YAŞAR UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

MASTER THESIS

# DEMAND PREDICTION IN CLOTHING INDUSTRY WITH USING NEURAL NETWORKS

CANER KIVANÇ HEKİMOĞLU

THESIS ADVISOR: ASSOC. PROF. DR. MEHMET SÜLEYMAN ÜNLÜTÜRK

COMPUTER ENGINEERING MASTERS PROGRAM

PRESENTATION DATE: 26.12.2017

BORNOVA / İZMİR
DECEMBER 2017

We certify that, as the jury, we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

**Jury Members:** **Signature:**

Assoc. Prof. Dr. Mehmet Süleyman ÜNLÜTÜRK
Yaşar University ........................

Asst. Prof. Dr. İbrahim ZİNCİR
Yaşar University ........................

Asst. Prof. Dr. Samsun Başarıcı
Aydın Adnan Menderes University ........................

---------------------------------------------------------------

Prof. Dr.Cüneyt GÜZELİŞ
Director of the Graduate School

## ABSTRACT

DEMAND PREDICTION IN CLOTHING INDUSTRY WITH USING NEURAL NETWORKS

Hekimoğlu, Caner Kıvanç

MSc. Computer Engineering

Advisor: Assoc. Prof. Dr. Mehmet Süleyman ÜNLÜTÜRK

December 2017

This research presents an artificial neural network that can be used as a fashion consultant for fashion design companies. In this study, machine learning techniques have been employed in the fashion domain. A software application that using neural network has been implemented as a fashion consultant for accepting or rejecting garment design samples. Moreover, this model learns customer preferences through the usage of feed-forward neural network with backpropagation and SVM model. This neural network application scores the company's customized fashion designs based on its customer's preferred fashion style history of garment purchases. According to the score, the company can decide to send or not to send the garment design sample to its customer for the review process, which saves a lot of time and source for the company. Our study results demonstrate that feed forward neural network with backpropagation and SVM model can be used effectively as a fashion consultant.

**Key Words:** Success Rate, Prediction, Artificial Neural Networks, Textile

# ÖZ

HAZIR GİYİM SEKTÖRÜ İÇİN TASARLANAN MODELLERİN BEĞENİSİNİN YAPAY SİNİR AĞLARI KULLANILARAK ÖNGÖRÜLMESİ

Hekimoğlu, Caner Kıvanç

Yüksek Lisans Tezi, Bilgisayar Mühendisliği

Danışman: Doç Dr. Mehmet Süleyman Ünlütürk

Aralık 2017

Bu araştırma, moda tasarım şirketleri için bir moda danışmanı olarak kullanılabilecek bir yapay sinir ağı sunmaktadır. Bu çalışmada, makine öğrenme teknikleri moda alanında kullanılmıştır. Sinir ağları tekniğiyle geliştirilen bir yazılım, hazır giyim tasarım örneklerini kabul veya reddetmek için moda danışmanı olarak uygulanmıştır. Ayrıca, bu model, geri yayılımlı sinir ağı devresi ve SVM modeli kullanarak müşteri tercihlerini öğrenmektedir. Bu sinir ağı uygulaması, müşterinin geçmişinde tercih ettiği moda tarzına dayanarak, müşteriye özel olarak hazırlanan özel moda tasarımlarını puanlandırmaktadır. Skora göre, şirket; giyim tasarım örneğini inceleme süreci için müşterisine göndermeye veya göndermemeye karar verebilir ve bu karar şirkete zaman kazandırır ve kaynak harcamasını azaltır. Çalışmanın sonuçlarına göre, geri yayılım sinir ağı ve SVM modeli bir moda danışmanı olarak etkin bir şekilde kullanılabilir.

**Anahtar Kelimeler:** Beğeni, Tahminleme, Yapay Sinir Ağları, Tekstil, Renk Analizi

## ACKNOWLEDGEMENTS

First of all, I would like to thank my supervisor Mehmet Süleyman ÜNLÜTÜRK for his guidance and patience during this study.

I would like to express my enduring love to my parents, who are always supportive, loving and caring to me in every possible way in my life.

I gratefully thank Kamil BİLİR for providing necessary garment sample data for this project.

I would like to thank all my research assistant friends at Yaşar University who always supported me and helped me with best of their abilities.

Caner Kıvanç Hekimoğlu

Izmir, December 2017

## TEXT OF OATH

I declare and honestly confirm that my study, titled “DEMAND PREDICTION IN CLOTHING INDUSTRY WITH USING NEURAL NETWORKS” and presented as a Master’s Thesis, has been written without applying to any assistance inconsistent with scientific ethics and traditions. I declare, to the best of my knowledge and belief, that all content and ideas drawn directly or indirectly from external sources are indicated in the text and listed in the list of references.

Caner Kıvanç Hekimoğlu

Signature

…………………………………..

December 26,2017

# TABLE OF CONTENTS

# LIST OF FIGURES

## LIST OF TABLES

## SYMBOLS AND ABBREVIATIONS

ABBREVIATIONS:

| | | |
|---|---|---|
| SVM | - | Support Vector Machine |
| NN | - | Neural Network |
| GF | - | Gabor Filters |
| RM | - | Red Value’s Mean Value |
| RS | - | Red Value’s Standard Deviation |
| GS | - | Green Value’s Mean Value |
| GM | - | Green Value’s Standard Deviation |
| BS | - | Blue Value’s Mean Value |
| BM | - | Blue Value’s Standard Deviation |
| IGA | - | Interactive Genetic Algorithm |

# CHAPTER 1
# INTRODUCTION

International fashion design companies always try to create a new trend to reach customers' demands like other competitors in the business. That's why they need competitive and innovative fashion design teams. Those design companies make product designs for big companies that are in the ready-to-wear clothing sector. Regardless of how big the companies are they always need to get the special series design, retail clothes, and ready-to-wear clothes. For this process, these big companies trace different methodologies for their shortcomings of the jobs. One of the ways is outsourcing their design jobs to partially smaller fashion design companies. These smaller fashion design companies create special series, thematic series or seasonal series of clothing with respect to demanded request from the bigger company. All garment designs are inspected by the fashion critics and the final decisions for each garment will be made by this qualified people.

For the scope of this thesis, modeling of acceptance; sales volume and profitability will be realized with a system that learns the modeling decisions of past years.

For the scope of this thesis, model of acceptance organized by information of accepted and rejected samples of garments that have a certain volume of sales in the past years and it will continue to learn with future garment sample information. The buying process consists of these steps;

- Fashion design company transmits garment design to the customer,
- Customer evaluates the design through images and decides for physical samples
- Fashion design company manufactures the sample of design and sends samples
- Customer companies' fashion evaluators rate the product and decide between buying the sample or not.

This thesis aims to make this process shorter, more efficient and more educated with the assistance of the decision-making model that is proposed in this paper.

Also, international fashion design companies who sell garment designs to other big fashion design companies have problems with these topics:

- Subjective evaluation methods lower the profitability
- Difficulty of understanding the customer's decision-making methods in different sale periods
- Desire to prevent unnecessary sample production for rejected garment samples
- Desire to learn the customer decision-making process or generate a model

Current model on this thesis suggests a solution to these problems to improve garment design process. In case of using this model properly, international fashion design companies will be able to save human resources, time and garment material. Furthermore, with better acceptance rate, the trust of the client to the company will be increased.

The rest of the thesis is organized as follows: General information and background of the materials and algorithms that are used in this thesis are given in section 2. Previous studies about the fashion sector using artificial neural networks and other methodologies are explained in section 3. In section 4, system architecture of the methods which are developed for the thesis, the dataset and information gathering techniques are explained and clarified. In the last section, general evaluation of the system and future works are summarized.

CHAPTER 2

BACKGROUND

## 2.1. ARTIFICIAL NEURAL NETWORKS

Artificial neural networks are a computational model based on biological neural networks. It consists of an interconnected group of artificial neurons and processes information using a connectionist approach to computation. In most cases, an ANN is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase.(Gardner & Dorling, 1998) In figure 2.1 simple neural network model is shown.

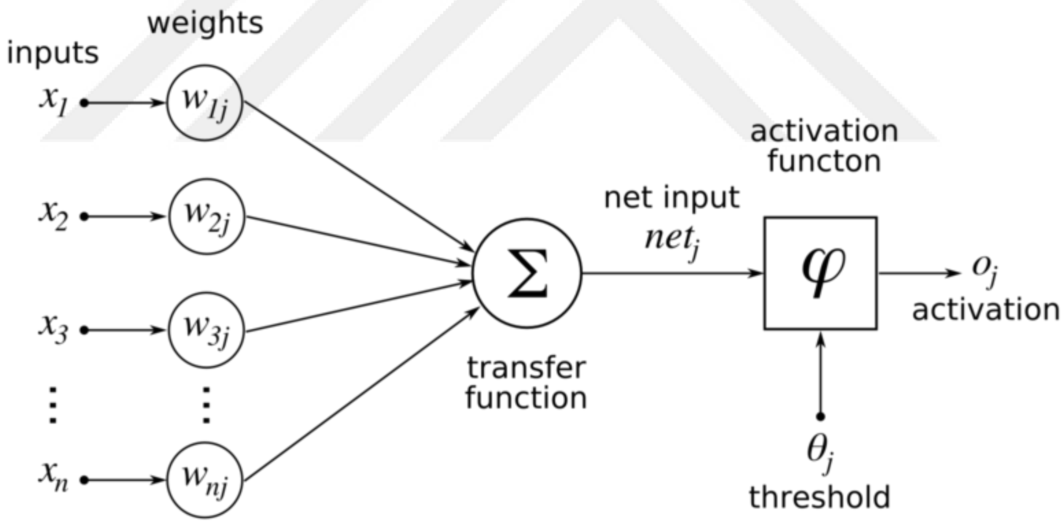


**Figure 2.1** Simple neural network model

### 2.1.1. Basic Elements of a Neural Network

Below the basic elements involved in neural networks are explained

### 2.1.2. Layers

Most of the neural networks have three basic layers. Input, hidden and output layers. The input units number in input layer depends on types and number of features in the dataset. The unit number in hidden layer can be determined by the user. Output unit number in output layer depends on the result set of the output in a data set. (Larose, 2005)

#### 2.1.2.1. Input Layer

The input layer is a hypothetical layer. It retrieves the property values from the data set and passes these values to the hidden layer without any processing.

#### 2.1.2.2. Hidden Layer

Units in the hidden layer show the source of nonlinearity in the behavior of NN, due to their nonlinear behavior. Whenever a new hidden layer is added to the neural network, local minimum points to which the error function can be inserted in the training process are added. For this reason, a single hidden layer network is created first then if this network is insufficient then a new hidden layer may be added to the neural network.(Gries & Schneider, 2008).

#### 2.1.2.3. Output Layer

For binary classification problems, it is common to use a single unit to which a "threshold" value is assigned that will classify classes in the output layer. A single output unit can be used for problems with more than one class. In many networks, computing units produce output in equation 2.1. Where y is the output, w is weight value of a node in weight layer, x is input value in the input layer.

$$y = f\left(\sum_{i} w_i \, x_i\right) \tag{2.1}$$

### 2.1.3. Weights

Each connection between nodes has a weight (e.g., $w_{1j}$ in Figure 2.1) associated with it. At initialization, these weights are randomly assigned to values between zero and one. These values formed when neural networks trained with learning method.

### 2.1.4. Transfer Function

The behavior of a neural network depends on both the weights and the transfer function that is specified for the units. The most common transfer function is weighted sum function. Weighted multiplication, maximum, minimum or cumulative addition methods can also be used. In equation 2.2 weighted sum function is shown. Where net input y is the transfer function's value, w is weight value of a node in weight layer, x is input value in the input layer.

$$net\ input_j = \sum_i w_{ij}\, x_{ij} \tag{2.2}$$

### 2.1.5. Activation Function

This function processes the net entry into the cell to determine if the cell will produce the corresponding input. The activation function is usually chosen to be a non-linear function. The most common activation function is sigmoid function. The equation of sigmoid function is shown in equation 2.3.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$

In figure 2.2 sigmoid function's graphic is shown. The sigmoid function returns an output bounded between 0 and 1.

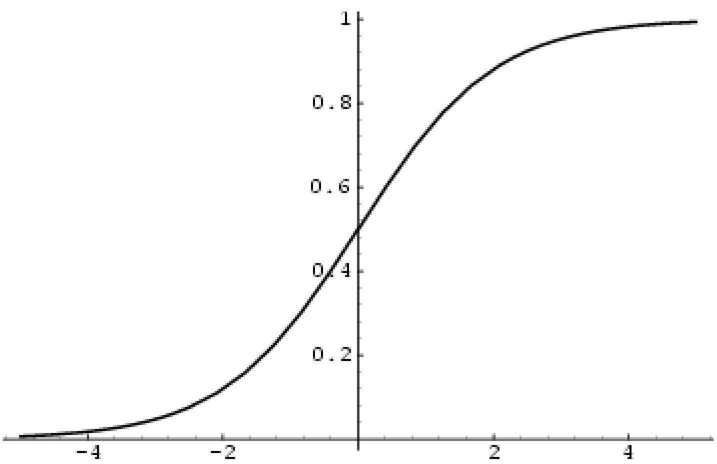


**Figure 2.2** Sigmoid function

### 2.1.6. Architecture of Neural Networks

Different types of connections cause different behavior of the neural network. Two architecture types are mainly used in artificial neural networks. These are feed-forward and backward architectures.

#### 2.1.6.1. Feed-Forward Neural Networks

When the neural network has no loops between its outputs to inputs, it is called feed-forward neural networks. Neural network performs a static mapping between inputs and outputs, independent of previous system states, only depending on the current inputs at hand(Lavine & Blank, 2009). In figure 2.3 an example of a feed-forward neural network is shown.

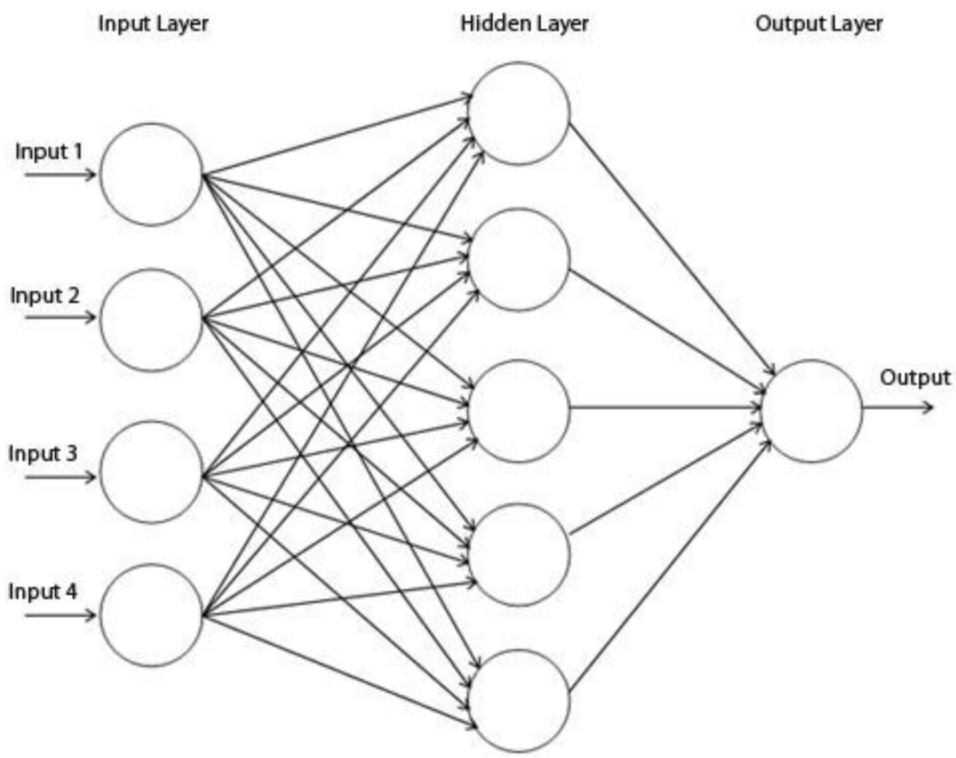


**Figure 2.3** Feed-forward neural network example

#### 2.1.6.2. Recurrent Feedback Neural Networks

Recurrent neural networks are fundamentally different from feedforward architectures. They are not only operating on an input space but also they trace what already has been processed by the network(Boden, 2001).As shown in Figure 2.4 signals can travel in both directions by introducing loops in the network Due to the feedback connections, the network enters a new state by changing the inputs.

