

**MULTI-SCALE BINARY SIMILARITY
A LOCAL BINARY PATTERN VARIANT FOR
FACE RECOGNITION**



A Thesis

by

Ahmet Tavlı

Submitted to the
Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in the
Department of Computer Science

Özyeğin University
August 2018

Copyright © 2018 by Ahmet Tavlı

**MULTI-SCALE BINARY SIMILARITY
A LOCAL BINARY PATTERN VARIANT FOR
FACE RECOGNITION**

Approved by:

Assistant Professor M. Furkan Kırac,
Advisor
Department of Computer Science
Özyeğin University

Associate Professor Ali Fuat Alkaya,
Jury Member
Department of Computer Science
Özyeğin University

Associate Professor Hürevren Kılıç,
Coadvisor
Department of Computer Engineering
Doğuş University

Associate Professor Murat Şensoy,
Jury Member
Department of Computer Science
Özyeğin University

Date Approved: 1 August 2018

Assistant Professor Reyhan Aydoğan,
Jury Member
Department of Computer Science
Özyeğin University



To my beloved parents

ABSTRACT

Face recognition problem were studied for more than four-decade, and many descriptors and neural network architectures have been proposed since then. The aim is simple, extract features from the same subjects for training and test face image sets, if the proposed method was accurate, the extracted features categorized under the same label. However, the problem starts with the illumination effect on the images; the illumination effect may cause the extracted features for the same subject to be classified with the different labels. Therefore, illumination and other environmental impacts should be removed for accurate classification. One solution for eliminating environmental effect is using Local Binary Pattern (LBP) descriptor. LBP is an illumination invariant, computationally simple, and highly discriminative visual descriptor. Therefore, LBP based descriptors have been developing for more than a two-decade for solving face recognition problem. LBP's computationally simple property make it applicable to different types of computer vision problems, also there are many examples of LBP variants either achieved state-of-the-art results in a particular application or complementary to the LBP. Having been inspired from the results, in this thesis, an LBP variant descriptor, Multi-scale Binary Similarity approach is proposed. MSBS encodes face image characteristic by analyzing the pixel relationships in selected components. The encoded features of the MSBS trained with Support Vector Machines (SVM) and tested with AT&T, Extended Yale B, Georgia Tech and MNIST datasets. The results show that MSBS outperforms most of the proposed approaches in the literature.

ÖZETÇE

Yüz tanıma problemi üzerine kırk yıldan fazla bir süredir çalışılmaktadır ve bu problemi çözmek için birçok algoritma ve sinir ağ yapısı önerilmiştir. Yüz tanımadaki amaç oldukça basittir, eğer önerilen metod, aynı kişiye ait eğitim ve test setlerinden alınmış örneklerden çıkarılan özellikleri aynı etiket altında kategorize edebiliyorsa, bu o algoritmanın doğruluğunu gösterir. Öte yandan problem ise resimler üzerindeki ışıklandırma efekti ile başlar. Işıklandırma efekti, aynı kişiye ait iki farklı örnekten çıkarılmış özellikleri farklı etiket altında kategorize edilmesine, bu resimleri farklı kişilere aitmiş gibi gösterilmesine sebep olabilir. Bu yüzden doğru sınıflandırma yapabilmek için, ışıklandırma ve diğer çevresel etkiler ortadan kaldırılmalıdır. Çevresel etkilerin ortadan kaldırılması için bir çözüm yerel ikili örüntü (LBP) algoritmasını kullanmaktır. LBP ışıklandırma efektinden etkilenmeyen, kolay hesaplanabilen, örneklerden çıkarılan özellikleri başarılı bir şekilde ayırt edebilen bir görsel anahtar nokta tanımlayıcıdır. Bu yüzden LBP temelli tanımlayıcılar yirmi yılı aşkın süredir, yüz tanıma problemini çözmek için geliştirilmektedir. LBP'nin kolay hesaplanabilme özelliği onun farklı problemlere uygulanabilir olmasını sağlamıştır, hatta LBP türevlerinin farklı uygulamalarda en iyi sonucu aldığı veya LBP ile birlikte kullanılarak, LBP'nin sınıflandırma doğruluğunu artırdığı çok örnek vardır. Bu sonuçlardan esinlenerek, bu tezde, bir LBP varyantı tanımlayıcı olan Çok Ölçekli İkili Benzerlik algoritması (ÇÖİB) önerilmiştir. ÇÖİB tanımlayıcı algoritması, yüz resmindeki karakteristiği, seçilen alanlar içerisindeki, pikseller arasındaki ilişkiyi kodlayarak ortaya koyar. ÇÖİB'in çıkardığı özellikler, Destek Vektör Makinesi (SVM) ile eğitilmiştir ve AT&T, Extended Yale B, Georgia Tech verisetleri ile test edilmiştir. Sonuç olarak ÇÖİB algoritması, literatürde önerilmiş yaklaşımların çoğundan daha iyi sınıflandırmıştır.

ACKNOWLEDGMENT

During my thesis study, I have had the privilege and honor of working with Dr. Furkan Kır a , his reviews and comments improved my thesis.

I would like to thank Dr. H revren Kılı  for initiating this thesis idea. His support for helping me to complete my thesis is invaluable.

Thanks to my best friend, Tansoy Aksoylar, and Mahir Atmı , Barı   zcan, Okan Tunalı, Jareth Moyo. Tansoy’s suggestion motivated me through my thesis time. Mahir revised and debugged my thesis software patiently. Barı  shared his experience with me. Okan was informed me the valuable references for my thesis. Jareth comments on my journal paper was valuable. Working with them was memorable for me at OzU.

Special thanks to my parents. Their help initiated my research in this university. I am so grateful for my parents’ encouragement and sacrifices during my thesis time. They were supportive at all time. They helped me to go through all challenges without giving up. This work has been done at Vision and Graphics Laboratory at  zyeđin University and supported with funds from Scientific and Technological Research Council of Turkey (TUBITAK) through the ”1007 - Kgys Developing Smart Software” scholarship.

TABLE OF CONTENTS

DEDICATION	iii
ABSTRACT	iv
ÖZETÇE	v
ACKNOWLEDGMENT	vi
LIST OF TABLES	ix
LIST OF FIGURES	x
I INTRODUCTION	1
II PREVIOUS WORK	3
2.1 Local Binary Pattern	3
2.1.1 Texture Spectrum	3
2.1.2 Uniform Patterns	8
2.1.3 Rotationally Invariant Local Binary Pattern	11
2.2 Local Binary Pattern Variants	12
2.2.1 Preprocessing	12
2.2.2 Neighborhood Topology	13
2.3 Convolutional Neural Networks	22
2.4 Visual Descriptors and CNNs	32
III PROPOSED METHOD	33
3.1 Overview	33
3.2 Multi-Scale Binary Similarity	33
3.2.1 Neighborhood Template	39
3.2.2 Neighborhood Template Extraction from Accumulated Score Matrix	40
3.2.3 MSBS Features Extraction	41
3.2.4 Variants of MSBS	43

IV EXPERIMENTS	51
4.0.1 AT&T	51
4.0.2 Extended Yale B	55
4.0.3 Georgia Tech Face Dataset	60
4.0.4 MNIST	70
V CONCLUSION	74
VITA	86



LIST OF TABLES

1	Uniform Patterns Examples	10
2	Comparison of Neighborhood Topology Algorithms	23
3	Summary of LeNet-1, LeNet-4, and LeNet-5	29
4	AT&T SVM Results	52
5	Preprocessing Chain Applied AT&T SVM Results	52
6	Subset Numbers.	57
7	Extended Yale B SVM Linear Kernel Results	61
8	Extended Yale B Polynomial Kernel Results	61
9	Extended Yale B SVM RBF Kernel Results	62
10	Preprocessing Chain Applied Extended Yale B SVM Linear Kernel Results	63
11	Preprocessing Chain Applied Extended Yale B Polynomial Kernel Results	63
12	Preprocessing Chain Applied Extended Yale B SVM RBF Kernel Results	63
13	Recognition rates of different methods on the Extended Yale B dataset	63
14	Georgia Tech SVM Results	64
15	Preprocessing Chain Applied Georgia Tech SVM Results	64
16	MNIST SVM Results	72

LIST OF FIGURES

1	A simple Co-occurrence matrix calculation demonstration.	4
2	A simple Texture Unit Calculation.	6
3	The Original LBP Calculation.	8
4	Texture Spectrum and LBP Histograms comparison. (Brodatz Images and Texture Spectrum Histograms adapted from [15])	9
5	Local Primitives detected by the LBP (Figure adapted from [1]).	10
6	Rotated Image effect on obtaining the pattern (Figure adapted from [1]).	11
7	The difference between Original and Rotated Image Patterns.	11
8	The steps of Preprocessing Chain	12
9	The effect of Preprocessing Chain on Extended Yale B database	13
10	ELBP Demonstration. A: long axis of the ellipse; B: the short axis of the ellipse; m: number of neighboring pixels. From selected center pixel, A and B distance away, m neighbors are selected.	14
11	The different Neighborhood Topologies (Figure adapted from [9]): (a) circle; (b) ellipse; (c) parabola; (d) hyperbola; (e) Archimedean spiral	14
12	Quinary Pattern Example.	15
13	LLBP Calculation Example.	16
14	TPLBP code with $\alpha=2$, $S=8$, $\omega=3$	17
15	The difference between TPLBP and FPLBP (Figure adapted from [6]).	19
16	Example of Sampling the Neighbor Pixels with radius 1 (Figure adapted from [38]).	21
17	RDLBP Approach (Left), ADLBP (Right) (Figure adapted from [38]).	21
18	Neocognitron Network Model	23
19	Logistic Regression Cost Function	24
20	The Difference of Finding the Global Minimum in non-convex and convex functions	25
21	The Architecture of LeNet5	26
22	The Result of the C1 Convolutional Layer (Figure adapted from [44])	27

23	LeNet5 recognition of visual patterns form unusual, distorted and noisy images (Figure adapted from [27])	28
24	The Architecture of LeNet1	29
25	Commonly used activation functions, from left to right, Sigmoid ($y = \frac{1}{1+e^{-x}}$), Tanh ($y = \frac{e^x - e^{-x}}{e^x + e^{-x}}$) and ReLU ($y = \max(0, x)$)	31
26	Window size effect on Max and Average Pooling	31
27	The difference between traditional (left) and MSBS(right) approach for dividing images.	37
28	Major Patch decomposed into several minor patches	38
29	Example Score Matrix Calculation	40
30	Selecting Coordinates from Score Matrix	41
31	Major Patch Initialization	41
32	Pivot Patch Initialization	42
33	Minor Patch Initialization	42
34	MSBS Formulation	43
35	Creating MSBS Histogram from the Feature Matrix	44
36	AT&T Training Examples	51
37	AT&T Test Examples	51
38	Preprocessing Chain Applied AT&T Training Examples	51
39	Preprocessing Chain Applied AT&T Test Examples	51
40	Comparing MSBS Results with other descriptors using AT&T dataset	53
41	Comparing AT& T Kernel Results	56
42	The Effect of Azimuth Degrees on Image, elevation degree is 0°	56
43	The Effect of Elevation Degrees on Image, azimuth degree is 0°	57
44	Azimuth and Elevation Degrees high Images	57
45	First 11 Class Ambient Images of Extended Yale B	57
46	Examples of Severe Illumination Images in Extended Yale B dataset	58
47	Extended Yale B Subset 1 Examples	59
48	Extended Yale B Preprocessing Chain Applied Subset 1	59
49	Extended Yale B Subset 2 Examples	60

50	Extended Yale B Preprocessing Chain Applied Subset 2	60
51	Extended Yale B Subset 3 Examples	60
52	Extended Yale B Preprocessing Chain Applied Subset 3	60
53	Extended Yale B Subset 4 Examples	61
54	Extended Yale B Preprocessing Chain Applied Subset 4	61
55	Extended Yale B Subset 5 Examples	62
56	Extended Yale B Preprocessing Chain Applied Subset 5	62
57	Comparing Extended Yale B Subset Means	64
58	Comparing Extended Yale B Test2 Kernel Results	65
59	Comparing Extended Yale B Subset 3 Kernel Results	66
60	Comparing Extended Yale B Test4 Kernel Results	67
61	Comparing Extended Yale B Subset5 Kernel Results	68
62	Georgia Tech Face Recognition Training Examples	69
63	Georgia Tech Face Recognition Test Examples	69
64	Preprocessed Chain Applied Georgia Tech Face Recognition Training Examples	69
65	Preprocessed Chain Applied Georgia Tech Face Recognition Test Ex- amples	69
66	Comparing MSBS Results with other descriptors using Georgia Tech dataset	70
67	Comparing Georgia Tech Kernel Results	71
68	MNIST Examples	73
69	Preprocessed Examples	73

CHAPTER I

INTRODUCTION

The way the humans act in the daily life is based on perceiving and interpreting the events. We can recognize faces, read handwritten characters, identify foods, etc.. While we are accomplishing these actions, we analyze the information received from the environment through our vision. At least 50% of human brain is dedicated to vision [1]. Therefore, vision is crucial for people. Vision is also essential for machines. Computer vision helps us to track multiple objects, detect, recognize or classify them; it makes human-computer interaction more useful. Computer Vision capabilities can be summarized with Human Behavior Analysis (HBA) [2]. HBA consists of two parts, models, and analysis. The second part is divided into pattern recognition, machine learning, and statistics. Pattern Recognition is the act of analyzing and categorizing raw data [3].

Pattern Recognition is the machine imitation for what most humans do in daily life, perceiving the distinguishing characteristic of raw data from the environment and interpreting obtained feature for classification. Pattern Recognition consists of four-steps, taking raw data, preprocessing, feature extraction and classification [3]. The first step is receiving input data, and it can be an image or integer matrix. The second phase is preprocessing. The input data may contain unwanted artifacts, to remove these effects, multiple filters are applied. The third step is feature extraction, finding the distinguishing characteristic or pattern from input data. The fourth step is classification; extracted features are categorized due to their unique properties. If first three steps are effectively accomplished, possible highest classification accuracy can be achieved. Otherwise, the selected features must be changed with new ones, until

highest accuracy is achieved. This can be an exhaustive long iterative loop. Therefore, feature selection is essential for classification accuracy. One possible solution for selecting discriminative features is preprocessing the input data. For this reason, preprocessing is considered to be useful [1] and preprocessing algorithm aims to increase classification or detection accuracy. Preprocessing can be a chain of multiple filters [4] or a descriptor [5]. The preprocessing algorithm can be designed for a particular application, or descriptor. There are nine LBP variant development directions cited in the literature, those are preprocessing, neighborhood topology, thresholding & encoding, multiscale analysis, handling rotation, handling color, complementary descriptors, feature selection and learning, and also the other methods inspired by LBP [1]. In this thesis, we focused on preprocessing and neighborhood topology directions by using preprocessing chain [4], which proved to be an effective preprocessing method and LBP descriptor with a broad range of application areas, e.g., face analysis [6], face recognition [7], handwritten digit recognition [8], bioimaging techniques, such as biopsied organ tissue and blood cell phenotype classification [9], radiological techniques such as the magnetic resonance imaging [10].

We proposed a directly applicable, visual descriptor, called Multi-scale Binary Similarity (MSBS), for face recognition. MSBS is designed to extract interest points from the input image. The term interest points refer to candidate feature [11]; it can be an edge, corner or spot. The extracted feature descriptor chooses the best matching interest points. In this study, MSBS extracts interest points from the input data, and LBP used to select among interest points. Extracted features were classified using SVM. MSBS was tested with AT&T [12], Extended Yale B [13] and Georgia Tech [14] face recognition datasets.

CHAPTER II

PREVIOUS WORK

2.1 Local Binary Pattern

The original LBP descriptor was first proposed by Ojala et al. in the comparative study of texture measures [15]. The study consisted of LBP method explanation and comparison with other current descriptors. Ojala et al. profoundly influenced and embraced the idea of texture spectrum and modified the texture spectrum and named as LBP. Texture spectrum used three-level encoding structure 0, 1, 2 for the characterization of the image. Texture spectrum checks three conditions, whether the neighbor pixel is greater, equal to or lower than the center pixel, which will result in $3^8 = 6561$ histogram bin. Instead, Ojala et al. suggested the same method with binary encoding structure, which checks two conditions. The first condition is neighbor pixel either greater than or equal or the second condition, lower than the center pixel. The new descriptor was compared using Nearest-Neighbor Classifier with Gray-level difference method, Law's texture measures [16], center-symmetric covariance measures, Complementary feature pairs. The result was, LBP had the lowest error rate. LBP has reduced dimension, and highest classification rate compared to Texture Spectrum.

2.1.1 Texture Spectrum

Texture Spectrum was designed to extract principal features, which believed to be efficient at the characterization of the image. The characterized features are discriminative and higher classification accuracy achieved. The motivation behind the texture spectrum was Wang et al. disbelieved in the co-occurrence matrix because the co-occurrence matrix depends on pixel pair occurrences and background pixels.

The co-occurrence matrix calculation as in the following:

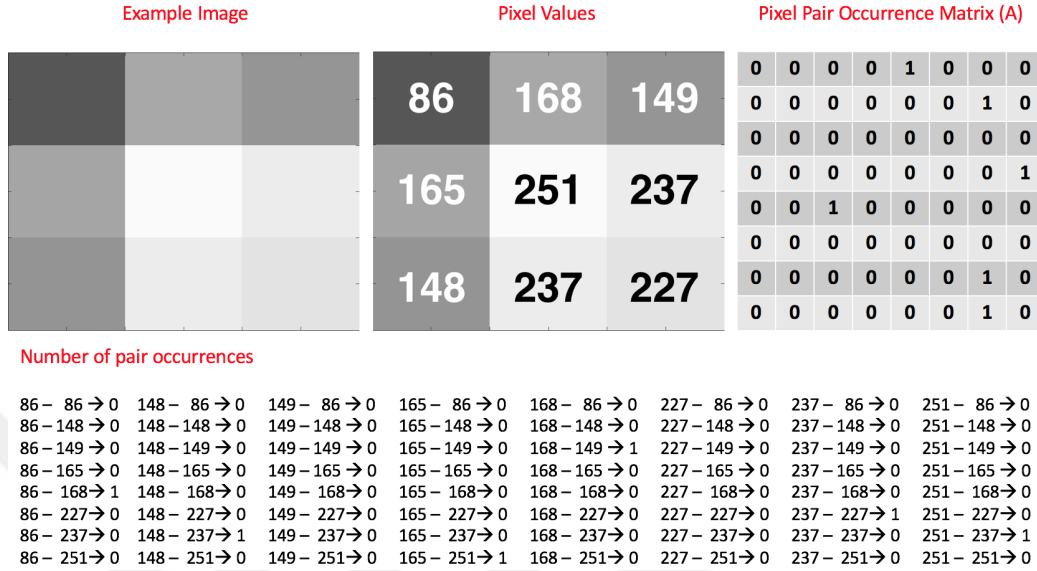


Figure 1: A simple Co-occurrence matrix calculation demonstration.

$$\text{The sum of the Matrix A values} = 6. \quad (1)$$

$$\text{Co-occurrence Matrix (C)} = \frac{1}{6} \text{A}. \quad (2)$$

As it can be interpreted from the above calculations, The co-occurrence matrix indicates the number of pixel values occurs in the selected spatial coordinate, and it was widely used for feature extraction. Wang et al. proposed the term texture spectrum, which did not depend on background pixel and the pair of pixel occurrences. Texture Spectrum consisted of texture units and labeling.

2.1.1.1 Texture Units and Labeling

Texture units (TU) was designed to extract the principal features from the selected region of the input image, which believed to best characterized the local region. As the name implies, texture means characterization of gray-level variation in an image [17].

The term unit referred to the set of eight elements, where each element (E) can only be 0, 1 or 2; therefore texture unit means calculating eight elements from the selected region in the gray-scale image. The term element should not be confused with pixel intensity (V), element value calculated by thresholding pixel values. The reason texture unit consisted of eight elements is because each element obtained from the 3x3 matrix. Texture unit calculation can be represented as in the following:

$$\begin{aligned}
 V &= \{V_0, V_1, \dots, V_8\} \\
 TU &= \{E_0, E_1, \dots, E_8\} \\
 E_i &= \begin{cases} 0, & \text{if } V_i < V_0 \\ 1, & \text{if } V_i = V_0 \\ 2, & \text{if } V_i > V_0 \end{cases} \quad (3)
 \end{aligned}$$

for $i = 1, 2, \dots, 8$, where V_0 is the center pixel intensity and V_i is the current neighbor pixel intensity.

Texture spectrum is the histogram of all texture units obtained from the image. To be more precise, from the selected 3x3 matrix, which consists, a center pixel and eight neighbors, each neighbor, starting from upper left spatial coordinate, thresholded with the center pixel in a clockwise direction to encode texture unit.

The encoded texture unit was converted into a single decimal value named as texture unit number (N_{TU}). Converting texture unit to decimal value is called labeling, using the following formula:

$$N_{TU} = \sum_{i=1}^8 3^{i-1} E_i, \quad N_{TU} \in \{0, 1, 2, \dots, 6560\} \quad (4)$$

For Instance; the calculated TU 00000200 converted as 486 value. The calculated decimal value used as bin value in texture spectrum distribution.

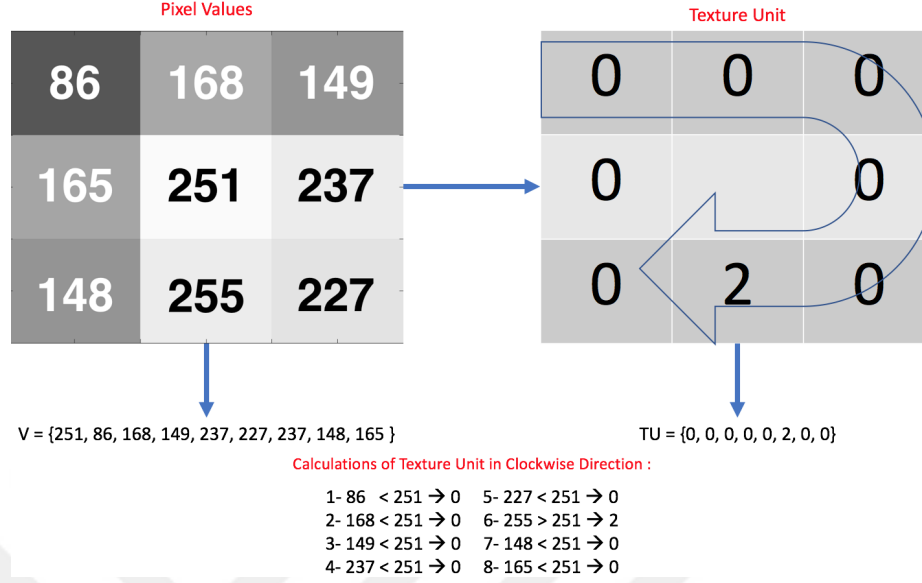


Figure 2: A simple Texture Unit Calculation.

2.1.1.2 Texture Classification

Texture spectrum classification accuracy is calculated as following [18]:

procedure TEXTURE CLASSIFICATION

- (1) For each image in the dataset, randomly select 30x30 pixels regions.
- (2) For each selected pixel region, calculate texture spectrum by overlapping 3x3 matrix inside.
- (3) For each image in the dataset, move sliding window 30x30 pixel inside the image. Each time the sliding window moves, calculate the texture spectrum, then move the sliding window 2 pixels right and down.
- (4) Calculate absolute difference between the calculated texture spectrum of the sliding window and computed texture spectrum of pixel regions ($D(i)$), using the below formula;

$$D(i) = \sum_{j=1}^{6561} |W(j) - S(i, j)|, \quad i = 1, 2, 3, 4 \quad (5)$$

$W(j)$: TU j occurrence value in the sliding window.

$S(i, j)$: TU j occurrence value in the pixel region i .

(5) The minimum D(i) values are categorized together as labels.

end procedure

Texture spectrum was tested on only four images in Brodatz dataset [19]. The average classification accuracy was calculated as 97.5%. However, texture spectrum tested on only four images and the dataset was not divided into training and test set. Therefore the value 97.5% interpreted as the training set accuracy.

