in figure 3.1. The whole ANN system in hardware was scaled by a factor of 10000 for the system's simplicity because the inputs, weights and biases contain fractional parts. Therefore, the fractional parts can be avoided by multiplying a scale factor, 10000 which causes a shift in positions of all inputs, weights and up to four digits from fractional part were considered. Only the integer parts of all data were considered after scaling. As a result, the network also provided scaled outputs. So scaling the ANN makes easier to design the hardware

system as the system does not require dealing any data with fractional part. Figure 3.2 presents the top level block diagram of hidden layer and output layer where the outputs of hidden layer are used as inputs to output layer. Each element of feature vector is applied to all neurons in hidden layer in serial and each hidden neuron provides output at same time which goes to every neuron in output layer. Each neuron in output layer performs parallel computation to provide the final output of the neural network.

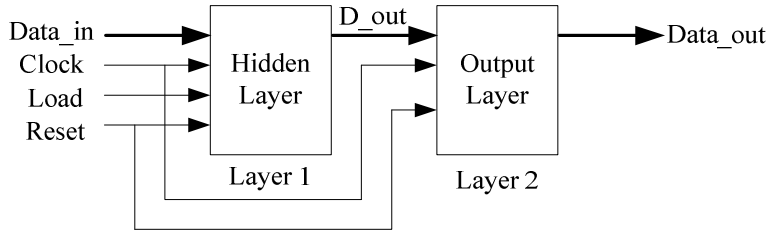

Figure 3.2 Top level block diagram of hidden layer and output layer

3.2.1 Hidden Layer Design

Figure 3.3 represents the block diagram for hidden layer where both input and weights is 18 bit long. Each CSD weight contains positive and negative nonzero elements and these weights are collected from Matlab system level simulation. Therefore, in Matlab, a piece of code is written which allows splitting the positive and negative nonzero elements of CSD weights and save these values in different places. These positive and negative nonzero elements for each neuron are transferred and saved in two ROMs: Weight_pos and Weight_neg as mentioned in figure 3.3. Therefore, each neuron has two weight ROMs associated with it which hold these two types of nonzero elements of CSD weights and the size of the ROM depends on the number of inputs of a neuron. The size of each weight ROM in hidden layer for both PCA and LDA

network is 39x18 as both networks has same number of inputs. Each neuron has a counter which is responsible for this necessary synchronization. In Figure 3.3, when an input enters in hidden layer, it goes to each neuron. Each hidden neuron contains a CSD multiplier which takes 18 bit binary input and positive, negative elements of corresponding CSD coefficient from ROM, The multiplier provides 36 bit multiplication result which is stored in a register. When new input is applied to that hidden neuron, the multiplication result for that input is accumulated with previously stored weighted input. The neuron provides the required address to the weight ROMs for each input by using a counter so that ROMs provide required weights which are used as coefficients of the multiplier. This procedure is continued until all inputs are applied in serial to hidden layer to get the output of a neuron. After applying all inputs to a hidden neuron, the result is forwarded to the tansig AF to get the final output of that neuron. A straight forward LUT based array was used to realize the implementation of AF where each output value corresponds to a unique input address from a series of uniformly spaced input-output values [46], [47]. LUT is also a simple and faster way to realize AF and this method is used for high performance hardware design though it consumes a lot of memory to store the look up table in hardware.

The number of hidden neurons for PCA-NN and LDA-NN was selected as 28 and 29 respectively which were found as optimal number of neurons for hidden layer from table 3.1. Figure 3.4 represents the schematic diagram from Xilinx synthesis tool for a neuron in hidden layer.

Figure 3.5 presents the block diagram for performing CSD multiplication. The multiplier receives three inputs: 18 bit binary multiplicand and positive, negative nonzero elements of CSD weight which are stored in data and sign registers respectively. These data and sign values are

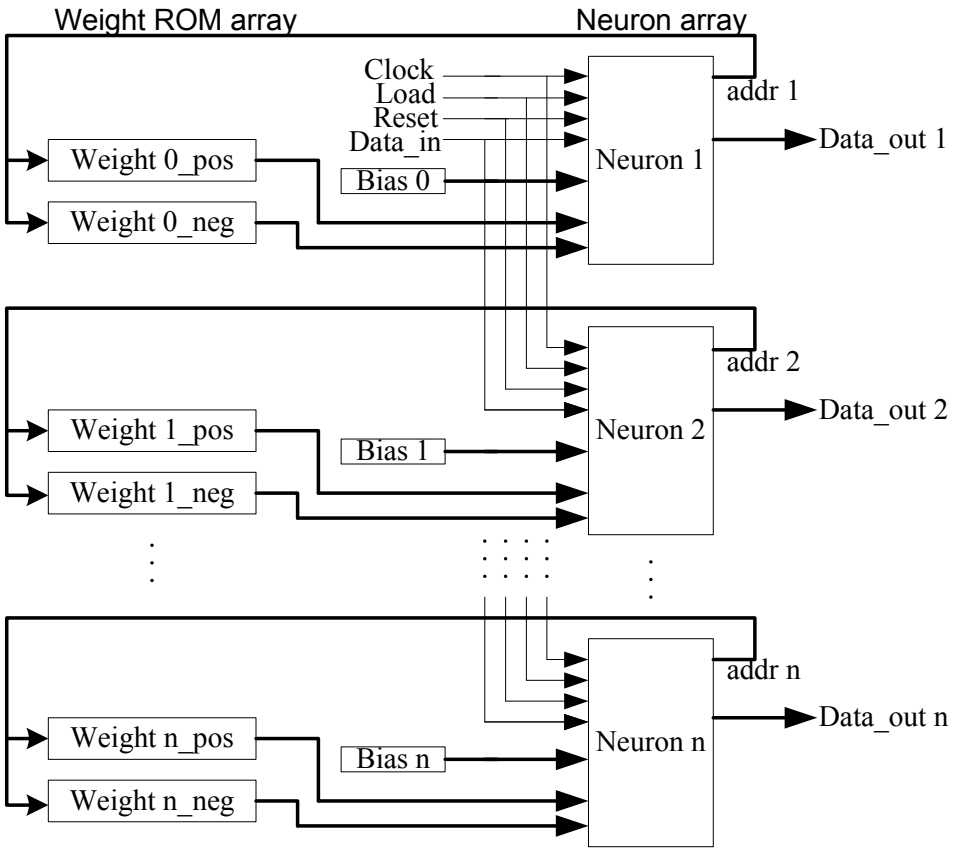

Figure 3.3 Block diagram of hidden layer of neural network in hardware implementation