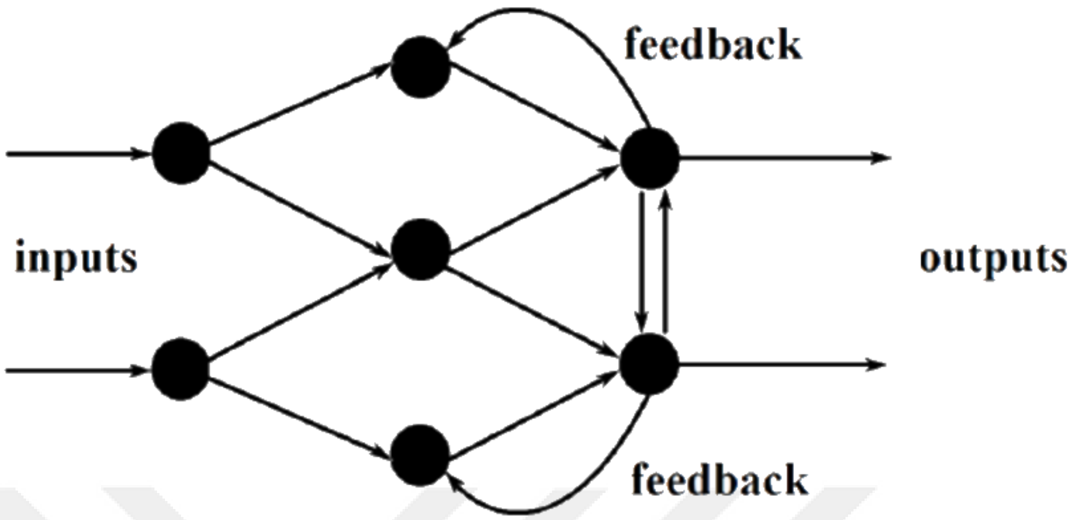


**Figure 2.4** Recurrent feedback neural network example

## 2.2. Support Vector Machines

SVM algorithm was invented and then used as a non-linear classifier by (Cortes & Vapnik, 1995). It is used as classification and solution of linear and nonlinear function approximation problems. SVM is used in text recognition (D. Chen et al., 2004), object recognition, and face recognition areas (E.~Osuna et al., 1997). We can divide SVM usage area into linear and non-linear.

### 2.2.1. Linear Support Vector Machines

Linear SVM uses to recognize linear patterns that are distinguishable or can be easily separated in low dimension(Pradhan, 2012). There are different approaches like hard margin and soft margin on linear SVM. These approaches can classify linearly separable data exceptionally well.

### 2.2.2. Non-Linear Support Vector Machines

When data can't be separated linearly, Nonlinear SVMs map this data to higher dimensional space which is called feature space. The reason for this extension is that an SVM that can create a nonlinear decision hypersurface will be able to classify nonlinearly separable data.(Wang, 2005). To map this data to higher dimensional spaces new dimension’s point must be calculated. That’s where the kernel trick helps us to generate this new dimension.

The Kernel trick is a very interesting and compelling tool. It is powerful because it provides a link from linearity to non-linearity to any algorithm. A lot of functions are available as kernel functions. With these functions, infinitely various points in dimensional space can be calculated. Below some kernel functions available from the existing literature are explained.

#### 2.2.2.1. Linear Kernel Function

The Linear kernel is the simplest kernel function. Linear kernel function allows us to pick out lines or hyperplanes in a higher dimension. As shown in Equation 2.4, it is given by the inner product of x and y coordinates with an optional constant c.

$$k(x,y) = x^T y + c \tag{2.4}$$

#### 2.2.2.2. Polynomial Kernel Function

The Polynomial kernel is a non-stationary kernel. Polynomial kernels are well suited for problems where all the training data is normalized. As shown in Equation 2.5 slope alpha, the constant c and the polynomial degree d used as adjustable parameters with coordinates x and y values.

$$k(x,y) = (ax^T y + c)^d \tag{2.5}$$

#### 2.2.2.3. Radial Basis Kernel Function

The Radial basis functions are well known in signal processing as a tool to smooth the images. Radial basis functions allow to pick out circles or hyperspheres in a higher dimension. As shown in Equation 2.6 sigma is used as an adjustable parameter to make equation nonlinear with coordinates x and y values. Sigma value is critical in this equation if it is set too high, equation will behave like a linear equation. If it is set too low the result will be significantly affected by outlier values.

$$k(x, y) = \exp(-\gamma \|x - y\|^2) \tag{2.6}$$

## 2.3. 2D Gabor Filters

2D Gabor filters are the type of filters in image processing that is used for edge detection, feature extraction texture analysis. Gabor filters are special classes of bandpass filters. They permit a certain ‘band’ from claiming frequencies while rejecting the others. In this thesis, Gabor filters are applied to gather features of the fabric’s images to feed the neural network. Formula for Gabor filter that is used in this thesis is;

$$g(x, y; \lambda, \theta, \Psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \exp\left(i\left(2\pi\frac{x'}{\lambda} + \Psi\right)\right) \tag{2.7}$$

In formula 2.7 : λ (lambda) is the wavelength of the sinusoidal factor, γ (gamma) is the spatial aspect ratio, Ψ (psi) phase offset, θ (theta) orientation of the normal to the parallel stripes of the Gabor function, σ (sigma) standard deviation of the Gaussian function used in the Gabor filter.

When a Gabor filter is applied to the image, it will return a response data in the edges where the texture continuity changes in an angle. Gabor filters are applied for angles that add to 180°, and each angle returns different results with respect to its orientation degree. If all responses add to every other orientations’ responses, the result will show edges of the data more clearly. In figure 2.5 graphic shows the response set of Gabor Filters for 12 directions and 3 scales in the frequency domain.

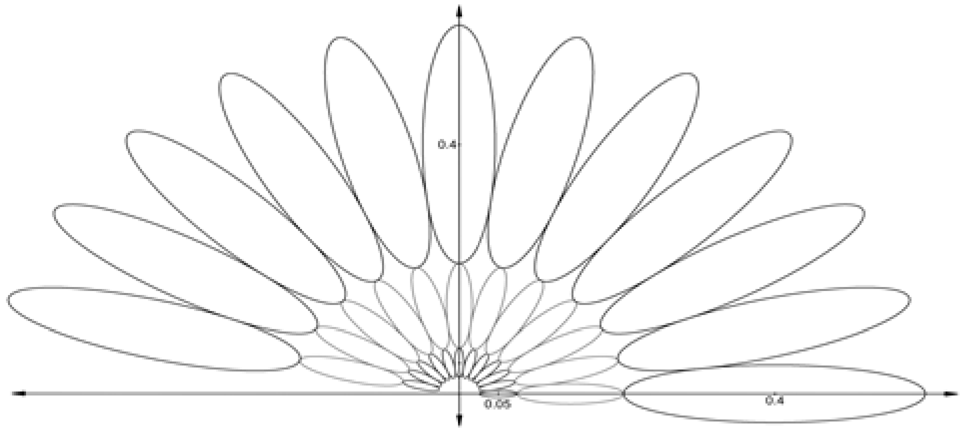


**Figure 2.5** Gabor Filter in Frequency Domain

In Figure 2.6 Gabor filters with 12 orientations results and the result of the Gabor filter's compared with original image is shown with the parameters lambda:10, sigma:4, gamma:0.6, psi:0. From top left to bottom right angles increase from 15 degrees to 180° with the amount of 15° increase.

**Figure 2.6** Gabor filters response for Garment Sample

**Figure 2.7** Comparison of original and result image

In figure 2.7 on the left original sample is shown, on the right all orientations of Gabor filters add up to the final picture. These results indicate us that Gabor filters are suitable for extracting features from 2-dimensional fabric images.

## 2.4. MACHINE LEARNING AND PROGRAMMING TOOLS

There are a lot of machine learning programs and programming tools. For this project Python with OpenCV library used for gathering the data from cloth photographs and Matlab 2017B version is used to process this data.

### 2.4.1. PYTHON

Python is an interpreter, interactive, object-oriented programming language. Modules, exceptions, dynamic typing, very high-level dynamic data types, and classes are consolidated (Python Docs, 2017). Python language is used for gathering image data by using OpenCV library. All data is stored as NumPy matrices. These matrices are used as train data to construct the artificial neural network.

### 2.4.2. OPENCV

OpenCV is aimed at providing the tools needed to solve computer-vision problems. It contains a blend of low-level image-processing functions and high-level algorithms such as end tracking, face detection, feature matching. (Bradski, 2000) In this project OpenCV Library let us extract the features from the 2-dimensional image data with Gabor filters and red, green and blue color features taken out from the image to get the mean and standard deviation of those RGB values.

### 2.4.3. MATLAB

MATLAB® is the high-level language and interactive environment used by millions of engineers and scientists around the world. Computational mathematics can be expressed by this matrix-based language (Matlab, 2017). Matlab has lots of useful toolboxes. In this project, Matlab Neural Network Library is used for constructing, training and testing an artificial neural network with using data that was gathered from image set. Also, SVM toolbox is used for Support Vector Machine construction and testing part.

# CHAPTER 3
# LITERATURE REVIEW

In fashion retail industry every important decision about fashion design and production processes comes down to sales forecasting, and this plays a very prominent role (Xia et al., 2012). Fashion design companies always need to provide right products at the right time, maintain good stock and fast design process. In design part, that’s why AI techniques are widely used to help designers to understand what the customer is thinking, what customers like and how to catch or create trends in the fashion sector. In their paper(Kokol et al., 2006), used machine learning techniques like decision trees to understand what are teens clothing fashion preferences. A survey data of specific generation’s fashion preference data processed with AI techniques, and it is used to model a specific clothing design area. And the results are used for improving the quality, cost, and speed of the designs to be made on that area by using this model’s output.

These AI techniques can also be used for learning a person’s fashion choice by looking up their recent fashion decisions. In her thesis, (WANG, 2014) created a neural network named “Style Me” to replace the personal fashion stylist concept in our minds. She analyzed earlier fashion choices of a person and built a neural network that suggests new combinations for that individual with higher acceptance rate. This shows that our machine learning and AI techniques can help us to decide what an individual wants to wear in his daily life that complies with his fashion sense.

Like (WANG, 2014), another example to find the best fashion outfit composition project is (Li et al., 2017)’s deep learning system. It evaluates and scores the garment’s aesthetic features and find the possible combination of other items based on this score. Garment’s aesthetic features are used as categories, colors, coherence, patterns, creative pairing choices, as well as styles for different ages and individuals. Those features are based on (Chen et al., 2012) work on Clothing semantic attributes. Which (Li et al., 2017) found that even though color values on the image has the highest individual impact, the best outcome can be obtained by combining these features of garment’s aesthetic features.

For my thesis, an only color attribute of the garment samples is used to feed the neural network and SVM Model. Raw RGB values of the garment samples are not used instead of processed, and normalized values of these RGB values are used. Images were processed with Gabor filtering technique. This technique is applied to various 2-dimensional image data to extract features.

In their paper, (Yang et al., 2003) used Gabor filters on 2-dimensional fingerprint images to enhance results. Another example of using Gabor filters for enhancing outcomes would be (Ji et al., 2004)'s work. They used Gabor filters for improving edge detection and object segmentation of 2-dimensional environment images.

In fashion Industry, (Bianconi & Fernández, 2007) show that Gabor filters can be used for classifying texture types and it is a widely adopted technique for texture analysis. However, parameter selection for Gabor filters is crucial, and all task is dependent on it. There are five essential parameters for Gabor filters; Frequency ratio, number of frequencies, number of orientations, orthogonal width (kernel size), gamma. In my thesis 12 orientation and three gamma value are used to generate 36 distinct images, and from these images, RGB values' mean and standard deviation values are used as train and test dataset.

CHAPTER 4

METHODS AND RESULTS

## 4.1. Original Image Set

Total of 17 children clothing samples are used as a dataset. 10 of them are accepted from international fashion design company, and 7 of them are rejected by the same international fashion design company. All images are 535-pixel width and 535-pixel height. In figure 4.1 some of the samples are shown. Top 3 image is accepted bottom 3 are rejected garment samples.


**Figure 4.1** Garment Samples

## 4.2. Generating Data from Images

In this thesis, 17 images from an international fashion design company are used to generate a dataset to feed the neural network. Those images passed on Gabor filters with various parameters and formed the sub-images from those Gabor filters. Those sub-images' processed with OpenCV library to get the mean and standard deviation of the RGB values of each one. For every sub-image, these mean and standard deviance values kept in a shirt class in python and "setMeanStd" function is used to get mean and deviance values of RGB values from the image. Python codes are below for this operation:

```
class Shirt:

    def __init__(self, name,path):
        self.name = name
        self.path=path
        self.img= cv2.imread(self.path)
        self.Rmean=0
        self.Bmean=0
        self.Gmean=0
        self.Rstd=0
        self.Bstd=0
        self.Gstd=0

        """b, g, and r are numpy.ndarray types"""
    def setMeanStd(self):
        b, g, r = cv2.split(self.img)
        ttl = self.img.size / 3 #divide by 3 to get the
number of image PIXELS
        B = float(np.sum(b)) / ttl #convert to float
        G = float(np.sum(g)) / ttl
        R = float(np.sum(r)) / ttl
        self.Bmean=B
        self.Gmean=G
        self.Rmean=R
        self.Bstd = np.std(b,ddof=1
        self.Rstd=np.std(r,ddof=1)
        self.Gstd=np.std(g,ddof=1)
```

For example, original samples final values shown in Table 4.1: Results of Mean and Standard deviance operations. First 10 sample is accepted last 7 samples are rejected.

| | Red Mean | Green Mean | Blue Mean | Red Std | Green Std | Blue Std | Accepted |
|---|---|---|---|---|---|---|---|
| 1 | 150.7391 | 145.5124 | 147.4055 | 8.230971 | 10.25395 | 14.06771 | Y |
| 2 | 147.9949 | 145.6617 | 138.472 | 7.933805 | 13.06357 | 14.019 | Y |
| 3 | 159.0324 | 134.3405 | 142.0713 | 19.44498 | 30.23556 | 28.99563 | Y |
| 4 | 159.3493 | 113.9278 | 116.6122 | 16.71193 | 39.94548 | 40.98684 | Y |
| 5 | 143.5791 | 135.8936 | 137.597 | 30.11022 | 28.46003 | 38.85749 | Y |
| 6 | 141.5739 | 133.4466 | 140.2085 | 31.06075 | 22.29711 | 20.95606 | Y |
| 7 | 144.9968 | 141.1245 | 144.3134 | 10.26491 | 9.659441 | 11.3398 | Y |
| 8 | 145.6599 | 140.7573 | 150.0762 | 20.31485 | 19.18293 | 17.89336 | Y |
| 9 | 146.1507 | 143.0434 | 136.8672 | 14.79369 | 16.838 | 20.29186 | Y |
| 10 | 150.276 | 137.2486 | 138.1783 | 14.00088 | 18.22782 | 19.29555 | Y |
| 11 | 144.8743 | 140.7005 | 148.3739 | 20.23862 | 19.98211 | 20.80327 | N |
| 12 | 143.7952 | 138.2358 | 142.6285 | 23.27407 | 22.68776 | 27.04497 | N |
| 13 | 148.9649 | 137.7756 | 139.7162 | 21.93902 | 23.11332 | 22.24217 | N |
| 14 | 103.5006 | 105.1163 | 118.3985 | 72.11238 | 67.49665 | 60.19049 | N |
| 15 | 150.2918 | 138.4237 | 140.7143 | 14.2749 | 18.35079 | 18.6898 | N |
| 16 | 140.6401 | 134.6778 | 139.8884 | 25.0406 | 27.85136 | 25.69047 | N |
| 17 | 124.5826 | 123.6446 | 124.5789 | 100.8114 | 100.9053 | 99.56993 | N |

**Table 4.1** Results of Mean and Standard deviance of garment samples

For a larger data set Gabor filters on original images with 12 different orientation and three different aspect ratio (0.2, 0.4, 0.6) are applied. Due to this orientations and aspect ratios total of 36 different images are generated. After generating those distinct images, RGB mean and standard deviation of those images are gathered, and a total of 216 features are generated for every one of the original image samples. For Gabor filtering, "ShirtGabor" class is created as a python class. First kernels are built with "build_filters_for_all" method, then every sub-image filtered with these kernels via "process_individual" method. Finally, in the main section, all 17 of garment sample images processed and their results stored in excel file. Python codes are below for this operation:

```
class Shirt:
def build_filters_for_all():
    filters = []
    i=0
    n=[0.2,0.4,0.6]
    while (i<3):
        for theta in np.arange(0, np.pi, np.pi / 12):#12
different angle
            kern = cv2.getGaborKernel((535, 535), 4.0,
theta, 10.0, n[i], 0, ktype=cv2.CV_32F)#535,535 image
size
            kern /= 1.5*kern.sum()
            filters.append(kern)
        i=i+1
    return filters

def procesindividual(img, kern):
    fimg= cv2.filter2D(img,cv2.CV_8UC3,kern)
    return fimg

if __name__ == '__main__':
    import sys
    print __doc__

filenames=["success/s","failure/f"]
last=[10,7]
n=["s","f"]
fs=0;
i=1
shirts=[]
writePath="C:/data/env1/"
while(i<=last[fs]):
    try:
        img_fn = sys.argv[1]
    except:
        img_fn =
'C:\\data\\env1\\'+filenames[fs]+str(i)+'.jpg'
    img = cv2.imread(img_fn)
    if img is None:
        print 'Failed to load image file:', img_fn
        sys.exit(1)
    filters = build_filters_for_all()
    for kern in filters:
        res1 = procesindividual(img,kern)
        sname=n[fs]+str(i)
        sh= ShirtGabor(sname,res1)
        sh.setMeanStd()
        shirts.append([sh.getExcel()])
        print(i)
        cv2.waitKey(0)
    i=i+1
```

Because of the size of the table, some parts of Gabor filter results are, shown in Table 4.2. Remaining data will be on Appendix pages.