2.1.1.3 Texture Spectrum Modification

Ojala et al. proposed each element of TU should be either 0 or 1, instead of 0, 1 or 2. This result reduced bin size 6561 (3^8) to 256 (2^8). The basic LBP operator can be summarized as follows:

$$LBP_{P,R} = \sum_{p=0}^{P-1} S_{LBP}(g_p - g_c) * 2^p \quad (6)$$

$$S_{LBP}(z) = \begin{cases} 1, & z \geq 0 \\ 0, & z < 0. \end{cases} \quad (7)$$

LBP was tested with Brodatz and images were used as in Ohanian and Dubes [20] study. Both images were classified using Nearest-Neighborhood Classification. Each extracted LBP features are distributed in 32 bin histogram, and as a similarity function, Kullback’s minimum cross entropy principle is used [21]. In equation 8, s is the sample, m is model distribution, n is a number of the bin, s_i , and m_i are respective sample and model probability at bin i.

$$D(s : m) = \sum_{i=1}^n s_i \log \frac{s_i}{m_i} \quad (8)$$

The significant modification of texture spectrum is reducing bin size 6561 to 256. One possible reason for this is, as Harwood et al. stated if histograms have too many bins, histograms become sparse and unstable, therefore reducing bins make

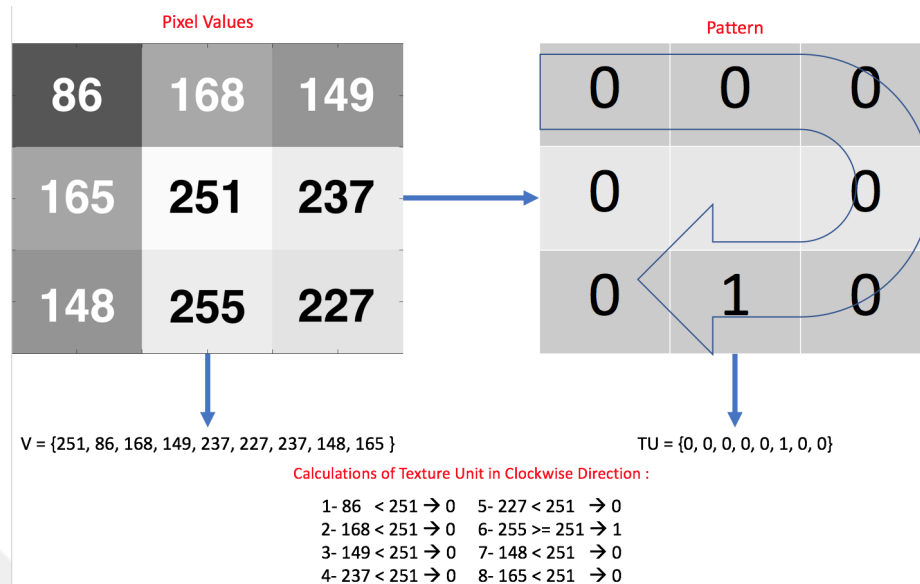


Figure 3: The Original LBP Calculation.

histogram more stable [22]. The minor modifications were made in terminology; the term element interpreted as bin, texture unit interpreted as the pattern. Other principals of texture spectrum were accepted as it is, for instance, each pattern has eight bins, the patterns calculated by thresholding neighbor and center pixels in a clockwise direction.

2.1.2 Uniform Patterns

During the development of LBP, calculated patterns were divided into uniform and non-uniform patterns. The reason for the division was, better results were obtained with uniform patterns. Uniform patterns had more discriminating power compared with non-uniform patterns. Uniform pattern has many advantages; it is less affected by noise, reduced the size of calculated labels and detected local primitives. The term local primitive refers to spot, spot/flat, Line end, edge, and corner [1]. The non-uniform patterns are collected into the single bin size in the histogram.

The detection of uniform pattern was based on a simple rule, the change in bits should be equal to or less than two. For instance, the pattern 01000000 is uniform.

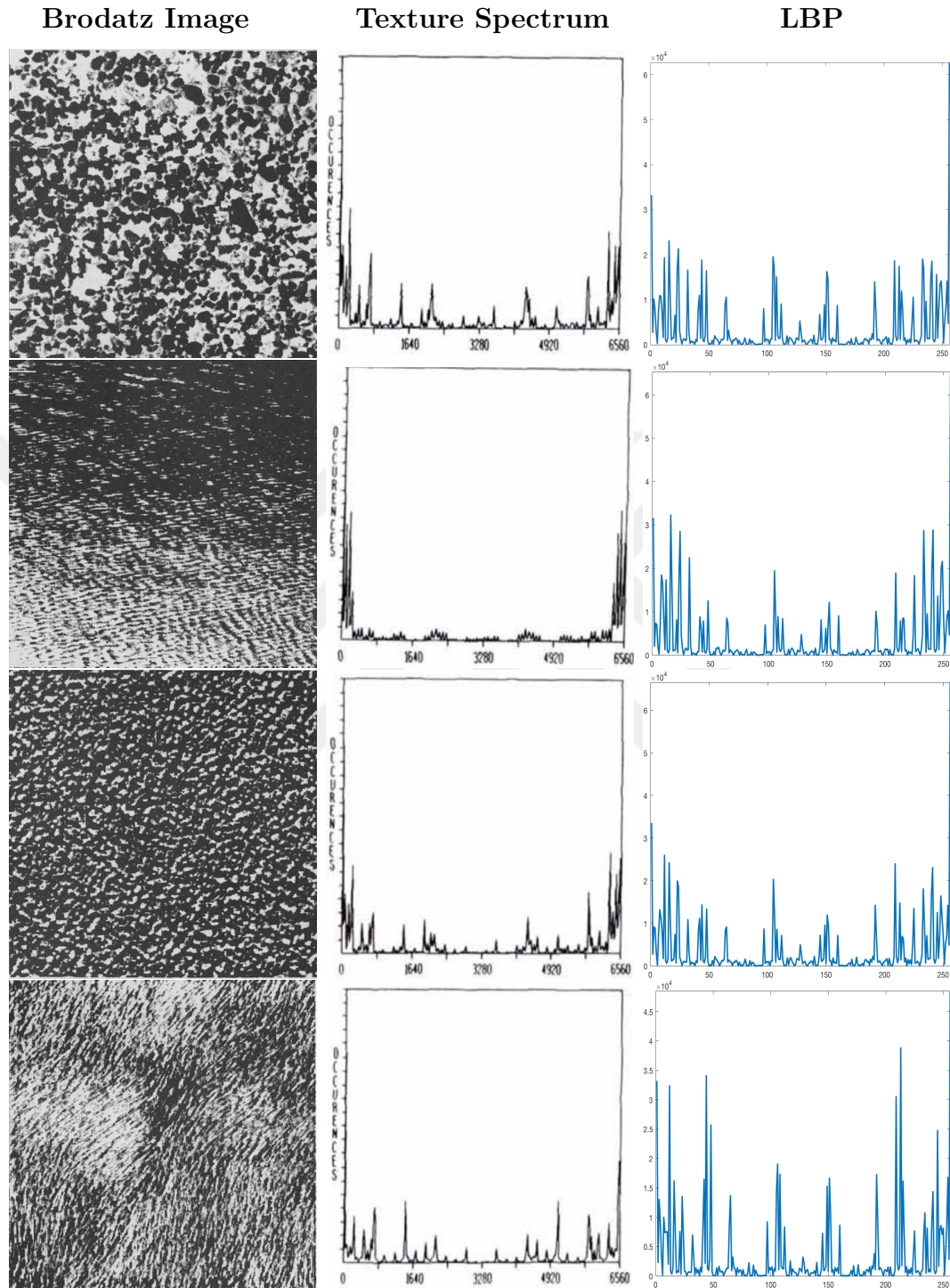


Figure 4: Texture Spectrum and LBP Histograms comparison. (Brodatz Images and Texture Spectrum Histograms adapted from [15])

The reason is the change in bits is two. The first change is from first to second bit $0 \rightarrow 1$ and the second change is from second to third bit $1 \rightarrow 0$. In a gray-scale image,

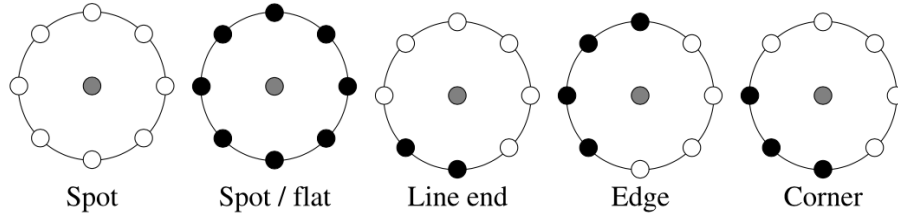
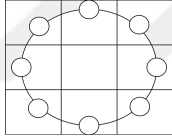
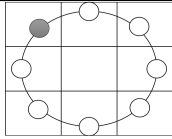
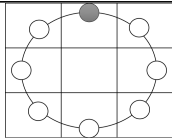
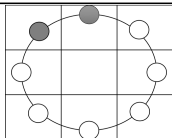
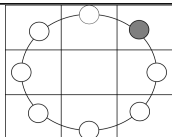
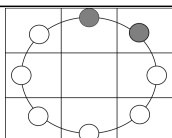


Figure 5: Local Primitives detected by the LBP (Figure adapted from [1]).

there is 58 uniform pattern (Table 1). Here in the above Figure, Spot pattern is 00000000, Spot/flat pattern is 11111111, Line end pattern is 00000110, Edge pattern is 00000110, Corner pattern is 00000111.

Table 1: Uniform Patterns Examples

Pattern Representation	Uniform Pattern	Decimal Value
	00000000	0
	00000001	1
	00000010	2
	00000011	3
	00000100	4
	00000110	6

2.1.3 Rotationally Invariant Local Binary Pattern

The effect of rotating an image changes the order of pixels; therefore the obtained pattern from the selected region is different from the unrotated image.

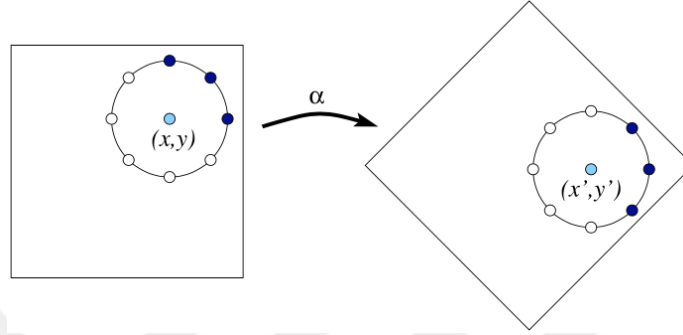


Figure 6: Rotated Image effect on obtaining the pattern (Figure adapted from [1]).

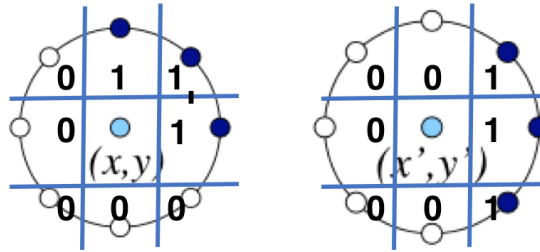


Figure 7: The difference between Original and Rotated Image Patterns.

In the original image, the obtained pattern is 01110000, and the rotated image received pattern is 001110000. Both original and rotated images pattern extracted from same pixel and region, however, the results are not same. Since results are not same, two images are interpreted as different images. Therefore rotational invariant mapping is required [1]. The idea of mapping is to rotate the obtained pattern to its minimum value. The result is both patterns are same. For instance; 01110000 rotated to 001110000, so both patterns have the same value. This rotation equalizes both patterns and can be shown in the formula:

$$LBP_{P,R}^{ri} = \min ROR(LBP_{P,R}, i) \quad (9)$$

2.2 Local Binary Pattern Variants

2.2.1 Preprocessing

The general purpose of preprocessing algorithm is to analyze the given input image pixel by pixel and obtain complementary information. A Preprocessing algorithm can be a filter, combination of filters, or even a descriptor. There are two distinctions between preprocessing and a descriptor. The first distinction is the range; the preprocessing algorithm considers all pixels, where the descriptor considers only selected region inside the image. The second distinction is the purpose, the preprocessing algorithm, aimed to encode image data, where descriptor extracts small and crucial information from the selected region. Some successful preprocessing algorithms have been shown in face recognition, facial appearance, and shape localization fields. In their pioneering work, Zhang et al. [23] combined Gabor filter with LBP, named their method as Local Gabor Binary Pattern Histogram Sequence (LGBPHS). LGBPHS tested on face recognition field. Given input image divided into local region, from each region LGBPHS histograms are extracted. Each histogram is concatenated to create the feature vector. LGBPHS reached the state-of-the-art performance in face recognition. Another preprocessing algorithm is Tan and Triggs preprocessing chain [4]. Preprocessing Chain three main steps are gamma correction, the difference of Gaussian filtering (DoG), masking (optional) and contrast equalization (Figure 8). As seen in Figure 9, Preprocessing Chain is useful for recognition.

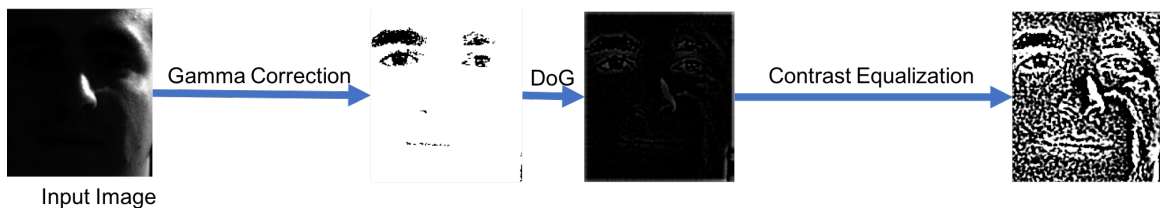


Figure 8: The steps of Preprocessing Chain

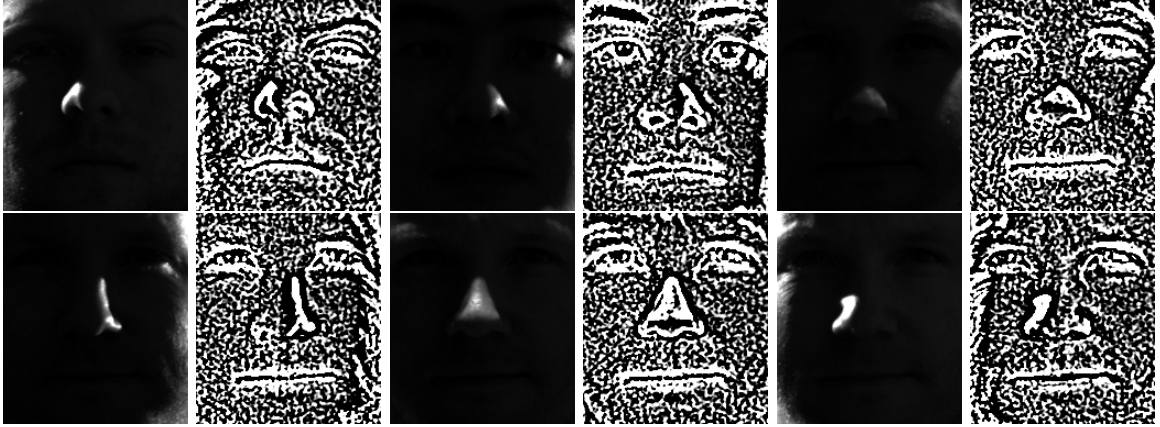


Figure 9: The effect of Preprocessing Chain on Extended Yale B database

2.2.2 Neighborhood Topology

2.2.2.1 Elongated LBP

LBP features are usually extracted from a circular or square neighborhood. As a result, the rotational-invariance is ensured. However, if the rotational-invariance is not needed or depending on the need of the application, neighborhood calculation can be different. For instance, Liao et al. [24] used elliptical neighborhood instead of a circular or square neighborhood. The reason is their study focused on face recognition. In face recognition problem, anisotropic structural information is the only concern. Therefore the best way to increase face recognition accuracy is extracting anisotropic structure from the face. An anisotropic structure can be eyes, mouth, etc.. For this reason, Liao et al. proposed two features, the Elongated LBP (ELBP) and Average Maximum Distance Gradient Magnitude (AMDGM). ELBP extracts anisotropic structure from an elliptical neighborhood, while AMDGM, for each pattern, captures gradient information. There are three parameters for ELBP: Long (A) and short axis (B) of the ellipse and the number of neighboring pixels (m) (Figure 10).

ELBP tested with FERET [26] and ORL [12]. The conclusion was, ELBP with AMDGM performed higher classification accuracy compared to LBP.

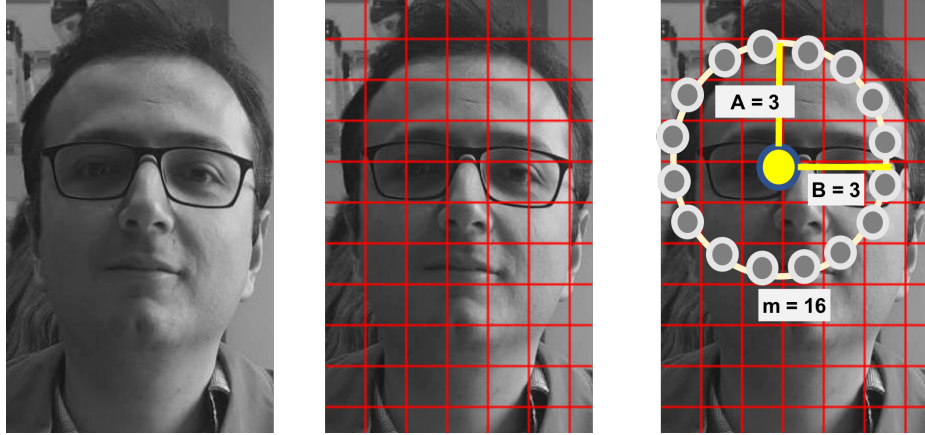


Figure 10: ELBP Demonstration. A: long axis of the ellipse; B: the short axis of the ellipse; m: number of neighboring pixels. From selected center pixel, A and B distance away, m neighbors are selected.

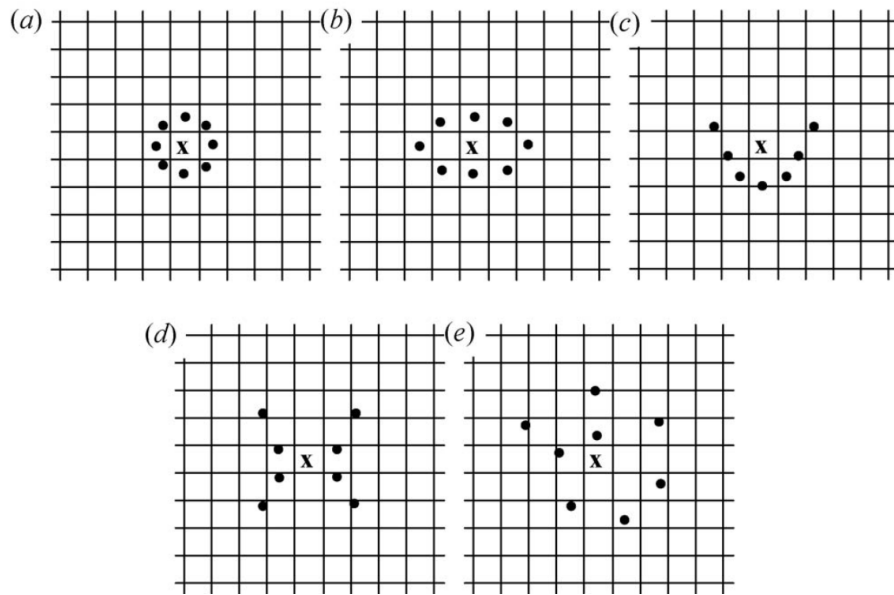


Figure 11: The different Neighborhood Topologies (Figure adapted from [9]): (a) circle; (b) ellipse; (c) parabola; (d) hyperbola; (e) Archimedean spiral

2.2.2.2 Quaternary Pattern

Nanni et al. [9] compared five different neighborhood (Figure 11) on three different datasets for medical image analysis; classification of pain expression (COPE) [28], 2D-HeLa dataset [29], pap smear diagnosis [30]. The term pain expression provides

crucial information for the baby's state. The conclusion was, elliptic neighborhood, encoded with quaternary patterns, performs the best classification for medical image analysis. Unlike binary pattern, the quinary pattern is encoded using five values (-2, -1, 0, 1, 2) and two threshold values (τ_1 , τ_2). The difference (d) calculated using neighbor pixel (ϵ) and center pixel (x). The quaternary pattern is split into four (-2, -1, 1, 2) binary patterns.

$$d = \begin{cases} 2, & v \geq x + \tau_2 \\ 1, & x + \tau_1 \leq v < x + \tau_2 \\ 0, & x - \tau_1 \leq v < x + \tau_1 \\ -1, & x - \tau_2 \leq v < x - \tau_1 \\ -2, & \text{otherwise} \end{cases} \quad (10)$$

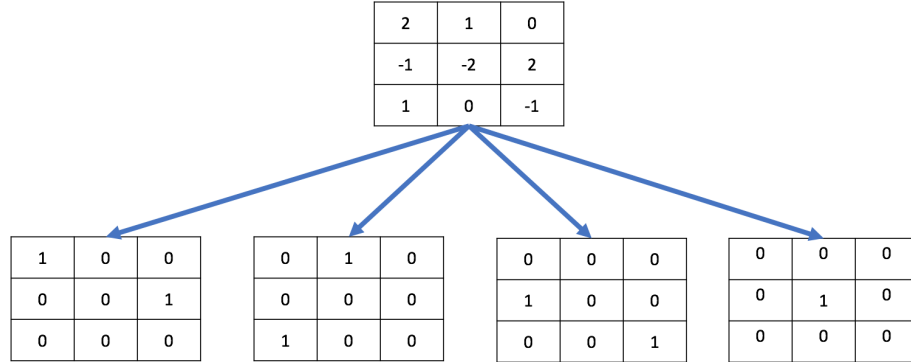


Figure 12: Quinary Pattern Example.

2.2.2.3 Local Line Binary Pattern

Petpon et al. [31] proposed Local Line Binary Pattern (LLBP). LLBP has two differences compared to LBP. The first difference is the neighborhood, LLBP calculated with the horizontal and vertical neighborhood. The second difference is encoding, and if the neighbor pixel is lower than the center pixel, then it is encoded as 1, otherwise

0. This approach is the opposite of LBP encoding. LLBP was tested on FERET and Yale Face Database B [13]. LLBP has higher classification compared to LBP. Equation 11, 12 and 13 show how LLBP is calculated.

$$LLBP_h(N, C) = \sum_{n=1}^{c-1} s(h_n - h_c) \cdot 2^{(c-n-1)} + \sum_{n=c+1}^N s(h_n - h_c) \cdot 2^{(n-c-1)}, \quad (11)$$

$$LLBP_v(N, C) = \sum_{n=1}^{c-1} s(v_n - v_c) \cdot 2^{(c-n-1)} + \sum_{n=c+1}^N s(v_n - v_c) \cdot 2^{(n-c-1)}, \quad (12)$$

$$LLBP_m = \sqrt{LLBP_h^2 + LLBP_v^2} \quad (13)$$



Figure 13: LLBP Calculation Example.

Each encoded bit multiply with its corresponding weights, both in horizontal and vertical directions. Each result is used to calculate the magnitude, which represents the feature of the image. Figure 13 shows how LLBP calculated. First, the image is divided into the grids. Second, The horizontal grid values are used for encoding binary pattern. For Instance, in figure 13, the middle image shows the center and neighbor pixel intensities. The binary pattern can be encoded as in the following; $252 > 232$ and $232 = 232$. Therefore upper neighborhood pattern is 00000011. Same

is also true for lower neighborhoods; $229 < 232$ and $226 < 232$. Therefore the pattern is 00000000. Now the encoded two binary pattern; 00000011 (3) and 00000000 (0) will be the new value of $LLBP_h(N, C)$, which is 3. The calculation of the $LLBP_v(N, C)$ is similar to the $LLBP_h(N, C)$. The encoded patterns are; 00000000 (0) and 00000001 (1), the sum is 1. The magnitude result will be $\sqrt{(3)^2+(1)^2} \approx 3.16$

2.2.2.4 Three-Patch and Four-Patch LBP

Wolf et al. [6] proposed Three-Patch LBP (TPLBP) and Four-Patch LBP (FPLBP) for face analysis. This is a new approach to find complementary information in a given image. Both descriptors find complementary information by finding similarities between neighboring pixels. Both TPLBP and FPLBP was inspired by Center-Symmetric LBP [34]. TPLBP have four parameters; radius (r) from a selected patch, r pixel distance selected circularly. The number of patches (S) is uniformly distributed in a ring, r distance. The size of the patch (ω), and the distance between two patches (α). TPLBP encoding is shown in formula 14.

$$TPLBP_{r,S,\omega,\alpha}(p) = \sum_i^S f(d(C_i, C_p) - d(C_{i+\alpha \bmod S}, C_p))2^i \quad (14)$$

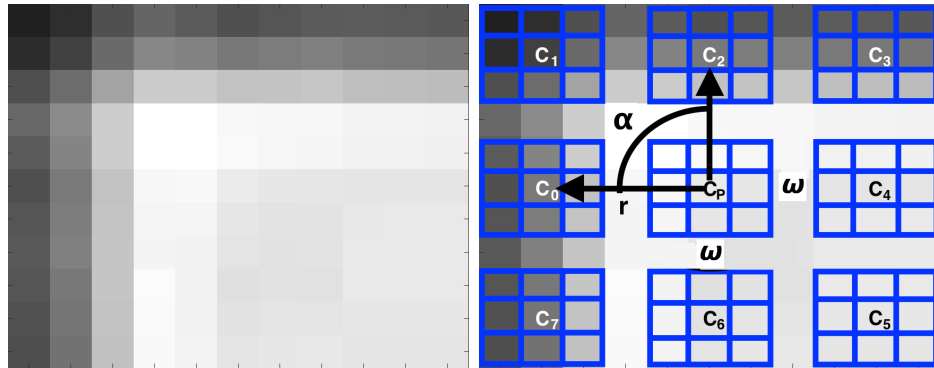


Figure 14: TPLBP code with $\alpha=2$, $S=8$, $\omega=3$.