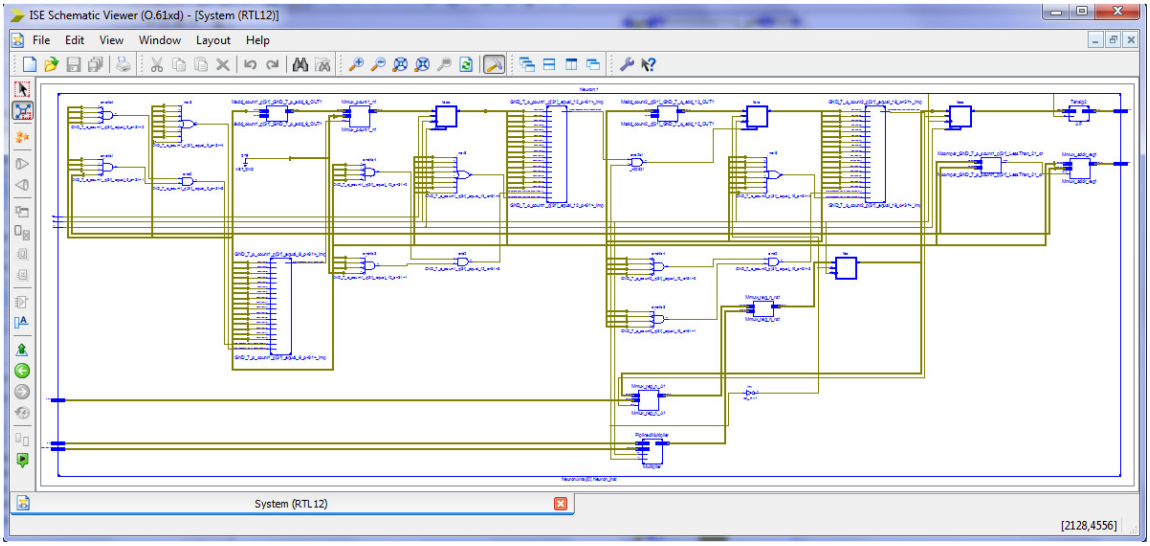

Figure 3.4 Schematic diagram of a neuron in hidden layer

supplied to the multiplier by corresponding neuron and used to represent a CSD number. The data register contains 1 for positive nonzero positions of weight element and the sign register contains 1 for the negative positions of nonzero elements in CSD number. The shift1 and shift2 both are 32 bit signals in shown in figure 3.5. These signals are found from 18 bit binary multiplicand where both signals are the 36 bit sign extended value of multiplicand and shift2 is left shifted by 1. The signal din is generated initially which is the two's complement value of shift1 or shift2 depending on the first and second bit positions of sign and data. These four signals shift1, shift2 and data, sign are used as inputs to shift and accumulator which performs two operations: a) it performs left shift by 2 on shift1 and shift2 and right shift by 2 on data and sign; b) next it performs addition or subtraction of shift1 or shift2 with din depending on the bit position of data and sign to provide dout. As 18 bit CSD coefficient contains maximum of 9 nonzero elements, therefore there can be maximum of nine shift and addition or subtraction operations are required for any 18 bit CSD multiplication and the final result is expressed as Product in figure 3.5. Thus the number of partial products is reduced by half for any n bit multiplier.

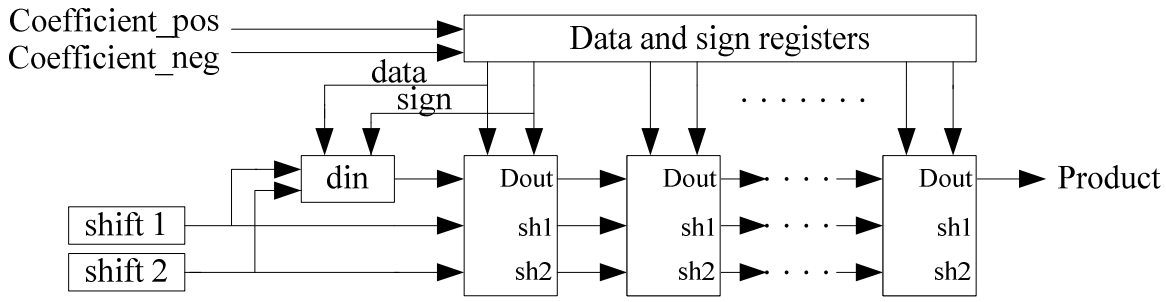

Figure 3.5 Block diagram for CSD multiplier

The RTL schematic diagram found from Xilinx synthesis tool for CSD multiplier is presented by figure 3.6.

