IMAGE PROCESSING FOR SURFACE TEXTURE PATTERN CLASSIFICATION

Master of Science Thesis

Khamis Salim BAMAMA

Eskişehir 2020

IMAGE PROCESSING FOR SURFACE TEXTURE PATTERN CLASSIFICATION

Khamis Salim BAMAMA

Master's Thesis

Department of Electrical and Electronics Engineering

Programme in Telecommunications Engineering

Supervisor: Prof. Dr. Ömer Nezih GEREK

Eskişehir

Anadolu University

Institute of Graduate Programs

August 2020

ABSTRACT

IMAGE PROCESSING FOR SURFACE TEXTURE PATTERN CLASSIFICATION

Khamis Salim BAMAMA

Department of Electrical and Electronics Engineering

Programme in Telecommunications Engineering

Anadolu University, Institute of Graduate Programs, August 2020

Supervisor: Prof. Dr. Ömer Nezih GEREK

Conventional pattern classification systems have mostly employed binary classification methods where training features are extracted from multiple classes. This approach faces a challenge when it comes to systems which deal with imbalanced distribution of class samples, a typical characteristic of defect detection systems where there also exists a wide spectrum of possible defects. Using binary classifiers in such a scenario is bound to introduce uncertainties with respect to classifier performance as defects which had not been used in the training stage are encountered. One class classifiers have been proposed to overcome this challenge by using only normal samples to train the classifier. This thesis provides a comprehensive analysis of one class (i.e. unitary) classification to provide an empirical evaluation of the effects of defect spectrum in the feature space. Different unitary classifiers were compared to common binary classifiers and experimental results showed significant instability in the performance of the binary classifiers when classes occupy different regions in the feature space relative to the training classes. The performance of unitary classifiers was stable in all defect type scenarios.

Keywords: Unitary classification, Image texture analysis, Automated fabric defect detection, Pattern classification

ÖZET

YÜZEY DOKU ÖRÜNTÜSÜ SINIFLANDIRMA AMAÇLI GÖRÜNTÜ İŞLEME

Khamis Salim BAMAMA

Elektrik-Elektronik Mühendisliği Anabilim Dalı

Telekomünikasyon Mühendisliği Bilim Dalı

Anadolu Üniversitesi, Lisansüstü Eğitim Enstitüsü, Ağustos 2020

Danışman: Prof. Dr. Ömer Nezih GEREK

Klasik örüntü sınıflandırma sistemleri genellikle tüm sınıflardan özniteliklerin çıkarıldığı ikili sınıflandırma metodlarını kullanır. Son derece farklı arıza türlerini içeren hata tespiti gibi durumlarda bu tür ikili sınıflandırıcılar veri miktarında dengesizlik sorunu yaşarlar. Bu tür problemlerde eğer önceden rastlanmamış tür bir hata verisi ile karşılaşılırsa ikili sınıflandırıcılar belirsizlik ve hataya mahkumdur. Tek sınıflı sınıflandırıcılar, yalnızca normal sınıfa ait örneklerle eğitilerek bu sorunu aşabilmektedir. Bu tezde hatalı örnek sayısına bağlı olarak ikili ve tek sınıflı sınıflandırıcıların kapsamlı bir karşılaştırması gerçek veriler üzerinde sunularak ampirik değerlendirme yapılmaktadır. Farklı tek sınıflı yöntemler farklı ikili sınıflandırıcılar ile karşılaştırıldığında hata vektörlerinin öznitelik uzayında dağınık olması durumunda tek sınıflı sınıflandırıcının ikili sınıflandırıcıya göre çok ciddi miktarda performans avantajı elde ettiği deneysel olarak gözlenmiştir. Tek sınıflı sınıflandırıcı performansı tüm hata türleri için stabil kalmaktadır..

Anahtar Sözcükler: Tek sınıflı sınıflandırma, Görüntü doku analizi, Otomatik kumaş hatası tespiti, Örüntü sınıflandırma.

ACKNOWLEDGEMENTS

First and foremost, I wish to extend my deepest gratitude to God the Almighty for the health and strength he bestowed upon me to achieve this milestone.