This method extracts features from original data set to make high-quality compact discriminative features with Gabor filters. This method inspired after (Simo-Serra & Ishikawa, 2016)'s method. In their research, they collected meaningful data from weak images. These images gathered from the generic image database, and they were noisy data. Significant features of the images are extracted via joint ranking method to sweep off the noise from image data.

| | Gabor Result Gamma:0.2,Theta:15° | | | | | | Gabor Result Gamma:0.2,Theta:30° | | | | | | Accepted |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Red Mean | Green Mean | Blue Mean | Red Std | Green Std | Blue Std | Red Mean | Green Mean | Blue Mean | Red Std | Green Std | Blue Std | |
| 1 | 100.4915 | 97.01561 | 98.29159 | 8.367516 | 9.922067 | 12.53539 | 100.494 | 97.01685 | 98.29487 | 8.170726 | 9.746017 | 12.39904 | Y |
| 2 | 98.66193 | 97.13284 | 92.34569 | 8.364848 | 12.64959 | 13.20427 | 98.68295 | 97.17156 | 92.38603 | 9.013172 | 13.10159 | 13.72314 | Y |
| 3 | 106.5045 | 90.36429 | 95.44786 | 20.89312 | 25.66809 | 24.79574 | 106.4638 | 90.28445 | 95.37173 | 20.47664 | 25.22221 | 24.49602 | Y |
| 4 | 106.2426 | 76.09092 | 77.86504 | 15.01307 | 31.37304 | 32.21492 | 106.2473 | 76.07641 | 77.84542 | 14.7098 | 30.36019 | 31.11571 | Y |
| 5 | 97.82867 | 92.65015 | 94.96055 | 36.44999 | 33.83792 | 43.35462 | 97.43984 | 92.20857 | 94.87526 | 34.48064 | 32.1649 | 42.03263 | Y |
| 6 | 94.60968 | 89.04508 | 93.8799 | 31.31704 | 21.98476 | 22.42905 | 94.4764 | 88.98579 | 93.57612 | 28.89098 | 20.2527 | 20.00962 | Y |
| 7 | 96.76577 | 94.15112 | 96.2935 | 12.7999 | 10.13563 | 11.78786 | 96.83134 | 94.19367 | 96.33215 | 13.62937 | 10.73233 | 12.48025 | Y |
| 8 | 97.24747 | 93.96968 | 100.1161 | 19.6144 | 18.78851 | 18.0815 | 97.17605 | 93.9053 | 100.0797 | 20.28801 | 19.43799 | 18.56348 | Y |
| 9 | 97.77541 | 95.73112 | 91.73017 | 14.77345 | 16.32027 | 18.32385 | 97.64767 | 95.59182 | 91.61108 | 13.64376 | 15.25885 | 17.32997 | Y |
| 10 | 100.181 | 91.50264 | 92.12686 | 15.25743 | 19.29845 | 20.59205 | 100.1805 | 91.50061 | 92.12671 | 14.66151 | 18.62714 | 19.82915 | Y |
| 11 | 97.05805 | 94.29354 | 99.37291 | 25.396 | 24.96386 | 25.85635 | 97.06767 | 94.30549 | 99.38315 | 25.48053 | 25.01693 | 25.9293 | N |
| 12 | 96.6331 | 92.88061 | 95.96426 | 31.5458 | 30.25072 | 33.76269 | 96.53778 | 92.85301 | 95.88563 | 30.86289 | 29.88299 | 33.13193 | N |
| 13 | 99.56152 | 92.14321 | 93.42346 | 23.81265 | 23.91034 | 23.61512 | 99.42536 | 91.98653 | 93.28402 | 21.42862 | 21.59881 | 21.25065 | N |
| 14 | 70.74765 | 71.54167 | 79.76223 | 50.95818 | 48.08352 | 43.4554 | 70.74304 | 71.52652 | 79.81069 | 51.30978 | 48.52213 | 44.26053 | N |
| 15 | 100.2217 | 92.88101 | 94.40887 | 14.82252 | 19.64146 | 20.71007 | 100.2007 | 92.50152 | 94.0133 | 13.55234 | 17.32803 | 18.20086 | N |
| 16 | 94.16856 | 90.22158 | 93.65965 | 21.42051 | 24.04378 | 22.72453 | 93.97413 | 90.01418 | 93.47595 | 20.36077 | 22.81979 | 21.48491 | N |
| 17 | 86.82115 | 86.33249 | 86.72075 | 75.92933 | 75.83363 | 74.96742 | 85.80831 | 85.29142 | 85.70244 | 74.03393 | 73.95615 | 73.09947 | N |

**Table 4.2** First 2 orientation of 0.2 gamma values' RGB mean and standard deviance data.

## 4.3. Feed Forward Neural Network Training Method

Matlab program is used for building the feed-forward neural network. After generating data sets to make the tests, we allocate 75% of the data to feed the neural network which is 8 of accepted and 5 of rejected sample data to train the neural network, and 25% of the data which is 2 accepted samples and 2 of the rejected samples to test the accuracy of the neural network as a rule of thumb. Before feeding the neural network, all sample data are normalized with MATLAB prestd method. The normalization method is as follows;

P - All sample values

Meanp - Mean of all sample values

Stdp – Standard deviations of all sample values

PN Normalised Samples: PN = (P –Meanp)/Stdp for each sample.

The features of the neural network are selected as:

- 1 Hidden layer selected with 20 hidden neurons
- Training method Trainingdx is selected which is gradient descent with momentum and adaptive learning rate backpropagation method. This function is a network training function that updates weight and bias values according to gradient descent momentum and an adaptive learning rate.
- Maximum iteration is 5000 iterations.
- Minimum Error rate is set as 0.01
- Performance evaluation is made with mean squared error method

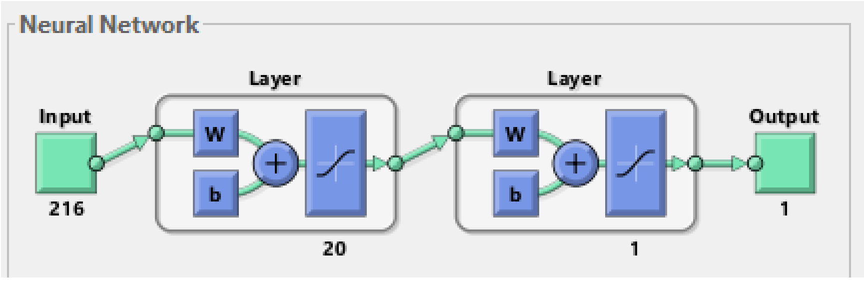


**Figure 4.2** General scheme for neural networks with 216 inputs

In the code below, first training and test set is generated from P matrix which is RGB mean and standard deviance values of Gabor filtered sub-images. “prestd” method used to generate normalized values. Neural network features such as max iteration, termination criteria are given and network started with “sim” method. The building code for the neural network is below.

```
P=P';
PP = P(:,3:13);
PP=[ PP P(:,14:15)];
T2=[ 0.9*ones(1,8) -0.9 * ones(1,5)];
PT=P(:, 1:2);
PT=[ PT P(:,16:17)];
TT=[0.9 * ones(1,2) -0.9*ones(1,2)];

[pn,meanp,stdp,tn,meant,stdt] = prestd(PP,T2); % training
normalization
net= newff(minmax(pn),[20 1],{'tansig',
'tansig'},'traingdx');
net=init(net);

net.trainParam.epochs=5000;
net.trainParam.goal=0.01;
net.trainParam.show=20;

[net,tr]=train(net,pn,tn);

p2n = trastd(PT,meanp,stdp); % testing normalization
an=sim(net,p2n);
A2 = poststd(an,meant,stdt)
```

Accepted records’ output values are selected as 0.9, and rejected values are selected as -0.9. This allows us rather a big interval between input and output values and makes predictions smoother.

### 4.4. Support Vector Machine Training Method

We implemented SVM method with using MATLAB program. The same approach that is used with the feed-forward neural network is also used for generating training and test sets. The same method for normalizing data is used (MATLAB prestd method). Because of our data dimensions, it can't be drawn on a simple hyperplane, and we should use reproducing kernel approach. For these kernels, there are different kernel functions. In Matlab, we can use three different kernel function types for SVM kernel function. These are:

- Linear kernel function, (meaning dot product)
- Polynomial kernel function
- RBF (Gaussian radial basis function kernel)

With these three kernel function, three distinct SVM model is generated, and their performance is evaluated with confusion matrix method.

In the code below, normalized values which are created with "prestd" method from neural network code used as "spn" and "T2" matrices. 3 SVM created with different kernel functions which are Linear, Polynomial and Radial Basis Function. After training these SVMs' training and test, data accuracy is calculated via confusion matrices and accuracy values printed to screen. The building code for SVM models are below:

```
SvmLinear =
fitcsvm(spn,T2','Standardize',false,'KernelFunction','linear',...
    'KernelScale','auto')
SvmPolynomial =
fitcsvm(spn,T2','Standardize',false,'KernelFunction','polynomial',..
.
    'KernelScale','auto')
SvmRbf =
fitcsvm(spn,T2','Standardize',false,'KernelFunction','rbf',...
    'KernelScale','auto')

LinearTrain=SvmLinear.predict(spn)
PolyTrain=SvmPolynomial.predict(spn)
RBFTrain=SvmRbf.predict(spn)

LinearTest=SvmLinear.predict(sp2n)
PolyTest=SvmPolynomial.predict(sp2n)
RBFTest=SvmRbf.predict(sp2n)


[CMLinearTrain,order] = confusionmat(T2',LinearTrain)
[CMPolyTrain,order] = confusionmat(T2',PolyTrain)
```

```
[CMRBFTrain,order] = confusionmat(T2',RBFTrain)

[CMLinearTest,order] = confusionmat(TT',LinearTest)
[CMPolyTest,order] = confusionmat(TT',PolyTest)
[CMRBFTest,order] = confusionmat(TT',RBFTest)


accLinearTrain=
(CMLinearTrain(1,1)+CMLinearTrain(2,2))/(CMLinearTrain(1,1)+CMLinear
Train(2,2)+CMLinearTrain(1,2)+CMLinearTrain(2,1))

accLinearTest=
(CMLinearTest(1,1)+CMLinearTest(2,2))/(CMLinearTest(1,1)+CMLinearTes
t(2,2)+CMLinearTest(1,2)+CMLinearTest(2,1))

accPolyTrain=
(CMPolyTrain(1,1)+CMPolyTrain(2,2))/(CMPolyTrain(1,1)+CMPolyTrain(2,
2)+CMPolyTrain(1,2)+CMPolyTrain(2,1))

accPolyTest=
(CMPolyTest(1,1)+CMPolyTest(2,2))/(CMPolyTest(1,1)+CMPolyTest(2,2)+C
MPolyTest(1,2)+CMPolyTest(2,1))

accRBFTrain=
(CMRBFTrain(1,1)+CMRBFTrain(2,2))/(CMRBFTrain(1,1)+CMRBFTrain(2,2)+C
MRBFTrain(1,2)+CMRBFTrain(2,1))

accRBFTest=
(CMRBFTest(1,1)+CMRBFTest(2,2))/(CMRBFTest(1,1)+CMRBFTest(2,2)+CMRBF
Test(1,2)+CMRBFTest(2,1))
```

## 4.5. Results

We implemented the algorithm with Python and Matlab on Windows platform. For feed-forward neural network training:

Neural network trained for 216 input nodes which are feed with Gabor filtering results of 13 distinct garment samples. 8 sample of this set was accepted by international fashion design company, 5 of them was rejected by the same company. Trained neural networks’ mean squared error result is 0.015991, and the values are in table 4.3.

| Expected Result | NN Result |
|---|---|
| 0.9 | 0.7595 |
| 0.9 | 0.7596 |
| 0.9 | 0.7595 |
| 0.9 | 0.7595 |
| 0.9 | 0.7595 |
| 0.9 | 0.7594 |
| 0.9 | 0.7594 |
| 0.9 | 0.7594 |
| -0.9 | -0.9999 |
| -0.9 | -1 |
| -0.9 | -0.9999 |
| -0.9 | -0.9999 |
| -0.9 | -0.9998 |
| MSE | 0.015991 |

**Table 4.3** Training Results for NN

For testing part there are 4 samples. 2 of them was accepted, and 2 of them was rejected. Test results are given to neural network, and the mean squared error result is 0.04825692, and the values are in table 4.4.

| Expected Result | NN Result |
|---|---|
| 0.9 | 0.6918 |
| 0.9 | 1.0273 |
| -0.9 | -0.6733 |
| -0.9 | -0.6135 |
| MSE | 0.04825692 |

**Table 4.4** Testing Results for NN

Overall information about experiments is in figure 4.3.