The above TPLBP can be calculated as follows:

$$\begin{aligned}
\text{TPLBP}_{\tau,8,3,2}(p) = & f(d(C_0, C_p) - d(C_2, C_p))2^0 + \\
& f(d(C_1, C_p) - d(C_3, C_p))2^1 + \\
& f(d(C_2, C_p) - d(C_4, C_p))2^2 + \\
& f(d(C_3, C_p) - d(C_5, C_p))2^3 + \\
& f(d(C_4, C_p) - d(C_6, C_p))2^4 + \\
& f(d(C_5, C_p) - d(C_7, C_p))2^5 + \\
& f(d(C_6, C_p) - d(C_0, C_p))2^6 + \\
& f(d(C_7, C_p) - d(C_1, C_p))2^7
\end{aligned}$$

$$f(x) = \begin{cases} 1, & \text{if } x \geq \tau \\ 0, & \text{if } x < \tau \end{cases} \quad (15)$$

It can be interpreted from the Figure 14; There are nine patches distributed in a given image. The center patch and eight neighbor patches. Each time three patch selected for encoding, the center patch and two neighbor patches. The first neighbor patch is the left patch of center patch. The second neighbor patch is α patch away from the first neighbor patch. Each neighbor center pixel thresholded with central patch center pixel. If the result is bigger than defined τ value 1, otherwise 0 (Equation 15). In FPLBP there are two center patches; therefore two radii are set. FPLBP calculated as follow (Equation 16):

$$\text{FPLBP}_{r_1, r_2, S, \omega, \alpha}(p) = \sum_i^{S/2} f(d(C_{1i}, C_{2, i+\alpha \bmod S}) - d(C_{1, i+S/2}, C_{2, i+S/2+\alpha \bmod S}))2^i \quad (16)$$

Both TPLBP and FPLBP tested with Labeled Faces in the Wild [32]. When TPLBP and FPLBP combined with each other, performs better than LBP. Figure 15 shows the results of TPLBP and FPLBP.

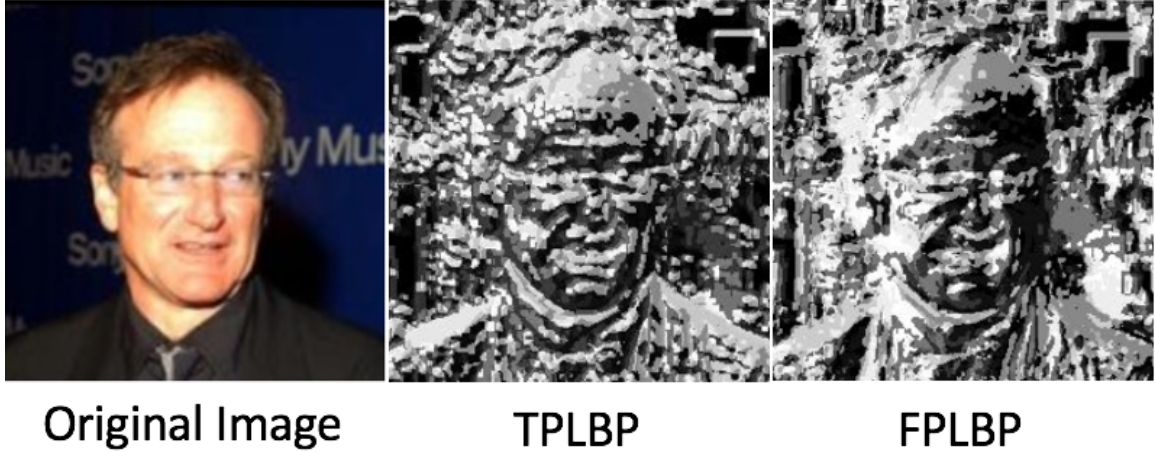


Figure 15: The difference between TPLBP and FPLBP (Figure adapted from [6]).

2.2.2.5 Angular and Radial Difference Based LBP

Liu et al. [35] proposed angular-difference based local binary pattern (ADLBP) and radial-difference based local binary pattern (RDLBP). Both descriptor thresholds the neighbor pixels within each other, instead of thresholding each neighbor pixel with the center pixel. The motivation behind both ADLBP and RDLBP was to find the complementary information using the neighboring pixels difference. Both descriptors starts with dividing images into patches. From each patch, first center pixel spatial coordinate found, based on the center pixel spatial coordinate, neighbor pixels uniformly distributed in the ring of radius r . As it can be interpreted from figure 16, the spatial coordinate of the center pixel of 7×7 image is $(4, 4)$. The neighbor pixels uniformly distributed around the $(4, 4)$ spatial coordinate with radius 1. They divided their approach into two parts; radius and angular. The radius approach was thresholding neighborhood tuples in the same radii. The angular approach was thresholding neighborhood tuples in the same angle (Figure 17). For each neighbor tuple in the clockwise direction, the difference was calculated. However, there is a constraint in neighborhood selection. At each level, for instance, in figure 16, red, green and blue colors represents level, only multiples of 8 neighbors are selected (Equation 17). This

constraint represented for ADLBP (Equation 18), and RDLBP (Equation 19). In both equation 18 and 19, the parameters r , δ , q are same. The parameter r radius specifies where the neighbor pixels starts and the distance between the circular neighbor pixels. The parameter δ specifies how many neighbor pixels will be distributed in r . The parameter q specifies how many neighbor pixels will be selected. The binary pattern evaluation for both descriptors is nearly same, similar to LBP, if the thresholded value is greater or equal to 0, encoded as 1, otherwise 0. Both descriptors equations is nearly same, in equations 20 and 21. The 'S' represents the sign, Liu et al. symbolized the thresholding calculation as $_S$ symbol. The only difference is ADLBP $_S$, calculate the difference of the neighbor pixels in the same angular distance, where RDLBP $_S$ calculate the difference of the neighbor pixels in the same radius. After binary pattern obtained, same as LBP, if the thresholded value is greater than or equal to 0, encoded as 1 or 0 (Equation 22). The encoded pattern converted into decimal value. Decimal values were used to obtain the histogram. From each patch, histograms were obtained. Finally obtained histograms were used for classification. Both ADLBP and RDLBP was the result of the two extensive research [35], [38]. In their previous research [38], Liu et al. stated the LBP's limitations. One of the major limitations is LBP's assumption on thresholding neighbor pixels with the center pixel is independent of the center pixel. However, the independence is not guaranteed [39]. The question aroused from this assumption; Could LBP miss textural information? Therefore they considered the thresholding of neighbor pixels as well. As a result, they came up with ADLBP and RDLBP.

$$x_{r,8q} = [x_{r,8q,0}, \dots, x_{r,8q,8q-1}]^T \quad (17)$$

$$\phi_{r,\delta,q}^{AngSign} = [\phi_{r,\delta,q,0}^{AngSign}, \dots, \phi_{r,\delta,q,8q-1}^{AngSign}]^T \quad (18)$$

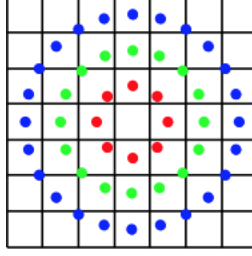


Figure 16: Example of Sampling the Neighbor Pixels with radius 1 (Figure adapted from [38]).

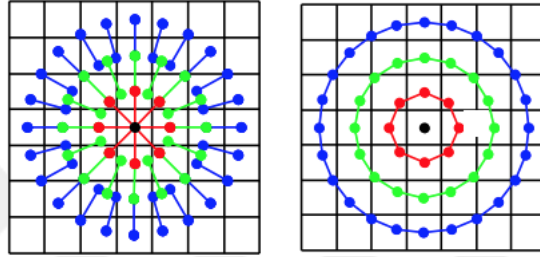


Figure 17: RDLBP Approach (Left), ADLBP (Right) (Figure adapted from [38]).

$$\underline{\Delta}_{r,\delta,q}^{Rad} = [\Delta_{r,\delta,q,0}^{Rad}, \dots, \Delta_{r,\delta,q,8q-1}^{Rad}]^T \quad (19)$$

$$ADLBP_S = \sum_{n=0}^{7} s(\phi_{r,\delta,q,n}^{AngSign}) 2^n \quad (20)$$

$$RDLBP_S = \sum_{n=0}^{7} s(\phi_{r,\delta,q,n}^{RadSign}) 2^n \quad (21)$$

$$s(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \quad (22)$$

2.2.2.6 Comparing Previous Studies

In this section, the summary of the previous neighborhood topology algorithm are compared (Table 2). The illumination invariance is probably the most striking component of LBP. Therefore the first component compared was illumination invariant. The most descriptor influenced by LBP already illumination invariant. The second component compared was being computationally simple. The proposed algorithm may reach the highest accuracy, however, how the highest accuracy reached? If the answer is, with a simple and efficient algorithm, then the computationally simple component checked. The third component compared was flexible neighborhood topology. If the neighborhood selection is parametric, then this component was checked. For instance, ELBP neighborhood selection based on two parameter, where LLBP neighborhood selection was static. The fourth component is being rotational invariant. Generally, in face recognition application, the rotational invariant is not a necessity. Therefore most descriptor, focused on other components. If the proposed algorithm considered rotational invariant, then this component was checked. The fifth component is whether the proposed algorithm was used with another descriptor. Since LBP is very flexible, it is possible to use with other image descriptors. If the proposed algorithm used with another algorithm, then this component was checked. The sixth component is highly discriminative. If the proposed algorithm, surpasses the LBP accuracy, or multiple algorithms accuracy, then this component was checked.

2.3 Convolutional Neural Networks

Convolutional Neural Networks often abbreviated as ConvNet or CNNs, is part of the neural network and common form of Deep Neural Networks [40], which was inspired by animal visual cortex organization [41]. The discovery of animal visual cortex organization started with Hubel and Wiesel experiments on monkeys. [42]. Hubel and Wiesel found light was detected by the receptive fields of visual cortex in animals.

Table 2: Comparison of Neighborhood Topology Algorithms

	ELBP	LLBP	TPLBP	FPLBP	RDLBP	ADLBP
Illumination Invariant	✓	✓	✓	✓	✓	✓
Computationally Simple		✓				
Flexible Neighborhood Topology	✓		✓	✓	✓	✓
Rotational Invariant		✓				
Used with Other Descriptor	✓					
Highly Discriminative	✓	✓			✓	✓
Application Area	Face Recognition	Face Recognition	Face Analysis	Face Analysis	Face Recognition	Face Recognition
Dataset	FERET [26] AT&T [12]	Yale Face Database B [13] FERET [26]	LFW [32]	LFW [32]	Extended Yale B [13] CAS-PEAL-R1 [33]	Extended Yale B [13] CAS-PEAL-R1 [33]

Visual cortex has the following structure: Lateral Geniculate Body (LGB) → simple cells → complex cells → lower order hypercomplex cells → higher order hypercomplex cells. Fukushima [43] converted this discovery to the architecture, named Neocognitron (Figure 18).

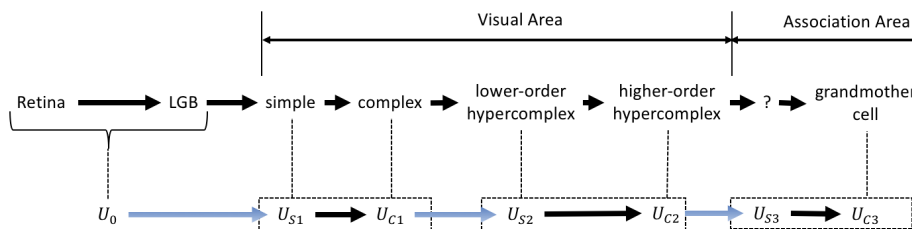


Figure 18: Neocognitron Network Model

Neocognitron has three layers; each layer consisted of the simple cell (U_S) and complex cell (U_C). From each layer, features are extracted and passed to the next layer. In Figure 18 there is a '?' in the architecture. The reason for that is Hubel and Wiesel did not tell what kind of cell exists after higher order higher complex cell. Therefore Fukushima inserted question mark instead removing the part from architecture. Neocognitron was designed for the improvement of the unsupervised learning, as Fukushima defined as "self-organized by learning without a teacher."

Unfortunately, the time neocognitron published, there wasn't enough data to test the neocognitron. Therefore, Fukushima was tested on digit images, and neocognitron successfully extracted the features and classified based on its extracted features. Although Fukushima described neocognitron as a hypothesis for visual pattern recognition, it was accepted as the predecessor of CNN [44]. Successful classification using neural network required preprocessing of the dataset as a preliminary step. For instance, acquisition, binarization, and segmentation [45]. However, preprocessing a large-scale dataset can be very hard due to its size. Therefore Lecun et al. [46] proposed neural network with backpropagation. Backpropagation learning aim was to minimize the measure between output vector to the desired vector by adjusting weight with repetitive steps [47].

$$J(\Theta) = -\frac{1}{m} \sum_{i=1}^m \sum_{k=1}^K \left[y_k^{(i)} \log((h_{\Theta}(x^{(i)}))_k) + (1 - y_k^{(i)}) \log(1 - (h_{\Theta}(x^{(i)}))_k) \right] + \frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (\Theta_{j,i}^{(l)})^2 \quad (23)$$

The neural network cost function ($J(\Theta)$) is the generalization of the logistic regression cost function [48]. If we look at figure 19, we can analyze $J(\Theta)$ better.

$$J(\theta) = \underbrace{-\frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \log h_{\theta}(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)})) \right]}_{\text{Cost Function}} + \underbrace{\frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2}_{\text{Regularization Term}}$$

Figure 19: Logistic Regression Cost Function

The logistic regression cost function $J(\theta)$ is similar to the linear regression optimization function (Equation 24). The idea is $h_{\theta}(x^{(i)})$ represents the prediction of the i 'th sample and $y^{(i)}$ represents the actual value of the i 'th sample. Therefore, if the result of $h_{\theta}(x^{(i)}) - y^{(i)}$ is close to zero, this means predicted value is very close to the actual value. When we take the square, we calculate the euclidian distance between

two samples. The coefficient $\frac{1}{2m}$ is to make calculation easier [49].

$$\text{Linear Regression Optimization Function} = \min \frac{1}{2m} \sum_i^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 \quad (24)$$

Now, we can analyze the $J(\theta)$. The first difference is the logarithm of $h_{\theta}(x^{(i)}) - y^{(i)}$ was taken. The reason is we $h_{\theta}(x)$ is a non-linear function, and if we take the square of a non-linear function, a non-convex cost function appears. Gradient descent function cannot guarantee to find the global minimum in a non-convex function. Therefore we take the logarithm of the result, to guarantee gradient descent for finding the global minimum (Figure 20). The second difference is regularization term $\frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$. The regularization term adds up the weights of other eliminated features. For instance, defining an object requires 1000 parameter. The cost function is similar to $J(\theta) = \theta_0 + \theta_1 x + \dots + \theta_{999} x^{999}$. However, some features are less important. Therefore instead of multiplying with 1000 parameter, adding a regularization term to the cost function is more feasible and gives the approximate result when the less crucial features added. The summation formula of regularization term starts from 1 ends with n. The reason for this, the bias term θ_0 is not added. The neural network $J(\Theta)$ is similar to the logistic regression $J(\theta)$. The only difference is for each neuron in the layer, $J(\theta)$ calculated. Therefore if we have k neuron in the single layer, we obtain k output.

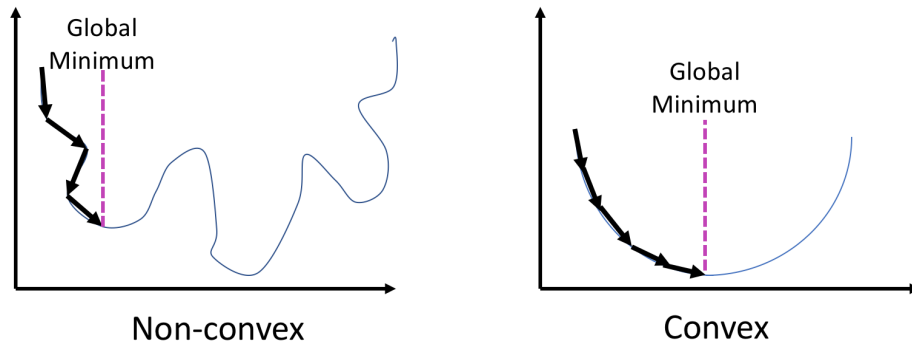


Figure 20: The Difference of Finding the Global Minimum in non-convex and convex functions

Suppose we have L layer in our network with m number of training samples. First, the activation function of each neuron was calculated during forward propagation. The reason is we have to observe how our actual result is close to correct output. Therefore we calculate the difference of last layer result from the correct output. If the output is close to zero, this can be interpreted as the activation function suitable for classifying the current label. Otherwise, activation function needs to be updated, until the actual result is close to the correct output. LeCun et al. [46] applied the neural network with backpropagation to zip code recognition. They tested neural network with backpropagation on U.S. Mail dataset, and the result was 99% correct classification accuracy. Having inspired by the result, LeCun et al. [27] proposed LeNet-5 (Figure 21).

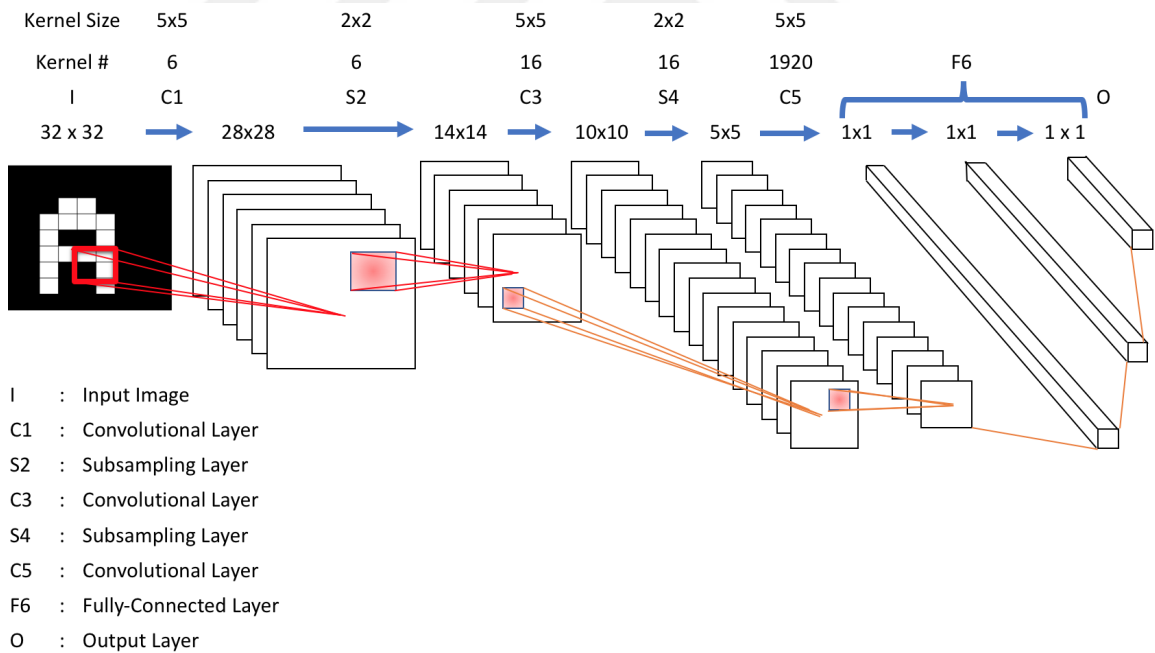


Figure 21: The Architecture of LeNet5

LeNet-5 has seven layers (C1, S2, C3, S4, C5, F6, Output), can be trained with backpropagation and obtain the effective representation of the original image (Figure 23). LeNet-5 consisted of three types of layer, convolutional(C1, C3, C5), pooling

(S2, S4) and fully-connected (F6). The first layer is C1; it is a convolutional layer with six feature maps. Feature maps generate high-level abstraction of the data [40]. The term high-level abstraction of data refers to preserving essential information. The result of C1 shows the learned weights from the example image (Figure 22).

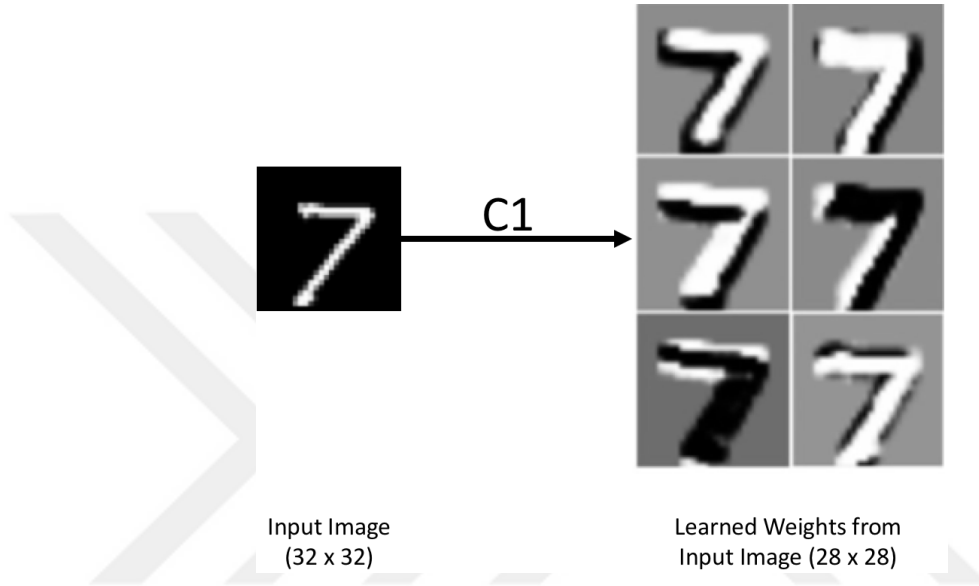


Figure 22: The Result of the C1 Convolutional Layer (Figure adapted from [44])

The output of C1 layer pass to the S2 subsampling layer. S2 is the pooling layer. The pooling layer aim is to make input image reduce resolution, so the image features aligned into the center. Therefore in the subsampling layer, the image size is halved. The result of S2 connected to C3, and after C5 layer the result is connected to fully-connected layer F6. The fully connected layer aims to find the global semantic information by connecting to every single neuron of the last convolutional layer to current layer [44]. Global semantic information refers to finding the unique generalized representation of the current image using the least resource. The output layer was used to classify the output of the fully-connected layer. However, there is some constraint in LeNet-5 architecture. First not every pooling layer output is given input to the convolutional layer. According to LeCun et al. [46], it broke the symmetry of the network and forced to extract different features. Therefore, starting from the

third convolutional layer, they took only first six feature maps. However, the reason for selecting first six feature map is still ambiguous.