I would like to sincerely thank my supervisor Prof. Dr. Ömer Nezih GEREK for the advice and leadership in the writing of this thesis, I cannot imagine how difficult it would have been to complete my work without his guidance. I would also like to thank my thesis committee members: Assoc. Prof. Dr. Tansu FİLİK and Assoc. Prof. Dr. Kemal ÖZKAN for their insightful comments and direction.

Words are not enough to express my gratitude and appreciation for my parents, Salim Mwatsaka and Nuru Juma for their unconditional love and support throughout my life, thank you for always believing in me. Special thanks to my siblings for their endless encouragement and being there for me all these years. I also extend my sincere thanks to my entire extended family and friends for having me in their thoughts and prayers throughout my graduate studies.

To everyone who was there for me and with me, I wish to appreciate your support in making this journey a successful one. Thank you and may God bless you all.

Khamis Salim BAMAMA

07/08/2020

STATEMENT OF COMPLIANCE WITH ETHICAL PRINCIPLES AND RULES

I hereby truthfully declare that this thesis is an original work prepared by me; that I have behaved in accordance with the scientific ethical principles and rules throughout the stages of preparation, data collection, analysis and presentation of my work; that I have cited the sources of all the data and information that could be obtained within the scope of this study, and included these sources in the references section; and that this study has been scanned for plagiarism with "scientific plagiarism detection program" used by Anadolu University, and that "it does not have any plagiarism" whatsoever. I also declare that, if a case contrary to my declaration is detected in my work at any time, I hereby express my consent to all the ethical and legal consequences that are involved.

Khamis Salim BAMAMA

CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

GLOSSARY OF SYMBOLS AND ABBREVIATIONS

AE	:	Auto Encoder
DTB	:	Decision Tree Binary Classifier
ICA	:	Independent component analysis
KNN	:	K-Nearest neighbor
KNNB	:	K Nearest Neighbor Binary Classifier
MoG	:	Mixture of Gaussian
MSE	:	Mean Square Error
NNB	:	Neural Network Binary Classifier
OCC	:	One Class Classifier
PCA	:	Principal Component Analysis
PDF	:	Probability Density Function
RFB	:	Random Forest Binary Classifier
SVMB	:	Support Vector Machine Binary Classifier

1. INTRODUCTION

1.1. Background and Motivation

Surface texture analysis has seen an increase in interest within the realms of pattern recognition applications. Of importance is the practical applications they have in automated defect detection systems where traditionally done by humans introduced not only bias but affected production due to fatigue and inconsistency. This has exacerbated the need for automatic detection of defects using pattern classification systems. A typical pattern classification system comprises of a preprocessing stage, feature extraction stage and a subsequent classification stage [1]. To establish patterns within surface textures a feature extraction stage extracts features that are separable enough to distinguish between different patterns. The goal of Feature extraction is to extract useful characteristics from the data. A feature is a measurement specifying some quantifiable property (i.e., color, texture, or shape) of an object like an image or sub-image. The classification stage involves assuming a model and using training patterns to learn the unknown parameters of the model. The learning approaches include; supervised, semi supervised and reinforcement Learning [1].

Conventionally, supervised binary classification methods have been the dominant candidates when it comes to classification application. These classifiers assume the presence of equally balanced data classes [2]. An important feature of the supervised binary classifiers is the need to be fed with all possible types of defects that can arise in a practical setup. Such a classifier therefore does not cope well with scenarios where samples from other classes are not available, or very few samples are available due to reasons such as difficulty of collection, high computational cost, infrequent event [3]. In surface texture defect detection applications, not only are the defect samples under sampled but they also exhibit a wide spectrum; in textile manufacture for instance, the defect spectrum ranges from holes in the fabric, cracks, thread defects, folds, oil patches etc. [4]. In the manufacturing process of wood veneer, defects include knots, branches, and cores. In manufacturing plants, defect samples could be expensive to obtain. In all these scenarios even when some defect samples are available, they might not be sufficient to generalize the whole defect spectrum in a practical setup.