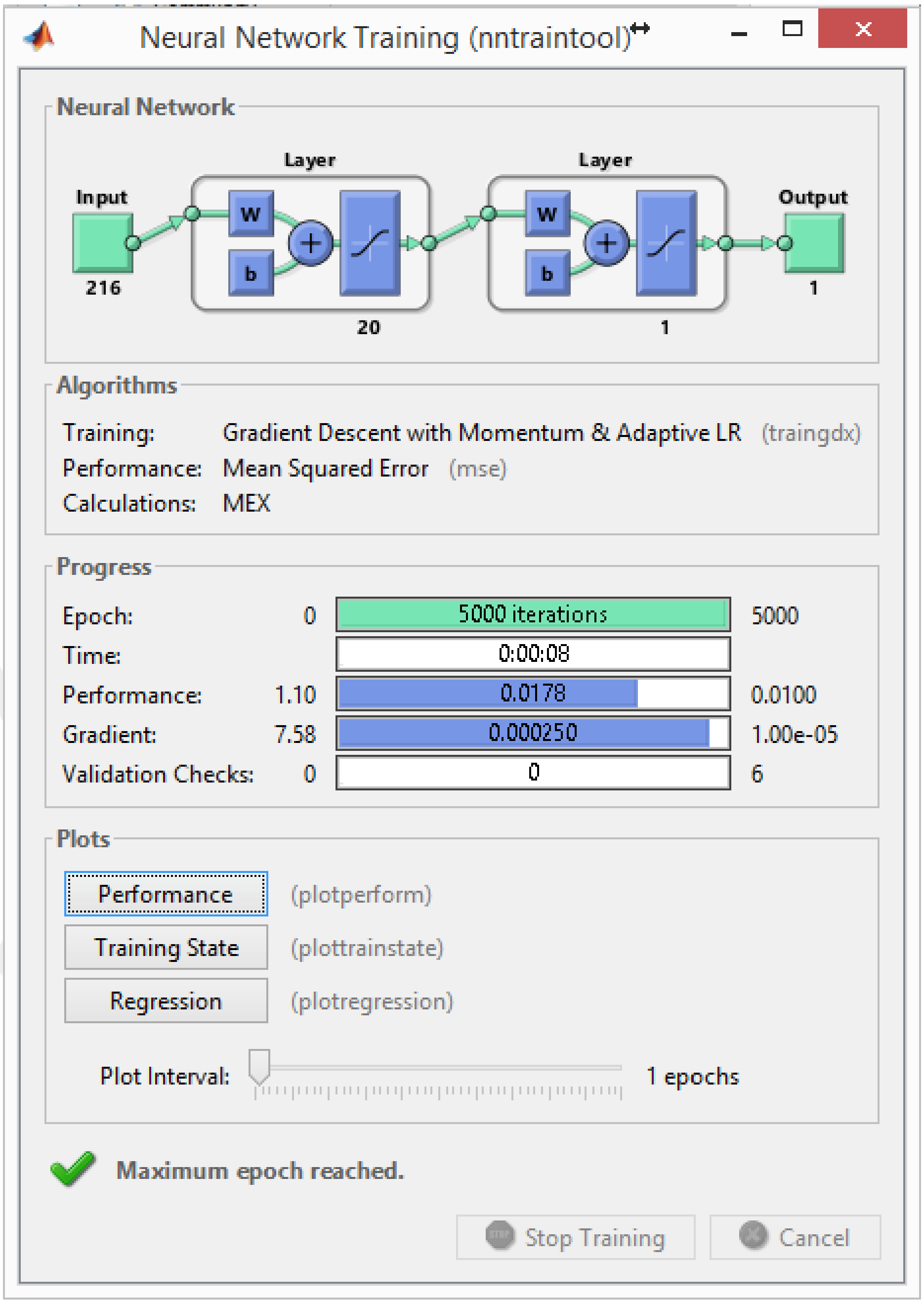


**Figure 4.3** General Result scheme for Neural network on Matlab

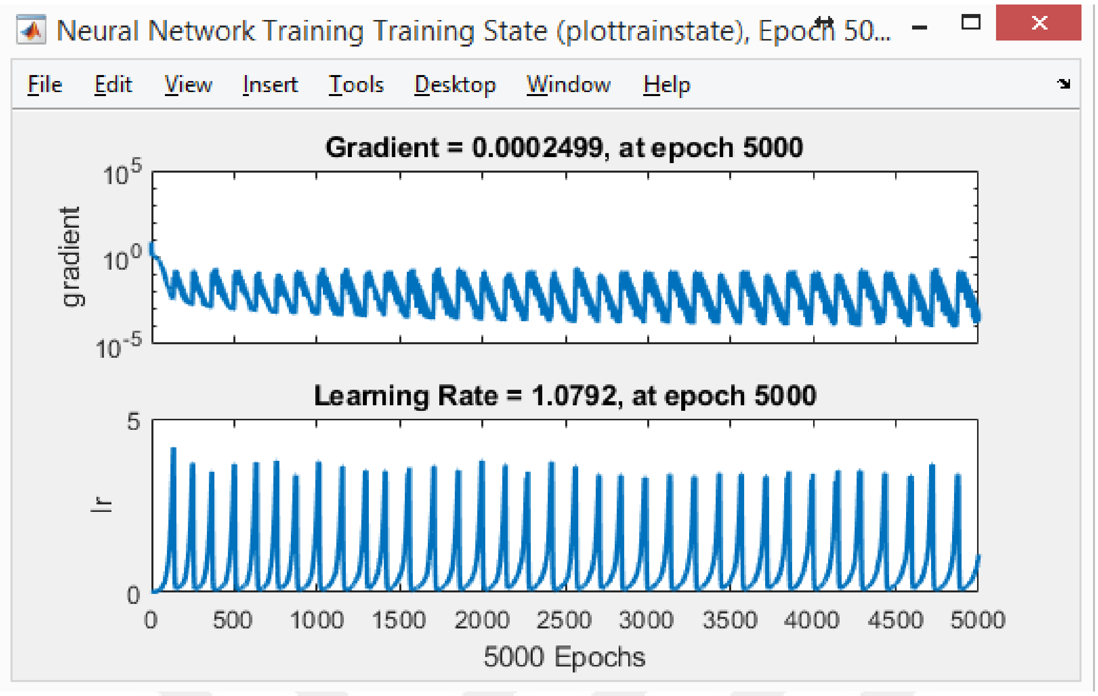


**Figure 4.4** Gradient Descent, and Learning rate plot on Matlab

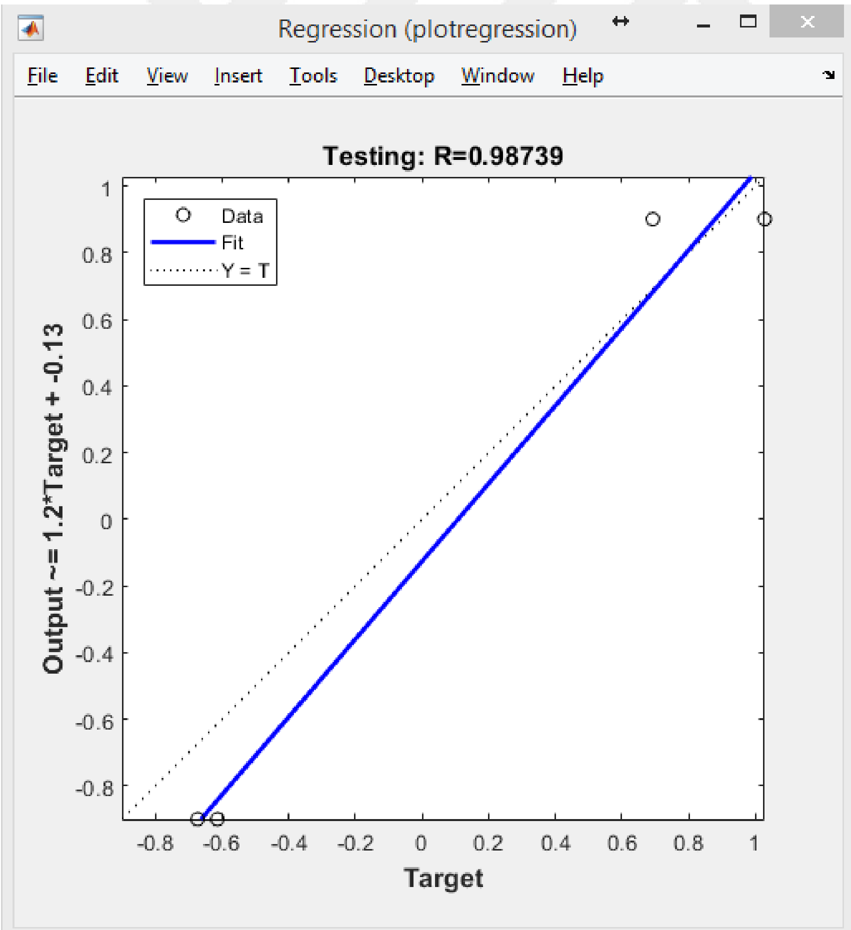


**Figure 4.5** Testing Regression Plot on Matlab

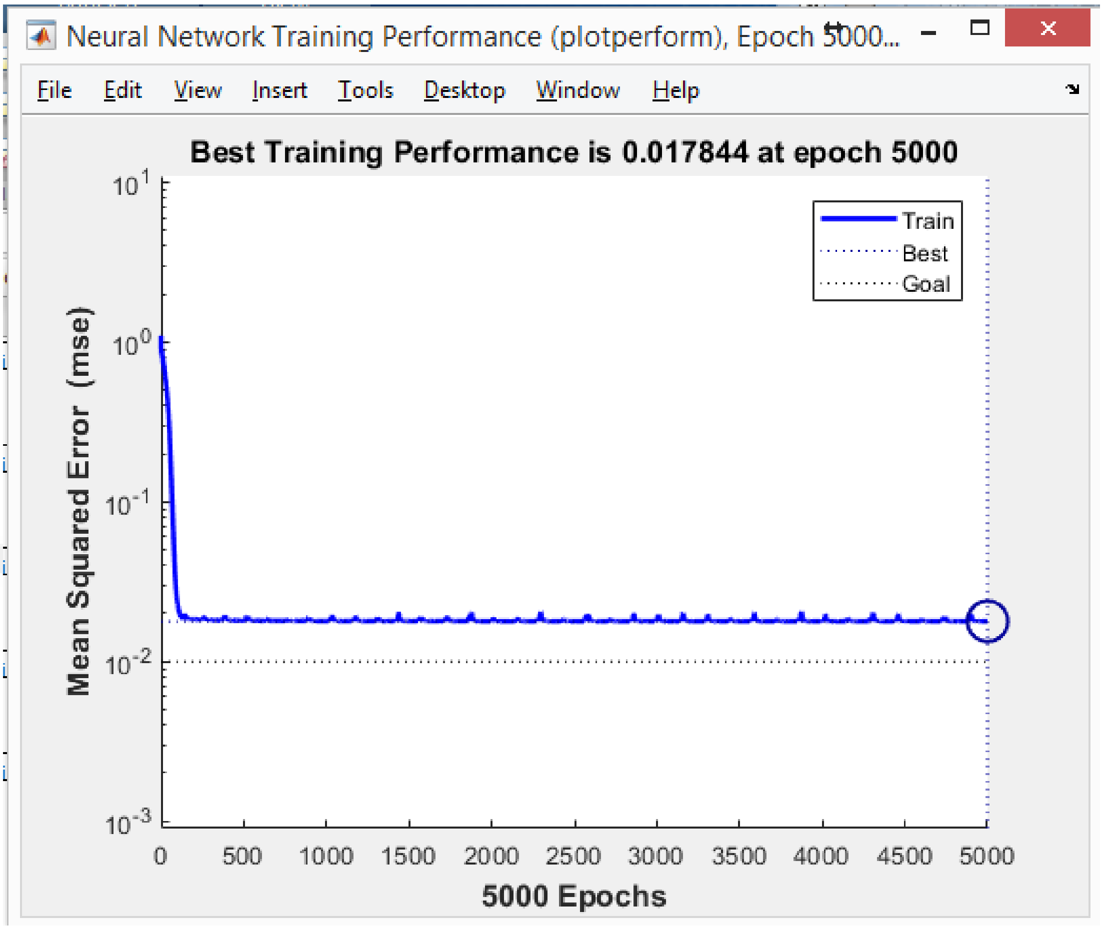


**Figure 4.6** Training Performance for NN

From Figure 4.4 we can clearly see that there is a continuous learning in 5000 iteration and gradient changes every epoch. From Figure 4.5 we can observe that test results and expected results are almost linear. Figure 4.6 displays that in first 100 epochs neural network finds very close result to end result. But it never reaches the fitness condition which is 0.01. That means we need more sample to feed the neural network in order to pass the expected fitness condition.

Another test metric for this neural network would be hypothesis testing(Masters, 1993). For training results of the network, we can generate a density function with sigma value with 0.25 for both accepted and rejected results. And find an area which separates both functions. At figures 4.6 we can clearly see that both results are perfectly separated with our neural network. We can say that if the neural network result is less than 0, it will be rejected and if it is greater than 0 it will be accepted. (Unluturk et al.,

2011) have also implemented this approach in their research to get training result and find the deciding factor.

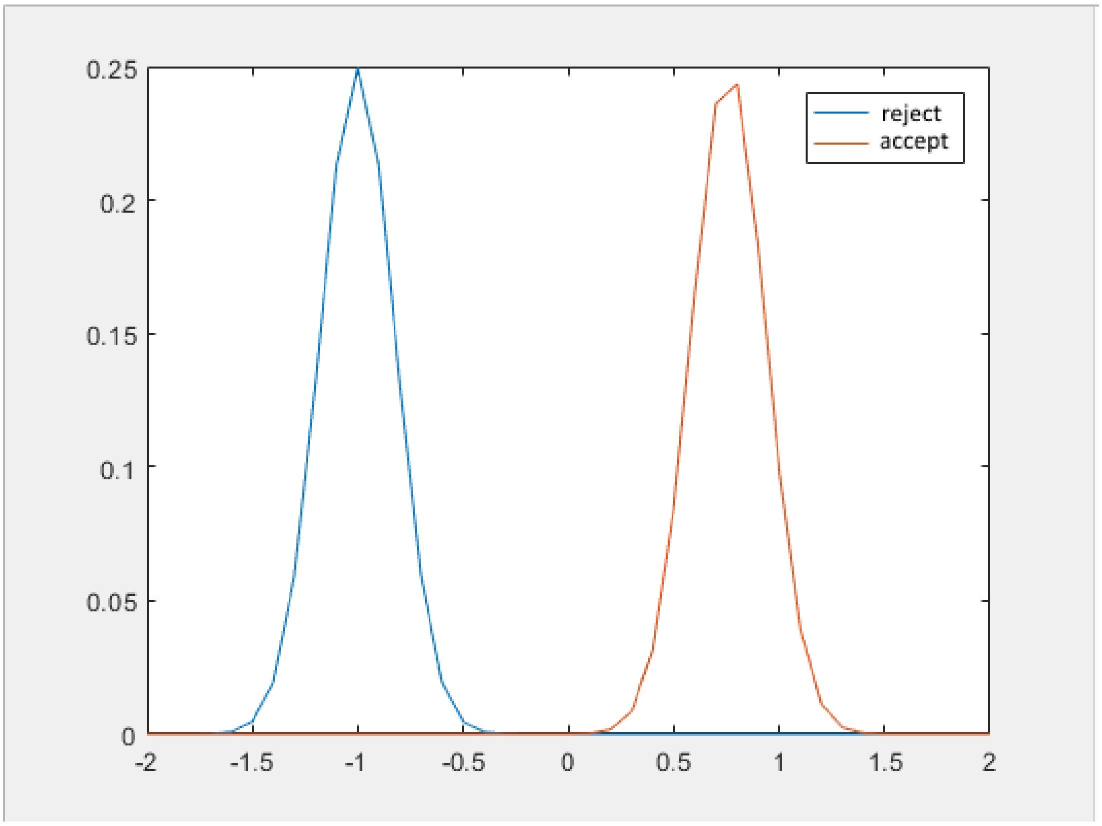


**Figure 4.7** Hypothesis testing results

For the SVM method, we used same training and testing data. In Matlab fitcsvm method is used to evaluate the Support Vector Machine method. 0.9 value is used to specify the accepted value and -0.9 is given to determine a rejected value. For performance evaluation, confusion matrix method is used. Results are below:

| Expected | LinearTrain | PolyTrain | RBFTrain |
|---|---|---|---|
| 0.9 | 0.9 | 0.9 | 0.9 |
| 0.9 | 0.9 | 0.9 | 0.9 |
| 0.9 | 0.9 | 0.9 | 0.9 |
| 0.9 | 0.9 | 0.9 | 0.9 |
| 0.9 | 0.9 | 0.9 | 0.9 |
| 0.9 | 0.9 | 0.9 | 0.9 |
| 0.9 | 0.9 | 0.9 | 0.9 |
| 0.9 | 0.9 | 0.9 | 0.9 |
| -0.9 | 0.9 | -0.9 | -0.9 |
| -0.9 | 0.9 | -0.9 | -0.9 |
| -0.9 | 0.9 | -0.9 | 0.9 |
| -0.9 | -0.9 | -0.9 | -0.9 |
| -0.9 | 0.9 | -0.9 | 0.9 |

**Table 4.5** SVM Training results for all Kernel Functions

| Expected | LinearTest | PolyTest | RBFTest |
|---|---|---|---|
| 0.9 | 0.9 | 0.9 | 0.9 |
| 0.9 | 0.9 | 0.9 | 0.9 |
| -0.9 | 0.9 | -0.9 | 0.9 |
| -0.9 | -0.9 | -0.9 | 0.9 |

**Table 4.6** SVM Testing results for all Kernel Functions

| CMLinearTrain | Predicted: Rejected | Predicted: Accepted |
|---|---|---|
| Rejected | 1 | 4 |
| Accepted | 0 | 8 |
| Accuracy = 0.6923 | | |

| CMPolyTrain | Predicted: Rejected | Predicted: Accepted |
|---|---|---|
| Rejected | 5 | 0 |
| Accepted | 0 | 8 |
| Accuracy = 1 | | |

| CMLinearTest | Predicted: Rejected | Predicted: Accepted |
|---|---|---|
| Rejected | 1 | 1 |
| Accepted | 0 | 2 |
| Accuracy = 0.75 | | |

| CMPolyTest | Predicted: Rejected | Predicted: Accepted |
|---|---|---|
| Rejected | 2 | 0 |
| Accepted | 0 | 2 |
| Accuracy = 1 | | |

| CMRBFTrain | Predicted: Rejected | Predicted: Accepted |
|---|---|---|
| Rejected | 3 | 2 |
| Accepted | 0 | 8 |
| Accuracy = 0.8462 | | |

| CMRBFTest | Predicted: Rejected | Predicted: Accepted |
|---|---|---|
| Rejected | 0 | 2 |
| Accepted | 0 | 2 |
| Accuracy = 0.5 | | |

**Table 4.7** SVM Results Confusion Matrix and Accuracy Values

In Table 4.5 SVM training results are given and in Table 4.6 SVM testing results are presented. Table 4.7 shows the comparison of the different SVM models' confusion matrices and accuracy values which uses different SVM kernel functions. It is clear that polynomial kernel function is superior to other kernel functions. With 100% accuracy value for training and testing data, it is better to use polynomial kernel function than other kernel functions.

As a result of this feed forward neural network and SVM model experiments, it is clear that both feed-forward neural network model and SVM model with polynomial kernel function can classify the test samples correctly. Both models can be used for this sample data but with a bigger garment design dataset feed forward neural network approach would be easier to retrain and feed the garment design data with newer garment samples.

# CHAPTER 5
# CONCLUSIONS AND FUTURE WORK

In this thesis, a small garment data from an international fashion design company is analyzed in order to improve the garment design process and developed a feed-forward neural network with backpropagation and SVM model. This small garment data passed on Gabor filters with different gamma values and orientation in order to improve and expand features. To process this data both feed-forward neural network and SVM model techniques are performed.

The results indicate that both models can provide accurate results for this small set of accepted and rejected garment designs. Those models can be used to make this process shorter, more efficient and more educated.