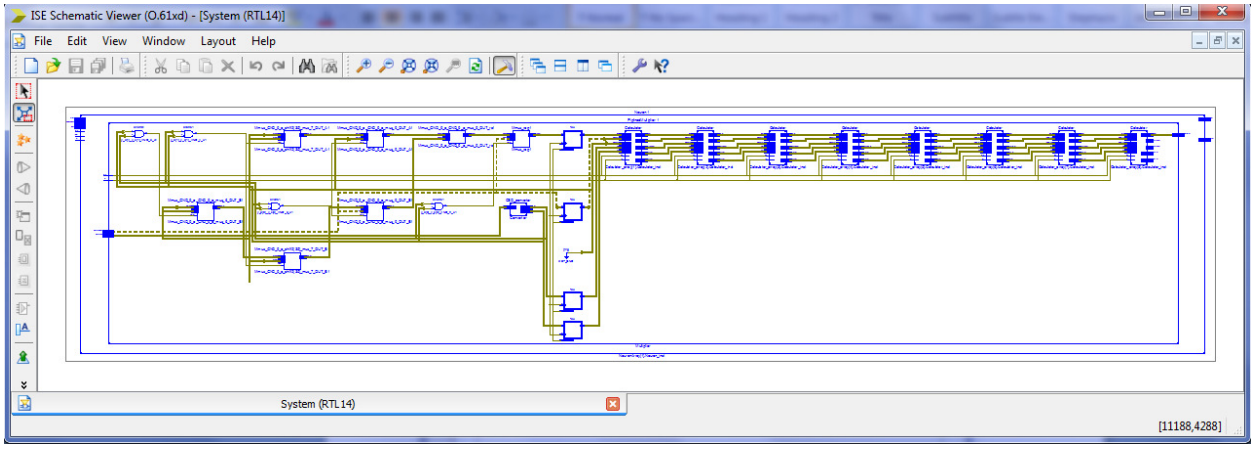

Figure 3.6 RTL schematic diagram of CSD multiplier

3.2.2 Output Layer

The output layer contains total of 6 neurons for 200 test images. The architecture of this layer is designed to work in fully parallel where each neuron has n number of multipliers depending on the number of inputs to this layer. For PCA-NN, the number of hidden neurons is selected as 28 from system level. Therefore each output neuron for PCA-NN contains 28 multipliers. Each output neuron has its own weight ROM of size 28x18 to hold the weights for PCA-NN and 29x18 for LDA-NN. When all inputs enter in to output layer, they go to each output neuron. Figure 3.7 presents the block diagram for a neuron in output layer where each neuron contains n number of multipliers which is equal to the number of inputs. Therefore, all multipliers of each output neuron perform parallel multiplications and provide the multiplication outputs at same time. Again, each output neuron has its own bias associated with it. An adder tree was used to add multiplication results and bias of a neuron in output layer. The final step

consists of applying the 32 bit output (produced by the adder tree) as input address to tansig activation function to produce the final output of a neuron in output layer.

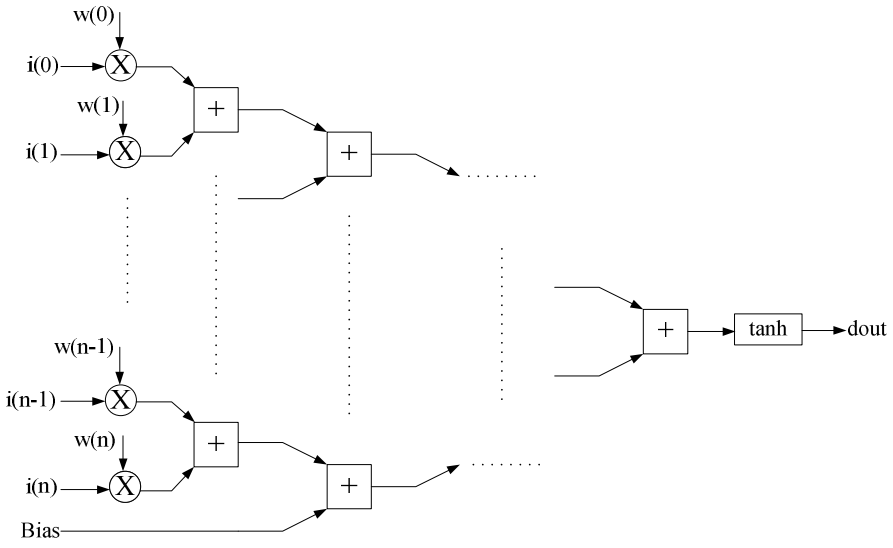

Figure 3.7 Block diagram of a neuron in output layer

CHAPTER IV

RESULTS AND DISCUSSION

4.1 Performance Analysis

NN based face recognition with CSD coefficients were implemented where the NN system was synthesized by using Xilinx ISE for a Xilinx FPGA device, XC6VLX550T. This chapter contains the results found from system level implementation, hardware implementation, comparison between system and hardware level results. The last part of this chapter contains the comparison between NN with CSD multiplier and binary multiplier in hardware platform.

4.1.1 Accuracy Analysis of System Level Implementation

In system level implementaiton, a multiplier was designed to perform CSD multiplication in Matlab as discussed in section 3.1 and it helps to realize how the system may perform in hardware platform. Then 39 input feature elements of test images were applied to the network both for PCA and LDA. The results found from system level Implementation is represented in table 4.1:

	Nonzero elements reduced by	PCA-NN		LDA-NN	
		Classification accuracy (%)	Epoch	Classification accuracy (%)	Epoch
With CSD system	1	92.0	19	94.0	20
	2	92.0	18	93.5	17
	3	87.5	22	87.5	19
	4	76.5	21	77.0	21
Without CSD system	--	92.0	18	94.0	19

Table 4.1: Accuracy and other parameter comparison using Matlab for face recognition

The last row of table 4.1 presents the results found from Matlab where CSD system was not used and others contain the results when CSD coefficients and CSD multiplier were used. Here, the nonzero elements from CSD coefficients were reduced from 1 to 4. Therefore these weigths were not accurate because they are not exactly same to the original weights produced by network. For each reduction, the recogntion accuracy were counted. From table 4.1, we can see that classification accuracy for PCA-NN after reducing single nonzero element is 92% which is same to the result found from if CSD system was not used. Moreover, the recognition rate started to decrease gradually for both PCA-NN and LDA-NN when the number of nonzero elements of CSD weights was reduced. The accuracy dropped by 16.8% for PCA-NN if nonzero elements were reduced by 1 and 4. For LDA-NN, the recognition rate for both CSD and without CSD system is same and here the accuracy also dropped 18.1% for 1 and 4 nonzero elements reduced weights. In table 4.1, epoch mentions the number of iterations required for the network while learning and we can see that it requires almost same learning iteration for the network to learn for both PCA-NN and LDA-NN.

4.1.2 Hardware Based Performance

To realize the full benefit of any system, it is necessary to implement the system in hardware level. NN with CSD coefficients for human face recognition was implemented in hardware by using VHDL coding. A description of the implemented design was discussed in chapter III.

4.1.2.1 Accuracy Realization of Designed System from Simulation

ModelSim simulator was used to verify if the VHDL description of NN really performs quite similar to system level performance. At first, the network was designed using VHDL and

then test images were used to test the accuracy of the designed system. Table 4.2 presents the result in terms of accuracy for NN with CSD coefficients for face recognition:

Nonzero elements of CSD coefficient reduced by	Accuracy for PCA-NN (%)		Accuracy for LDA-NN (%)	
	From simulation	From system level	From simulation	From system level
1	92.0	92.0	93.5	94.0
2	91.5	92.0	92.5	93.5
3	87.0	87.5	86.0	87.5
4	75.5	76.5	76.0	77.0

Table 4.2 Accuracy comparison between hardware and software implementation of PCA-NN and LDA-NN with CSD coefficients

From table 4.2, we can see that the recognition accuracies for face images from simulation are almost similar to the accuracy found from Matlab with using CSD multiplier. The accuracies for PCA-NN and LDA-NN system from ModelSim simulation started to reduce gradually with the reduction of nonzero elements from CSD coefficients. For PCA-NN, the recognition accuracy dropped by 17.9% for 1 and 4 nonzero bit reduction from CSD coefficients in ModelSim simulation and from system level, it was 16.8% for the same number of bit reduction from coefficients. In table 4.2, for ModelSim simulation of LDA-NN, the accuracies were found 93.5% and 76.0% for nonzero bits reduction from weights by 1 and 4 respectively and the accuracy rate was fallen by 18.7% for the same number of bit reduction but in system level simulation, it was 18.1%.