1.2. Problem Statement

This study was inspired by an industrial problem from the lumber and wood processing industry. A furniture manufacturing firm in Turkey was seeking to automate its quality inspection operation to detect anomalies on processed wood surfaces. A challenging aspect of the problem was the fact that samples of these so-called anomalies on the surfaces of the wood were not sufficient to capture all the possible types of defects that may arise during the manufacturing process. To address the problem of imbalanced samples and wide spectrum of defects in such defect detection applications, we proposed a semi supervised based unitary classification approach for surface texture defect detection where the training uses examples from only the normal non-defective samples a phenomenon referred to in literature as one-class classification, outlier detection, novelty detection, or concept learning [2]. The study by Alam et al. [5] which provides state of the art for researchers in the space of unitary classifiers mentions in addition to the data set class imbalance problem, issues such as class distribution skew, absent features and small disjunction which affects performance of binary classifiers. Alam et al. also observed that there is a relative insufficiency of deep learning methods for anomaly detection [5]. We adopted the Tilda textile data set since it modeled our problem well due to the wide spectrum of defects available.

1.3. Contribution

In this thesis we seek to evaluate empirically, the effect of outlier class spectrum (location on the feature space) on the performance of binary classifiers against the performance of unitary classifiers in surface texture defect detection applications. The assumption here is that defects which have the different textural characteristics are found in different areas on the feature space.

1.4. Thesis Roadmap

The rest of the thesis outline includes a related work chapter 2 where past work that has been done in the field of texture analysis is investigated from a pattern classification perspective. Different stages and processes that occur in the classification process will be highlighted. Chapter 3 contains a comprehensive review of unitary classification and provides a comparison between unitary and binary classification. In chapter 4 the system design will be explained in detail including all the building blocks

of the classification process. Chapter 5 will highlight the process of evaluating the performance of the unitary classifiers as they are compared to binary classifiers. Chapter 6 gives inferences from the results obtained and concludes the study by summarizing the main points of focus in the study. Recommendation for future work will be provided in chapter 7. Lastly the cited works will be enumerated.

2. RELATED WORK

2.1. Introduction

Surface texture classification has seen significant interest for the potential that it has shown in the automation of texture inspection in fields such as textile manufacture where inspection by humans which has been the norm is prone to error [6]. In their book, Duda et al. [1] break down the classification problem into pre-processing, feature extraction and classification stages. The pre-processing stage is crucial in preparing the sensed signal in this case an image for the feature extraction stage. The feature extraction then extracts relevant features which perform best models the class labels being investigated. The output of the feature extraction stage is a feature extracted image or a feature vector which then becomes the input to the classification stage. This section presents a breakdown of the different methods used in texture analysis.

In the study of [7] the method of classification used only normal samples by extracting independent components from the training samples and according to the Euclidean distance from the mean feature vector of a test sample (of the independent components) a sample was classified. The main idea was that defective samples would deviate more from the mean feature vector since they have a different structure that is not included in the learning phase's independent component filters' extraction. The classification process was a thresholding approach where the threshold was set automatically using the spread of the incoming test data sample.

Due to the lack of example outliers, estimating the error of the second kind becomes a challenge [2]. In the support vector data description proposed by [2], the volume of the description (a boundary of a hyperspace) is used to minimize the chance of accepting outliers.

This study proposes to tackle this problem from both the classifier side and feature extraction. Improving the quality of features is one way to improve the performance of any classifier. However, this usually comes with increasing computational cost. In texture analysis specifically, the choice of a feature extraction technique is of critical importance. The more computationally expensive methods provide the best performance in terms of reducing the type two errors.

2.2. Texture Analysis Overview

Before embarking on any algorithm development for a texture analysis based application, two important factors need to be well defined: the underlying characteristics of the texture and the type of the defects [7]. According to Sezer et.al this is because texture analysis methods perform differently with respect to the type of texture and defect, thus a method that performs well in one scenario can fail in another.

Texture defects can be categorized into three classes: intensity defects, geometrical defects, and a combination of the two. Intensity defects exhibit a noticeable change in gray level intensity while geometrical defects change the spatial correlation between pixels. When defects exhibit a wide spectrum, higher-order statistics methods are preferred to meet the performance required than simple thresholding techniques [7].

2.3. Preprocessing

Preprocessing is considered as the process of preparing the image for the stages that are to follow and is often used in classification applications [8]. However, restraint must be applied not to degrade the image by removing important image features during the preprocessing stage. This is especially true in applications such as medical imaging applications and classification of highly texturized images. Examples of preprocessing techniques include low pass filtering, morphological operations. An important part of preprocessing is segmenting the image to isolate the areas of interest to compute the features from.