For more accurate prediction, additional features of garments can be added to improve the feed forward neural network or SVM model. Such as:

- Emotion information of a garment.

  In his book, (Kobayashi, 1991) and his team after intensive research have matched 130 basic colors and over 10000 combinations to 180 key image words allowing to group colors and map those groups to certain emotions. By doing these colors can be combined with the same group of other groups to be in better harmony. In their work (N. Y. Kim et al., 2007) used this information and indexed textiles and created a new emotion-based indexing system.

- Tactile fabric comfort information

  The works of (Sztandera, 2009) and (Gonca Özçelik Kayseri, 2012) shows that tactile fabric comfort information or hand feeling of clothing is essential and distinct information of a garment. When buying clothing first and instinctive thing to do is touching it. And by touching, we can gather complex parameters from the garment. (Sztandera, 2009) says that there are 17 distinct parameters to get feeling information of fabric. That shows us there can be more features to consider for our training and test set.

- General fashion knowledge about garment

- Pattern information of garment

> As (S. Kim et al., 2006) say in their research there are patterns in most of the garments and these patterns can be indexed with different emotions based on its shape. They propose in their work that there can be nine distinct pattern types and they can map those patterns to various emotions.

With this extra features and the possible addition of sample data, feed-forward neural network and SVM model can be retrained and possibly find accurate results.

In order to move this study one step further, after improving our decision mechanism system, a fashion design application with using a genetic algorithm or evolutionary algorithm can be used to generate data for decision mechanism. (H. S. Kim & Cho, 2000) proposed an interactive genetic algorithm (IGA) driven application that generates garment designs. Those garments were produced by this system, and as a decision mechanism system, it uses human's response which in that case are fashion designers. IGA approach with our decision mechanism system can be used by international fashion design companies to generate and evaluate potential decision for their garment design.

## REFERENCES


Bianconi, F., & Fernández, A. (2007). Evaluation of the effects of Gabor filter parameters on texture classification. *Pattern Recognition*, *40*(12), 3325–3335. https://doi.org/10.1016/j.patcog.2007.04.023

Boden, M. (2001). A guide to recurrent neural networks and backpropagation. *Electrical Engineering*, (2), 1–10. https://doi.org/10.1.1.16.6652

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, *25*(11), 120–123.

Chen, D., Odobez, J. M., & Bourlard, H. (2004). Text detection and recognition in images and video frames. *Pattern Recognition*, *37*(3), 595–608. https://doi.org/10.1016/j.patcog.2003.06.001

Chen, H., Gallagher, A., & Girod, B. (2012). Describing clothing by semantic attributes. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 7574 LNCS, pp. 609–623). https://doi.org/10.1007/978-3-642-33712-3_44

Cortes, C., & Vapnik, V. (1995). Support-Vector Networks. *Machine Learning*. https://doi.org/10.1023/A:1022627411411

E.~Osuna, R.~Freund, & F.~Girosi. (1997). Training Support Vector Machines: An Application to Face Detection. In *Proceedings of CVPR'97, Puerto Rico*. Retrieved from ftp://ftp.ai.mit.edu/pub/cbcl/cvpr97-face.ps.gz

Gardner, M. W., & Dorling, S. R. (1998). Artificial neural networks (the multilayer perceptron) - a review of applications in the atmospheric sciences. *Atmospheric Environment*, *32*(14–15), 2627–2636. https://doi.org/10.1016/S1352-2310(97)00447-0

Gonca Özçelik Kayseri, N. Ö. and G. S. M. (2012). *Sensorial Comfort of Textile Materials, Woven Fabrics, Prof. Han-Yong Jeon (Ed.)Intech*. https://doi.org/10.5772/37596

Gries, D., & Schneider, F. B. (2008). *Fundamentals of the New Artificial Intelligence*. *Texts in Computer Science*. https://doi.org/10.1007/978-1-84628-839-5

Ji, Y., Chang, K. H., & Hung, C.-C. (2004). Efficient edge detection and object segmentation using Gabor filters. *Proceedings of the 42nd Annual Southeast Regional Conference - ACM*, 454–459. https://doi.org/10.1145/986537.986651

Kim, H. S., & Cho, S. B. (2000). Application of interactive genetic algorithm to fashion design. *Engineering Applications of Artificial Intelligence*, *13*(6), 635–644. https://doi.org/10.1016/S0952-1976(00)00045-2

Kim, N. Y., Shin, Y., & Kim, E. Y. (2007). Emotion-based textile indexing usinig neural networks. In *Proceedings of the International Symposium on Consumer Electronics, ISCE*. https://doi.org/10.1109/ISCE.2007.4382230

Kim, S., Kim, E. Y., Jeong, K., & Kim, J. (2006). Emotion-Based Textile Indexing Using Colors , Texture and Patterns. In *ISVC 2006, LNCS 4292* (pp. 9–18).

Kobayashi, S. (1991). *Color Image Scale*. Kosdansha International.

Kokol, P., Verlič, M., & Križmarić, M. (2006). Modelling teens clothing fashion preferences using machine learning. *WSEAS Transactions on Information Science and Applications*, *3*(10), 2054–2065.

Larose, D. T. (2005). *Discovering knowledge in data: an introduction to data mining*. *Statistics* (Vol. 1st). https://doi.org/10.1016/j.cll.2007.10.008

Lavine, B. K., & Blank, T. R. (2009). 3.18 - Feed-Forward Neural Networks. *Comprehensive Chemometrics*, 571–586. https://doi.org/10.1016/B978-044452701-1.00026-0

Li, Y., Cao, L., Zhu, J., & Luo, J. (2017). Mining Fashion Outfit Composition Using An End-to-End Deep Learning Approach on Set Data, *9210*(c), 1–10. https://doi.org/10.1109/TMM.2017.2690144

Masters, T. (1993). *Practical Neural Networks Recipes in C++*. *Book*.

Matlab. (2017). Matlab Documents. Retrieved from https://www.mathworks.com/help/?s_tid=hp_ff_l_doc

Pradhan, A. (2012). SUPPORT VECTOR MACHINE-A Survey. *International Journal of Emerging Technology and Advanced Engineering*, *2*(8), 82–85. Retrieved from http://www.ijetae.com/files/Volume2Issue8/IJETAE_0812_11.pdf

Python Docs. (2017). Python Documents. Retrieved from docs.python.org/3/faq/general.html

Simo-Serra, E., & Ishikawa, H. (2016). Fashion Style in 128 Floats: Joint Ranking and Classification Using Weak Data for Feature Extraction. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 298–307). https://doi.org/10.1109/CVPR.2016.39

Sztandera, L. M. (2009). Tactile fabric comfort prediction using regression analysis. *WSEAS Transactions on Computers*, *8*(2), 292–301.

Unluturk, M. S., Unluturk, S., Pazir, F., & Kuscu, A. (2011). Process neural network method: Case study I: Discrimination of sweet red peppers prepared by different methods. *Eurasip Journal on Advances in Signal Processing*, *2011*. https://doi.org/10.1155/2011/290950

WANG, H. (2014). MACHINE FASHION: AN ARTIFICIAL INTELLIGENCE BASED CLOTHING FASHION STYLIST.

Wang, L. (2005). Support Vector Machines: Theory and Applications. *Studies in Fuzziness and Soft Computing*, 431. https://doi.org/10.1007/3-540-44673-7_12

Xia, M., Zhang, Y., Weng, L., & Ye, X. (2012). Knowledge-Based Systems Fashion retailing forecasting based on extreme learning machine with adaptive metrics of inputs. *Knowledge-Based Systems*, *36*, 253–259. https://doi.org/10.1016/j.knosys.2012.07.002

Yang, J., Liu, L., Jiang, T., & Fan, Y. (2003). A modified Gabor filter design method for fingerprint image enhancement. *Pattern Recognition Letters*, *24*(12), 1805–1817. https://doi.org/10.1016/S0167-8655(03)00005-9

# CURRICULUM VITAE

Caner Kıvanç Hekimoğlu was born in 1993, Burdur – Turkey. He holds a Bachelor of Science degree from Yaşar University, Computer Engineering Department after graduating in 2014. After starting his Master of Science (MSc.) education at Yaşar University, he started to study on Neural Networks and applications, data mining, application development. He simultaneously started as a research assistant at Yaşar University in 2014, and he continues his job since then.

# APPENDIX

Original garment images RGB Mean Standard deviation values after processing with Gabor Filter for 12 orientation and 3 gamma values

| | Gabor Result Gamma:0.2,Theta:15degree | | | | | | Gabor Result Gamma:0.2,Theta:30degree | | | | | | Gabor Result Gamma:0.2,Theta:45degree | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RM | GM | BM | RS | GS | BS | RM | GM | BM | RS | GS | BS | RM | GM | BM | RS | GS | BS |
| 1 | 100.491 | 97.016 | 98.292 | 8.368 | 9.922 | 12.535 | 100.494 | 97.017 | 98.295 | 8.171 | 9.746 | 12.399 | 100.492 | 97.008 | 98.280 | 8.109 | 9.643 | 12.375 |
| 2 | 98.662 | 97.133 | 92.346 | 8.365 | 12.650 | 13.204 | 98.683 | 97.172 | 92.386 | 9.013 | 13.102 | 13.723 | 98.697 | 97.186 | 92.400 | 10.080 | 14.083 | 14.718 |
| 3 | 106.505 | 90.364 | 95.448 | 20.893 | 25.668 | 24.796 | 106.464 | 90.284 | 95.372 | 20.477 | 25.222 | 24.496 | 106.468 | 90.313 | 95.399 | 20.865 | 25.026 | 24.583 |
| 4 | 106.243 | 76.091 | 77.865 | 15.013 | 31.373 | 32.215 | 106.247 | 76.076 | 77.845 | 14.710 | 30.360 | 31.116 | 106.239 | 76.025 | 77.808 | 14.021 | 29.442 | 30.235 |
| 5 | 97.829 | 92.650 | 94.961 | 36.450 | 33.838 | 43.355 | 97.440 | 92.209 | 94.875 | 34.481 | 32.165 | 42.033 | 97.058 | 91.960 | 94.518 | 32.025 | 30.432 | 40.511 |
| 6 | 94.610 | 89.045 | 93.880 | 31.317 | 21.985 | 22.429 | 94.476 | 88.986 | 93.576 | 28.891 | 20.253 | 20.010 | 94.461 | 88.977 | 93.507 | 27.922 | 20.336 | 19.568 |
| 7 | 96.766 | 94.151 | 96.294 | 12.800 | 10.136 | 11.788 | 96.831 | 94.194 | 96.332 | 13.629 | 10.732 | 12.480 | 96.855 | 94.216 | 96.352 | 13.700 | 11.097 | 13.022 |
| 8 | 97.247 | 93.970 | 100.116 | 19.614 | 18.789 | 18.081 | 97.176 | 93.905 | 100.080 | 20.288 | 19.438 | 18.563 | 97.161 | 93.894 | 100.067 | 21.040 | 20.190 | 19.273 |
| 9 | 97.775 | 95.731 | 91.730 | 14.773 | 16.320 | 18.324 | 97.648 | 95.592 | 91.611 | 13.644 | 15.259 | 17.330 | 97.661 | 95.613 | 91.640 | 13.898 | 15.655 | 17.829 |
| 10 | 100.181 | 91.503 | 92.127 | 15.257 | 19.298 | 20.592 | 100.180 | 91.501 | 92.127 | 14.662 | 18.627 | 19.829 | 100.192 | 91.520 | 92.153 | 15.121 | 18.911 | 20.104 |
| 11 | 97.058 | 94.294 | 99.373 | 25.396 | 24.964 | 25.856 | 97.068 | 94.305 | 99.383 | 25.481 | 25.017 | 25.929 | 97.197 | 94.432 | 99.504 | 26.159 | 25.680 | 26.663 |
| 12 | 96.633 | 92.881 | 95.964 | 31.546 | 30.251 | 33.763 | 96.538 | 92.853 | 95.886 | 30.863 | 29.883 | 33.132 | 96.535 | 92.859 | 95.899 | 31.016 | 30.073 | 33.275 |
| 13 | 99.562 | 92.143 | 93.423 | 23.813 | 23.910 | 23.615 | 99.425 | 91.987 | 93.284 | 21.429 | 21.599 | 21.251 | 99.384 | 91.944 | 93.238 | 20.285 | 20.515 | 20.126 |
| 14 | 70.748 | 71.542 | 79.762 | 50.958 | 48.084 | 43.455 | 70.743 | 71.527 | 79.811 | 51.310 | 48.522 | 44.261 | 70.721 | 71.520 | 79.754 | 52.589 | 49.795 | 45.215 |
| 15 | 100.222 | 92.881 | 94.409 | 14.823 | 19.641 | 20.710 | 100.201 | 92.502 | 94.013 | 13.552 | 17.328 | 18.201 | 100.197 | 92.317 | 93.835 | 12.895 | 15.716 | 16.345 |
| 16 | 94.169 | 90.222 | 93.660 | 21.421 | 24.044 | 22.725 | 93.974 | 90.014 | 93.476 | 20.361 | 22.820 | 21.485 | 93.836 | 89.861 | 93.339 | 20.312 | 22.636 | 21.262 |
| 17 | 86.821 | 86.332 | 86.721 | 75.929 | 75.834 | 74.967 | 85.808 | 85.291 | 85.702 | 74.034 | 73.956 | 73.099 | 84.935 | 84.373 | 84.837 | 71.742 | 71.686 | 70.808 |

| Gabor Result Gamma:0.2,Theta:60degree | | | | | | Gabor Result Gamma:0.2,Theta:75degree | | | | | | Gabor Result Gamma:0.2,Theta:90degree | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RM | GM | BM | RS | GS | BS | RM | GM | BM | RS | GS | BS | RM | GM | BM | RS | GS | BS |
| 100.497 | 97.011 | 98.281 | 8.849 | 10.218 | 12.900 | 100.548 | 97.074 | 98.346 | 11.083 | 12.331 | 14.749 | 100.608 | 97.154 | 98.435 | 13.603 | 14.824 | 16.971 |
| 98.694 | 97.191 | 92.427 | 11.424 | 15.586 | 16.196 | 98.719 | 97.235 | 92.482 | 12.367 | 16.847 | 17.618 | 98.721 | 97.244 | 92.483 | 12.686 | 17.409 | 18.197 |
| 106.595 | 90.460 | 95.564 | 22.372 | 25.892 | 25.603 | 106.730 | 90.678 | 95.757 | 24.147 | 27.176 | 27.028 | 106.823 | 90.813 | 95.864 | 24.886 | 27.800 | 27.723 |
| 106.240 | 76.015 | 77.800 | 13.994 | 29.596 | 30.499 | 106.241 | 76.067 | 77.847 | 15.223 | 31.116 | 32.182 | 106.255 | 76.217 | 77.986 | 16.890 | 32.596 | 33.744 |
| 96.795 | 91.765 | 94.065 | 30.215 | 28.540 | 38.994 | 97.249 | 91.800 | 94.677 | 32.346 | 29.032 | 40.389 | 97.484 | 91.671 | 95.218 | 34.556 | 29.896 | 42.720 |
| 94.561 | 89.070 | 93.609 | 28.847 | 22.315 | 21.580 | 94.677 | 89.213 | 93.782 | 30.637 | 24.578 | 23.962 | 94.613 | 89.183 | 93.786 | 31.228 | 25.148 | 24.704 |
| 96.787 | 94.172 | 96.313 | 13.358 | 11.358 | 13.415 | 96.733 | 94.145 | 96.294 | 13.882 | 12.444 | 14.485 | 96.765 | 94.186 | 96.331 | 14.332 | 13.115 | 14.969 |
| 97.256 | 93.987 | 100.136 | 22.366 | 21.544 | 20.692 | 97.469 | 94.201 | 100.292 | 24.257 | 23.515 | 22.918 | 97.431 | 94.171 | 100.267 | 24.249 | 23.526 | 23.022 |
| 97.752 | 95.723 | 91.818 | 14.566 | 16.686 | 18.975 | 97.799 | 95.800 | 91.987 | 15.954 | 18.438 | 20.588 | 97.925 | 95.917 | 92.108 | 18.040 | 20.281 | 22.284 |
| 100.266 | 91.617 | 92.261 | 16.321 | 19.752 | 20.877 | 100.354 | 91.762 | 92.408 | 18.131 | 21.093 | 22.208 | 100.422 | 91.867 | 92.525 | 19.337 | 22.166 | 23.329 |
| 97.152 | 94.392 | 99.462 | 26.186 | 25.728 | 26.755 | 97.247 | 94.493 | 99.562 | 27.096 | 26.676 | 27.702 | 97.239 | 94.495 | 99.556 | 28.086 | 27.669 | 28.712 |
| 96.505 | 92.818 | 95.863 | 31.676 | 30.783 | 34.024 | 96.584 | 92.897 | 95.971 | 32.443 | 31.618 | 35.013 | 96.656 | 92.971 | 96.089 | 33.882 | 33.047 | 36.548 |
| 99.353 | 91.908 | 93.201 | 19.799 | 20.149 | 19.795 | 99.359 | 91.922 | 93.218 | 20.371 | 20.791 | 20.469 | 99.413 | 91.992 | 93.290 | 21.787 | 22.186 | 21.908 |
| 75.373 | 75.219 | 80.883 | 61.270 | 59.279 | 56.852 | 92.193 | 91.317 | 93.657 | 87.540 | 86.144 | 84.324 | 94.201 | 93.293 | 95.076 | 93.253 | 92.150 | 90.840 |
| 100.208 | 92.311 | 93.834 | 13.132 | 15.607 | 16.180 | 100.204 | 92.326 | 93.854 | 14.360 | 16.824 | 17.515 | 100.226 | 92.394 | 93.928 | 16.426 | 18.884 | 19.602 |
| 93.959 | 89.982 | 93.411 | 22.884 | 24.928 | 23.559 | 94.844 | 90.866 | 94.107 | 29.052 | 30.858 | 29.442 | 95.826 | 91.887 | 94.918 | 33.568 | 35.216 | 33.817 |
| 84.900 | 84.335 | 84.803 | 70.817 | 70.743 | 69.839 | 85.219 | 84.656 | 85.096 | 70.977 | 70.881 | 69.907 | 85.463 | 84.908 | 85.323 | 71.361 | 71.258 | 70.249 |