There are 40 classes for face database and each class contains 10 identical face images. If maximum number of corrected faces per class is considered for both PCA-NN and LDA-NN, then the recognition accuracy started to increase. For example, if NN for PCA provides 8 correct outputs for a class and NN for LDA provides 9 correct outputs for that same class, then

maximum correct images was considered as 9 for that specific class. In this way, for each class, highest correct faces were counted and it was found that the total recognition rate improved by a significant amount for both PCA and LDA network. Table 4.3 represents the final results for maximum corrected images for 40 classes in hardware platform for the first nonzero element reduction from coefficients where the total recognition rate for 40 classes is 95.2 which mean the rate improves by 3.4% for PCA network and 1.8% for LDA network:

PCA-NN (%)	LDA-NN (%)	Maximum correct faces for 40 classes (%)
92.0	93.5	95.2

Table 4.3 Improvement in total recognition rate after considering maximum number of correct faces for 40 classes in hardware platform

4.1.2.2 Resource Requirements

XC6VLX550T FPGA device from Xilinx Virtex 6 family was used for implementation. The use of CSD coefficients in NN reduces the area requirements of the network because CSD coefficients contains maximum half nonzero elements of total bit width. If the nonzero bit is reduced from the coefficients, the area requirement also started to reduce more. Table 4.4 represents the resources required for each nonzero bit reduction from coefficients of both PCA-NN and LDA-NN. From this table, we can see that the number of slice registers and slice LUTS are reduced by 19.47% and 51.86% respectively for nonzero elements of CSD coefficients reduced by 1 and 4 of PCA-NN. For LDA-NN, the number of slice registers and LUTS are reduced by 20.71% and 52.33% respectively for the same number of nonzero elements reduction from the coefficients of this network.

Nonzero bits reduced by	PCA-NN		LDA-NN	
	# of slice registers	# of slice LUTs	# of slice registers	# of slice LUTs
1	7366	59964	7668	61818
2	6922	49373	7173	51134
3	6466	39189	6654	39851
1	5932	28862	6080	29466

Table 4.4: Resource requirements for PCA and LDA network from Xilinx synthesis

Therefore there is a great reduction of area with the reduction of nonzero elements from network's CSD coefficients but here we have to sacrifice the accuracy because the network's accuracy also starts to reduce with the reduction of nonzero elements from CSD weights. The objective of this work is to reduce the area requirement of NN with CSD coefficients in comparison to corresponding binary network for face recognition application. Therefore a signed binary multiplier was used instead of CSD multiplier to observe the network performance. Both CSD and binary multiplier were synthesized and the results are shown in following table where we can see that CSD multiplier requires 0 slice registers and 19% less slice LUTs in comparing to binary multiplier.

Type of multiplier	# of slice registers	# of slice LUTs
CSD	0	422
Binary	95	521

Table 4.5: Resources required by multipliers

Figure 4.1.a and 4.1.b show the graphical representations of table 4.4. From these figures, we can see that the number of slice registers and slice LUTs reduces gradually and almost

linearly with the reduction of nonzero elements from CSD coefficients from both networks. Moreover, the resource requirement for PCA-NN is less than LDA-NN because PCA network has 28 neurons where as LDA network had 29 neurons in hidden layer. Therefore LDA network requires more resources for saving the positive and negative nonzero elements of CSD coefficients in ROM as discussed in previous chapter.

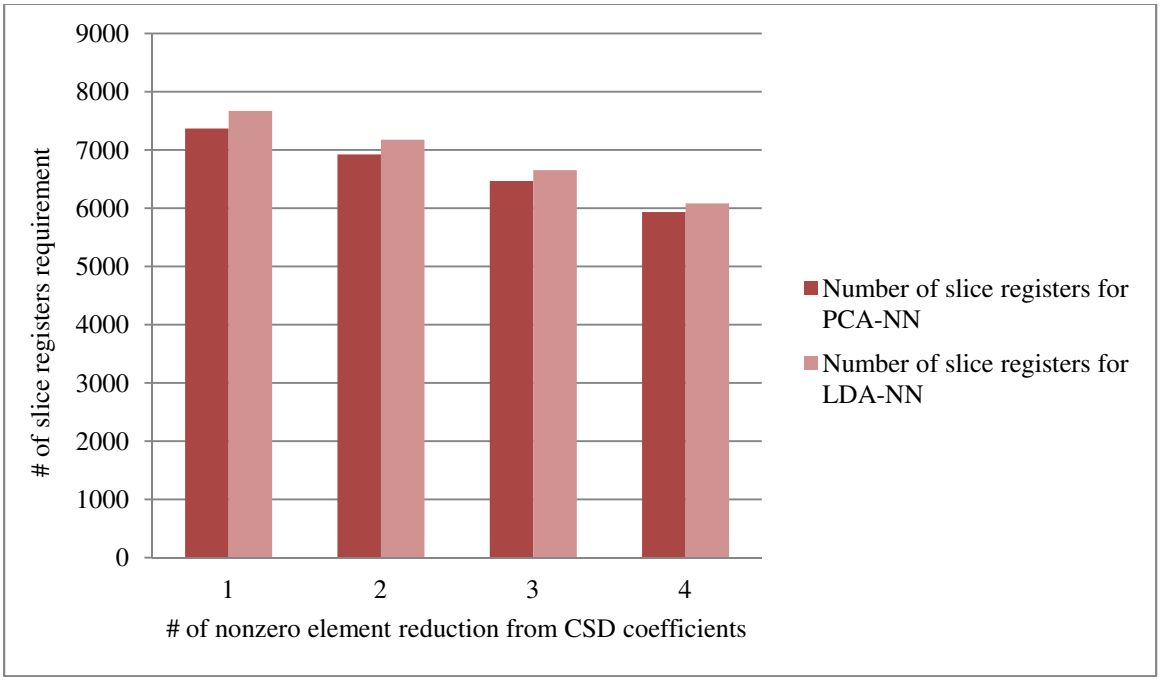

Figure 4.1.a Number of nonzero elements reduction vs. slice registers requirement