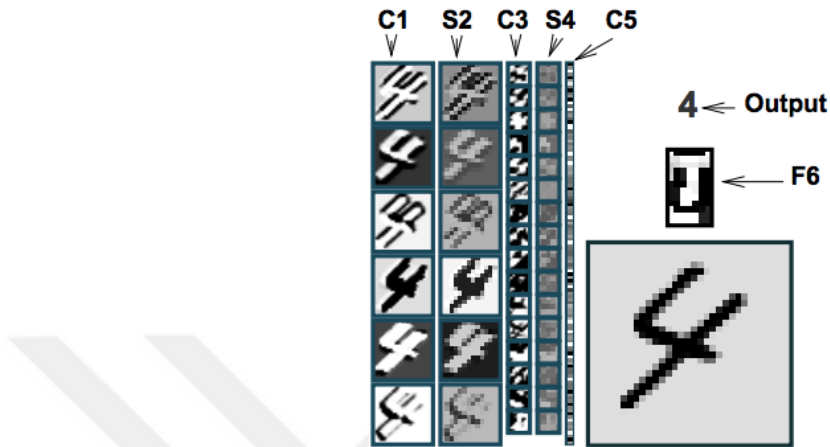


Figure 23: LeNet5 recognition of visual patterns form unusual, distorted and noisy images (Figure adapted from [27])

The second question is why LeNet-5 has seven layers? LeCun et al. [46] had been experimenting on optical character recognition dataset US Postal Service zip codes (USPS)., starting with LeNet-1. They described LeNet-1 as a small network [50]. LeNet-1 achieved 1.7% error rate on USPS. LeNet-1 has six layers, three convolutional, two pooling layer and one output layer (Figure 24).

The success of LeNet-1 motivated LeCun et al. [50] built LeNet-4. LeNet-4 has six layers, similar to LeNet-1, three convolutional, two pooling and one output layer. LeNet-4 achieved 1.1% error rate on USPS. The success of LeNet-4 motivated LeCun et al. [50] LeNet-5, which achieved 0.9% error rate on USPS. The difference between them shown in Table 3. The reason is Lecun et al. [50] discovered as layer and connection size increases, the error rate decreases, resulting in better classification accuracy. Therefore LeNet-5 has seven layers.

LeNet-5 is the first successful application of convolutional neural network on both optical character recognition and digit recognition [27]. Other researchers were interested in more complex areas, such as large-image and video classification. However,

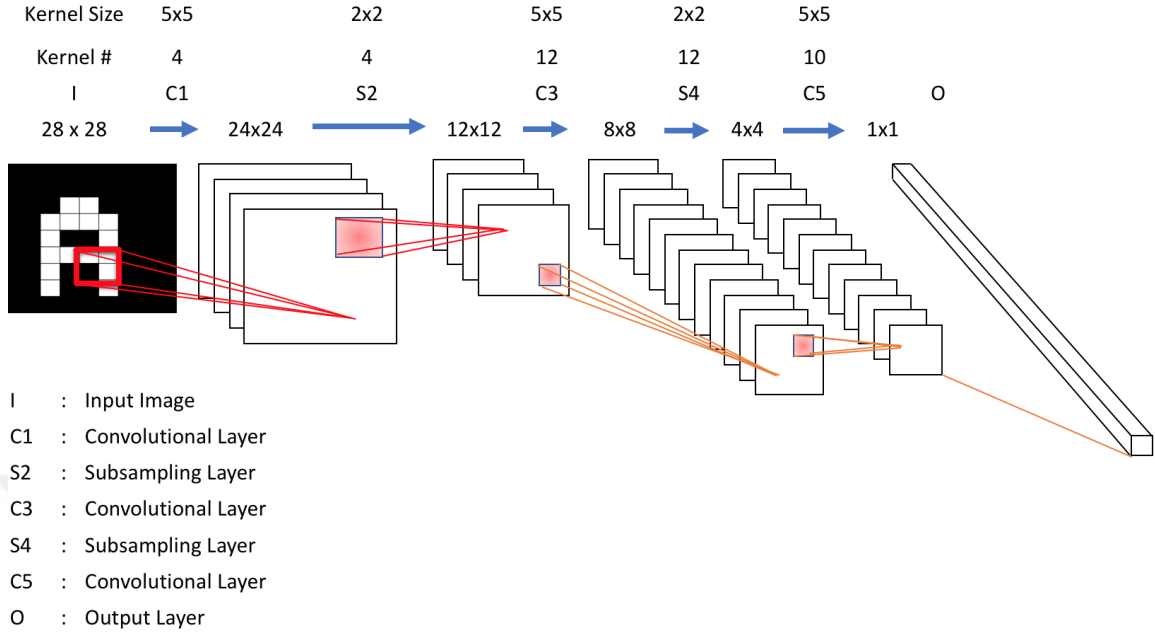


Figure 24: The Architecture of LeNet1

Table 3: Summary of LeNet-1, LeNet-4, and LeNet-5

	LeNet-1	LeNet-4	LeNet-5
Input Image Size	16x16	32 x 32	32 x 32
Total Connection Size		260.000	340.000
Layer Size	6	6	7
Accuracy	98.3%	98.9%	99.1%

due to large training data and computing power, their networks did not perform well [44]. Since 2006, researchers concentrated on difficulties encountered in CNN. One of the researchers' group, Krizhevsky et al. [51] proposed AlexNet. AlexNet is the winner of ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [52]. ILSVRC is computer vision competition, focused on the large dataset of object detection and image classification. AlexNet has some similarity with LeNet-5, the architecture is deeper than LeNet-5. After the proposal of AlexNet, many networks designed to improve the performance of AlexNet. For Instance, ZFNet [53], VGGNet [54], GoogleNet [55] and ResNet [56]. The difference between them is, CNN architectures are getting deeper. For instance, the winner of ILSVRC 2015 ResNet

is 20 times deeper than AlexNet [44]. There are some similar components between each CNN architecture, feature map calculation, activation function, pooling, fully connected and output layers, respectively.

The first component is the feature map calculation. The feature map calculation is selecting a patch inside the image, multiply with the randomly generated weight and added with randomly generated bias. This mathematical operation continues until every pixel inside the image filter. For instance, assume we want to find the feature in the l 'th layers k 'th feature map with spatial coordinate i, j ($z_{i,j,k}^l$). We select an area (patch) in the l 'th layer, represented as $x_{i,j}^l$. Now, random weight and bias values should be generated. The weight value multiplied with selected patch, $w_k^{lT} * x_{i,j}^l$ and bias value added $w_k^{lT} * x_{i,j}^l + b_k^l$. The result will give the feature value of the selected region. The feature value is the unique information obtained from the region. The reason for multiplying with weights is because we want to compare the importance of each filtered region. When we multiply weight with our $x_{i,j}^l$, we show its importance in the current layer and scale the data in the specified interval, for instance, -1 to 1. Therefore multiplication with weights reduce the complexity and make the network easier to train. The reason for adding is because the bias value shifts the activation function to the center.

The second component is the calculation of the activation functions. There are three commonly used activation functions, sigmoid, tanh [57] and linear rectified unit (ReLU) [58] as shown in Figure 25. Activation function detects the nonlinear features from the input feature map.

The third component is pooling. Pooling applied each channel separately, for making the network shift invariant by reducing dimensionality. The result is network learns robust features from the input channel. There are two typical pooling operation, max pooling [59] and average pooling [60]. Max Pooling finds the maximum

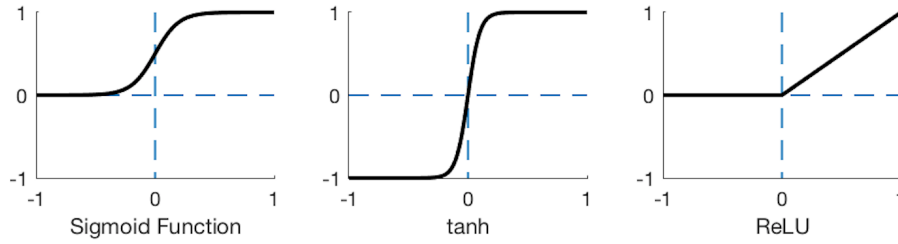


Figure 25: Commonly used activation functions, from left to right, Sigmoid ($y = \frac{1}{1+e^{-x}}$), Tanh ($y = \frac{e^x - e^{-x}}{e^x + e^{-x}}$) and ReLU ($y = \max(0, x)$)

value in the kernel and places it in the center pixel location. Average pooling calculates the mean value in the kernel and places it in the center pixel location. The kernel is a non-overlapping window (Figure 26).

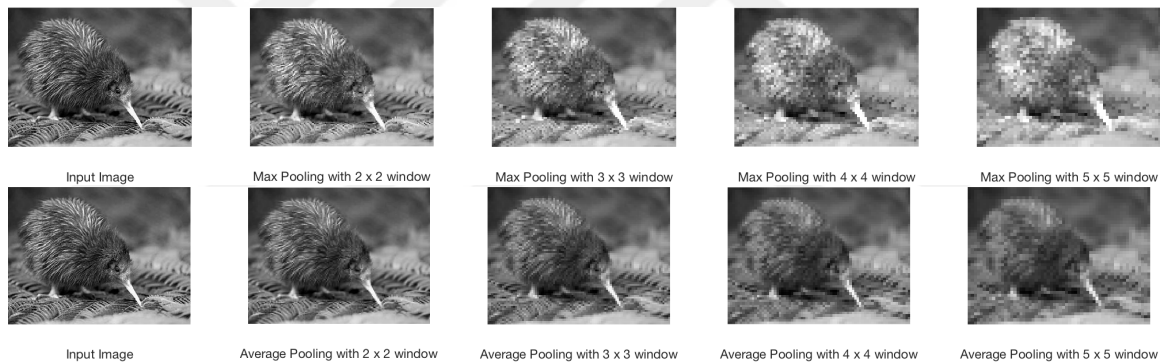


Figure 26: Window size effect on Max and Average Pooling

The fourth component is the fully-connected layer. So far, the network learned every detail from input image during feature map \rightarrow activation function \rightarrow pooling layer steps. Now it is time to creating the character of the image from learned features. Fully-connected layer gathers all neurons from the previous layer and connects to the current layer to generate the characteristic of the image.

The fifth component is the output layer. Output layer is responsible for the classification of the features. Generally, Softmax or Multi-class SVMs are used [61].

2.4 *Visual Descriptors and CNNs*

CNNs have been developing more than a three-decade, its root is even older compared to visual descriptors. CNN works like the human brain and also classify better than human[51]. There are two significant reasons why for practitioners started to develop visual descriptors, descriptor-based methods are computationally simple and require less computation power. As a result, the algorithm can be tested instantly. Visual descriptors are simple, extracts the discriminative features from the input image and feed into the classification algorithm. If the obtained features are highly discriminative, then the method performs well. However, CNN is not simple, first of all, CNN has an architecture. Based on the architecture it might have multiple convolutional, pooling and fully-connected layers, for extracting highly discriminative features. Therefore CNN requires large training data and computing power [44].

Visual descriptors cannot be compared with CNNs; they can only be the part of CNN. There are several options, a visual descriptor can be the part of convolution operation, or it can be a separate layer. Recently, practitioners have been combining visual descriptors with CNN. For instance, Local Binary Pattern CNN (LBCNN) [62] and LBPNet [63]. In LBCNN, there are separate layers for LBP operation, wherein LBPNet, LBP used as convolution operation. LBCNN was tested on MNIST [27] 99.51%, SVHN [64] 94.50%, CIFAR-10 [65] 92.99%. LBPNet tested on FERET [26] and LFW [32] 94.04%.

CHAPTER III

PROPOSED METHOD

3.1 Overview

In this chapter, the proposed algorithm Multi-scale Binary Similarity (MSBS) and its variants are presented in detail. First, MSBS approach for feature extraction is explained. Then MSBS stages are explained in detail.

3.2 Multi-Scale Binary Similarity

LBP has many qualifying attributes, thresholding each neighbor pixel intensity to its center pixel intensity make it illumination invariant. Normalizing the binary code to its minimum value make it rotational invariant. Selecting either circular or square neighbor make it flexible in neighborhood topology. Placing obtained patterns based on its uniformity make it flexible in histogram size. Computationally simple attributes make it applicable to many real-time applications. LBP can be used together with other image descriptors [66]. LBP is easily adapted to different types of problem, and it is highly discriminative texture operator. However, LBP still can be improved based on application. In specific application areas, LBP's classification accuracy can still be increased for instance, face recognition. LBP's illumination invariant property naturally avoids the unwanted lighting effect in a given face image. Next step is selecting the discriminative feature from the face image. In literature, there were many studies proposed for facial feature extraction.

Liao and Chung [24] mentioned that anisotropic information (mouth, eyes, nose,

etc..) is the discriminative characteristic for a given face image. First, they questioned the circular neighborhood topology which was part of the rotational invariance. However, rotational invariance is not needed for given face image. Therefore, they proposed elongated LBP (ELBP). ELBP has three parameters A, B, and m. From a selected center pixel, A pixel distance in horizontal and B pixel distance in the vertical direction are covered. From covered area m pixels are selected. The order of selecting pixels are same with LBP. Liao and Chung discovered that LBP had the lack of gradient information. The gradient information of a standard image shows the direction of rapid changes in an image.

According to Liao and Chung, the gradient information shows the direction of anisotropic information. Anisotropic information means having a physical property which has a different value when measured in different directions [67]. For instance, a repetitive pattern structure, texture, the isotropic information is essential, but wood along the grass, the anisotropic information is necessary. When Elongated LBP used, the missing anisotropic information obtained. For instance, eye, a mouth is an anisotropic structure. Our method also gets the anisotropic information. However, the basic LBP cannot obtain isotropic information, because it uses circular neighborhood for extracting features. Therefore, Liao and Chung proposed Average Maximum Distance Gradient Magnitude (AMDGM). AMDGM first thresholds the neighbor pixel intensities with center pixel intensity then calculate the distance from neighbors to the center pixel. The calculated values are divided with each other. The maximum divided values were selected as features. The drawback of this study is only uniform patterns are considered. Liao et al. [25] stated that a large amount of useful patterns turns into non-uniform ones. The non-uniform patterns are not considered by conventional LBP since Ojala et al. [11] stated that 90% of texture information obtained by the uniform pattern. However, in face recognition dataset, non-uniform patterns are also important.

Nanni et al. [9] studied on Medical Image Analysis. They analyzed LBP variants with different neighborhoods. They discovered quinary encoding using elliptic neighborhood accurately classify medical images. They tested each LBP, local ternary pattern and elliptic quinary pattern with five different topologies, circle, ellipse, parabola and archimedean spiral. Petpon et al. studies on face recognition. Instead of using the traditional circular symmetric neighborhood, they considered the vertical and horizontal pixels from the selected center pixel. Approach named as local line binary pattern (LLBP). Results show that LLBP performs slightly better than LBP.

Wolf et al. focused on finding the complementary information from given the face image. In a previous study, it was stated that the patch based approach provided state of the art in learning of faces [68]. This motivated Wolf et al. for designing a patch based descriptor. The term patch refers to square neighborhood operator. S additional patches were placed on the image. The image is divided into the center patch and (S-1) neighbor patches. Each patch is $w \times w$ size, where w is the integer value. Each time distance between patches was calculated. Wolf et al. calculated the distance of three patches, abbreviated as (TPLBP) and four patches (FPLBP). Selection of patches is based on neighbor patches distance to center patch. For instance, in TPLBP, two neighbor patch distance to center patch was calculated.

Liu et al. studied the effect of thresholding neighbor pixels on classification. Instead of thresholding each neighbor pixel intensity with the center pixel intensity, they threshold neighbor pixel based on the same position. The position is separated into twofold. First, the neighbor pixels in the same radius are thresholded, then neighbors in the same angular angle thresholded. The first approach named as radial difference local binary pattern (RDLBP). The second approach named as angular distance local binary pattern (ADLBP). Feature extraction is similar to LBP. If the thresholded neighbor intensities greater or equal to zero, current bit is one, otherwise zero. Liu et al. achieved state of the art result in Extended Yale B dataset.

After analyzing the previous works, we gain the insight of taking non-uniformity into consideration is useful, deciding on neighborhood topology may increase the classification accuracy; the patch-based approach provides the state of the art learning and, the anisotropic information is essential for face recognition. It is a fact that, there are relations between pixels in the dataset since each dataset has the specific characteristic. For instance, in a face recognition dataset, anisotropic information (two eyes, one nose) is the characteristic. The distance between each anisotropic information can be approximated. Standard LBP approach neglected this anisotropic information and used the same neighborhoods. The previous works were used a static way for finding anisotropic information based on their algorithm. We questioned this static approach for finding anisotropic information. We believe that unique neighborhood information is required for face recognition classification. This motivated us and using, for each pixel in the dataset, we extracted a neighborhood template based on measured pixel neighborhood importance. Each neighbor pixel assigned a value based on a score. After calculation, the value of each neighbor represents its importance for classification. Therefore best K neighbor selected. Selecting here, refers to getting the best K neighbor spatial coordinate inside the neighborhood window. After deciding about important neighbors, we need to decide about their order of selection. In previous studies, texture spectrum, basic and conventional LBP descriptors used counter-clockwise approach for picking up neighbor pixel values and achieved the state-of-the-art classification accuracy. Therefore, we used counter-clockwise approach for ordering the neighbor pixel spatial coordinates. Each ordered spatial coordinates were selected inside the moving window. Each corresponding spatial coordinate intensity was used for creating the decided pattern. Created patterns were used for establishing the histogram. Histogram values of each label used in classification. We named this approach as Multi-Scale Binary Similarity (MSBS), since it considers neighborhood topology inside windows of different sizes.

MSBS algorithm is designed to analyze the pixel relationships and represent it as a binary vector. Similar to previous studies [23], [35], MSBS divides the input image into regions and extracts features from regions, represented by histograms. Then all histograms are concatenated for representing the image characteristics. Unlike the traditional approaches, MSBS algorithm has some differences for feature extraction. The first difference is, instead of dividing the image into fixed regions, MSBS obtains image regions, using a moving window (Figure 27). This moving window is named as the major patch (P_{maj}) and defined by a radius value, named r_1 . P_{maj} starts the movement in the horizontal and continues with the vertical until all pixels are covered. As a result, MSBS covers all local regions for further processings.

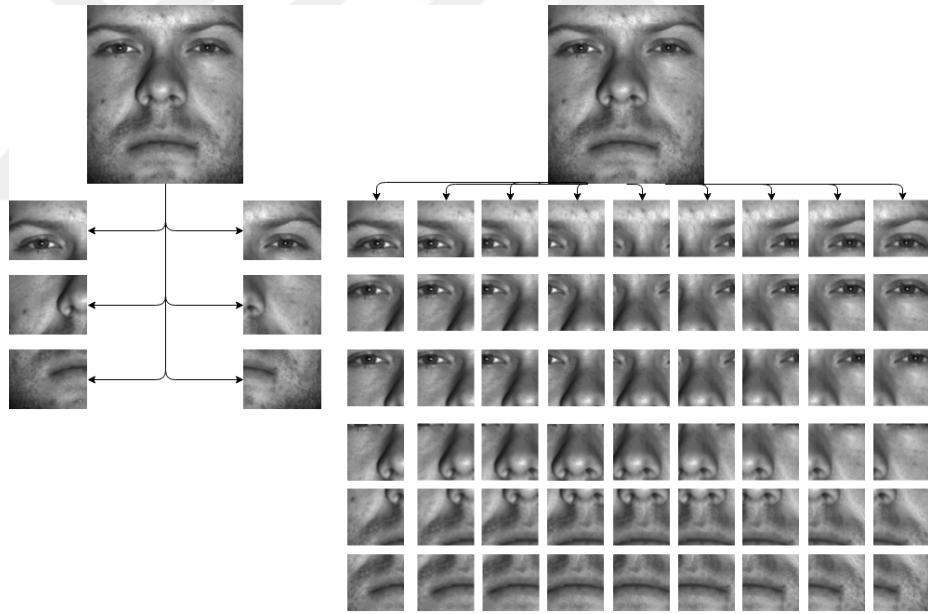


Figure 27: The difference between traditional (left) and MSBS(right) approach for dividing images.

From each local region, the P_{maj} in our case, the pixel relationships are analyzed. The analysis starts with selecting two more regions, the minor patch (P_{min}) and the pivot patch (P_{piv}) both declared with radius r_2 . The difference is, P_{min} moves inside the P_{maj} , and P_{piv} fixed at r_2 radius, the square area from major patch center

pixel (C_{maj}). P_{min} , similar to P_{maj} movement, moves in the horizontal direction and continues with the vertical direction, until all pixels covered inside the P_{maj} window.

Each P_{min} neighbor pixel is thresholded with P_{piv} neighbor pixels. This thresholding has shown some similarity with LBP; if P_{min} center pixel (C_{min}) is greater than or equal to P_{piv} center pixel (C_{piv}), P_{min} neighbors which are greater than or equal to P_{piv} , are encoded as 1, otherwise 0. As a result, a binary pattern is obtained (Figure 28).

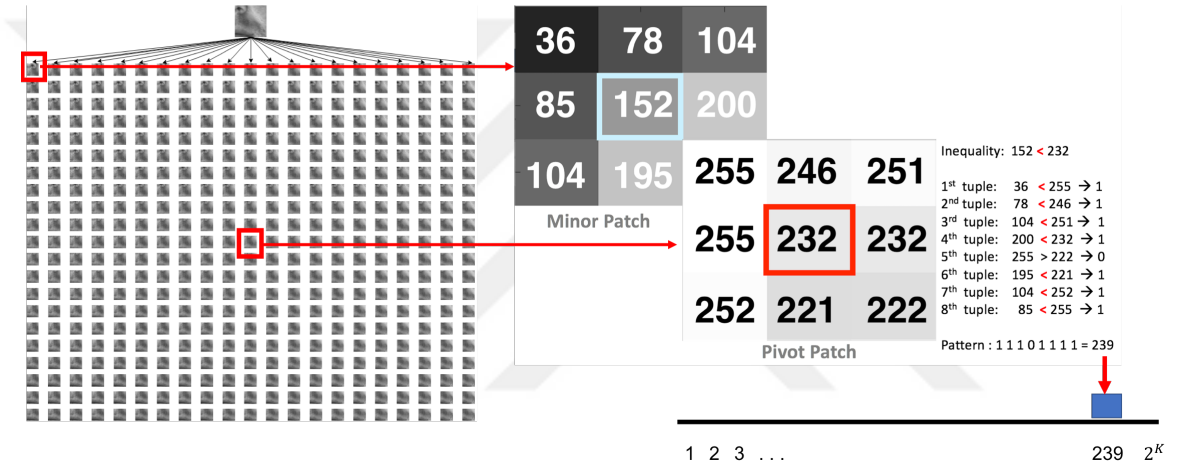


Figure 28: Major Patch decomposed into several minor patches

3.2.0.1 The movement of Major Patch

P_{maj} size is $(2 \times r_1 + 1) \times (2 \times r_1 + 1)$ pixels for a given r_1 value. The total movement of the P_{maj} in an image of size height (H) and width (W) can be calculated as below;

$$\text{Total Moves} = (H - (2 \times (r_1 + r_2) + 1)) * (W - (2 \times (r_1 + r_2) + 1)) \quad (25)$$

So far, we have explained the base calculation of MSBS. We used this base approach for the main steps of MSBS calculation. MSBS has three main steps, neighborhood template calculation, spatial coordinates selection and MSBS feature extraction.

3.2.1 Neighborhood Template

The basic LBP selects all the neighbors in a counter-clockwise direction. Each selected neighbor pixel value is thresholded with center pixel. This approach shows basic LBP has no rule for neighbor pixel selection. We questioned this approach and created computationally simple score matrix. Score matrix calculation uses the base approach and defined by r_2 radius. P_{\min} is compared with the P_{piv} . First, the inequality between center pixels determined. The inequality as shown before, either C_{\min} is greater than, equal or less than C_{piv} . Each neighbor pixel that supports the inequality will get score +1, otherwise -1 (Algorithm 1). The value is stored in the same spatial coordinates with compared pixels (Figure 29). Accumulated score matrix is calculated by using all images in the dataset.