| Gabor Result Gamma:0.2,Theta:105degree | | | | | | Gabor Result Gamma:0.2,Theta:120degree | | | | | | Gabor Result Gamma:0.2,Theta:135degree | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RM | GM | BM | RS | GS | BS | RM | GM | BM | RS | GS | BS | RM | GM | BM | RS | GS | BS |
| 100.533 | 97.066 | 98.340 | 10.917 | 12.396 | 14.651 | 100.494 | 97.011 | 98.279 | 8.549 | 10.182 | 12.690 | 100.487 | 97.004 | 98.268 | 7.712 | 9.415 | 12.164 |
| 98.684 | 97.159 | 92.380 | 10.968 | 15.943 | 16.645 | 98.671 | 97.134 | 92.346 | 9.639 | 14.716 | 15.434 | 98.666 | 97.131 | 92.340 | 8.933 | 13.820 | 14.449 |
| 106.577 | 90.468 | 95.534 | 23.159 | 26.549 | 26.397 | 106.540 | 90.403 | 95.486 | 21.612 | 25.260 | 24.978 | 106.536 | 90.387 | 95.440 | 21.028 | 24.521 | 24.096 |
| 106.242 | 76.113 | 77.894 | 15.884 | 31.836 | 32.949 | 106.247 | 76.039 | 77.833 | 14.192 | 30.218 | 31.230 | 106.240 | 76.018 | 77.812 | 13.390 | 29.705 | 30.698 |
| 97.850 | 91.961 | 95.645 | 36.468 | 31.046 | 45.001 | 98.222 | 92.369 | 96.441 | 38.283 | 33.195 | 47.459 | 98.266 | 92.565 | 96.771 | 38.670 | 33.910 | 47.839 |
| 94.458 | 89.003 | 93.564 | 29.564 | 23.413 | 22.865 | 94.423 | 88.978 | 93.508 | 27.561 | 21.042 | 20.416 | 94.432 | 88.990 | 93.535 | 27.519 | 20.167 | 19.813 |
| 96.704 | 94.116 | 96.264 | 13.024 | 11.566 | 13.345 | 96.732 | 94.125 | 96.258 | 12.078 | 10.260 | 11.953 | 96.757 | 94.153 | 96.278 | 11.878 | 9.819 | 11.316 |
| 97.301 | 94.034 | 100.168 | 21.643 | 20.915 | 20.400 | 97.196 | 93.926 | 100.088 | 20.431 | 19.675 | 19.001 | 97.166 | 93.898 | 100.075 | 20.497 | 19.703 | 18.874 |
| 97.942 | 95.915 | 92.058 | 17.144 | 19.111 | 21.078 | 97.826 | 95.788 | 91.861 | 15.260 | 17.194 | 19.290 | 97.848 | 95.794 | 91.796 | 14.534 | 16.333 | 18.470 |
| 100.255 | 91.661 | 92.329 | 17.879 | 21.129 | 22.289 | 100.209 | 91.556 | 92.203 | 15.667 | 19.507 | 20.729 | 100.179 | 91.507 | 92.134 | 14.569 | 18.739 | 20.029 |
| 97.238 | 94.484 | 99.543 | 27.155 | 26.784 | 27.857 | 97.151 | 94.387 | 99.453 | 25.535 | 25.198 | 26.237 | 97.162 | 94.400 | 99.465 | 25.109 | 24.771 | 25.714 |
| 96.584 | 92.916 | 95.983 | 33.852 | 33.218 | 36.539 | 96.558 | 92.891 | 95.900 | 32.443 | 31.650 | 34.846 | 96.628 | 92.954 | 95.946 | 31.826 | 30.858 | 33.914 |
| 99.419 | 91.995 | 93.298 | 21.747 | 22.143 | 21.845 | 99.459 | 92.026 | 93.322 | 21.742 | 21.995 | 21.764 | 99.512 | 92.070 | 93.372 | 21.713 | 21.766 | 21.564 |
| 90.872 | 89.955 | 92.369 | 85.259 | 83.703 | 81.726 | 74.412 | 74.364 | 80.409 | 60.559 | 58.311 | 55.507 | 71.062 | 71.945 | 80.177 | 53.288 | 50.365 | 45.941 |
| 100.195 | 92.334 | 93.863 | 14.872 | 17.731 | 18.607 | 100.195 | 92.336 | 93.866 | 13.390 | 16.477 | 17.504 | 100.200 | 92.382 | 93.906 | 12.879 | 16.335 | 17.294 |
| 94.813 | 90.850 | 94.045 | 29.408 | 31.253 | 29.842 | 93.963 | 89.994 | 93.431 | 22.690 | 25.040 | 23.648 | 93.843 | 89.881 | 93.355 | 19.733 | 22.475 | 21.214 |
| 85.629 | 85.088 | 85.511 | 72.010 | 71.912 | 70.920 | 85.615 | 85.079 | 85.528 | 72.539 | 72.461 | 71.539 | 85.544 | 84.982 | 85.425 | 73.632 | 73.582 | 72.663 |

| Gabor Result Gamma:0.2,Theta:150degree | | | | | | Gabor Result Gamma:0.2,Theta:165degree | | | | | | Gabor Result Gamma:0.2,Theta:180degree | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RM | GM | BM | RS | GS | BS | RM | GM | BM | RS | GS | BS | RM | GM | BM | RS | GS | BS |
| 100.487 | 97.003 | 98.268 | 7.563 | 9.271 | 12.086 | 100.487 | 97.003 | 98.269 | 7.851 | 9.446 | 12.200 | 100.487 | 97.005 | 98.273 | 8.264 | 9.834 | 12.498 |
| 98.663 | 97.125 | 92.330 | 8.306 | 13.021 | 13.601 | 98.660 | 97.114 | 92.320 | 7.998 | 12.404 | 13.063 | 98.659 | 97.110 | 92.320 | 7.982 | 12.322 | 12.948 |
| 106.549 | 90.355 | 95.402 | 20.490 | 24.224 | 23.688 | 106.538 | 90.371 | 95.419 | 20.676 | 25.061 | 24.240 | 106.491 | 90.344 | 95.421 | 20.683 | 25.798 | 24.810 |
| 106.233 | 76.024 | 77.816 | 13.536 | 30.155 | 31.181 | 106.235 | 76.084 | 77.861 | 14.116 | 31.169 | 32.230 | 106.235 | 76.098 | 77.894 | 14.697 | 31.698 | 32.641 |
| 97.649 | 92.188 | 95.681 | 35.920 | 32.066 | 44.812 | 97.257 | 91.899 | 94.732 | 34.750 | 31.556 | 42.859 | 97.611 | 92.455 | 94.707 | 36.488 | 33.552 | 43.243 |
| 94.468 | 89.007 | 93.626 | 29.561 | 21.429 | 21.905 | 94.556 | 89.102 | 94.062 | 31.877 | 22.999 | 23.805 | 94.630 | 89.161 | 94.264 | 32.748 | 23.216 | 24.148 |
| 96.742 | 94.151 | 96.279 | 11.704 | 9.741 | 11.191 | 96.717 | 94.125 | 96.258 | 11.560 | 9.721 | 11.253 | 96.717 | 94.120 | 96.253 | 12.010 | 9.825 | 11.480 |
| 97.227 | 93.954 | 100.094 | 21.250 | 20.459 | 19.781 | 97.313 | 94.038 | 100.167 | 20.832 | 20.018 | 19.401 | 97.299 | 94.026 | 100.172 | 19.744 | 18.986 | 18.326 |
| 97.958 | 95.880 | 91.850 | 15.258 | 16.807 | 18.647 | 98.030 | 95.954 | 91.932 | 16.184 | 17.602 | 19.281 | 97.971 | 95.915 | 91.907 | 16.097 | 17.608 | 19.436 |
| 100.181 | 91.502 | 92.125 | 14.342 | 18.726 | 20.158 | 100.182 | 91.507 | 92.132 | 14.995 | 19.524 | 21.132 | 100.193 | 91.513 | 92.133 | 15.881 | 20.164 | 21.694 |
| 97.068 | 94.306 | 99.377 | 24.401 | 24.046 | 24.940 | 96.974 | 94.215 | 99.291 | 24.388 | 23.940 | 24.874 | 97.055 | 94.292 | 99.366 | 25.432 | 24.961 | 25.919 |
| 96.674 | 92.917 | 95.950 | 31.778 | 30.569 | 33.663 | 96.750 | 92.928 | 96.040 | 32.079 | 30.613 | 34.061 | 96.864 | 93.035 | 96.204 | 32.272 | 30.704 | 34.347 |
| 99.533 | 92.086 | 93.387 | 22.151 | 22.201 | 22.033 | 99.610 | 92.175 | 93.465 | 23.423 | 23.590 | 23.379 | 99.713 | 92.302 | 93.570 | 24.813 | 24.974 | 24.692 |
| 70.868 | 71.805 | 80.115 | 51.471 | 48.605 | 44.373 | 70.911 | 71.764 | 80.000 | 51.001 | 47.850 | 43.184 | 70.087 | 70.973 | 79.342 | 49.492 | 46.578 | 41.858 |
| 100.195 | 92.411 | 93.918 | 12.164 | 16.265 | 17.258 | 100.196 | 92.589 | 94.098 | 12.808 | 17.603 | 18.672 | 100.239 | 92.890 | 94.401 | 14.719 | 19.816 | 20.948 |
| 93.938 | 89.996 | 93.458 | 19.435 | 22.507 | 21.296 | 94.162 | 90.228 | 93.654 | 20.821 | 24.041 | 22.882 | 94.244 | 90.323 | 93.739 | 21.846 | 24.854 | 23.629 |
| 86.228 | 85.672 | 86.065 | 75.765 | 75.693 | 74.757 | 87.115 | 86.612 | 87.012 | 77.120 | 77.049 | 76.193 | 87.328 | 86.848 | 87.249 | 76.971 | 76.903 | 76.076 |

| Gabor Result Gamma:0.4,Theta:15degree | | | | | | Gabor Result Gamma:0.4,Theta:30degree | | | | | | Gabor Result Gamma:0.4,Theta:45degree | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RM | GM | BM | RS | GS | BS | RM | GM | BM | RS | GS | BS | RM | GM | BM | RS | GS | BS |
| 100.492 | 97.010 | 98.283 | 7.367 | 8.910 | 11.472 | 100.494 | 97.015 | 98.288 | 7.207 | 8.761 | 11.343 | 100.491 | 97.005 | 98.274 | 7.021 | 8.477 | 11.184 |
| 98.662 | 97.117 | 92.325 | 7.112 | 11.095 | 11.665 | 98.670 | 97.153 | 92.365 | 7.795 | 11.611 | 12.242 | 98.690 | 97.167 | 92.375 | 8.601 | 12.249 | 12.859 |
| 106.347 | 90.142 | 95.230 | 19.020 | 23.843 | 23.003 | 106.262 | 90.013 | 95.100 | 18.137 | 23.204 | 22.457 | 106.254 | 90.016 | 95.099 | 18.493 | 23.083 | 22.611 |
| 106.234 | 76.012 | 77.788 | 13.850 | 29.965 | 30.732 | 106.238 | 76.019 | 77.794 | 13.599 | 28.936 | 29.593 | 106.233 | 75.976 | 77.765 | 12.783 | 27.935 | 28.649 |
| 97.163 | 92.012 | 93.924 | 32.910 | 30.652 | 39.077 | 96.773 | 91.585 | 93.836 | 30.360 | 28.277 | 37.734 | 96.609 | 91.476 | 93.633 | 28.537 | 27.344 | 36.721 |
| 94.504 | 89.001 | 93.665 | 27.914 | 19.486 | 19.875 | 94.416 | 88.974 | 93.515 | 25.255 | 17.546 | 16.878 | 94.411 | 88.968 | 93.485 | 24.612 | 17.647 | 16.542 |
| 96.701 | 94.113 | 96.249 | 11.219 | 8.958 | 10.337 | 96.760 | 94.153 | 96.288 | 12.239 | 9.627 | 11.099 | 96.789 | 94.182 | 96.312 | 12.308 | 9.943 | 11.605 |
| 97.175 | 93.897 | 100.074 | 17.180 | 16.400 | 15.660 | 97.123 | 93.858 | 100.061 | 17.756 | 16.913 | 15.990 | 97.122 | 93.853 | 100.060 | 18.240 | 17.438 | 16.447 |
| 97.601 | 95.552 | 91.516 | 13.227 | 14.593 | 16.562 | 97.510 | 95.450 | 91.425 | 12.213 | 13.577 | 15.541 | 97.533 | 95.473 | 91.449 | 12.590 | 14.067 | 16.069 |
| 100.181 | 91.499 | 92.122 | 13.179 | 16.795 | 17.838 | 100.180 | 91.496 | 92.121 | 12.567 | 16.183 | 17.179 | 100.181 | 91.501 | 92.126 | 12.940 | 16.463 | 17.501 |
| 96.758 | 93.988 | 99.086 | 22.289 | 21.939 | 22.657 | 96.754 | 93.981 | 99.080 | 22.195 | 21.805 | 22.531 | 96.872 | 94.103 | 99.189 | 23.489 | 23.062 | 23.890 |
| 96.119 | 92.392 | 95.378 | 27.572 | 26.477 | 29.810 | 96.106 | 92.417 | 95.388 | 27.060 | 26.330 | 29.340 | 96.072 | 92.368 | 95.356 | 27.263 | 26.552 | 29.522 |
| 99.425 | 91.988 | 93.273 | 21.475 | 21.724 | 21.364 | 99.355 | 91.905 | 93.202 | 18.885 | 19.193 | 18.789 | 99.333 | 91.881 | 93.176 | 18.118 | 18.488 | 18.016 |
| 70.017 | 70.892 | 79.331 | 47.805 | 44.834 | 40.216 | 70.484 | 71.315 | 79.677 | 49.472 | 46.778 | 42.620 | 69.918 | 70.812 | 79.282 | 47.248 | 44.430 | 39.972 |
| 100.202 | 92.723 | 94.243 | 13.382 | 18.131 | 19.067 | 100.197 | 92.354 | 93.867 | 12.029 | 15.400 | 15.999 | 100.195 | 92.284 | 93.813 | 11.338 | 13.918 | 14.371 |
| 94.034 | 90.060 | 93.516 | 20.003 | 22.308 | 20.986 | 93.859 | 89.881 | 93.357 | 18.631 | 20.852 | 19.522 | 93.779 | 89.782 | 93.276 | 18.382 | 20.489 | 19.112 |
| 86.122 | 85.603 | 85.995 | 72.906 | 72.814 | 71.947 | 84.860 | 84.310 | 84.766 | 69.928 | 69.902 | 69.087 | 84.386 | 83.791 | 84.309 | 67.641 | 67.633 | 66.825 |