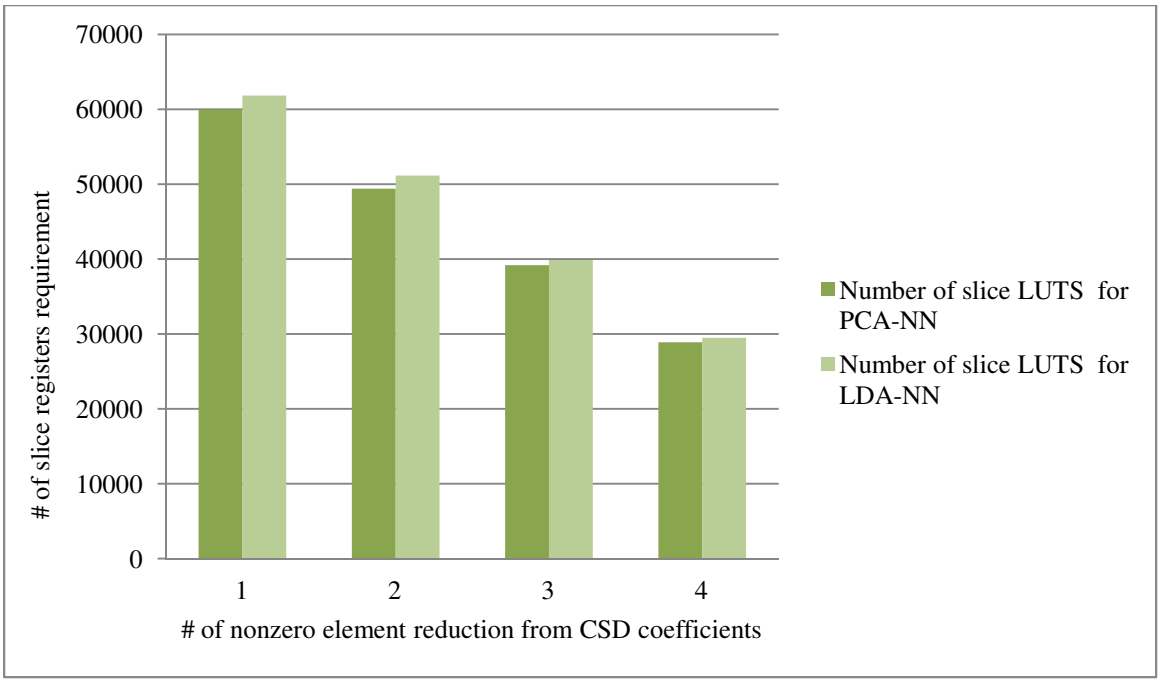

Figure 4.1.b Number of nonzero elements reduction vs. slice LUTs requirement

4.1.2.3 Timing Performance

The time requirements for PCA-NN and LDA-NN are presented in table 4.6 where the time started to reduce gradually with the reduction of nonzero elements from CSD coefficients.

Nonzero elements reduced by	PCA-NN		LDA-NN	
	Minimum time (ns)	Maximum operating frequency (MHz)	Minimum time (ns)	Maximum operating frequency (MHz)
1	7.980	125.3	7.981	125.3
2	7.092	141.0	7.092	141.0
3	6.194	161.4	6.194	161.4
4	5.278	189.5	5.278	189.5

Table 4.6 Time requirements for PCA-NN and LDA-NN from Xilinx synthesis

From the above table, we can see that both network speed up by 33.9% for nonzero elements reduction by 1 and 4. But here we have to sacrifice the accuracy of the network because from table 4.2, the accuracy also started to decrease gradually with the reduction of nonzero bits from CSD coefficients of PCA-NN and LDA-NN.

From table 4.6, we can see that that both PCA and LDA network require almost same processing time. Recalling the architecture of hidden layer and output layer from chapter III, hidden layer receives feature elements in serial but all hidden neurons perform computations in parallel. The output neurons in output layer receive input at a same time and perform parallel computations. As all neurons in both networks perform parallel computations and both of them have same number of feature elements, therefore both networks require same processing speed.

The time requirements for PCA-NN and LDA-NN from table 4.3 are plotted in figure 4.2.a and 4.2.b respectively with the reduction of nonzero elements from PCA-NN and LDA-NN where we can see that the processing time reduces almost linearly for each nonzero bit reduction from CSD coefficients of both networks.