Algorithm 1 Neighborhood Template Calculation

Require: P_{\min} : A moving window with size $(2 \times r_2 + 1)^2$

P_{piv} : A grid area with size $(2 \times r_2 + 1)^2$

r_2 : Minor and Pivot patches radius

$C_{\min} \leftarrow P_{\min}^{(r_2+1, r_2+1)}$ -Initialize P_{\min} center pixel
 $C_{\text{piv}} \leftarrow P_{\text{piv}}^{(r_2+1, r_2+1)}$ -Initialize P_{piv} center pixel
 $L \leftarrow (2 \times r_2 + 1)$ -Edge of the both P_{\min} and P_{piv}

-Let score total S_T be a matrix of size $L \times L$.

```

for  $s \leftarrow 1$  to  $L$  do
  for  $t \leftarrow 1$  to  $L$  do
    if  $(C_{\min} \geq C_{\text{piv}}) \ \& \ (P_{\min}^{(s, t)} \geq P_{\text{piv}}^{(s, t)})$  then
       $S_T^{(s, t)} \leftarrow S_T^{(s, t)} + 1$ 
    else if  $(C_{\min} < C_{\text{piv}}) \ \& \ (P_{\min}^{(s, t)} < P_{\text{piv}}^{(s, t)})$  then
       $S_T^{(s, t)} \leftarrow S_T^{(s, t)} + 1$ 
    else
       $S_T^{(s, t)} \leftarrow S_T^{(s, t)} - 1$ 
    end if
  end for
end for

```

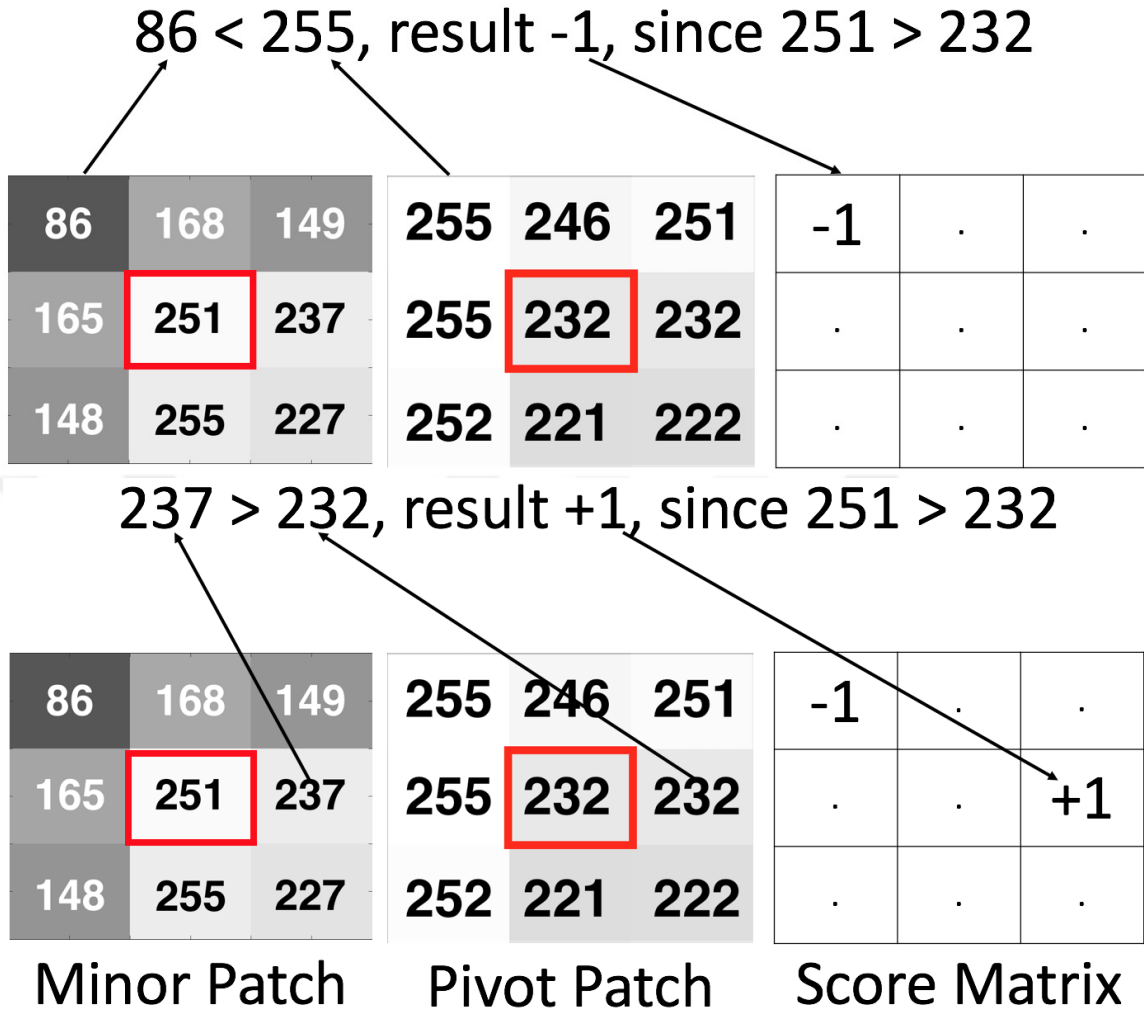


Figure 29: Example Score Matrix Calculation

3.2.2 Neighborhood Template Extraction from Accumulated Score Matrix

The accumulated score matrix shows the weights of each pixel in the P_{\min} . The weights represent the importance of pixel's spatial coordinate for classification. The highest scored K neighbors are selected in clockwise direction. The coordinates are stored in ordered set T which defines the neighborhood template (Figure 30).

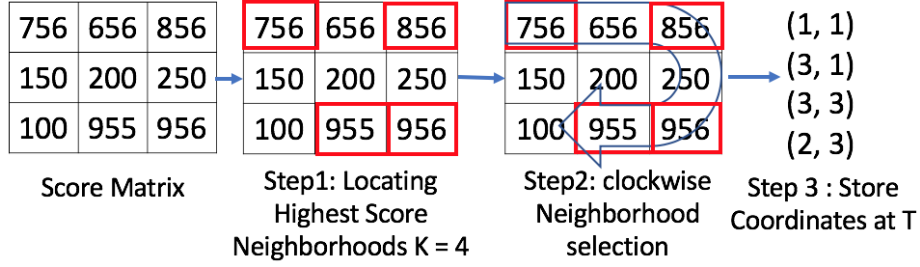


Figure 30: Selecting Coordinates from Score Matrix

3.2.3 MSBS Features Extraction

MSBS feature extraction is similar to neighborhood template extraction. MSBS has three main components; P_{maj} , P_{piv} and P_{min} . Each component is established in the following order; first, the center pixel spatial coordinate is located, then from the located coordinate, a square neighborhood is selected. The square neighbor selection depends on radius parameter. The first component is the P_{maj} . C_{maj} is located at (r_1, r_1) . From the initial pixel spatial coordinates $(0, 0)$, C_{maj} is $(1+r_1, 1+r_1)$ distance away. The square neighborhood of P_{maj} depends on r_1 . Each pixel, r_1 distance away from C_{maj} , is major patch neighbor (Figure 31).

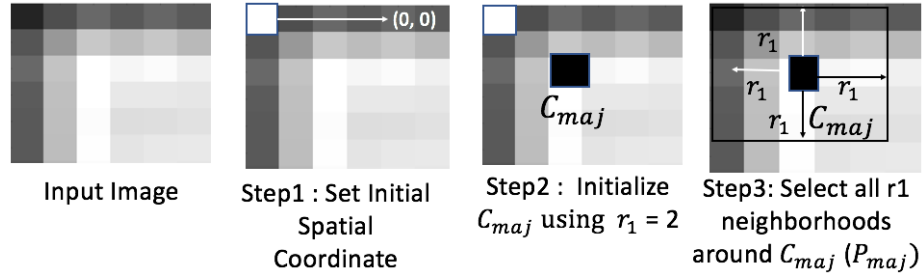


Figure 31: Major Patch Initialization

The second component is the P_{piv} . C_{piv} has the same spatial coordinate at C_{maj} . The square neighborhood of P_{piv} depends on r_2 . Each pixel r_2 distance away from C_{piv} is pivot patch neighbor (Figure 32). The third component is the P_{min} . C_{min} is at the initial pixel in P_{maj} . Each pixel r_2 distance away from C_{min} is P_{min} neighbor

(Figure 33).

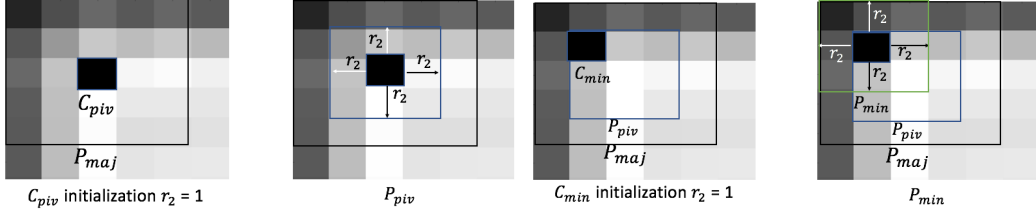


Figure 32: Pivot Patch Initialization **Figure 33:** Minor Patch Initialization

After all patches are located, P_{\min} and P_{piv} are analyzed to obtain encoding. The first rule of analysis is center pixel thresholding. Both C_{\min} and C_{piv} are thresholded, respectively. Based on the inequality condition, each neighbor tuples (P_{\min} and P_{piv} , respectively) are compared. Neighbor coordinates are already calculated in neighborhood template section. The selected tuples supports the inequality encoded as 1, else 0. The resulting pattern holds the new extracted interest point values. The next step is, C_{\min} visit each pixel of P_{maj} . After visiting done, P_{maj} moves one pixel at a time, until there is no movement left for P_{maj} (Algorithm 2).

3.2.3.1 MSBS Formulation

MSBS encodes the neighborhood tuples of minor and pivot patches, based on the comparison between C_{\min} and C_{piv} , respectively. As formulated in Algorithm 1, if each neighbor tuple supports the direction of thresholded center pixels, then the pattern is encoded as 1, otherwise 0. Formulation has two steps. First step is subtracting P_{\min} from P_{piv} . The result of subtraction stored in the temporary patch (P_t) (Figure 34). Second step is comparing each sign of the neighbor pixel value with center pixel sign (Equation ??). If they have the same sign, the result is 1, otherwise 0 (Equation ??). The process is explained in Algorithm 3.

$$msbsPattern(r_1, r_2, K) = \sum_{i=0 \wedge L_i \in T}^{K-1} S_{MSBS}(P_t^{L_i}, P_t^c) \times 2^i \quad (26)$$

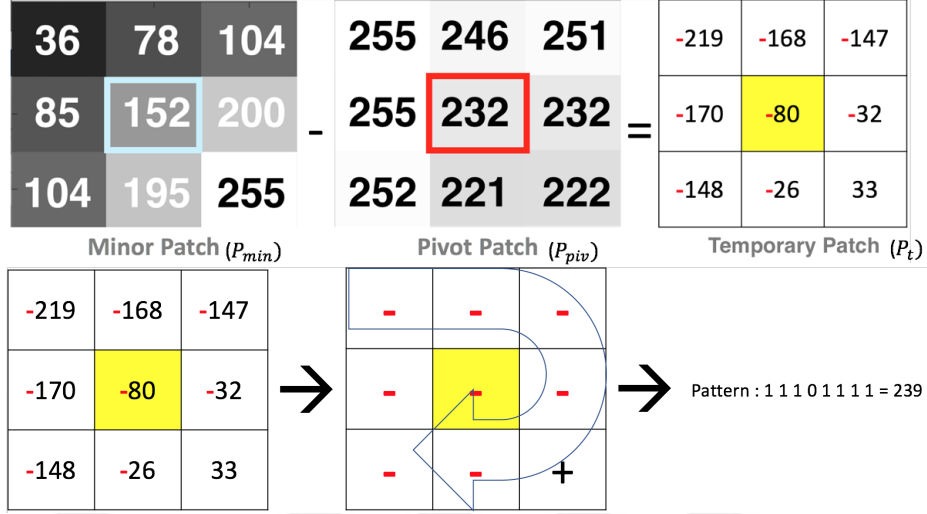


Figure 34: MSBS Formulation

Remember that the ordered set T is constructed by using all dataset images.

L_i is the i^{th} member of T and $P_t^{L_i}$ is the temporary patch value, defined by L_i .

$$S_{MSBS}(x, y) = \begin{cases} 1, & x \times y \geq 0 \\ 0, & x \times y < 0 \end{cases} \quad (27)$$

3.2.4 Variants of MSBS

3.2.4.1 MSBS with Feature Matrix

The process is same as MSBS feature calculation, to make calculation faster, the feature matrix is created during MSBS features extraction step. The process is simple, the binary patterns converted to decimal value and placed on the feature matrix. The feature matrix has the same size as the input image. Instead of placing the decimal value of binary pattern directly on MSBS histogram, the value is placed on the feature matrix on minor patch spatial coordinate. MSBS features are extracted from the feature matrix and placed on the histogram within defined cell size (Figure 35). Instead of placing features one-by-one, multiple features are placed on the histogram. Creating feature matrix is almost similar to Algorithm 2. The only difference is that

the decimal value is set to feature matrix (Algorithm 5)(the difference is shown in red). Then feature matrix is divided into defined cell sizes, and from each cell size histograms are evaluated (Algorithm 6).

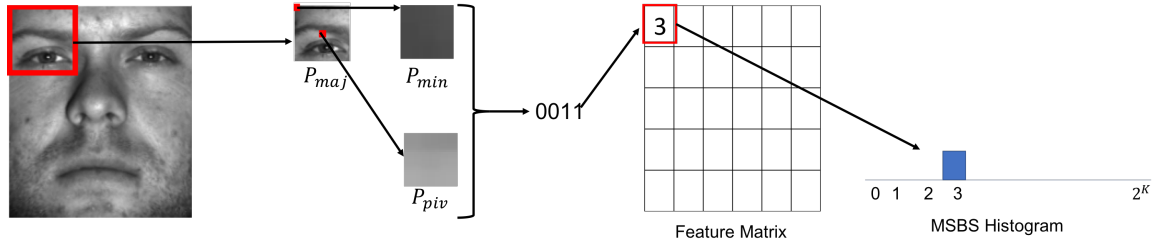


Figure 35: Creating MSBS Histogram from the Feature Matrix

3.2.4.2 MSBS extract features from Major Patch

In the initial approach, features extracted from the moving window, which was declared with r_2 . We wanted to see the effect of the features extracted from the major patch on classification. This approach suggests extracting features from P_{maj} instead of P_{min} . Therefore, neighborhood template calculation (Algorithm 4) and feature extraction are modified based on this approach (Algorithm 7). In previous version, moving window is not exceeding P_{maj} borders. However, in this version each pixel in P_{maj} will be P_{min} center pixel. Therefore P_{min} exceeds P_{maj} borders. Same as MSBS, each time P_{min} compared with P_{piv} , and the best K neighbor selected inside P_{maj} . The calculation of the spatial coordinates is same. MSBS feature extraction is similar to neighbor template calculation, binary pattern is obtained from each comparison with P_{min} and P_{piv} . Note that, MSBS feature extraction from the major patch produces identical patterns and histogram results as a consequence.

Algorithm 2 MSBS Feature Extraction

Require: **I** : A gray-scale image.

H × **W** : dimensions of the gray-scale image.

r_1 : Major patch radius

r_2 : Minor and Pivot patches radius

K : Neighbor number

D : Distance from center pixel to neighbor pixels with the dimension $2 \times K$

$Y_{maj} \leftarrow H - (2 \times r_1 + 1)$ - P_{maj} movement in Y direction

$X_{maj} \leftarrow W - (2 \times r_1 + 1)$ - P_{maj} movement in X direction

$L \leftarrow 2 \times (r_1 - r_2)$ - P_{min} movement in P_{maj}

-Let msbs Pattern (P_{msbs}) be a vector of size K.

-Let msbs Histogram (H_{msbs}) be a dynamic array.

for $i \leftarrow 1$ **to** Y_{maj} **do**

for $j \leftarrow 1$ **to** X_{maj} **do**

 - Assume C_{ymaj} and C_{xmaj} represents the P_{maj} spatial coordinates inside I

for $k \leftarrow 1$ **to** L **do**

for $l \leftarrow 1$ **to** L **do**

$Y_{min} \leftarrow C_{ymaj}^{(k, l)} : C_{ymaj}^{(k, l)} + 2 \times r_2$ - P_{min} y coordinates

$X_{min} \leftarrow C_{xmaj}^{(k, l)} : C_{xmaj}^{(k, l)} + 2 \times r_2$ - P_{min} x coordinates

$P_{min} \leftarrow I(Y_{min}, X_{min})$ - P_{min} intensity values

$C_{min} \leftarrow P_{min}^{(r_2+1, r_2+1)}$ - P_{min} center pixel intensity

for $m \leftarrow 1$ **to** K **do**

$Y_{current} \leftarrow D(m, 1) + r_2 + 1$ -Current Neighbor Y coordinate

$X_{current} \leftarrow D(m, 2) + r_2 + 1$ -Current Neighbor X coordinate

$N_{current} \leftarrow P_{min}^{(Y_{current}, X_{current})}$ -Current Neighbor

if $N_{current} \geq C_{min}$ **then**

$P_{msbs}^{(m)} \leftarrow 1$

else

$P_{msbs}^{(m)} \leftarrow 0$

end if

$v \leftarrow 0$

for $q \leftarrow 1$ **to** size of P_{msbs} **do**

$v \leftarrow v + P_{msbs}^{(q)} \times 2^{(q-1)}$

end for

$H_{msbs}^{(v)} \leftarrow H_{msbs}^{(v)} + 1$

end for

end for

end for

end for

end for

Algorithm 3 MSBS Simplified Version

Require: P_{\min} : A moving window with size $(2 \times r_2+1)^2$

P_{piv} : A grid area with size $(2 \times r_2+1)^2$

r_2 : Minor and Pivot patches radius

K : Neighbor number

$P_t \leftarrow P_{\min} - P_{\text{piv}}$

$C_t \leftarrow P_t^{(r_2+1, r_2+1)}$ -Initialize P_t center pixel

-Let msbs Pattern P_{msbs} be a vector of size K .

for $i \leftarrow 1$ **to** K **do**

if $P_t^{(i)} \times C_t \geq 0$ **then**

$P_{\text{msbs}}^{(i)} \leftarrow 1$

else

$P_{\text{msbs}}^{(i)} \leftarrow 0$

end if

end for

for $i \leftarrow 1$ **to** K **do**

$v \leftarrow v + P_{\text{msbs}}^{(i)} \times 2^{(i-1)}$

end for

$C_{\min} \leftarrow v$ -Set decimal value to P_{\min} center pixel

Algorithm 4 MSBS Vectorized Approach for Neighborhood Template Calculation

Require: P_{\min} : A moving window with size $(2 \times r_2+1)^2$

P_{piv} : A grid area with size $(2 \times r_2+1)^2$

r_2 : Minor and Pivot patches radius

$C_{\min} \leftarrow P_{\min}^{(r_2+1, r_2+1)}$ -Initialize P_{\min} center pixel

$C_{\text{piv}} \leftarrow P_{\text{piv}}^{(r_2+1, r_2+1)}$ -Initialize P_{piv} center pixel

$L \leftarrow 2 \times (r_2+1)$ -Edge of the both P_{\min} and P_{piv}

if $C_{\min} \geq C_{\text{piv}}$ **then**

$P \leftarrow \text{sum}(\text{sum}(P_{\min} \geq P_{\text{piv}}))$ -Sum of the condition supported values

$N \leftarrow \text{sum}(\text{sum}(P_{\min} < P_{\text{piv}}))$ -Sum of the condition unsupported values

else

$P \leftarrow \text{sum}(\text{sum}(P_{\min} < P_{\text{piv}}))$ -Sum of the condition supported values

$N \leftarrow \text{sum}(\text{sum}(P_{\min} \geq P_{\text{piv}}))$ -Sum of the condition unsupported values

end if

$S \leftarrow P-N$ -Score value is the difference between supported and unsupported values

Algorithm 5 MSBS Feature Matrix

Require: **I** : A gray-scale image.

H \times **W** : dimensions of the gray-scale image.

r_1 : Major patch radius

r_2 : Minor and Pivot patches radius

K : Neighbor number

D : Distance from center pixel to neighbor pixels with the dimension $2 \times K$

$Y_{maj} \leftarrow H - (2 \times r_1 + 1)$ - P_{maj} movement in Y direction

$X_{maj} \leftarrow W - (2 \times r_1 + 1)$ - P_{maj} movement in X direction

$L \leftarrow 2 \times (r_1 - r_2)$ - P_{min} movement in P_{maj}

$Y_{featurematrix} \leftarrow H - 2 \times r_2$

$X_{featurematrix} \leftarrow W - 2 \times r_2$

-Let msbs pattern (P_{msbs}) be a vector of size **K**.

-Let msbs histogram (H_{msbs}) be a dynamic array.

-Let feature matrix (M_F) be a matrix of size $Y_{featurematrix} \times X_{featurematrix}$

for $i \leftarrow 1$ **to** Y_{maj} **do**

for $j \leftarrow 1$ **to** X_{maj} **do**

 -Assume C_{ymaj} and C_{xmaj} represents the P_{maj} spatial coordinates inside **I**

for $k \leftarrow 1$ **to** **L** **do**

for $l \leftarrow 1$ **to** **L** **do**

$Y_{min} \leftarrow C_{ymaj}^{(k, l)} : C_{ymaj}^{(k, l)} + 2 \times r_2$ - P_{min} y coordinates

$X_{min} \leftarrow C_{xmaj}^{(k, l)} : C_{xmaj}^{(k, l)} + 2 \times r_2$ - P_{min} x coordinates

$P_{min} \leftarrow I(Y_{min}, X_{min})$ - P_{min} intensity values

$C_{min} \leftarrow P_{min}^{(r_2+1, r_2+1)}$ - P_{min} center pixel intensity

for $m \leftarrow 1$ **to** **K** **do**

$Y_{current} \leftarrow D(m, 1) + r_2 + 1$ -Current Neighbor Y coordinate

$X_{current} \leftarrow D(m, 2) + r_2 + 1$ -Current Neighbor X coordinate

$N_{current} \leftarrow P_{min}^{(Y_{current}, X_{current})}$ -Current Neighbor

if $N_{current} \geq C_{min}$ **then**

$P_{msbs} \leftarrow 1$

else

$P_{msbs} \leftarrow 0$

end if

$v \leftarrow 0$ -Initialize decimal value to 0

for $q \leftarrow 1$ **to** size of P_{msbs} **do**

$v \leftarrow v + P_{msbs}^{(q)} \times 2^{(q-1)}$

end for

$M_F^{(Y_{min}(1), X_{min}(1))} \leftarrow v$ -Store decimal value inside Feature Matrix

end for

end for

end for

end for

end for

Algorithm 6 MSBS Extract Features From Feature Matrix

Require: \mathbf{M}_F : Feature Matrix.

$\mathbf{H} \times \mathbf{W}$: dimensions of the preprocessed image.

$\mathbf{Y}_{\text{cell}} \times \mathbf{X}_{\text{cell}}$: dimensions of the cell size

\mathbf{K} : Neighbor number

$Y_{\text{movement}} \leftarrow \text{floor}(\mathbf{H}/\mathbf{Y}_{\text{cell}})$ -Cell movement in Y direction

$X_{\text{movement}} \leftarrow \text{floor}(\mathbf{W}/\mathbf{X}_{\text{cell}})$ -Cell movement in X direction

$Y_{\text{counter}} \leftarrow 1$

$X_{\text{counter}} \leftarrow 1$

-Let msbs Histogram (H_{msbs}) be a vector of size 2^K .

-Let total Histogram (H_{total}) be a dynamic array.

for $i \leftarrow 1$ **to** Y_{movement} **do**

for $j \leftarrow 1$ **to** X_{movement} **do**

$X_{\text{range}} \leftarrow (X_{\text{counter}} : X_{\text{counter}} + X_{\text{cell}})$

$Y_{\text{range}} \leftarrow (Y_{\text{counter}} : Y_{\text{counter}} + Y_{\text{cell}})$

$S_{\text{area}} \leftarrow P(Y_{\text{range}}, X_{\text{range}})$

 -Select Area from P based on X and Y ranges

for $k \leftarrow 1$ **to** height of S_{area} **do**

for $l \leftarrow 1$ **to** width of S_{area} **do**

$V \leftarrow S_{\text{area}}^{(k, l)}$

$H_{\text{msbs}}^{(V)} \leftarrow H_{\text{msbs}}^{(V)} + 1$

end for

end for

$H_{\text{total}} \leftarrow H_{\text{total}} + H_{\text{msbs}}$

 -Add current histogram to the total histogram

end for

end for

Algorithm 7 MSBS Feature Extraction from Major Patch

Require: **I** : A gray-scale image.