| Gabor Result Gamma:0.4,Theta:60degree | | | | | | Gabor Result Gamma:0.4,Theta:75degree | | | | | | Gabor Result Gamma:0.4,Theta:90degree | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RM | GM | BM | RS | GS | BS | RM | GM | BM | RS | GS | BS | RM | GM | BM | RS | GS | BS |
| 100.491 | 97.005 | 98.270 | 7.578 | 8.864 | 11.489 | 100.514 | 97.033 | 98.299 | 9.136 | 10.324 | 12.755 | 100.588 | 97.126 | 98.401 | 12.729 | 13.881 | 15.964 |
| 98.676 | 97.150 | 92.371 | 9.796 | 13.669 | 14.261 | 98.698 | 97.175 | 92.406 | 10.663 | 14.829 | 15.597 | 98.693 | 97.184 | 92.412 | 11.350 | 15.809 | 16.539 |
| 106.382 | 90.140 | 95.246 | 19.784 | 23.851 | 23.470 | 106.457 | 90.268 | 95.366 | 21.370 | 25.125 | 24.855 | 106.581 | 90.463 | 95.519 | 22.891 | 26.054 | 25.899 |
| 106.236 | 75.972 | 77.761 | 12.662 | 27.981 | 28.774 | 106.241 | 76.005 | 77.786 | 13.486 | 29.259 | 30.227 | 106.251 | 76.142 | 77.918 | 15.702 | 31.290 | 32.391 |
| 96.293 | 91.239 | 92.938 | 25.580 | 25.056 | 33.718 | 96.712 | 91.357 | 93.695 | 28.721 | 25.915 | 36.220 | 96.934 | 91.262 | 94.152 | 30.701 | 26.694 | 38.415 |
| 94.450 | 89.000 | 93.515 | 25.524 | 19.522 | 18.643 | 94.588 | 89.150 | 93.692 | 27.547 | 22.134 | 21.553 | 94.478 | 89.087 | 93.672 | 28.193 | 22.568 | 22.271 |
| 96.718 | 94.123 | 96.259 | 11.627 | 9.992 | 11.776 | 96.678 | 94.099 | 96.240 | 11.922 | 10.767 | 12.691 | 96.717 | 94.141 | 96.282 | 13.056 | 12.184 | 13.768 |
| 97.160 | 93.891 | 100.079 | 19.339 | 18.553 | 17.643 | 97.346 | 94.078 | 100.216 | 21.598 | 20.885 | 20.187 | 97.337 | 94.069 | 100.189 | 21.696 | 21.033 | 20.496 |
| 97.623 | 95.573 | 91.606 | 13.393 | 15.208 | 17.357 | 97.644 | 95.622 | 91.732 | 13.902 | 16.304 | 18.568 | 97.790 | 95.764 | 91.875 | 16.809 | 18.859 | 20.810 |
| 100.229 | 91.567 | 92.206 | 14.068 | 17.232 | 18.190 | 100.304 | 91.680 | 92.325 | 15.640 | 18.434 | 19.377 | 100.358 | 91.777 | 92.421 | 17.581 | 20.213 | 21.288 |
| 96.749 | 93.980 | 99.073 | 22.733 | 22.361 | 23.193 | 96.901 | 94.138 | 99.221 | 23.956 | 23.604 | 24.496 | 96.835 | 94.077 | 99.163 | 25.089 | 24.754 | 25.631 |
| 96.064 | 92.350 | 95.336 | 27.816 | 27.110 | 30.100 | 96.105 | 92.410 | 95.392 | 28.159 | 27.507 | 30.695 | 96.195 | 92.474 | 95.519 | 30.030 | 29.283 | 32.689 |
| 99.321 | 91.863 | 93.158 | 17.683 | 18.145 | 17.720 | 99.319 | 91.865 | 93.163 | 17.930 | 18.467 | 18.068 | 99.347 | 91.912 | 93.210 | 19.812 | 20.304 | 19.996 |
| 69.786 | 70.724 | 79.284 | 49.590 | 46.655 | 42.038 | 85.261 | 84.565 | 88.026 | 74.311 | 72.486 | 70.460 | 93.710 | 92.831 | 94.704 | 92.534 | 91.432 | 90.113 |
| 100.205 | 92.298 | 93.824 | 11.641 | 14.022 | 14.422 | 100.196 | 92.293 | 93.821 | 12.344 | 14.813 | 15.403 | 100.216 | 92.353 | 93.887 | 15.212 | 17.658 | 18.252 |
| 93.804 | 89.808 | 93.292 | 19.766 | 21.768 | 20.401 | 94.384 | 90.392 | 93.740 | 25.303 | 27.050 | 25.639 | 95.592 | 91.603 | 94.654 | 31.900 | 33.402 | 31.963 |
| 84.297 | 83.698 | 84.216 | 67.088 | 67.055 | 66.272 | 84.766 | 84.189 | 84.652 | 67.792 | 67.728 | 66.864 | 85.031 | 84.452 | 84.880 | 68.354 | 68.259 | 67.306 |

| Gabor Result Gamma:0.4,Theta:105degree | | | | | | Gabor Result Gamma:0.4,Theta:120degree | | | | | | Gabor Result Gamma:0.4,Theta:135degree | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RM | GM | BM | RS | GS | BS | RM | GM | BM | RS | GS | BS | RM | GM | BM | RS | GS | BS |
| 100.506 | 97.028 | 98.298 | 8.981 | 10.474 | 12.715 | 100.487 | 97.003 | 98.267 | 7.253 | 8.775 | 11.239 | 100.488 | 97.003 | 98.267 | 6.691 | 8.335 | 11.032 |
| 98.671 | 97.121 | 92.327 | 9.154 | 13.696 | 14.363 | 98.665 | 97.111 | 92.322 | 8.138 | 12.826 | 13.520 | 98.663 | 97.111 | 92.322 | 7.586 | 12.082 | 12.680 |
| 106.331 | 90.067 | 95.140 | 20.459 | 24.416 | 24.128 | 106.340 | 90.125 | 95.221 | 19.189 | 23.323 | 22.997 | 106.335 | 90.101 | 95.158 | 18.903 | 22.766 | 22.289 |
| 106.240 | 76.026 | 77.811 | 14.127 | 29.962 | 30.982 | 106.237 | 75.995 | 77.788 | 12.829 | 28.484 | 29.389 | 106.228 | 75.974 | 77.765 | 12.210 | 28.125 | 29.014 |
| 96.880 | 91.239 | 94.000 | 31.215 | 26.461 | 39.092 | 97.578 | 91.919 | 94.855 | 34.170 | 29.872 | 42.394 | 97.524 | 92.023 | 95.402 | 34.560 | 30.586 | 43.585 |
| 94.400 | 88.967 | 93.485 | 26.248 | 20.439 | 19.815 | 94.397 | 88.967 | 93.477 | 24.281 | 18.393 | 17.565 | 94.399 | 88.972 | 93.499 | 24.161 | 17.517 | 16.791 |
| 96.664 | 94.089 | 96.229 | 11.034 | 9.818 | 11.418 | 96.687 | 94.097 | 96.226 | 10.593 | 9.018 | 10.459 | 96.721 | 94.127 | 96.249 | 10.517 | 8.752 | 9.968 |
| 97.211 | 93.945 | 100.111 | 18.685 | 17.983 | 17.410 | 97.127 | 93.862 | 100.056 | 17.754 | 17.020 | 16.249 | 97.117 | 93.852 | 100.057 | 17.804 | 17.037 | 16.152 |
| 97.794 | 95.745 | 91.802 | 15.570 | 17.307 | 19.223 | 97.637 | 95.582 | 91.585 | 13.404 | 15.167 | 17.320 | 97.658 | 95.599 | 91.563 | 12.921 | 14.545 | 16.663 |
| 100.210 | 91.574 | 92.233 | 15.351 | 18.537 | 19.614 | 100.198 | 91.533 | 92.170 | 13.372 | 16.944 | 18.033 | 100.177 | 91.493 | 92.116 | 12.413 | 16.274 | 17.386 |
| 96.854 | 94.088 | 99.166 | 24.131 | 23.825 | 24.759 | 96.796 | 94.024 | 99.111 | 22.258 | 21.989 | 22.859 | 96.832 | 94.064 | 99.148 | 22.352 | 22.080 | 22.844 |
| 96.128 | 92.446 | 95.433 | 29.935 | 29.581 | 32.679 | 96.061 | 92.352 | 95.311 | 28.122 | 27.539 | 30.467 | 96.102 | 92.413 | 95.362 | 28.010 | 27.311 | 30.074 |
| 99.342 | 91.902 | 93.202 | 19.258 | 19.798 | 19.448 | 99.357 | 91.909 | 93.203 | 19.440 | 19.809 | 19.518 | 99.394 | 91.941 | 93.242 | 19.433 | 19.582 | 19.310 |
| 82.282 | 81.684 | 85.657 | 69.877 | 67.936 | 65.963 | 69.699 | 70.625 | 79.193 | 49.380 | 46.350 | 41.702 | 70.585 | 71.507 | 79.875 | 49.312 | 46.333 | 41.968 |
| 100.193 | 92.301 | 93.829 | 12.766 | 15.591 | 16.320 | 100.193 | 92.295 | 93.822 | 11.692 | 14.674 | 15.584 | 100.195 | 92.328 | 93.852 | 11.586 | 14.864 | 15.654 |
| 94.272 | 90.273 | 93.618 | 25.625 | 27.514 | 26.007 | 93.815 | 89.818 | 93.297 | 19.714 | 22.024 | 20.664 | 93.783 | 89.793 | 93.287 | 17.886 | 20.329 | 19.048 |
| 85.107 | 84.545 | 84.985 | 69.093 | 69.019 | 68.098 | 85.067 | 84.515 | 84.992 | 69.142 | 69.100 | 68.280 | 84.848 | 84.258 | 84.747 | 69.449 | 69.429 | 68.572 |

| Gabor Result Gamma:0.4,Theta:150degree | | | | | | Gabor Result Gamma:0.4,Theta:165degree | | | | | | Gabor Result Gamma:0.4,Theta:180degree | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RM | GM | BM | RS | GS | BS | RM | GM | BM | RS | GS | BS | RM | GM | BM | RS | GS | BS |
| 100.489 | 97.004 | 98.268 | 6.615 | 8.267 | 11.007 | 100.488 | 97.004 | 98.268 | 6.862 | 8.395 | 11.076 | 100.487 | 97.003 | 98.266 | 7.291 | 8.804 | 11.432 |
| 98.662 | 97.110 | 92.318 | 7.070 | 11.425 | 11.958 | 98.660 | 97.108 | 92.314 | 6.881 | 10.801 | 11.441 | 98.658 | 97.106 | 92.313 | 6.802 | 10.735 | 11.358 |
| 106.368 | 90.082 | 95.149 | 18.410 | 22.248 | 21.708 | 106.345 | 90.076 | 95.143 | 18.477 | 23.241 | 22.410 | 106.298 | 90.076 | 95.166 | 18.611 | 24.107 | 23.097 |
| 106.227 | 75.973 | 77.761 | 12.393 | 28.546 | 29.457 | 106.229 | 76.008 | 77.780 | 12.952 | 29.759 | 30.746 | 106.230 | 76.013 | 77.812 | 13.508 | 30.324 | 31.235 |
| 96.897 | 91.561 | 94.479 | 31.124 | 27.846 | 39.814 | 96.644 | 91.352 | 93.479 | 30.027 | 27.277 | 37.522 | 96.985 | 91.837 | 93.605 | 32.704 | 30.303 | 38.601 |
| 94.414 | 88.989 | 93.531 | 25.844 | 18.711 | 18.815 | 94.474 | 89.037 | 93.846 | 28.763 | 20.685 | 21.606 | 94.530 | 89.103 | 94.121 | 30.271 | 21.245 | 22.353 |
| 96.702 | 94.120 | 96.245 | 10.340 | 8.770 | 9.984 | 96.674 | 94.093 | 96.223 | 10.122 | 8.656 | 9.956 | 96.681 | 94.094 | 96.224 | 10.371 | 8.619 | 10.009 |
| 97.141 | 93.870 | 100.064 | 18.783 | 18.026 | 17.191 | 97.239 | 93.958 | 100.114 | 18.233 | 17.455 | 16.814 | 97.269 | 93.991 | 100.149 | 17.375 | 16.642 | 15.975 |
| 97.750 | 95.675 | 91.603 | 13.979 | 15.298 | 17.012 | 97.898 | 95.821 | 91.736 | 15.416 | 16.476 | 18.023 | 97.805 | 95.744 | 91.691 | 15.146 | 16.394 | 18.112 |
| 100.178 | 91.496 | 92.117 | 12.312 | 16.247 | 17.433 | 100.181 | 91.500 | 92.121 | 12.915 | 17.093 | 18.542 | 100.191 | 91.507 | 92.125 | 14.309 | 18.098 | 19.418 |
| 96.772 | 94.004 | 99.091 | 21.348 | 21.087 | 21.791 | 96.698 | 93.929 | 99.024 | 21.105 | 20.742 | 21.512 | 96.761 | 93.990 | 99.080 | 22.575 | 22.166 | 22.983 |
| 96.126 | 92.367 | 95.330 | 27.910 | 26.908 | 29.730 | 96.179 | 92.380 | 95.406 | 28.411 | 27.173 | 30.412 | 96.347 | 92.531 | 95.611 | 28.781 | 27.412 | 30.956 |
| 99.421 | 91.965 | 93.268 | 19.884 | 20.006 | 19.779 | 99.434 | 91.985 | 93.270 | 21.048 | 21.386 | 21.143 | 99.526 | 92.102 | 93.378 | 22.942 | 23.222 | 22.848 |
| 70.666 | 71.632 | 79.996 | 49.800 | 47.000 | 42.868 | 69.945 | 70.868 | 79.333 | 47.209 | 44.019 | 39.418 | 69.589 | 70.587 | 79.166 | 46.774 | 43.709 | 38.912 |
| 100.193 | 92.329 | 93.846 | 10.694 | 14.454 | 15.232 | 100.194 | 92.439 | 93.948 | 11.103 | 15.693 | 16.505 | 100.221 | 92.726 | 94.219 | 13.474 | 18.399 | 19.303 |
| 93.843 | 89.880 | 93.357 | 17.823 | 20.609 | 19.390 | 94.020 | 90.044 | 93.495 | 19.391 | 22.339 | 21.168 | 94.117 | 90.153 | 93.590 | 20.722 | 23.469 | 22.215 |
| 85.038 | 84.439 | 84.891 | 70.958 | 70.911 | 69.977 | 86.441 | 85.911 | 86.331 | 74.011 | 73.943 | 73.056 | 86.891 | 86.387 | 86.778 | 74.554 | 74.477 | 73.632 |

| Gabor Result Gamma:0.6,Theta:15degree | | | | | | Gabor Result Gamma:0.6,Theta:30degree | | | | | | Gabor Result Gamma:0.6,Theta:45degree | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RM | GM | BM | RS | GS | BS | RM | GM | BM | RS | GS | BS | RM | GM | BM | RS | GS | BS |
| 100.501 | 97.014 | 98.280 | 5.746 | 7.232 | 9.770 | 100.502 | 97.014 | 98.283 | 5.775 | 7.295 | 9.803 | 100.502 | 97.014 | 98.278 | 5.478 | 6.799 | 9.370 |
| 98.664 | 97.115 | 92.322 | 5.390 | 8.646 | 9.223 | 98.667 | 97.124 | 92.336 | 5.873 | 9.447 | 10.018 | 98.676 | 97.143 | 92.351 | 6.641 | 9.650 | 10.160 |
| 106.152 | 89.813 | 94.918 | 16.112 | 21.057 | 20.289 | 106.064 | 89.659 | 94.766 | 14.430 | 20.230 | 19.493 | 106.083 | 89.683 | 94.794 | 14.795 | 20.113 | 19.528 |
| 106.229 | 75.935 | 77.722 | 12.117 | 27.686 | 28.312 | 106.234 | 75.950 | 77.737 | 11.855 | 26.658 | 27.188 | 106.236 | 75.950 | 77.745 | 11.111 | 25.643 | 26.246 |
| 96.299 | 91.171 | 92.682 | 27.451 | 25.659 | 32.266 | 96.097 | 90.968 | 92.640 | 23.877 | 22.342 | 30.895 | 96.081 | 90.942 | 92.513 | 23.431 | 22.637 | 30.490 |
| 94.395 | 88.972 | 93.538 | 21.735 | 15.442 | 15.139 | 94.387 | 88.967 | 93.480 | 19.570 | 14.071 | 13.067 | 94.386 | 88.965 | 93.476 | 19.532 | 14.127 | 12.842 |
| 96.672 | 94.090 | 96.222 | 8.458 | 6.987 | 7.945 | 96.700 | 94.112 | 96.240 | 9.698 | 7.861 | 8.904 | 96.717 | 94.124 | 96.250 | 9.875 | 8.295 | 9.404 |
| 97.118 | 93.847 | 100.060 | 13.533 | 12.813 | 12.041 | 97.118 | 93.846 | 100.064 | 13.687 | 12.970 | 12.137 | 97.121 | 93.846 | 100.064 | 14.031 | 13.365 | 12.468 |
| 97.466 | 95.408 | 91.323 | 10.509 | 11.613 | 13.522 | 97.447 | 95.375 | 91.287 | 9.683 | 10.875 | 12.858 | 97.446 | 95.375 | 91.279 | 10.187 | 11.402 | 13.367 |
| 100.180 | 91.495 | 92.119 | 10.214 | 13.246 | 13.973 | 100.179 | 91.492 | 92.119 | 9.867 | 12.940 | 13.639 | 100.177 | 91.491 | 92.119 | 9.964 | 12.990 | 13.765 |
| 96.592 | 93.812 | 98.932 | 16.765 | 16.568 | 16.993 | 96.589 | 93.806 | 98.928 | 16.437 | 16.162 | 16.579 | 96.606 | 93.826 | 98.943 | 18.758 | 18.479 | 18.976 |
| 95.872 | 92.171 | 95.103 | 20.236 | 19.385 | 22.167 | 95.885 | 92.189 | 95.116 | 20.742 | 20.349 | 22.880 | 95.880 | 92.155 | 95.109 | 19.974 | 19.506 | 22.036 |
| 99.327 | 91.870 | 93.164 | 17.520 | 17.983 | 17.535 | 99.315 | 91.854 | 93.153 | 15.395 | 15.847 | 15.372 | 99.310 | 91.847 | 93.145 | 14.867 | 15.378 | 14.808 |
| 69.073 | 70.115 | 78.918 | 42.847 | 39.854 | 35.325 | 70.156 | 71.026 | 79.462 | 46.909 | 44.391 | 40.384 | 69.049 | 70.097 | 78.913 | 41.658 | 38.784 | 34.438 |
| 100.191 | 92.486 | 94.003 | 10.652 | 15.434 | 16.226 | 100.191 | 92.278 | 93.808 | 9.788 | 12.472 | 12.699 | 100.192 | 92.280 | 93.810 | 9.128 | 11.435 | 11.814 |
| 93.818 | 89.833 | 93.306 | 17.718 | 19.555 | 18.231 | 93.764 | 89.768 | 93.261 | 15.986 | 17.906 | 16.564 | 93.755 | 89.752 | 93.251 | 15.641 | 17.459 | 16.108 |
| 84.745 | 84.180 | 84.628 | 65.487 | 65.495 | 64.748 | 83.770 | 83.168 | 83.715 | 61.620 | 61.680 | 60.956 | 83.605 | 83.000 | 83.559 | 59.793 | 59.866 | 59.184 |