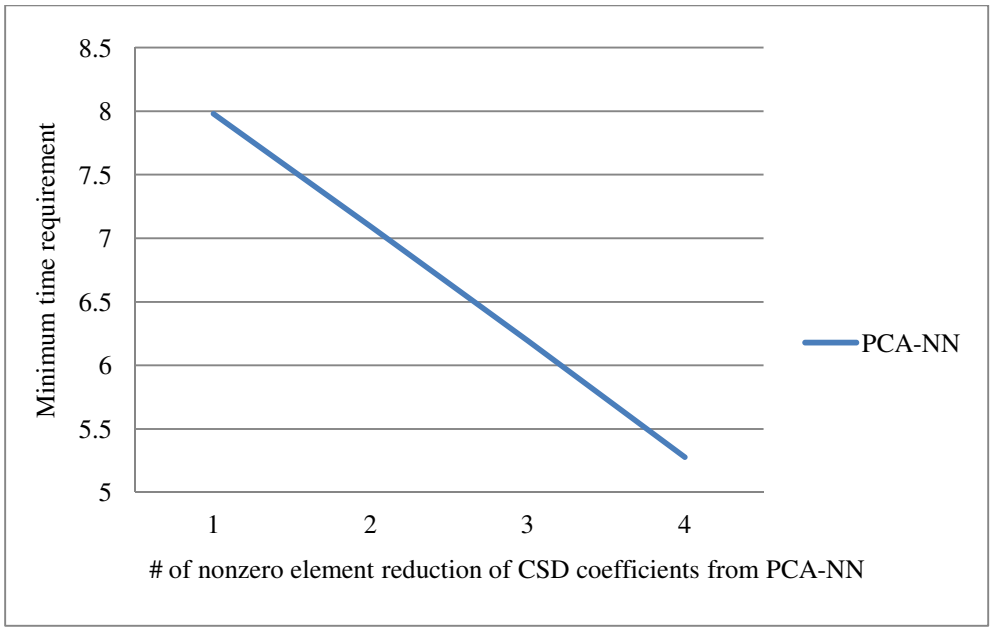

Figure 4.2.a Nonzero elements reduction vs. minimum time requirement for PCA-NN

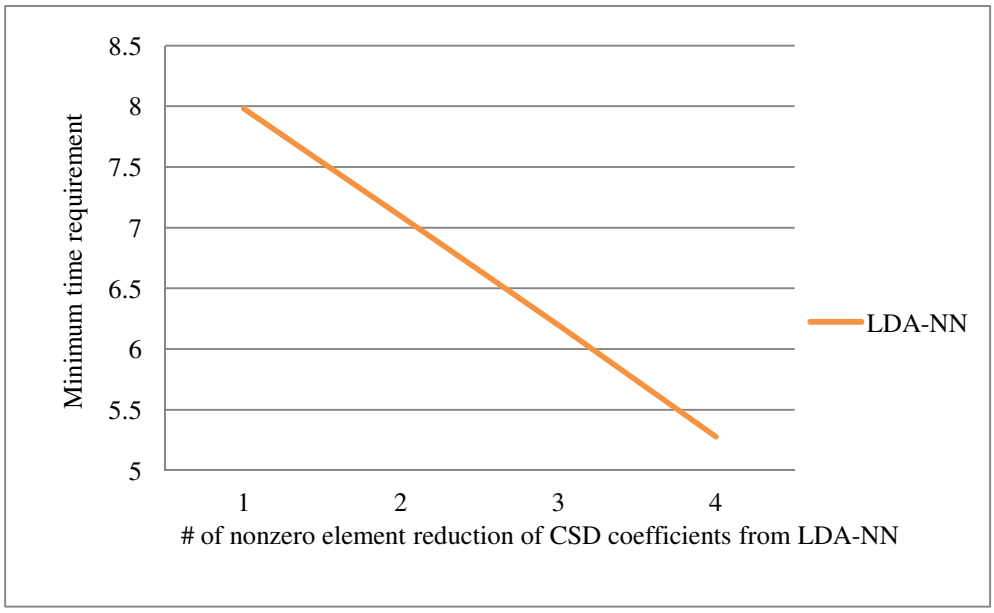

Figure 4.2.b Nonzero elements reduction vs. minimum time requirement for LDA-NN

4.1.2.4 Comparison of Hardware Implemented CSD Based NN with Binary NN

To realize how the design of hardware based NN with CSD coefficients performs in comparison to the network with binary system; NN with binary coefficients was used to compare the results. Binary number system is widely used in most of the hardware based systems. Therefore a signed binary multiplier was used instead of CSD multiplier where other network parameters are kept same and the results for resource and time requirement are shown in table 4.7.

Coefficient system	PCA-NN			LDA-NN		
	Number of slice registers	Number of slice LUTS	Required time (ns)	Number of slice registers	Number of slice LUTS	Required time (ns)
CSD	7366	59964	7.980	7668	61818	7.981
Binary	21973	78205	5.456	22792	80262	5.456

Table 4.7 Resource and time requirements for PCA-NN, LDA-NN with CSD and binary coefficient system from Xilinx synthesis

From the above table we can see that the number of required slice registers for PCA-NN and LDA-NN are 66.4% and 66.3% respectively which shows that both networks require less than half slice registers than corresponding binary network. For any hardware based system, the number of required slice register is an important parameter and the low requirement of slice registers indicates more compactness of the network. Therefore if the number of slice registers is reduced, it implies significant reduction in area. Moreover CSD based PCA and LDA networks require 23.3% and 22.9% less number of slice LUTs respectively than in comparison to

corresponding network with binary coefficients. But network with CSD coefficients requires little bit more time than the time required for the network with binary coefficients. Though CSD system requires some more time, but it helps to reduce the area of hardware based neural network by almost half thus making the size of the network smaller which is the objective of this thesis. In most real time application of neural network with face recognition, the size of face database is very large and it requires large area for hardware based large NN system. Therefore the advantage of NN with CSD coefficients in hardware platform is that it is more suitable for large face database with size almost double than corresponding network with binary coefficients.

CHAPTER V

CONCLUSION AND FUTURE WORKS

In this thesis, neural network classifier with CSD coefficients was used for face recognition application. Hardware implementation of this classifier was accomplished for realizing the full benefits of this application. Area and time are two most important parameters in any hardware design and binary system is widely used in designing hardware system. Thus CSD coefficients were used instead of binary coefficients to minimize the area requirement of neural network to make the network more compact because CSD contains minimum possible nonzero element among all signed digit number system. As a result, when the weights of neural network are saved by using CSD system for face recognition application, it consumes very low space and requires performing lowest possible number of partial products and additions to get weighted input which saves the area also. Therefore it was possible to reduce the area for PCA-NN and LDA-NN by almost half than its binary counterpart while much not compromising the speed of overall network in comparison to binary neural network system. The results from FPGA implementation resemble a significant improvement in space requirement for the recognition of face images for both PCA-NN and LDA-NN after using CSD coefficients than binary coefficients while both PCA and LDA networks also maintained satisfactory accuracies almost similar to the system level. Moreover, it was found that there was a significant improvement in recognition rate by considering maximum number of correct faces from PCA-NN and LDA-NN for all classes. The results also show that there was a gradual reduction in resources, time and recognition rate with each nonzero element reduction from the coefficients of both networks.

5.1 Contribution of Proposed Work

Nowadays, area and speed are two important issues in any hardware based systems of almost any real life applications. Most of the recent technologies are very compact in terms of space requirement. Face recognition is an advance technique with a very high demand in numerous applications in real world and the size of neural networks classifier with face recognition application become very large because of the large database of face images. Thus the method proposed in this thesis is proven to be useful for neural networks with face recognition application in real time hardware system as this method significantly reduce the size of the required silicon area for NN. Therefore FPGA implementation of the proposed idea can be used to generate a promising real time face recognition system by reducing the size of the network while much not compromising the speed required of the system.