H × **W** : dimensions of the gray-scale image.

r_1 : Major patch radius

r_2 : Minor and Pivot patches radius

K : Neighbor number

D : Distance from center pixel to neighbor pixels with the dimension $2 \times K$

$Y_{maj} \leftarrow H - (2 \times (r_1 + r_2) + 1)$ - P_{maj} movement in Y direction

$X_{maj} \leftarrow W - (2 \times (r_1 + r_2) + 1)$ - P_{maj} movement in X direction

$C_{maj} \leftarrow I(r_1 + 1, r_1 + 1)$ - P_{maj} center pixel intensity

$c \leftarrow 1$ -Initialize histogram counter

-Let msbs Pattern (P_{msbs}) be a vector of size D.

-Let msbs Histogram (H_{msbs}) be a dynamic array.

for $i \leftarrow 1$ **to** Y_{maj} **do**

for $j \leftarrow 1$ **to** X_{maj} **do**

$P_{maj} \leftarrow I(C_{maj}^{(1)} - r_1 : C_{maj}^{(1)} + r_1, C_{maj}^{(2)} - r_1 : C_{maj}^{(2)} + r_1)$

for $k \leftarrow 1$ **to** size of **D** **do**

$p \leftarrow P_{maj}(D(k, 1), D(k, 2))$ -Get current pixel intensity

$P_{selected}^{(k)} \leftarrow p$ -Add intensity to the selected pixel

end for

for $l \leftarrow 1$ **to** size of $P_{selected}$ **do**

if $P_{selected}^{(l)} \geq C_{maj}$ **then**

$P_{msbs}^{(l)} \leftarrow 1$

else

$P_{msbs}^{(l)} \leftarrow 0$

end if

end for

$v \leftarrow 0$

for $m \leftarrow 1$ **to** size of P_{msbs} **do**

$v \leftarrow v + P_{msbs}^{(m)} \times 2^{(m-1)}$

 -Convert binary pattern to decimal

end for

$H_{msbs}^{(c)} \leftarrow v$

-Add decimal value to histogram

if $j \leftarrow X_{maj}$ **then**

$C_{maj}^{(2)} \leftarrow r_1 + r_2 + 1, C_{maj}^{(1)} \leftarrow C_{maj}^{(1)} + 1$

else

$C_{maj}^{(2)} \leftarrow C_{maj}^{(2)} + 1$

end if

50

end for

end for

CHAPTER IV

EXPERIMENTS

4.0.1 AT&T

AT&T, formerly known as ORL, face recognition dataset [12] consisted of 400 grey-scale images of 40 subjects. Each subject has ten samples. Each image captured under the nominal lighting condition, with the size of 92×112 pixels. For each subject, five randomly selected samples are placed in the training set (Figure 36) and the rest are put into the test set (Figure 37). Both training and test sets have consisted of 200 images. We also applied the preprocessing chain for both training (Figure 38) and test sets (Figure 39), to observe how it will affect the classification accuracy.



Figure 36: AT&T Training Examples



Figure 37: AT&T Test Examples



Figure 38: Preprocessing Chain Applied AT&T Training Examples



Figure 39: Preprocessing Chain Applied AT&T Test Examples

The main reason we divided the training and test set equally, is because we want to compare our results with the previous study [69]. We tested our algorithm using LIB-SVM [70] with linear, polynomial and rbf kernels respectively. The best results were shown in bold in Table 4.

Table 4: AT&T SVM Results

AT&T- 40 Classes		SVM Accuracy on Test Set with 10-Fold Cross Validation						
Each class having 10 samples	Method				Unscaled	PCA	Whitening	Whitened PCA
Each Class Image 92 x 112 pixel	r ₁	r ₂	K	Cell Size	Accuracy	Accuracy	Accuracy	Accuracy
Linear Kernel	2	1	4	45x45	95%	93.5% (144→27)	98%	97% (144→27)
Polynomial Kernel Degree 3	2	1	4	45x45	93.5%	89.5%(144→27)	97.5%	93%(144→27)
RBF Kernel Degree 3	2	1	4	45x45	94%	91%(144→27)	98%	97.5%(144→27)

Table 5: Preprocessing Chain Applied AT&T SVM Results

AT&T- 40 Classes		SVM Accuracy on Test Set with 10-Fold Cross Validation						
Each class having 10 samples	Method				Unscaled	PCA	Whitening	Whitened PCA
Each Class Image 92 x 112 pixel	r ₁	r ₂	K	Cell Size	Accuracy	Accuracy	Accuracy	Accuracy
Linear Kernel	2	1	4	45x45	84.5%	82% (144→22)	94%	93% (144→22)
Polynomial Kernel Degree 3	2	1	4	45x45	82.5%	80.5% (144→22)	94.5%	92% (144→22)
RBF Kernel Degree 3	2	1	4	45x45	83%	77.5% (144→22)	95%	93% (144→22)

MSBS was compared with various descriptors and achieved the highest classification accuracy (Figure 40).

The results of Table 4 and 5 reveal two facts. The first one is, the preprocessing chain did not improve the classification accuracy. For each kernel, the preprocessing chain accuracy is lower than the normal accuracy (Figure 41). One possible reason is the images are visible, and there is no illumination effect on the images. Therefore, the preprocessing chain had no impact on classification accuracy. The second fact is, highest classification accuracy was achieved when whitening applied to both training and test features. In previous face recognition studies [72], [73], [74] authors have stated that the Whitened PCA technique was useful for improving the classification accuracy. In face recognition, PCA’s eigenvectors mostly encode illumination, rather than discriminating information. The whitening normalization applied to the PCA features to reduce the negative influence of eigenvectors on classification. For instance, MSBS extracted feature matrix size is 200×144 , where 200 is the number of images in

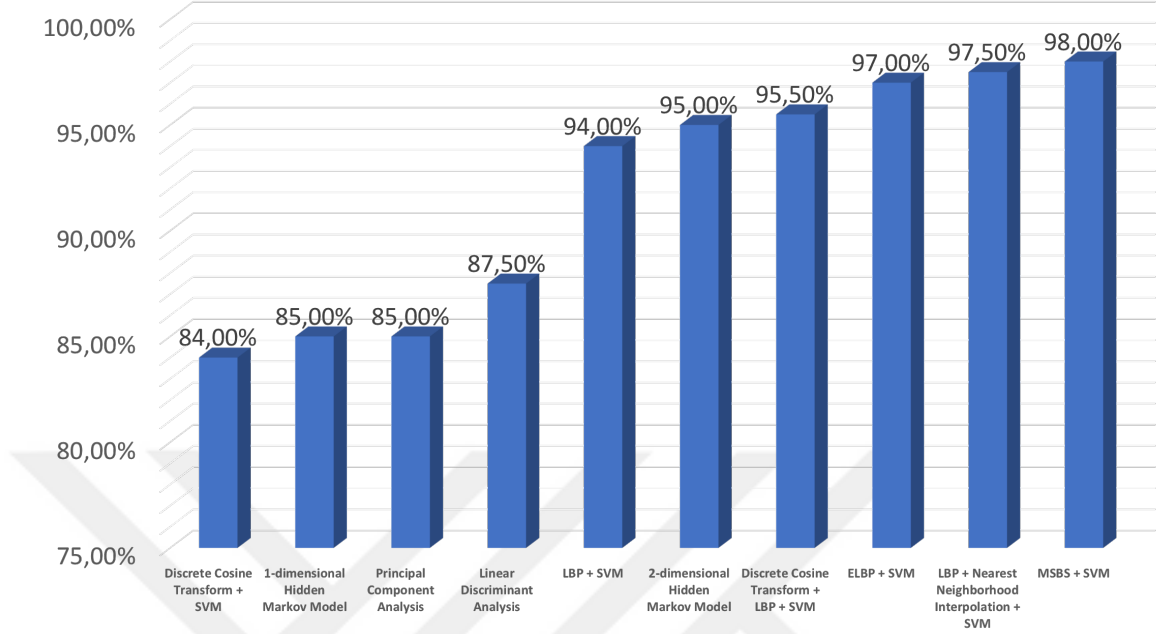


Figure 40: Comparing MSBS Results with other descriptors using AT&T dataset

the set and 144 (N) is the feature vector. First, the feature vector mean is calculated (\bar{X}). Second, the covariance matrix (S) was calculated. Assume Q represents the N times \bar{X} , $Q = [\bar{X}, \bar{X}, \dots, \bar{X}]$ and F_{train} and F_{test} represent the training and tests feature matrices respectively.

$$S = \frac{(F_{\text{train}} - Q) \times (F_{\text{train}} - Q)^T}{N} \quad (28)$$

Each feature value is subtracted from the mean and multiplied with its transpose, then divided to feature vector size. The covariance matrix measures the linear association of the feature values. In theory, the + value represents the growing relationship and - value represents the decreasing relationship. The size of S is $N \times N$. Third, applying whitening noise to the data. Assume function Z selects the diagonal values in the matrix and the resulting training and test set represented as W_{training} , W_{test} respectively.

$$W_{\text{training}} = Z\left(\frac{1}{\sqrt{D(S) + \epsilon}}\right) \times F_{\text{train}} \quad W_{\text{test}} = Z\left(\frac{1}{\sqrt{D(S) + \epsilon}}\right) \times F_{\text{test}} \quad (29)$$

In the equation 29, the whitening part is represented with $Z\left(\frac{1}{\sqrt{D(S) + \epsilon}}\right)$ in the Formula. The Whitened PCA dimensionality reduction calculation is similar to PCA. First, the Singular Value Decomposition of the covariance matrix is calculated. Singular value decomposition divide matrix into the product of three matrices; Left Singular (U), Singular (D), and Right Singular (V). D matrix is used for the finding the dimensionality reduction coefficient. The idea is, if the result of the i'th total sum divided by the total sum of D is greater or equal than a constant named as energy preservation value, then the new reduced dimension is 'i' value. In all of our experiments, we kept the energy preservation value 95%. The algorithm is below:

Algorithm 8 PCA and Whitened PCA Dimensionality Reduction

Require: **D:** Singular Matrix

EPV: Energy Preservation Value

eigenSum \leftarrow sum(D)

D_{new} \leftarrow 0 -The New Dimension

for do i \leftarrow 1 **to** size of D

currentEnergy \leftarrow currentEnergy + D(i)

if then currentEnergy / eigenSum \geq EPV

D_{new} \leftarrow i

end if

end for

The new dimension (D_{new}) is applied to the Left Singular matrix (U) because U matrix contains the eigenvectors. The new dimensionality reduced matrix (G) will be the $G = U(1:D_{\text{new}})$, since the 95% energy preserved at D_{new} dimension. In this thesis, both PCA and WPCA were calculated as in the following formulas:

$$\begin{aligned} \text{PCA}_{\text{training}} &= \mathbf{G}^T \times \mathbf{F}_{\text{train}} & \text{PCA}_{\text{test}} &= \mathbf{G}^T \times \mathbf{F}_{\text{test}} \\ \text{WPCA}_{\text{training}} &= \left(\mathbf{Z} \left(\frac{1}{\sqrt{D(S + \epsilon)}} \right) \times \mathbf{G} \right)^T \times \mathbf{F}_{\text{train}} & \text{WPCA}_{\text{test}} &= \left(\mathbf{Z} \left(\frac{1}{\sqrt{D(S + \epsilon)}} \right) \times \mathbf{G} \right)^T \times \mathbf{F}_{\text{test}} \end{aligned}$$

4.0.2 Extended Yale B

Extended Yale B face recognition database [13] consisted 2452 grey-scale images of 38 subjects. Each subject has nearly 64 samples. Each image was captured under controlled lighting conditions, with the size of 168×192 pixels. There are two ways to discriminate dataset into training and test set. The first way is randomly selecting half of the images per subject for the training and the other half for the test set. The second way is to divide into five subsets. Subset 1 is the training set, and Subset 2-5 will be the test sets. The second way is considered to be a challenging setup [35]. We chose the second way, for demonstrating our algorithm usefulness and potential for face recognition. Dividing into subsets was based on the name of the image. For instance, let's consider the image name 'yaleB01_P00A-010E+00.pgm'. The first part 'yaleB01_P00' refers to the subject labeled as 01' subject inside extended yale B dataset. The second part 'A-010' refers to the image azimuth value is -10°. The third part 'E+00' refers to the image elevation degree is +0°. If elevation degree kept same and azimuth value gets higher or lower degree, darker images occur (Figure 42).

In Figure 42 only azimuth values changed, and all images elevation degree is 0°. Starting from left to right the corresponding azimuth values are -120°, -95°, -70°, -50°, -25°, -10°, +10°, +50°, +75°, +95°, +120°. As absolute azimuth values increases, the image becomes harder to recognize. The same fact is also true for elevation degrees (Figure 43). In Figure 43, starting from left to right the corresponding elevation values are -35°, -20°, +20°, +45°, +90°. If both azimuth and elevation degrees are high, the recognizing face image is also hard (Figure 44).

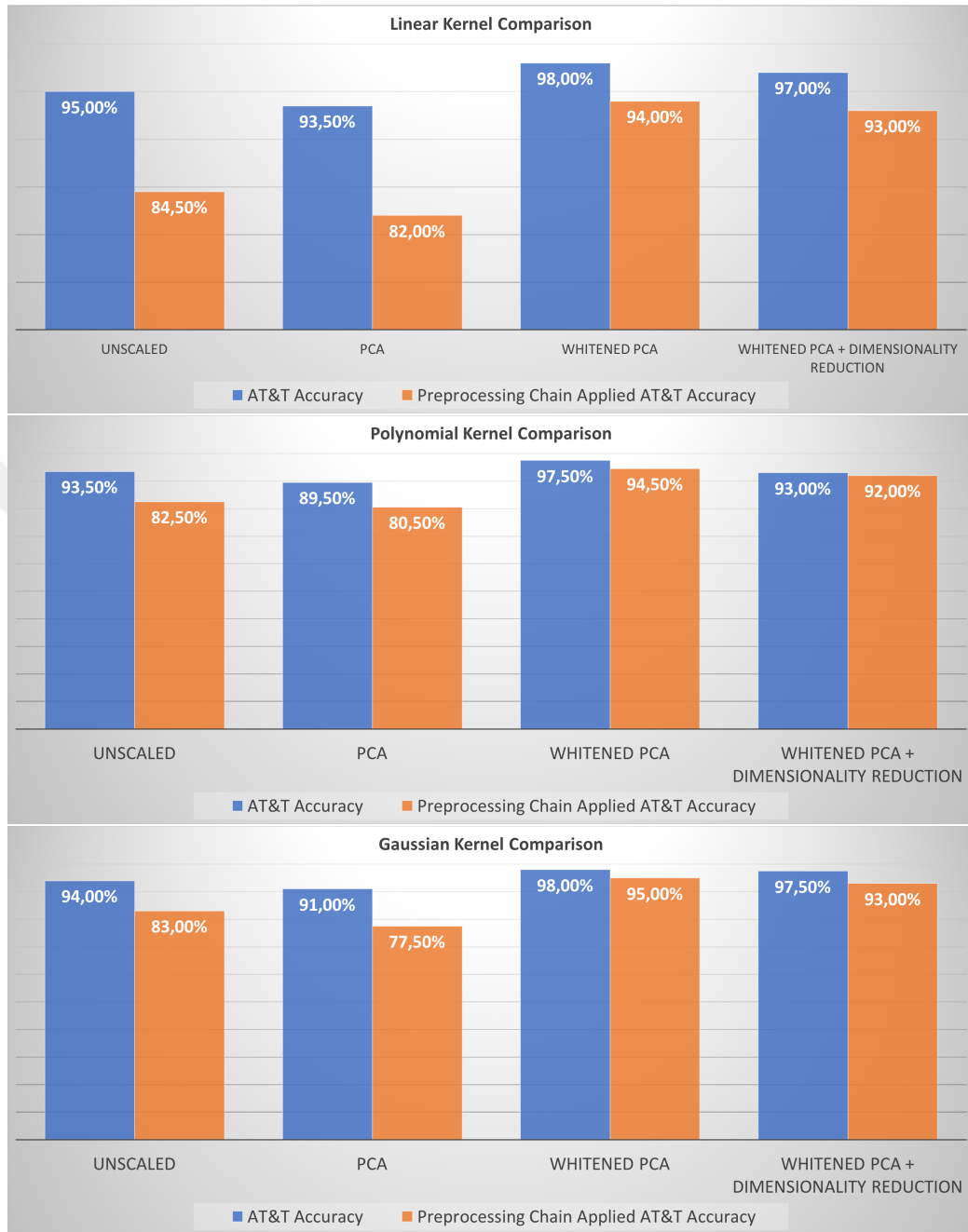


Figure 41: Comparing AT& T Kernel Results

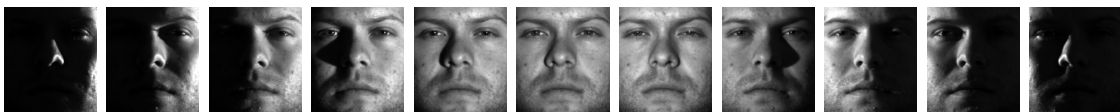


Figure 42: The Effect of Azimuth Degrees on Image, elevation degree is 0°

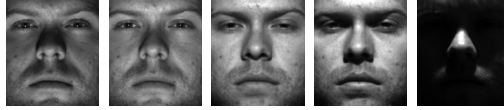


Figure 43: The Effect of Elevation Degrees on Image, azimuth degree is 0°

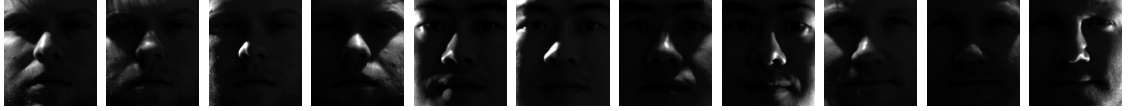


Figure 44: Azimuth and Elevation Degrees high Images

Liu et al. [35], Naseem et al. [36] and Tahir et al. [37] were used 2414 images instead of 2452. The reason was for each label there was an ambient image. (Figure 45).

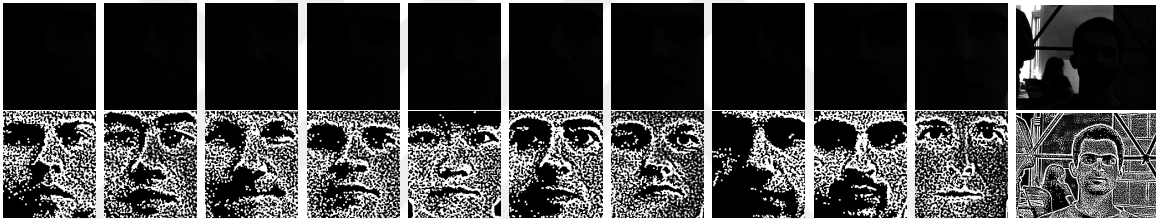


Figure 45: First 11 Class Ambient Images of Extended Yale B

They divided 2414 frontal face images into five subsets. Subset 1 consisting of nominal lighting condition images, Subsets 2 and 3, each consisting of slight-to-moderate luminance variation images, while subset 4 and 5, each consisting of depicting severe light variations images. Each subset number is shown in Table 6.

Table 6: Subset Numbers.

	# of Images
Subset 1	263
Subset 2	456
Subset 3	525
Subset 4	456
Subset 5	714

However, the explanation of subset division is not clear. For instance, what does

the term nominal lighting condition refers regarding azimuth and elevation? In the previous study [13], Subset 1 is up to 12° , Subset 2 is up to 25° , Subset 3 is up to 50° , and Subset 4 is up to 77° . We started with obtaining Subset 1. When we applied both azimuth and elevation values < 12 , we acquired 943 images. Some of the images were the examples of severe illumination (Figure 46).



Figure 46: Examples of Severe Illumination Images in Extended Yale B dataset

However, the subset 1 rule was under nominal lighting condition images were collected. So how do we separate the severe illuminated images from nominal lighting image? If we carefully look at the names of severe illuminated image ‘yaleB01_P00A-120E+00’ and nominal lighting image ‘yaleB01_P00A+010E+00’, the only difference is azimuth values. In the severe illuminated image, the azimuth value is -120° where nominal lighting azimuth value is $+10^\circ$. Therefore we added the rule $\text{azimuth} \geq 0$. When we applied $\text{azimuth} \geq 0$ and $\text{azimuth and elevation} < 12$, we acquired seven images per class which are 266 images in total. We continued with obtaining with Subset 2. When we applied both azimuth and elevation values $< 25^\circ$, we acquired 491 images. However, in previous studies, Subset 2 size is 453. None of the 491 images contains severely illuminated images. When we carefully examine, 38 images azimuth value is 0° . Therefore we added the rule $\text{azimuth} \neq 0$. When we applied $\text{azimuth} \neq 0$ and $\text{azimuth and elevation} \leq 25$, we obtained seven images per class which are 453 images. For Subset 3 and 4, we applied the same rule with [13], which is both azimuth and elevation values $\leq 50^\circ$, and $\leq 77^\circ$, respectively. The remaining images belong to Subset 5.

Same as in the previous studies [35], [36], the MSBS algorithm was trained on Subset 1 (263 samples) and tested on Subset 2-5. Subset 1 (Figure 47, 48) and Subset

2 (Figure 49, 50) are both images under nominal lighting condition, easily recognized by human eyes. Subset 3 (Figure 51, 52) is slightly darker compared to Subset 1 and 2 but still recognizable with a human eye. Subset 4 (Figure 53, 54) images were taken under severe lighting conditions, and few images are hard to recognize. Subset 5 (Figure 55) is the hardest set, some of the images are impossible to see or recognize with a human eye. The preprocessing chain enables Subset 5 (Figure 56) to be recognizable with a human eye.



Figure 47: Extended Yale B Subset 1 Examples **Figure 48:** Extended Yale B Preprocessing Chain Applied Subset 1

The results of Table 7 - 12 shows three facts. The first one is the preprocessing chain did improve the classification accuracy. For each kernel (Figure 58 - 61), the preprocessing chain accuracy is greater than the normal accuracy, except for Subset 2 polynomial kernel, where the Whitened PCA accuracies of both preprocessed and normal accuracies are same. (Table 8 and 11). The effect of the preprocessing chain is clearly can be seen in Figure 60 and 61. The second fact is the highest classification accuracies were obtained when Whitened PCA was applied. The third fact is, MSBS algorithm was higher than the previous results. (Table 13, Figure 57).