| Gabor Result Gamma:0.6,Theta:60degree | | | | | | Gabor Result Gamma:0.6,Theta:75degree | | | | | | Gabor Result Gamma:0.6,Theta:90degree | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RM | GM | BM | RS | GS | BS | RM | GM | BM | RS | GS | BS | RM | GM | BM | RS | GS | BS |
| 100.498 | 97.010 | 98.276 | 5.964 | 7.111 | 9.518 | 100.491 | 97.003 | 98.269 | 6.573 | 7.638 | 9.944 | 100.555 | 97.084 | 98.348 | 11.258 | 12.263 | 14.181 |
| 98.665 | 97.123 | 92.337 | 7.405 | 10.780 | 11.217 | 98.670 | 97.128 | 92.339 | 8.352 | 11.755 | 12.414 | 98.665 | 97.118 | 92.335 | 9.130 | 13.230 | 13.800 |
| 106.148 | 89.785 | 94.896 | 15.872 | 20.641 | 20.176 | 106.163 | 89.813 | 94.935 | 16.913 | 21.753 | 21.389 | 106.292 | 90.020 | 95.109 | 19.552 | 23.177 | 22.929 |
| 106.236 | 75.949 | 77.742 | 10.969 | 25.492 | 26.116 | 106.238 | 75.957 | 77.748 | 11.224 | 26.132 | 26.868 | 106.243 | 76.052 | 77.834 | 13.673 | 28.720 | 29.682 |
| 95.849 | 90.739 | 92.029 | 20.268 | 20.370 | 26.959 | 96.038 | 90.859 | 92.394 | 23.220 | 21.240 | 29.525 | 96.208 | 90.843 | 92.985 | 25.985 | 22.355 | 32.782 |
| 94.391 | 88.964 | 93.473 | 19.781 | 15.244 | 14.251 | 94.511 | 89.090 | 93.602 | 22.040 | 18.124 | 17.548 | 94.370 | 88.978 | 93.516 | 22.591 | 18.391 | 18.355 |
| 96.671 | 94.088 | 96.219 | 8.936 | 7.988 | 9.237 | 96.663 | 94.082 | 96.211 | 8.796 | 8.043 | 9.649 | 96.690 | 94.108 | 96.222 | 10.859 | 10.474 | 11.745 |
| 97.117 | 93.847 | 100.062 | 14.775 | 14.080 | 13.225 | 97.205 | 93.937 | 100.124 | 17.337 | 16.700 | 15.994 | 97.234 | 93.960 | 100.116 | 17.687 | 17.102 | 16.529 |
| 97.479 | 95.408 | 91.365 | 11.346 | 12.775 | 14.636 | 97.491 | 95.436 | 91.415 | 11.399 | 13.404 | 15.655 | 97.602 | 95.553 | 91.544 | 14.355 | 16.096 | 17.987 |
| 100.188 | 91.509 | 92.136 | 10.909 | 13.657 | 14.357 | 100.250 | 91.599 | 92.240 | 12.049 | 14.458 | 15.168 | 100.299 | 91.678 | 92.298 | 14.773 | 17.090 | 17.997 |
| 96.594 | 93.811 | 98.930 | 16.326 | 16.142 | 16.606 | 96.597 | 93.818 | 98.928 | 18.797 | 18.596 | 19.159 | 96.605 | 93.829 | 98.939 | 19.430 | 19.253 | 19.852 |
| 95.867 | 92.157 | 95.097 | 21.012 | 20.464 | 22.989 | 95.882 | 92.179 | 95.119 | 20.913 | 20.472 | 23.129 | 95.970 | 92.242 | 95.210 | 23.629 | 22.668 | 25.937 |
| 99.308 | 91.843 | 93.142 | 14.562 | 15.067 | 14.584 | 99.308 | 91.846 | 93.142 | 14.363 | 14.935 | 14.459 | 99.316 | 91.859 | 93.152 | 16.407 | 16.965 | 16.662 |
| 69.176 | 70.231 | 79.017 | 42.395 | 39.549 | 35.223 | 70.316 | 71.119 | 79.399 | 50.117 | 47.420 | 43.162 | 92.232 | 91.398 | 93.520 | 90.762 | 89.650 | 88.250 |
| 100.198 | 92.287 | 93.816 | 9.489 | 11.756 | 11.979 | 100.194 | 92.282 | 93.810 | 9.379 | 11.772 | 12.255 | 100.197 | 92.307 | 93.834 | 12.960 | 15.203 | 15.720 |
| 93.758 | 89.763 | 93.258 | 16.512 | 18.226 | 16.860 | 93.935 | 89.941 | 93.373 | 19.855 | 21.543 | 20.171 | 94.932 | 90.900 | 94.034 | 28.315 | 29.617 | 28.051 |
| 83.589 | 82.968 | 83.536 | 59.580 | 59.612 | 58.969 | 83.817 | 83.204 | 83.739 | 60.334 | 60.336 | 59.624 | 84.312 | 83.714 | 84.194 | 61.808 | 61.760 | 60.946 |

| Gabor Result Gamma:0.6,Theta:105degree | | | | | | Gabor Result Gamma:0.6,Theta:120degree | | | | | | Gabor Result Gamma:0.6,Theta:135degree | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RM | GM | BM | RS | GS | BS | RM | GM | BM | RS | GS | BS | RM | GM | BM | RS | GS | BS |
| 100.486 | 97.002 | 98.265 | 6.704 | 8.040 | 10.159 | 100.485 | 97.003 | 98.266 | 5.637 | 6.925 | 9.251 | 100.487 | 97.006 | 98.270 | 5.293 | 6.783 | 9.298 |
| 98.668 | 97.109 | 92.316 | 6.788 | 10.491 | 10.969 | 98.664 | 97.108 | 92.314 | 6.105 | 9.844 | 10.471 | 98.662 | 97.105 | 92.311 | 5.717 | 9.355 | 9.907 |
| 106.102 | 89.675 | 94.781 | 15.999 | 20.910 | 20.413 | 106.089 | 89.748 | 94.870 | 15.459 | 20.408 | 20.014 | 106.108 | 89.718 | 94.812 | 15.389 | 19.975 | 19.429 |
| 106.231 | 75.957 | 77.744 | 11.604 | 26.586 | 27.364 | 106.218 | 75.934 | 77.724 | 11.080 | 25.822 | 26.564 | 106.212 | 75.912 | 77.696 | 10.730 | 25.806 | 26.520 |
| 95.984 | 90.706 | 92.417 | 23.976 | 20.654 | 30.894 | 96.798 | 91.442 | 93.382 | 28.991 | 25.639 | 35.615 | 96.529 | 91.220 | 93.628 | 28.081 | 25.125 | 36.448 |
| 94.377 | 88.958 | 93.466 | 20.615 | 15.984 | 15.164 | 94.379 | 88.963 | 93.471 | 19.099 | 14.677 | 13.762 | 94.379 | 88.965 | 93.472 | 18.788 | 13.942 | 13.131 |
| 96.660 | 94.079 | 96.207 | 8.123 | 7.370 | 8.584 | 96.662 | 94.080 | 96.206 | 8.138 | 7.104 | 8.046 | 96.670 | 94.089 | 96.207 | 8.340 | 7.200 | 7.924 |
| 97.128 | 93.862 | 100.064 | 14.684 | 14.042 | 13.418 | 97.105 | 93.841 | 100.054 | 13.812 | 13.136 | 12.342 | 97.107 | 93.842 | 100.058 | 13.989 | 13.340 | 12.534 |
| 97.586 | 95.515 | 91.446 | 13.099 | 14.443 | 16.322 | 97.457 | 95.389 | 91.313 | 10.883 | 12.295 | 14.304 | 97.470 | 95.406 | 91.323 | 10.857 | 12.118 | 14.069 |
| 100.180 | 91.505 | 92.141 | 11.501 | 14.496 | 15.468 | 100.177 | 91.500 | 92.124 | 10.325 | 13.422 | 14.283 | 100.170 | 91.486 | 92.105 | 9.542 | 12.869 | 13.682 |
| 96.586 | 93.803 | 98.914 | 18.538 | 18.371 | 18.940 | 96.588 | 93.807 | 98.919 | 16.502 | 16.362 | 16.873 | 96.595 | 93.816 | 98.928 | 17.288 | 17.137 | 17.581 |
| 95.870 | 92.172 | 95.102 | 23.075 | 23.027 | 25.678 | 95.864 | 92.154 | 95.083 | 20.083 | 19.547 | 22.024 | 95.868 | 92.161 | 95.087 | 21.089 | 20.639 | 22.942 |
| 99.309 | 91.848 | 93.143 | 15.444 | 16.089 | 15.698 | 99.309 | 91.846 | 93.141 | 15.582 | 16.034 | 15.671 | 99.313 | 91.849 | 93.145 | 15.709 | 15.990 | 15.629 |
| 69.584 | 70.507 | 79.090 | 47.574 | 44.777 | 40.264 | 69.145 | 70.178 | 78.962 | 42.625 | 39.707 | 35.346 | 69.284 | 70.299 | 78.989 | 43.021 | 40.091 | 35.792 |
| 100.191 | 92.279 | 93.804 | 9.761 | 12.291 | 12.809 | 100.190 | 92.276 | 93.802 | 9.074 | 11.790 | 12.435 | 100.187 | 92.280 | 93.808 | 9.442 | 12.329 | 12.890 |
| 93.862 | 89.864 | 93.311 | 19.927 | 21.754 | 20.267 | 93.755 | 89.756 | 93.248 | 16.284 | 18.265 | 16.908 | 93.751 | 89.751 | 93.245 | 15.321 | 17.342 | 16.051 |
| 84.033 | 83.440 | 83.943 | 61.972 | 61.957 | 61.183 | 84.106 | 83.522 | 84.063 | 61.752 | 61.761 | 61.040 | 83.939 | 83.327 | 83.864 | 61.207 | 61.221 | 60.452 |

| Gabor Result Gamma:0.6,Theta:150degree | | | | | | Gabor Result Gamma:0.6,Theta:165degree | | | | | | Gabor Result Gamma:0.6,Theta:180degree | | | | | | Accept? |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RM | GM | BM | RS | GS | BS | RM | GM | BM | RS | GS | BS | RM | GM | BM | RS | GS | BS | |
| 100.490 | 97.007 | 98.270 | 5.343 | 6.836 | 9.392 | 100.491 | 97.006 | 98.272 | 5.498 | 6.935 | 9.512 | 100.490 | 97.006 | 98.269 | 5.785 | 7.184 | 9.721 | Y |
| 98.662 | 97.105 | 92.313 | 5.386 | 8.984 | 9.468 | 98.660 | 97.106 | 92.313 | 5.357 | 8.463 | 9.031 | 98.658 | 97.105 | 92.314 | 5.251 | 8.452 | 9.044 | Y |
| 106.155 | 89.770 | 94.869 | 15.381 | 19.564 | 19.089 | 106.128 | 89.744 | 94.847 | 15.012 | 20.293 | 19.513 | 106.112 | 89.756 | 94.851 | 15.374 | 21.281 | 20.360 | Y |
| 106.216 | 75.918 | 77.701 | 10.910 | 26.205 | 26.920 | 106.223 | 75.937 | 77.717 | 11.343 | 27.411 | 28.202 | 106.227 | 75.935 | 77.724 | 11.810 | 27.872 | 28.676 | Y |
| 96.161 | 90.937 | 93.079 | 25.053 | 22.347 | 32.751 | 96.063 | 90.876 | 92.393 | 23.280 | 21.393 | 29.249 | 96.185 | 91.061 | 92.395 | 26.795 | 25.168 | 31.492 | Y |
| 94.379 | 88.966 | 93.476 | 19.684 | 14.676 | 14.248 | 94.391 | 88.970 | 93.513 | 22.454 | 16.409 | 17.410 | 94.438 | 89.011 | 93.840 | 25.356 | 18.013 | 19.473 | Y |
| 96.663 | 94.083 | 96.208 | 8.088 | 7.194 | 8.037 | 96.663 | 94.081 | 96.208 | 7.769 | 6.821 | 7.696 | 96.669 | 94.087 | 96.215 | 7.906 | 6.736 | 7.644 | Y |
| 97.107 | 93.842 | 100.057 | 14.636 | 13.963 | 13.171 | 97.138 | 93.865 | 100.060 | 14.144 | 13.446 | 12.773 | 97.218 | 93.935 | 100.101 | 14.162 | 13.449 | 12.778 | Y |
| 97.510 | 95.441 | 91.335 | 11.581 | 12.522 | 14.136 | 97.665 | 95.598 | 91.472 | 13.446 | 14.202 | 15.500 | 97.556 | 95.494 | 91.407 | 12.675 | 13.636 | 15.127 | Y |
| 100.173 | 91.488 | 92.106 | 9.570 | 12.870 | 13.699 | 100.177 | 91.493 | 92.111 | 9.965 | 13.627 | 14.762 | 100.189 | 91.502 | 92.119 | 11.987 | 14.997 | 15.927 | Y |
| 96.591 | 93.811 | 98.923 | 16.112 | 16.003 | 16.366 | 96.588 | 93.805 | 98.922 | 15.492 | 15.319 | 15.795 | 96.592 | 93.810 | 98.926 | 17.186 | 16.911 | 17.405 | N |
| 95.878 | 92.150 | 95.077 | 20.545 | 19.663 | 21.987 | 95.881 | 92.157 | 95.101 | 21.659 | 20.651 | 23.334 | 95.965 | 92.215 | 95.190 | 22.339 | 21.204 | 24.414 | N |
| 99.350 | 91.887 | 93.184 | 16.382 | 16.625 | 16.299 | 99.338 | 91.869 | 93.154 | 17.272 | 17.816 | 17.504 | 99.342 | 91.892 | 93.185 | 19.487 | 19.919 | 19.419 | N |
| 70.401 | 71.379 | 79.783 | 47.517 | 44.823 | 40.777 | 69.001 | 70.065 | 78.885 | 42.150 | 39.107 | 34.604 | 69.318 | 70.323 | 78.984 | 44.229 | 41.095 | 36.335 | N |
| 100.186 | 92.273 | 93.798 | 8.657 | 11.761 | 12.373 | 100.187 | 92.291 | 93.814 | 8.798 | 12.937 | 13.320 | 100.198 | 92.436 | 93.947 | 11.008 | 15.728 | 16.236 | N |
| 93.760 | 89.780 | 93.265 | 15.555 | 17.965 | 16.736 | 93.795 | 89.812 | 93.292 | 17.064 | 19.390 | 18.179 | 93.874 | 89.891 | 93.356 | 18.600 | 20.745 | 19.487 | N |
| 83.844 | 83.224 | 83.750 | 61.623 | 61.631 | 60.853 | 84.924 | 84.320 | 84.799 | 65.776 | 65.751 | 64.874 | 86.142 | 85.611 | 86.006 | 68.637 | 68.578 | 67.777 | N |