5.2 Future Work

For further effectiveness of proposed work, the face recognition application can be implemented with different types of neural network and activation functions to observe the performance of the recognition system. The hardware system accuracy can also be improved by using other effective feature extraction methods. The performance of the system can be observed by using other standard face databases. Moreover, this recognition application can also be implemented by using analog and hybrid neural network classifier which offers some level of flexibility and takes some advantages of digital implementation.

REFERENCES

[1] M. Poliac, J. Zanetti, and D. Salerno, "Performance Measurements of Seismocardiogram Interpretation Using Neural Networks, Computer in Cardiology," IEEE Computer Society, 1993, pp. 573 - 576.

[2] U. Rucket, A. Funke, and C. Pintaske, "Accelerator board for Neural Associative Memories," IEEE Neurocomputing, Vol.5, No.1, 1993, pp. 39 - 49.

[3] B. Hassinbi, D. G. Stork, and G. J. Wolff, "Optimal brain surgeon and general network pruning," in Proc. IEEE Int. Joint Conf. Neural Netw., Vol. 2, 1992, pp. 441 – 444.

[4] B. Widrow and R. Winter, "Neural nets for adaptive filtering and adaptive pattern recognition," IEEE Computer, Vol. 21, no. 3, Mar. 1988, pp. 25 – 39.

[5] K. Fukushima, S. Miyake, and T. Ito, "Neocognitron: A neural network model for a mechanism of visual pattern recognition," IEEE Trans. Syst., Man, Cybern. , Vol. SMC-13, no. 5, 1983, pp. 826 – 834.

[6] S. Grossberg, E. Mingolla, and D. Todorovic, "A neural network architecture for preattentive vision," IEEE Trans. Biomed. Eng., Vol. 36, no.1, Jan. 1989, pp. 65 – 84.

[7] Ming-Hsuan Yang, D. J. Kriegman, and N. Ahuja, "Detecting Faces in Images: A Survey," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 24, Jan. 2002, pp. 34 - 58.

[8] http://www.face-rec.org/databases/, Last visited: October, 2012.

[9] http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html for downloading ORL face database.

[10] S. Haykin, Neural networks: a comprehensive foundation, Prentice Hall, July, 1998.

[11] D. E. Rumelhart and J. L. McClelland and the PDP Research Group, Parallel Distributed Processing, Vol. 1: Foundations, The MIT Press, July, 1987.

[12] B. Karlik and A. V. Olgac, "Performance Analysis of Various Activation Functions in Generalized MLP Architectures of Neural Networks," International Journal of Artificial Intelligence and Expert Systems (IJAE), Volume (1): Issue (4), 2011.

[13] M. Marvin, and S. Papert, Perceptrons: An Introduction to Computational Geometry. Cambridge, MA: MIT Press, 1969.

[14] M. T. Hagan, H. B. Demuth and M. H. Beale, Neural Network Design, Boston, MA: PWS Publishing, 1996.

[15] M. T. Hagan and M. B. Menhaj, "Training Feed-forward Networks with the Marquardt Algorithm," IEEE transactions on Neural Network, Vol. 5 No. 6, Nov. 1994, pp. 989 - 993.

[16] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," The Quarterly of Applied Mathematics, Vol. 2, 1944, pp. 164 - 168.

[17] D. W. Marquardt, "An algorithm for least-square estimation of nonlinear parameters," SIAM Journal on Numerical Analysis, Vol. 11, no. 2, 1963, pp. 431 - 441.

[18] R. Battiti, "First- and second order methods for learning: Between steepest descent and Newton's method," Neural Computation, Vol. 4, no. 2, 1992, pp. 141 - 166.

[19] Lih-Heng Chan, S. H. Salleh, and Chee-Ming Ting, “PCA, LDA and Neural Network for Face Identification,” 4th IEEE Conference on Industrial Electronics and Applications, May. 2009, pp. 1256 - 1259.

[20] Y. Jiang and P. Guo, “Comparative Studies of Feature Extraction Methods with Application to Face Recognition,” IEEE International Conference on Systems, Man and Cybernetics, Oct. 2007, pp. 3627 - 3632.

[21] H. Sahoolizadeh, and Y. A. Ghassabeh, “Face Recognition using Eigen-faces, Fisherfaces and Neural Networks,” 7th IEEE International Conference on Cybernetic Intelligent Systems, Sept. 2008, pp. 1 - 6.

[22] Lih-Heng Chan, S. Salleh, Chee-Ming Ting, and A. K. Ariff, “PCA AND LDA based face verification using back-propagation neural network,” 10th IEEE International Conference on Information Sciences Signal Processing and their Applications (ISSPA), May. 2010, pp. 728 - 732.

[23] W. Zhao, R. Chellappa, and A. Krishnaswamy, "Discriminant Analysis of Principal Components for Face Recognition," 3rd IEEE International Conference on Automatic Face and Gesture Recognition, Apr. 1998, pp. 336 - 341.

[24] M. A. Turk and A. P. Pentland, "Face Recognition Using Eigenfaces," IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Jun. 1991, pp. 586 - 591.

[25] Zhujie, and Y. L. Yu, "Face Recognition with Eigenfaces," Proceedings of the IEEE International Conference on Industrial Technology, Dec. 1994 pp. 434 - 438.

[26] M. Oravec, and J. Pavlovicova, "Face Recognition Methods Based on Principal Component Analysis and Feedforward Neural Networks," IEEE International Joint Conference on Neural Networks, Vol. 1, July. 2004.

[27] M. Agarwal, H. Agrawal, N. Jain, and M. Kumar, "Face Recognition using Principle Component Analysis, Eigenface and Neural Network," International Conference on Signal Acquisition and Processing, Feb. 2010, pp. 310 - 314.

[28] V. P. Kshirsagar, M. R. Baviskar, and M. E. Gaikwad, "Face Recognition Using Eigenfaces," IEEE 3rd International Conference on Computer Research and Development (ICCRD), Vol. 2, March 2011, pp. 302 - 306.

[29] J. Ruiz-del-Solar, and J. Quinteros, "Illumination Compensation and Normalization in Eigenspace-based Face Recognition: A comparative study of different pre-processing approaches," Pattern Recognition Letters, Vol. 29, No. 14, 2008, pp. 1966 - 1979.

[30] A. M. Martinez, and A. C. Kak, “PCA versus LDA,” IEEE Trans. Pattern Anal. Machine Intel., Vol. 23, 2004, pp. 228 - 233.