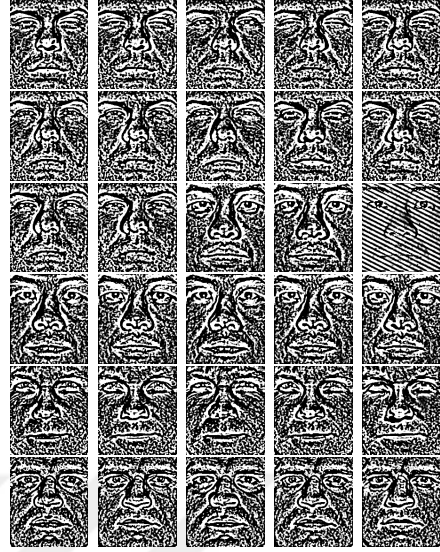


Figure 49: Extended Yale B Subset 2 Examples

Figure 50: Extended Yale B Preprocessing Chain Applied Subset 2



Figure 51: Extended Yale B Subset 3 Examples

Figure 52: Extended Yale B Preprocessing Chain Applied Subset 3

4.0.3 Georgia Tech Face Dataset

Georgia Tech Face Recognition database [14] consisted 750 colorful images of 50 subjects. Each subject having 15 samples. Each image was captured under nominal



Figure 53: Extended Yale B Subset 4 Examples **Figure 54:** Extended Yale B Preprocessing Chain Applied Subset 4

Table 7: Extended Yale B SVM Linear Kernel Results

Extended Yale B - 38 Classes		SVM Linear Kernel Accuracies						
Each class having around 64 samples Each Class Image 168 x 192 pixel	Method				Unscaled Accuracy	PCA Accuracy	Whitening Accuracy	Whitened PCA Accuracy
	r_1	r_2	K	Cell Size				
Test 2	2	1	4	7x7	99.5585%	99.5585% (10752→189)	99.5585%	99.5585% (10752→189)
Test 3	2	1	4	7x7	99.0476%	99.0476% (10752→189)	99.4286%	99.4286% (10752→189)
Test 4	2	1	4	7x7	52.8509%	52.8509% (10752→189)	76.5351%	76.3158% (10752→189)
Test 5	2	1	4	7x7	12.8852%	12.6050% (10752→189)	25.4902%	27.0308% (10752→189)

Table 8: Extended Yale B Polynomial Kernel Results

Extended Yale B - 38 Classes		SVM Polynomial Kernel Degree 3 Accuracies						
Each class having around 64 samples Each Class Image 168 x 192 pixel	Method				Unscaled Accuracy	PCA Accuracy	Whitening Accuracy	Whitened PCA Accuracy
	r_1	r_2	K	Cell Size				
Test 2	2	1	4	7x7	99.5585%	85.872% (10752→189)	99.7792%	96.0265% (10752→189)
Test 3	2	1	4	7x7	58.4762%	20% (10752→189)	94.0952%	44.7619% (10752→189)
Test 4	2	1	4	7x7	4.6053%	2.6316% (10752→189)	57.8947%	2.8509% (10752→189)
Test 5	2	1	4	7x7	2.6611%	2.6611% (10752→189)	28.5714%	2.6611% (10752→189)

lighting conditions, with the size of 640×480 pixels. We discriminate the dataset based on the previous studies [36] [71]; For each subject, first eight samples were placed into the training set and the rest placed into test set. Training set was consisted of 450 images (Figure 62) and test set was consisted of 300 images (Figure 63). All images

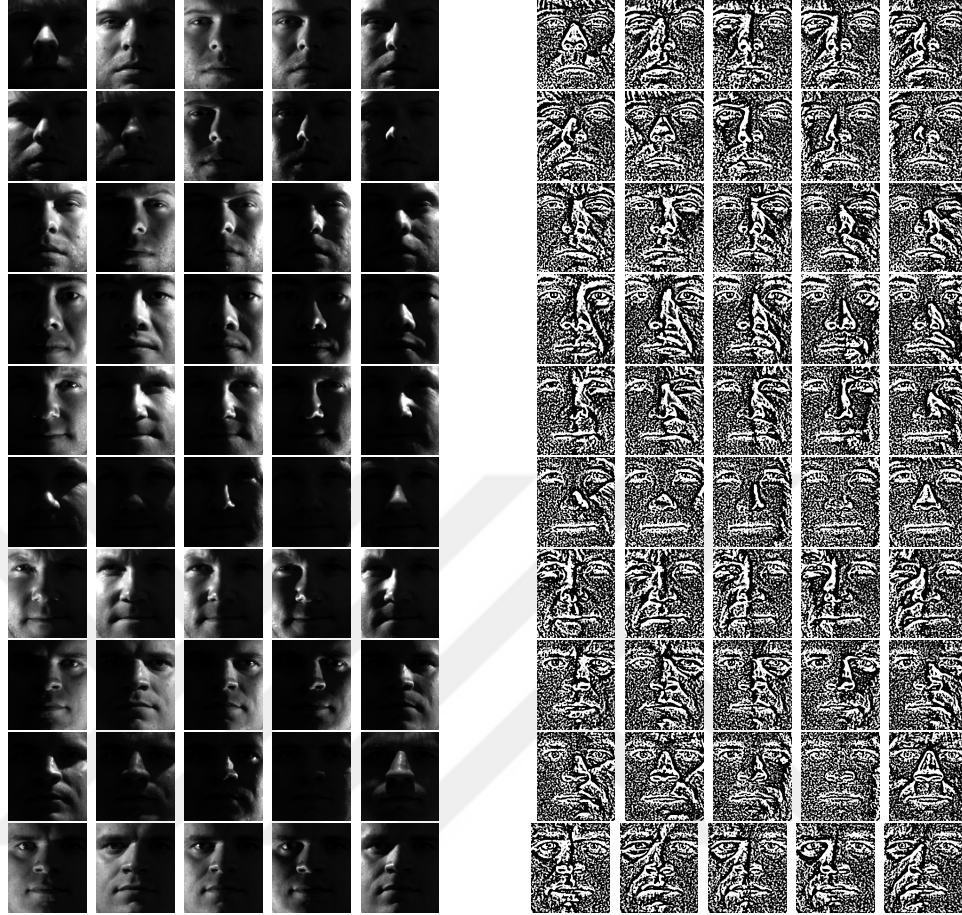


Figure 55: Extended Yale B Subset 5 Examples **Figure 56:** Extended Yale B Pre-processing Chain Applied Subset 5

Table 9: Extended Yale B SVM RBF Kernel Results

Extended Yale B - 38 Classes		SVM RBF Kernel Degree 3 Accuracies						
Each class having around 64 samples Each Class Image 168 x 192 pixel	Method				Unscaled Accuracy	PCA Accuracy	Whitening Accuracy	Whitened PCA Accuracy
	r_1	r_2	K	Cell Size				
Test 2	2	1	4	7x7	99.5585%	94.0397% (10752→189)	99.5585%	99.5585% (10752→189)
Test 3	2	1	4	7x7	98.4762%	91.619% (10752→189)	97.5238%	99.0476% (10752→189)
Test 4	2	1	4	7x7	57.8947%	55.0439% (10752→189)	50.8772%	72.5877% (10752→189)
Test 5	2	1	4	7x7	12.465%	23.3894% (10752→189)	17.0868%	23.3894% (10752→189)

are converted to gray-scale using the standard equation given below: [11]

$$I = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B. \quad (30)$$

In the equation, R represent the red channel, G represent green channel and B

Table 10: Preprocessing Chain Applied Extended Yale B SVM Linear Kernel Results

Extended Yale B - 38 Classes		Preprocessed Chain Applied SVM Linear Kernel Accuracies						
Each class having around 64 samples	Method				Unscaled Accuracy	PCA Accuracy	Whitening Accuracy	Whitened PCA Accuracy
	r ₁	r ₂	K	Cell Size				
Each Class Image 168 x 192 pixel								
Test 2	2	1	4	7x7	99.7792%	99.7792% (10752→194)	99.7792%	99.7792% (10752→194)
Test 3	2	1	4	7x7	99.619%	99.619% (10752→194)	99.619%	99.619% (10752→194)
Test 4	2	1	4	7x7	76.7544%	76.7544% (10752→194)	82.6754%	82.0175% (10752→194)
Test 5	2	1	4	7x7	58.9636%	59.8039% (10752→194)	64.7059%	64.7059% (10752→194)

Table 11: Preprocessing Chain Applied Extended Yale B Polynomial Kernel Results

Extended Yale B - 38 Classes		Preprocessed Chain Applied SVM Polynomial Degree 3 Kernel Accuracies						
Each class having around 64 samples	Method				Unscaled Accuracy	PCA Accuracy	Whitening Accuracy	Whitened PCA Accuracy
	r ₁	r ₂	K	Cell Size				
Each Class Image 168 x 192 pixel								
Test 2	2	1	4	7x7	99.7792%	99.5585% (10752→194)	99.7792%	98.0132% (10752→194)
Test 3	2	1	4	7x7	98.8571%	82.0952% (10752→194)	99.4286%	60%
Test 4	2	1	4	7x7	68.6404%	30.7018% (10752→194)	76.3158%	8.9912% (10752→194)
Test 5	2	1	4	7x7	48.4594%	22.1289% (10752→194)	58.2633%	6.0224% (10752→194)

Table 12: Preprocessing Chain Applied Extended Yale B SVM RBF Kernel Results

Extended Yale B - 38 Classes		Preprocessed Chain Applied SVM RBF Kernel Degree 3 Accuracies						
Each class having around 64 samples	Method				Unscaled Accuracy	PCA Accuracy	Whitening Accuracy	Whitened PCA Accuracy
	r ₁	r ₂	K	Cell Size				
Each Class Image 168 x 192 pixel								
Test 2	2	1	4	7x7	99.7792%	99.7792% (10752→194)	100%	99.7792% (10752→194)
Test 3	2	1	4	7x7	99.8095%	99.4286% (10752→194)	100%	99.619% (10752→194)
Test 4	2	1	4	7x7	91.6667%	83.114% (10752→194)	93.8596%	86.8421% (10752→194)
Test 5	2	1	4	7x7	82.493%	73.5294% (10752→194)	90.1961%	72.8291% (10752→194)

Table 13: Recognition rates of different methods on the Extended Yale B dataset

Method	Subset-2	Subset-3	Subset-4	Subset-5	Mean
PCA	98.5%	80.0%	15.8%	24.4%	54.7%
ADLBP (uniform pattern)	99.8%	89.5%	28.5%	12.5%	57.6%
LRC	100%	100%	83.27%	33.61%	79.2%
ADLBP	99.8%	99.6%	91.4%	67.1%	89.5%
RDLBP (uniform pattern)	99.8%	99.4%	91.9%	68.5%	89.9%
LBP_S (uniform pattern)	99.8%	99.6%	93.2%	77.7%	92.6%
LRC_Fused	100%	100%	88.97%	84.73%	93.4%
ADLBP_M (uniform pattern)	99.8%	99.6%	94.5%	88.7%	95.7%
MSBS	100%	100%	93.4%	90.2%	96%

represents the blue channel. After converting to grey-scale, we cropped image to size of 92×112 pixel [71]. We also applied preprocessing chain for both training (Figure 64) and test sets (Figure 65), to observe how preprocessed chain will effect

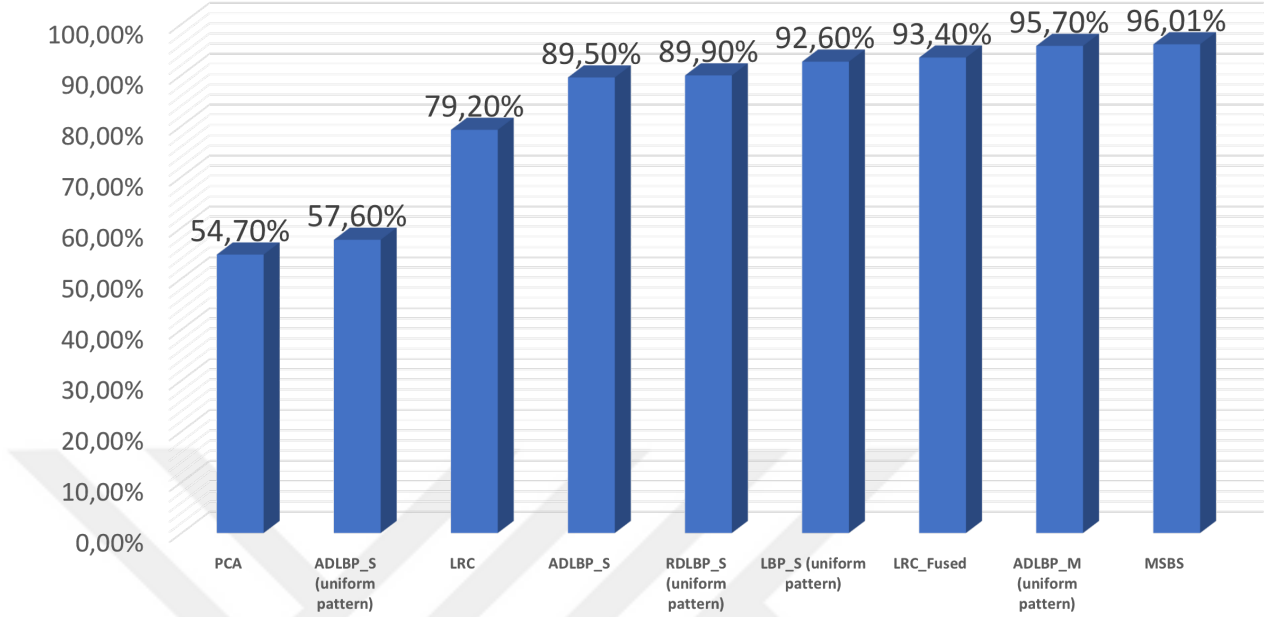


Figure 57: Comparing Extended Yale B Subset Means

the classification accuracy.

We tested our algorithm with LIB-SVM [70] with linear, polynomial and rbf kernels respectively. Polynomial and rbf degree were three and two-fold cross validation applied. The best results, for each kernel, highlighted bold in Table 14 and Table 15.

Table 14: Georgia Tech SVM Results

Georgia Tech- 50 Classes		SVM Accuracy on Test Set							
Each class having 15 samples Each Class Image 92 x 112 pixel		Method				Unscaled Accuracy	PCA Accuracy	Whitening Accuracy	Whitened PCA Accuracy
	r ₁	r ₂	K	Cell Size					
Linear Kernel	2	1	4	7x7	91.4286%	91.4286% (3584→283)	90.2857%	91.4286% (3584→283)	
Polynomial Kernel	2	1	4	7x7	90.2857%	85.7143% (3584→283)	91.1429%	91.1429% (3584→283)	
RBF Kernel	2	1	4	7x7	91.7143%	91.4286% (3584→283)	92.5714%	91.4286% (3584→283)	

Table 15: Preprocessing Chain Applied Georgia Tech SVM Results

Georgia Tech- 50 Classes		SVM Accuracy on Test Set							
Each class having 15 samples Each Class Image 92 x 112 pixel		Method				Unscaled Accuracy	PCA Accuracy	Whitening Accuracy	Whitened PCA Accuracy
	r ₁	r ₂	K	Cell Size					
Linear Kernel	2	1	4	7x7	91.4286%	92% (3584→310)	89.4286%	90.8571% (3584→310)	
Polynomial Kernel	2	1	4	7x7	91.1429%	87.1429% (3584→310)	87.7143%	76% (3584→310)	
RBF Kernel	2	1	4	7x7	92%	91.7143% (3584→310)	91.7143%	90.2857% (3584→310)	

MSBS result was compared with the previous results (Figure 66). MSBS classification accuracy (92.57%) is better than the previous results.



Figure 58: Comparing Extended Yale B Test2 Kernel Results

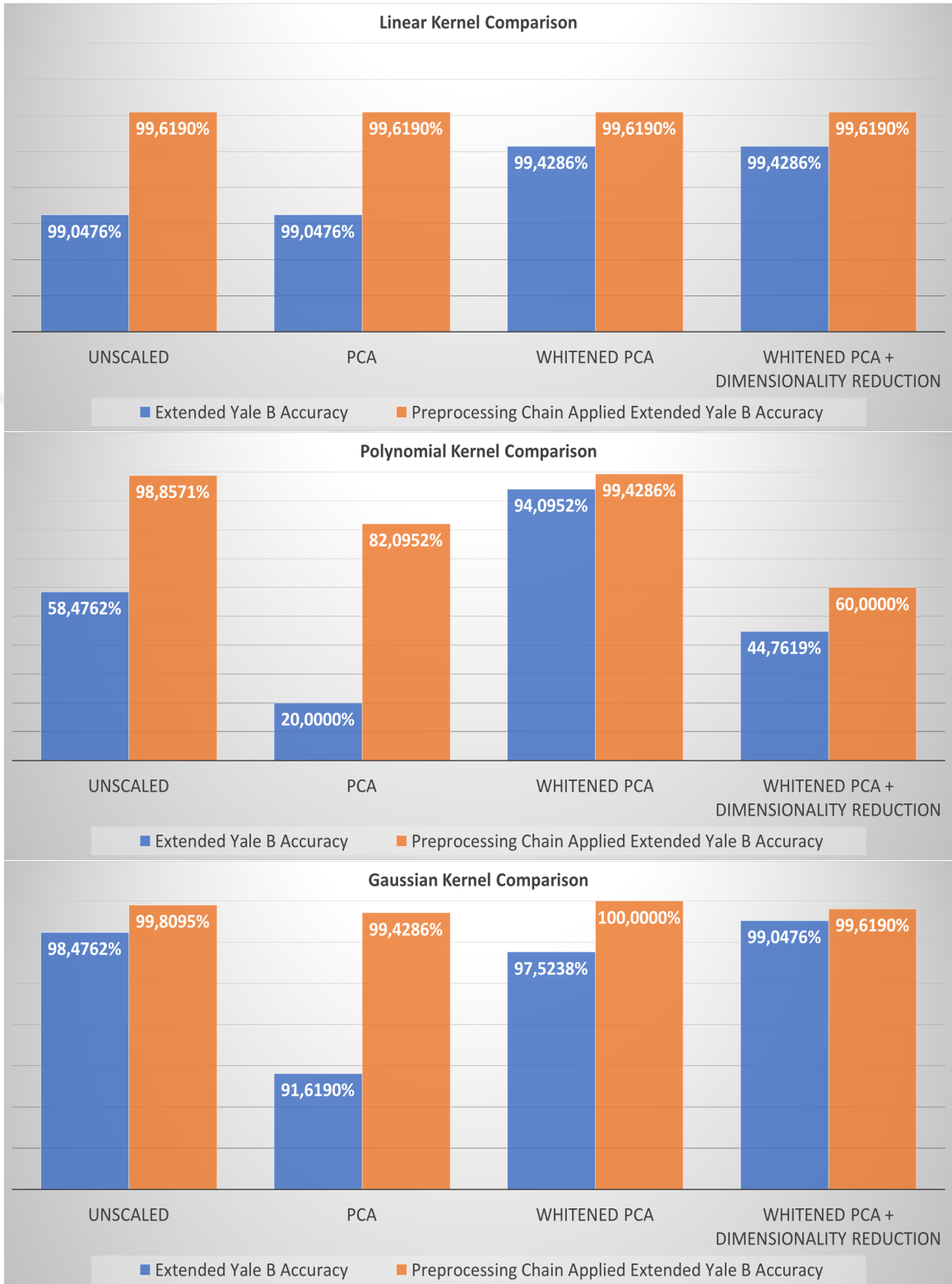


Figure 59: Comparing Extended Yale B Subset 3 Kernel Results

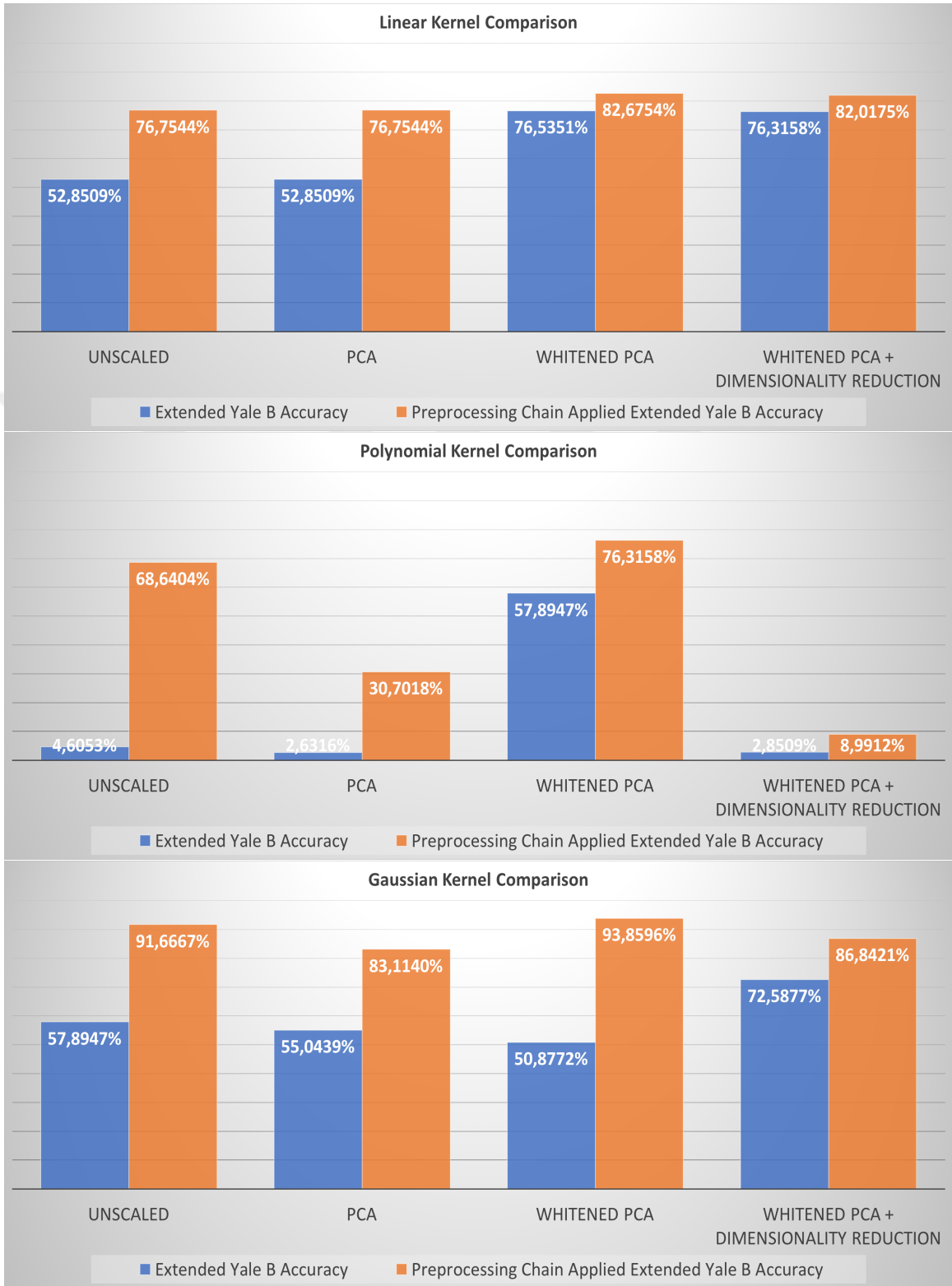


Figure 60: Comparing Extended Yale B Test4 Kernel Results

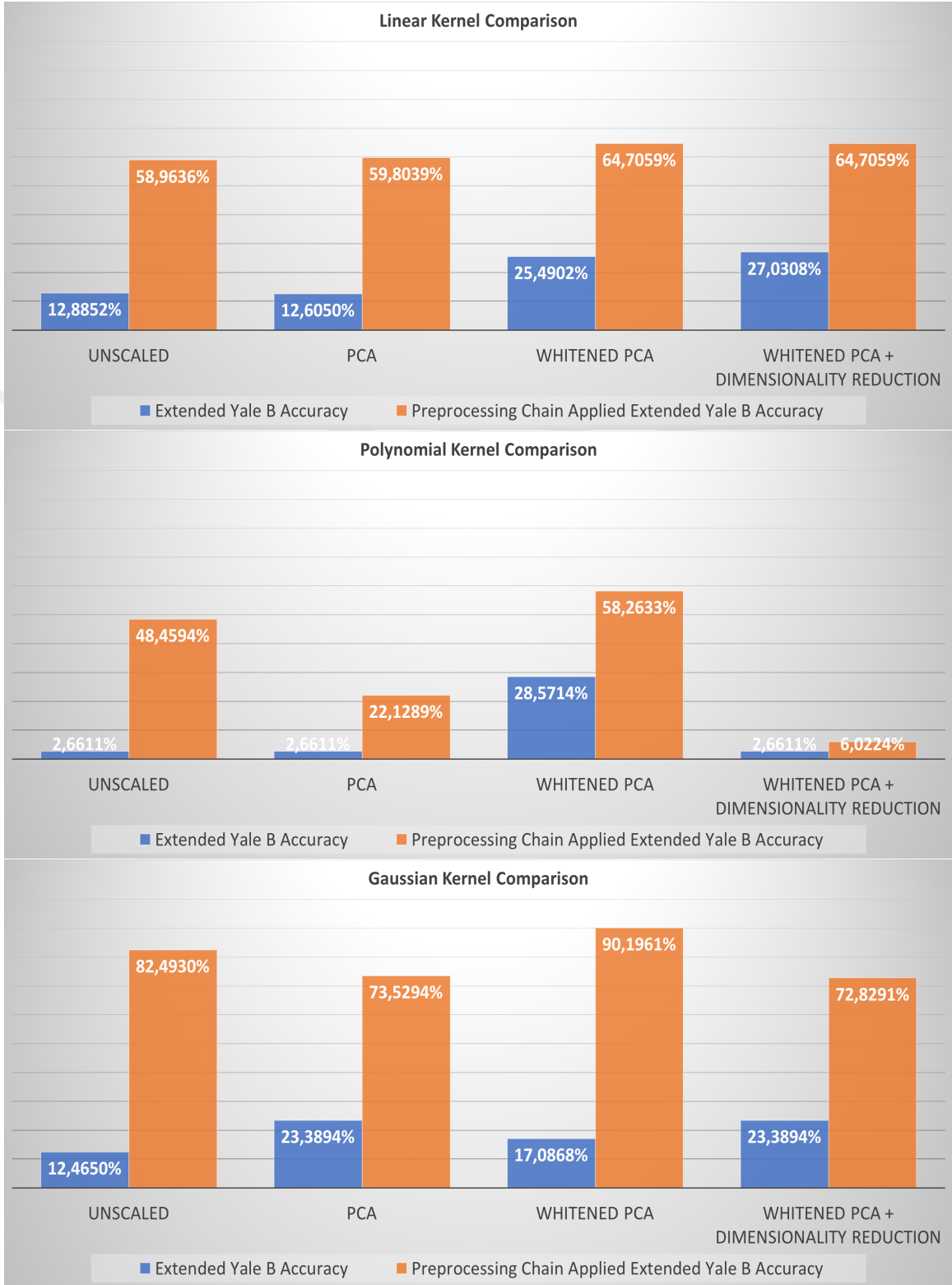


Figure 61: Comparing Extended Yale B Subset5 Kernel Results

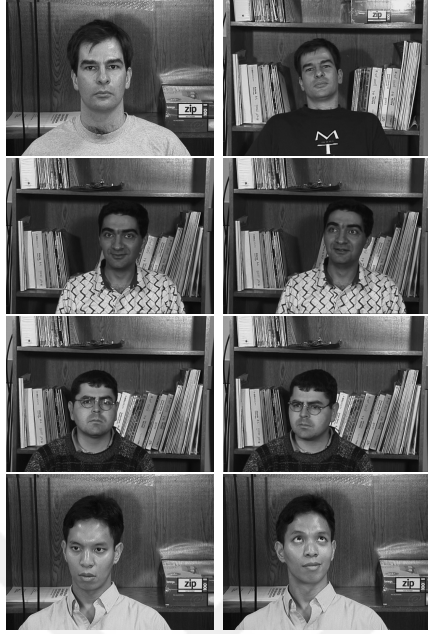


Figure 62: Georgia Tech Face Recognition Training Examples

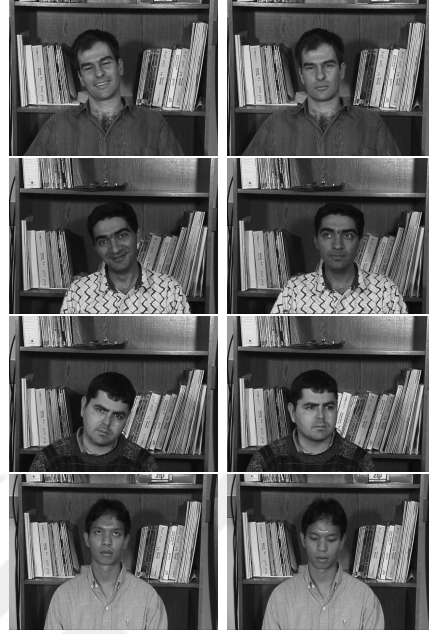


Figure 63: Georgia Tech Face Recognition Test Examples

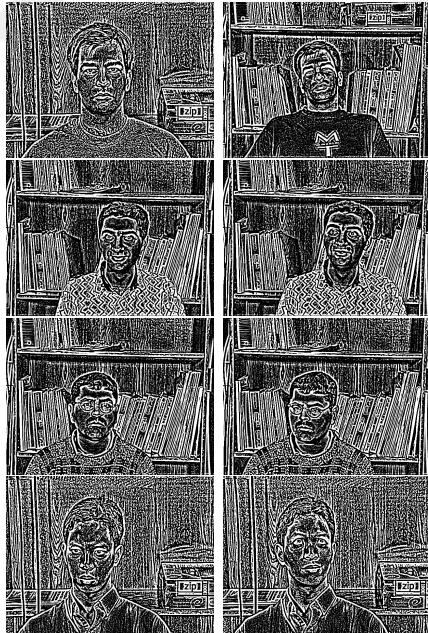


Figure 64: Preprocessed Chain Applied Georgia Tech Face Recognition Training Examples