2.4. Feature Extraction

This is the process of extracting features from an image which can then be used to classify the image. The features are required in a way that they can be separable. The features extracted at this stage are then put in a feature vector containing one or more features. The goal is to obtain a value that is representative of the feature being computed. These features then undergo suitability test to determine their ability to classify images. A feature could be as simple as a pixel value or an area of a segmented section.

Texture feature extraction experiences two important issues, firstly there is not a single method that is agreed upon as being the best since the texture manifestation in nature is too complex to characterize, performance of existing methods is therefore

dependent on the texture content and the type of processing. To achieve better results different methods are combined [9]. It is also the case that one method cannot always capture all the texture features, rather it has been shown that combining different methods improves the classification performance, how the different methods are combined then is an important consideration [10].

Image processing techniques used in texture feature extraction include the scale-invariant feature transform [11], speeded up robust feature [12], histogram of oriented gradients [13], local binary patterns (LBPs) [14], Gabor filters [15], Gray-level co-occurrence matrix (GLCM) [16]., gray-level difference matrix (GLDM), gray-level run-length matrix (GLRLM) [17] and others.

Gabor filters work best when defects take a vertical or horizontal orientation with respect to the background pattern. A challenge arises when the defects take similar orientation as the background pattern such as those occurring in marble surfaces. In this case the extractor should be able to tackle the problem of isolating background features from defect features while both exhibiting some level of randomness. Local Binary Patterns (LBP) allows detection of circular patterns and can be used to detect defects which take a circular pattern [18].

Different images require different features leading to the possibility of having more than one feature extraction technique in a single application as earlier stated. Where Gabor filter are good in one image classification application it could perform poorly in another. It is therefore important to understand the model of the pattern under review.

Features can be categorized into local and global features. Some feature extraction techniques such as local binary patterns make use of the local features but in so doing lose out on the contribution of global features. An approach which combines both techniques was proposed by [19].

Features can also be low level such as color, texture, and shape. Mid and High-level features include auto encoders, LBP, edge maps, object banks. Below is an extensive review of selected texture feature extraction techniques.

2.4.1. Haralick Features (GLCM)

Gray level co-occurrence matrix (GLCM) proposed by Haralick [16] is one of the most superior texture feature extraction methods. It represents the probability of finding two pixels within a neighborhood characterized by a distance d at an orientation θ;the most common values for d = 1 and θ = [0 degree;45 degree; 90 degree; 135 degree] but they are not limited to these values only [17].

Figure 2.1 shows an example of how the GLCM is calculated for a distance of 1 and 0° for a 3-by-3 image with 5 level pixel values [16].

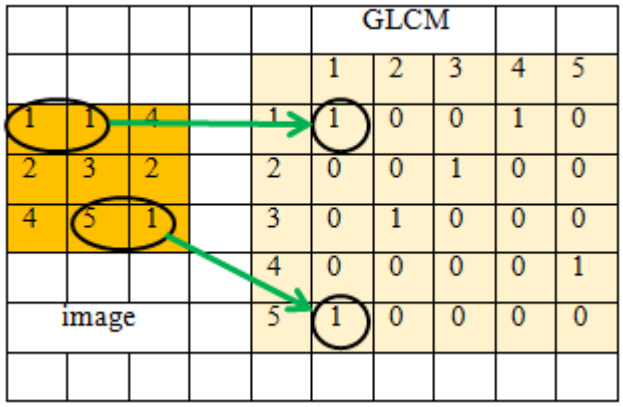


Figure 2.1. *GLCM calculation*

From the GLCM, a set of 13 measures of textural features are extracted. The six most commonly used features include angular second moment (ASM), contrast, correlation, homogeneity, entropy and dissimilarity [16].

$$Asm = \sum_{i,j} \{p(i,j)\}^2 \tag{2.1}$$

$$Con = \sum_{i,j} (i-j)^2 \, p(i,j) \tag{2.2}$$

$$Cor = \sum_{i,j} \frac{(i-u_i)(j-u_j)}{\sigma_i \sigma_j} \, p(i,j) \tag{2.3}$$

$$Hom = \sum_{i,j} \frac{p(i,j)}{1+|i-j|} \tag{2.4}$$

$$Ent = -\sum_{i,j} p(i,j)\log\big(p(i,j)\big) \tag{2.5}$$

$$Dis = \sum_{i,j} |i-j|\, p(i,j) \tag{2.6}$$

2.4.2. Local Binary Patterns (LBP)

Local Binary Pattern (LBP) provides an effective way to extract textural features. Introduced by [14] LBP boasts computational simplicity, gray scale and rotation invariant as its most useful characteristics. LBP has been successfully found application in texture classification, face recognition, target tracking and others.