[31] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, “Eigen faces vs. Fisher faces: Recognition using class specific linear projection,” IEEE Trans. Pattern Anal. Machine Intel. Vol. 19, May 1997, pp. 711 - 720.

[32] J. J. Weng, “Using discriminant eigen-features for image retrieval,” IEEE Trans. Pattern Anal. Machine Intel., Vol. 18, No. 8, 1996, pp. 831 - 836.

[33] R. M. Hewlitt, and E. S. Swartzlantler, Jr., “Canonical signed digit representation for FIR digital filters,” in Proc. of the IEEE workshop on Signal Processing Systems, 2000, pp. 416 - 426.

[34] A. D. Booth, “A Signed Binary Multiplication Technique,” Quarterly J. Mechanics and Applied Math., Vol. 4, 1951, pp. 236 – 240.

[35] C. K. Koc, "Parallel canonical recoding," Electronics Letters, Vol. 32, 1996, pp. 2063 - 2065.

[36] G. K. Ma and F. J. Taylor, "Multiplier Policies For Digital Signal Processing," IEEE ASSP Mag., Jan. 1990, pp. 6 - 20.

[37] L. Wanhammar, DSP Integrated Circuits, Academic Press, 1999.

[38] C. K. Koc, and Johnson, S., "Multiplication of signed-digit numbers," Electronics Letters, Vol. 30, 1994, pp. 840 - 841.

[39] K. K. Parhi, VLSI Digital Signal Processing Systems: Design and Implementation, John Wiley, 1999.

[40] T. M. Mitchell, Machine Learning, McGraw Hill, 1997.

[41] L. M. Reyneri, “Implementation issues of neuro-fuzzy hardware: Going towards HW/SW codesign,” IEEE Trans. Neural Netw., Vol. 14, no. 1, Jan. 2003, pp. 176–194.

[42] H. K. Kwan and C. Z., Tang, "A multilayer feedforward neural network model for digital hardware implementation," Proc. IEEE International Symposium on Circuits and Systems, Vo1.6, 1994, pp. 343 - 345.

[43] L. M. Reyneri, “Implementation issues of neuro-fuzzy hardware: Going towards HW/SW codesign,” IEEE Trans. Neural Network, Vol. 14, no. 1, Jan. 2003, pp. 176 – 194.

[44] M. Cristea and A. Dinu, “A new neural network approach to induction motor speed control,” in Proc. IEEE Power Electron. Specialist Conf., Vol. 2, 2001, pp. 784 – 788.

[45] http://www.xilinx.com/support/documentation/data_sheets/ds150.pdf, Last visited: October, 2012.

[46] H. K. Kwan, "Simple sigmoid-like activation function suitable for digital hardware implementation," Electronic Letters, Vol. 28, no. 15, July 1992, pp. 1379 - 1380.

[47] F. Piazza, A. Uncini, and M. Zenobi, "Neural networks with digital LUT activation functions," Proceedings of 1993 International Joint Conference on Neural Networks, IJCNN, Oct. 1993, pp. 1401 - 1404.

[48] R. C. Gonzalez, R. E. Woods, Digital Image Processing, second ed., Prentice Hall, 1992.

[49] S. Hariprasath, and T. N. Prabakar, “FPGA Implementation of Multilayer Feed Forward Neural Network Architecture Using VHDL,” International Conference on Computing, Communication and Application (ICCCA), Feb. 2012, pp. 1 - 6.

[50] http://content.imamu.edu.sa/Scholars/it/net/9114.pdf, Last visited: October, 2012.

[51] http://www.svms.org/finance/ShinLeeKim2005.pdf, Last visited: October, 2012.

[52] B. A. White, and M. I. Elmasry, "The Digi-Neocognitron: A Digital Neocognitron Neural Network Model for VLSI," IEEE Transactions on Neural Networks, Vol. 3, No. 1, Jan. 1992, pp. 73 - 85.

[53] H. Djahanshahi, "A robust hybrid VLSI neural network architecture for a smart optical sensor," PHD thesis, University of Windsor, 1998.

[54] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face recognition: a literature survey," ACM Computing Surveys, Vol. 35 No.4, 2003, pp. 339 - 458.

[55] A. Pukrittayakamee, "Fittinng functions and their derivatives with neural networks," PHD thesis, Oklahoma State University, 2009.

[56] S. W. Reitwiesner, “Binary Arithmetic,” Advances in Computers, Academic Vol. 1, 1966, pp. 231 - 308.

[57] Mi Lu., Arithmetic and Logic in Computer Systems, Wiley publishing, 2005, pp. 95 - 97.

[58] R. Guo and L. S. DeBrunner, "A Novel Fast Canonical-Signed-Digit Conversion Technique for Multiplication," IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), May 2011, pp. 1637 - 1640.

[59] A. Hu, A. and J. Al-Khalili, "Comparison of Constant Coefficient Multipliers for CSD and Booth Recoding," 14th International Conference on Microelectronics, Dec. 2002, pp. 66 - 69.

[60] L. Wanhammar, DSP Integrated Circuits, Academic Press, 1999, pp. 469 - 470.

[61] Sang-Min Kim, Jin-Gyun Chun, and K. K. Parhi, "Design of Low Error CSD Fixed-Width Multiplier," IEEE International Symposium on Circuits and Systems, Vol.1, 2002, pp. 69 - 72.

[62] Sang-Min Kim, Jin-Gyun Chung, and K. K. Parhi, "Low Error Fixed-Width CSD Multiplier With Efficient Sign Extension," IEEE Transactions on Circuits and Systems—II: Analog and Digital Signal Processing, Vol. 50, No. 12, Dec. 2003, pp. 984- 992.

[63] Shyh-Jye Jou and Hui-Hsuan Wang, "Fixed-Width Multiplier for DSP Application," International Conference on Computer Design, 2000, pp. 318 – 322.

VITA AUCTORIS

NAME:	Ayesa Parvin
PLACE OF BIRTH:	Dhaka, Bangladesh
YEAR OF BIRTH:	1986
EDUCATION:	University of Windsor, Windsor, Ontario, Canada M.A.Sc Electrical and Computer Engineering 2012 American International University-Bangladesh, Dhaka, Bangladesh B.Sc Electrical and Electronic Engineering 2009 Shiddeshwari Girls' College, Dhaka, Bangladesh 2005 Khilgaon Girls' High School, Dhaka, Bangladesh 2003