Figure 65: Preprocessed Chain Applied Georgia Tech Face Recognition Test Examples

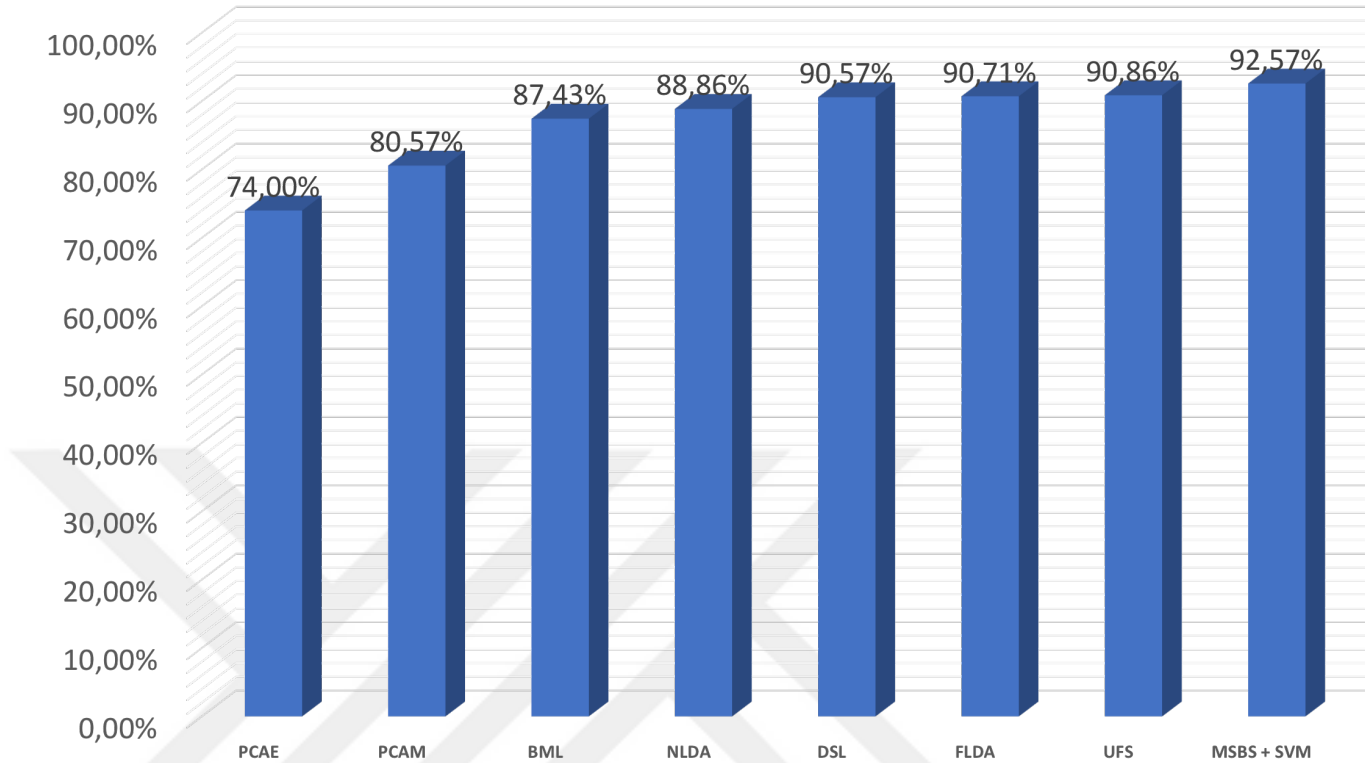


Figure 66: Comparing MSBS Results with other descriptors using Georgia Tech dataset

The results of table 14 and 15 shows that the WPCA increased the classification accuracy. Especially in table 14, the highest classification accuracy was obtained using gaussian kernel. Unlike the previous, AT&T and Extended Yale B datasets, Georgia Tech faces is not a cropped face images dataset. Each face image have a background. Therefore the background pixels might reduce the MSBS accuracy, by creating a noise in our feature vector. The preprocessed chain effect can be seen on PCA accuracy (Figure 67).

4.0.4 MNIST

MNIST, Optical Character Recognition (OCR) database [27] consisted 70.000 grey-scale images of 10 digits from 0-9. Each digit is having different size samples. There are 60.000 Training and 10.000 Test samples. As the name implied, MNIST is the



Figure 67: Comparing Georgia Tech Kernel Results

Modified version of NIST database. All the NIST data samples were normalized and centered to the 28 x 28 pixel size. MSBS algorithm was designed for face recognition. Therefore, MSBS was not expected to perform good on MNIST. However, MNIST was a test case for most of the algorithms [75].

MSBS was tested using fitcecoc [76] command, because LIB-SVM computation time takes long time on MNIST. The command ‘fitcecoc’ calculates error-correcting output codes (ECOC) for multiclass model. MSBS features were trained using SVM and the best results were obtained using the linear kernel (Table 16). We tested with linear and polynomial kernels, because radial-bases kernel (rbf) kernel takes long time for computation. We used the default polynomial kernel degree two. The preprocessing chain was also applied to the MNIST dataset (Figure 69), however preprocessed MNIST samples similar to MNIST samples (Figure 68), therefore we tested using MNIST samples.

Table 16: MNIST SVM Results

MNIST- 10 Classes		SVM Accuracy on Test Set						
Each class have different size samples Each Class Image 28 x 28 pixel	Method				Unscaled Accuracy	PCA Accuracy	Whitening Accuracy	Whitened PCA Accuracy
	r ₁	r ₂	K	Cell Size				
Linear Kernel	2	1	4	7x7	93.29%	91.79% (256→51)	93.26%	91.89% (256→51)
Polynomial Kernel	2	1	4	7x7	50.16%	59.53% (256→51)	11.35%	91.32% (256→51)

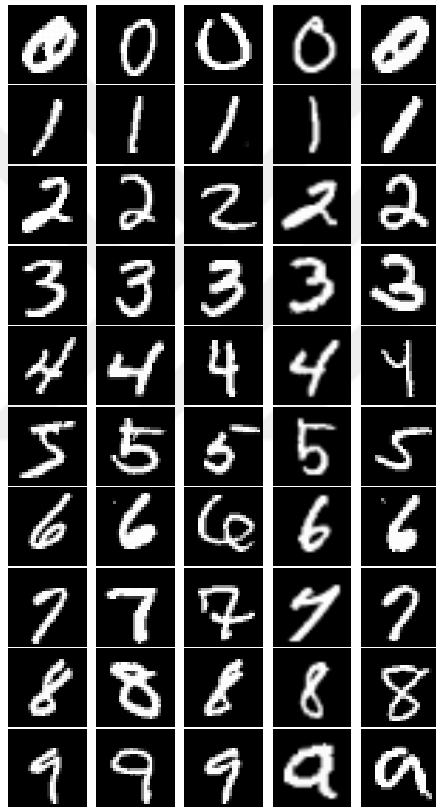


Figure 68: MNIST Examples

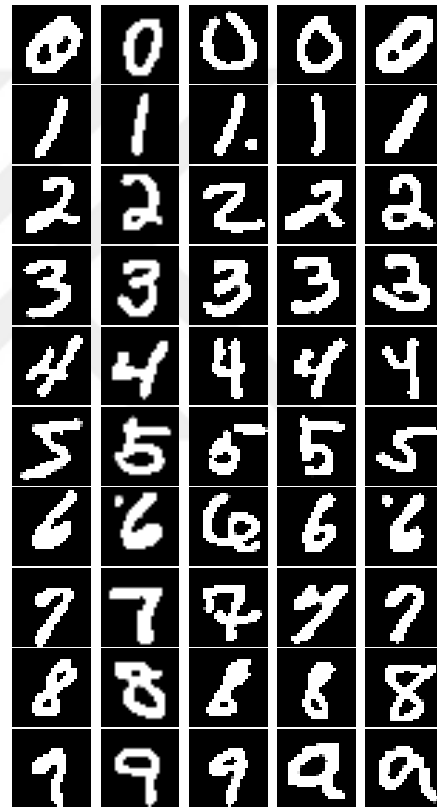


Figure 69: Preprocessed Examples

CHAPTER V

CONCLUSION

In a face recognition problem, the illumination is one of the challenges. Lighting on the image directly affects classification accuracy. The solution is either selecting illumination invariant visual descriptor or applying preprocessing method. Therefore, LBP is one of the solutions. LBP is illumination invariant, computationally simple and flexible at neighborhood pixel selection. However, LBP's has two significant limitations. The first significant limitation is selecting the uniform pattern. Researchers were discovered non-uniform patterns also contain the essential characteristic of the face image. Therefore neglecting or placing non-uniform pattern will be resulted in poor performance. In this study, all patterns were taken into consideration.

The second major limitation is the circular neighborhood. The circular neighborhood is one of the factors for the rotational invariant property. However, in face recognition problem, the rotational invariant property is not essential. The critical thing is extracting, characteristic (eye, nose, mouth, etc..) information. Since each face characteristic is in the different direction, with a different pixel value, researchers had named this fact as anisotropic information. MSBS captures anisotropic information by analyzing the neighborhood pixel relations, obtain pattern and give features to the classification algorithm.

Applying PCA usually increases the classification accuracy. However, in face recognition problem, researchers were discovered that PCA eigenvectors mostly encode illumination rather than characteristic information, instead Whitened PCA was suggested. The Whitening process normalizes the illumination effect and, PCA decreases the dimension; as a result, the performance increases. Therefore Whitened

PCA is used in this study. Most of the best results were obtained using both Whitening and Whitened PCA.

In this study, LBP's neighborhood topology variants are studied, and a new LBP based face recognition algorithm proposed. The proposed algorithm, named Multi-scale Binary Similarity are evaluated on standard datasets. MSBS classification accuracy compared with the previous methods, and its performance is better than most of the methods in literature.

As a future work, we will investigate the neighbor template extraction in a broader manner. We, currently, linearly score for similar neighbor behaviors. Scoring function can be incorporated a distance notion. Moreover, neighbor template extraction can be specialized for partial rotational invariance. While rotational invariance is not always desired, there are certain scenarios where it matters. For those special cases, we will investigate MSBS neighborhood template extraction with partial rotational invariance.

Bibliography

- [1] M. Pietikäinen, A. Hadid, G. Zhao, and T. Ahonen. *Computer Vision Using Local Binary Patterns*. Computational Imaging and Vision. Springer London, 2011.
- [2] Albert Ali Salah and Theo Gevers. *Computer analysis of human behavior*. Springer, 2011.
- [3] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Ed)*. Wiley, 2001.
- [4] X. Tan and B. Triggs. Enhanced local texture feature sets for face recognition under difficult lighting conditions. *IEEE Transactions on Image Processing*, 19(6):1635–1650, June 2010.
- [5] X. Li, W. Hu, Z. Zhang, and H. Wang. Heat kernel based local binary pattern for face representation. *IEEE Signal Processing Letters*, 17(3):308–311, March 2010.
- [6] Lior Wolf, Tal Hassner, and Yaniv Taigman. Descriptor based methods in the wild. In *Workshop on faces in real-life images: Detection, alignment, and recognition*, 2008.
- [7] Weilin Huang and Hujun Yin. Robust face recognition with structural binary gradient patterns. *Pattern Recognition*, 68:126 – 140, 2017.
- [8] N. Ilmi, W. T. A. Budi, and R. K. Nur. Handwriting digit recognition using local binary pattern variance and k-nearest neighbor classification. In *2016 4th International Conference on Information and Communication Technology (ICoICT)*, pages 1–5, May 2016.

- [9] Loris Nanni, Alessandra Lumini, and Sheryl Brahnam. Local binary patterns variants as texture descriptors for medical image analysis. *Artificial Intelligence in Medicine*, 49(2):117 – 125, 2010.
- [10] Solmaz Abbasi and Farshad Tajeripour. Detection of brain tumor in 3d mri images using local binary patterns and histogram orientation gradient. *Neurocomputing*, 219:526 – 535, 2017.
- [11] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, Jul 2002.
- [12] Ferdinando S Samaria and Andy C Harter. Parameterisation of a stochastic model for human face identification. In *Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on*, pages 138–142. IEEE, 1994.
- [13] Athinodoros S. Georghiades, Peter N. Belhumeur, and David J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE transactions on pattern analysis and machine intelligence*, 23(6):643–660, 2001.
- [14] ”Georgia Tech Face Database,” http://www.anefian.com/research/face_reco, 2007.
- [15] Timo Ojala, Matti Pietikäinen, and David Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, 29(1):51 – 59, 1996.
- [16] Kenneth I Laws. Textured image segmentation. Technical report, University of Southern California Los Angeles Image Processing INST, 1980.

- [17] Li Wang and DC He. A new statistical approach for texture analysis. *Photogrammetric Engineering and Remote Sensing*, 56(1):61–66, 1990.
- [18] Li Wang and Dong-Chen He. Texture classification using texture spectrum. *Pattern Recognition*, 23(8):905 – 910, 1990.
- [19] Phil Brodatz. *Textures: a photographic album for artists and designers*. Dover Pubns, 1966.
- [20] Philippe P. Ohanian and Richard C. Dubes. Performance evaluation for four classes of textural features. *Pattern Recognition*, 25(8):819 – 833, 1992.
- [21] Solomon Kullback. *Information theory and statistics*. Courier Corporation, 1997.
- [22] David Harwood, Timo Ojala, Matti Pietikäinen, Shalom Kelman, and Larry Davis. Texture classification by center-symmetric auto-correlation, using kullback discrimination of distributions. *Pattern Recognition Letters*, 16(1):1 – 10, 1995.
- [23] Wenchao Zhang, Shiguang Shan, Wen Gao, Xilin Chen, and Hongming Zhang. Local gabor binary pattern histogram sequence (lgbphs): a novel non-statistical model for face representation and recognition. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 1, pages 786–791 Vol. 1, Oct 2005.
- [24] Shu Liao and Albert C. S. Chung. *Face Recognition by Using Elongated Local Binary Patterns with Average Maximum Distance Gradient Magnitude*, pages 672–679. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.
- [25] S. Liao, M. W. K. Law, and A. C. S. Chung. Dominant local binary patterns for texture classification. *IEEE Transactions on Image Processing*, 18(5):1107–1118, May 2009.

- [26] P. Jonathon Phillips, Harry Wechsler, Jeffery Huang, and Patrick J. Rauss. The feret database and evaluation procedure for face-recognition algorithms. *Image and Vision Computing*, 16(5):295 – 306, 1998.
- [27] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.
- [28] Sheryl Brahnham, Loris Nanni, and Randall Sexton. *Introduction to Neonatal Facial Pain Detection Using Common and Advanced Face Classification Techniques*, pages 225–253. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.
- [29] Amina Chebira, Yann Barbotin, Charles Jackson, Thomas Merryman, Gowri Srinivasa, Robert F Murphy, and Jelena Kovačević. A multiresolution approach to automated classification of protein subcellular location images. *BMC bioinformatics*, 8(1):210, 2007.
- [30] Jan Jantzen, Jonas Norup, Georgios Dounias, and Beth Bjerregaard. Pap-smear benchmark data for pattern classification. *Nature inspired Smart Information Systems (NiSIS 2005)*, pages 1–9, 2005.
- [31] A. Petpon and S. Srisuk. Face recognition with local line binary pattern. In *2009 Fifth International Conference on Image and Graphics*, pages 533–539, Sept 2009.
- [32] Gary B. Huang, Vidit Jain, and Erik Learned-Miller. Unsupervised joint alignment of complex images. In *ICCV*, 2007.
- [33] W. Gao, B. Cao, S. Shan, X. Chen, D. Zhou, X. Zhang, and D. Zhao. The caspeal large-scale chinese face database and baseline evaluations. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 38(1):149–161, Jan 2008.

- [34] Marko Heikkilä, Matti Pietikäinen, and Cordelia Schmid. Description of interest regions with local binary patterns. *Pattern Recognition*, 42(3):425 – 436, 2009.
- [35] L. Liu, P. Fieguth, G. Zhao, and M. Pietikäinen. Extended local binary pattern fusion for face recognition. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 718–722, Oct 2014.
- [36] I. Naseem, R. Togneri, and M. Bennamoun. Linear regression for face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(11):2106–2112, Nov 2010.
- [37] M. A. Tahir, C. H. Chan, J. Kittler, and A. Bouridane. Face recognition using multi-scale local phase quantisation and linear regression classifier. In *2011 18th IEEE International Conference on Image Processing*, pages 765–768, Sept 2011.
- [38] Li Liu, Lingjun Zhao, Yunli Long, Gangyao Kuang, and Paul Fieguth. Extended local binary patterns for texture classification. *Image and Vision Computing*, 30(2):86 – 99, 2012.
- [39] Manik Varma and Andrew Zisserman. A statistical approach to material classification using image patch exemplars. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(11):2032–2047, November 2009.
- [40] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S. Emer. Efficient processing of deep neural networks: A tutorial and survey. *CoRR*, abs/1703.09039, 2017.
- [41] Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, and Fuad E. Alsaadi. A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11 – 26, 2017.

- [42] Hubel D. H. and Wiesel T. N. Receptive fields and functional architecture of monkey striate cortex. *The Journal of Physiology*, 195(1):215–243.
- [43] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, Apr 1980.
- [44] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, and Gang Wang. Recent advances in convolutional neural networks. *CoRR*, abs/1512.07108, 2015.
- [45] Ching-Huei Wang and Sargur N. Srihari. A framework for object recognition in a visually complex environment and its application to locating address blocks on mail pieces. *International Journal of Computer Vision*, 2(2):125–151, Jun 1988.
- [46] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, R. E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Handwritten digit recognition with a back-propagation network. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 396–404. Morgan-Kaufmann, 1990.
- [47] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA, 1986.
- [48] Andrew Ng. Cost function. <https://www.coursera.org/learn/machine-learning/supplement/afqGa/cost-function>, 2015. Accessed 07/05/18.
- [49] Andrew Ng. Linear Regression Cost Function <https://www.coursera.org/learn/machine-learning/lecture/rkTp3/cost-function>, 2015 Accessed 07/05/18.

- [50] Y. LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Müller, E. Säckinger, P. Simard, and V. Vapnik. Comparison of learning algorithms for handwritten digit recognition. In *INTERNATIONAL CONFERENCE ON ARTIFICIAL NEURAL NETWORKS*, pages 53–60, 1995.
- [51] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.
- [52] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.
- [53] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.
- [54] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.
- [55] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, June 2015.
- [56] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016.
- [57] Yann LeCun, Leon Bottou, Genevieve B. Orr, and Klaus Robert Müller. *Efficient BackProp*, pages 9–50. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

- [58] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA, 2010. Omnipress.
- [59] Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 111–118, USA, 2010. Omnipress.
- [60] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng. End-to-end text recognition with convolutional neural networks. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 3304–3308, Nov 2012.
- [61] Yichuan Tang. Deep learning using support vector machines. *CoRR*, abs/1306.0239, 2013.
- [62] Felix Juefei-Xu, Vishnu Naresh Boddeti, and Marios Savvides. Local binary convolutional neural networks. *CoRR*, abs/1608.06049, 2016.
- [63] M. Xi, L. Chen, D. Polajnar, and W. Tong. Local binary pattern network: A deep learning approach for face recognition. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3224–3228, Sept 2016.
- [64] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.
- [65] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

- [66] X. Wang, T. X. Han, and S. Yan. An hog-lbp human detector with partial occlusion handling. In *2009 IEEE 12th International Conference on Computer Vision*, pages 32–39, Sept 2009.
- [67] "anisotropic." Merriam-Webster.com. 2018. <https://www.merriam-webster.com> (14 May 2018).
- [68] E. Nowak and F. Jurie. Learning visual similarity measures for comparing never seen objects. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2007.
- [69] J. Aizan, E. C. Ezin, and C. Motamed. A face recognition approach based on nearest neighbor interpolation and local binary pattern. In *2016 12th International Conference on Signal-Image Technology Internet-Based Systems (SITIS)*, pages 76–81, Nov 2016.
- [70] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>
- [71] X. Jiang, B. Mandal, and A. Kot. Eigenfeature regularization and extraction in face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(3):383–394, March 2008.
- [72] Hieu V Nguyen, Li Bai, and Linlin Shen. Local gabor binary pattern whitened pca: A novel approach for face recognition from single image per person. In *International Conference on Biometrics*, pages 269–278. Springer, 2009.
- [73] Z. Lei, M. Pietikäinen, and S. Z. Li. Learning discriminant face descriptor. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(2):289–302, Feb 2014.

- [74] N. S. Vu and A. Caplier. Enhanced patterns of oriented edge magnitudes for face recognition and image matching. *IEEE Transactions on Image Processing*, 21(3):1352–1365, March 2012.
- [75] L. Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, Nov 2012.
- [76] MATLAB. *version R2016b*. Natick, Massachusetts: The MathWorks Inc., 2016.

VITA

Ahmet Tavlı completed his Bachelor of Science degree in Computer Engineering department at Yaşar University, İzmir. His specialization was Computer Vision. He is pursuing his Master of Science in Computer Science department at Özyeğin University, and his studies are concentrated on Object Recognition. His masters' thesis work was supported by the TUBITAK 1007 - Kgys Developing Smart Software.