For a center pixel *j* the LBP is calculated as follows

$$\boldsymbol{LBP_{P,R}(j) = \sum_{p=0}^{P-1} f(i_p - i_j)2^P} \tag{2.7}$$

Where *P* represents the neighborhood pixels around center pixel *j* and controls the quantization of the angular space while parameter *R* controls the spatial resolution [20], i_p, i_j are gray level values of the neighboring pixel and the center pixel respectively.

The function $f(x)$ is defined as

$$f(x) = \begin{cases} 1\ if\ x \geq 0 \\ 0\ if\ < 0 \end{cases} \tag{2.8}$$

LBP extracts local features from images using the neighborhood of a pixel expressed as P number of neighbors within a radius of R, the value of a central pixel and its neighboring pixels is then compared and a binary description of the neighboring pixel is made according to this comparison, if the neighboring pixel has an equal or higher gray level value than the center pixel, the binary descriptor is set to one, otherwise to zero [9]. This results in a P-bit binary number which can be uniform or non-uniform. A uniform binary number is where there are utmost two transitions within the binary number; for instance, 00011100 (two transitions) is a uniform pattern while 01110100 (four transitions) is not. From these binary codes the histogram of the operator can be computed and texture features extracted [21].

2.5. Window Selection

Apart from the specific methods used in textural feature extraction, window selection plays a critical role in the feature extraction process as it can influence both the accuracy and computation cost two parameters are of importance when it comes to window design, step size which specifies the nature of overlap to be used and the window size. When an overlapping window is used, defects have a less chance of being missed. The downside of using an overlapping window is the high computation cost. A larger window can detect texture features better but has poor performance in defect location compared to a smaller window [7].

According to [22] the choice of sub-window size depends on two factors: (i) the level of localization of the defects; and (ii) how well can the data contained in a window of such size represent the texture. Sezer et al. further highlighted two guidelines that need to be considered when determining the optimal window size: when the window size is too small, the surface texture cannot be modeled accurately, while a very large window size may lead to the defect not significantly affecting the feature vector. [7].

In [23] Using a block size 64x64 of pixels was found to locate defects more precisely than using a 128x128 block size. This was attributed to the fact that smaller blocks allow for more detailed delineation of defects and, therefore, for more accurate detections.

In [9] the authors proposed the use of variable window sizes with multiple scales to extract features by the GLCM method. Previous work on the effect of window size and shape found texture characterization to be more influenced by the window size than by its shape [24]. Previously, researchers have focused more on finding optimal window sizes by evaluating the texture methods by multiple window sizes and selecting the best performing size.

2.6. Dimensionality Reduction

A point of concern for any classification problem is the curse of dimensionality.

Usually the computational cost increases exponentially as the dimensionality increases and in effect requiring a large amount of data to get a reliable analysis, this phenomenon is referred to as the ‘curse of dimensionality’ [25]. To solve this issue, the number of features must be reduced. There are two main dimensionality reduction

techniques [26] [27]. A classical rule of thumb suggests that the number of training samples per class should be at least 5-10 times the dimensionality [28]. Figure 2.2 gives an overview of the dimensionality reduction methods [27].

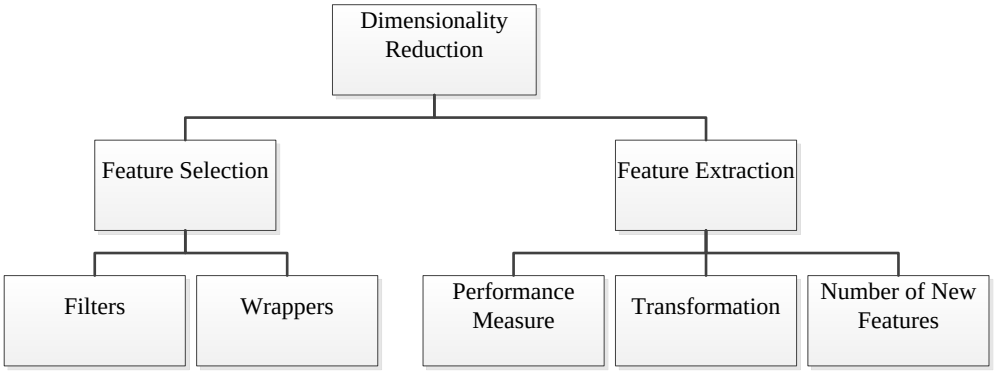


Figure 2.2. *Dimensionality reduction methods*

2.6.1. Feature Selection Dimensionality Reduction

In this technique, the features remain in the same domain and only a specified number of features are selected.

Hybrid methods also exist having recently been developed to benefit from the advantages of both filters and wrappers approaches [27].

2.6.2. Feature Extraction Dimensionality Reduction

In feature extraction the dataset is transformed to another domain to generate features that are more significant. The most common feature extraction approach is Principle Component Analysis (PCA) which is a linear transformation of data that minimizes the redundancy (measured through covariance) and maximizes the information (measured through the variance). Generally wrappers methods have higher classification accuracy as compared to feature extraction methods but are more computationally expensive than the feature extraction methods [27].

2.7. Data Splitting

Different data splitting methods exists in the literature including; cross-validation (CV), randomly selecting training samples and using the remaining for validation (methods such as bootstrap fall under this category) and systematically selecting the most representative samples based on the distribution of the data and leaving the rest for validation. [29]. Figure 2.3 shows a typical classifier data splitting process.

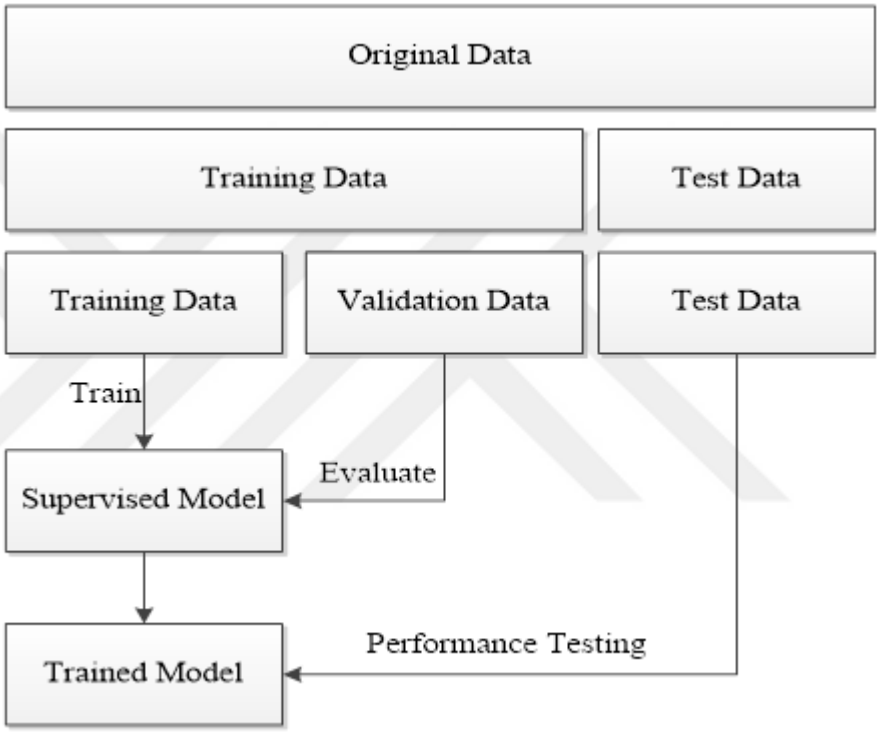


Figure 2.3. *Data splitting*

2.8. Classification

Machine learning techniques for classification applications can be done either using supervised or unsupervised learning. In supervised learning the features extracted from the feature extraction stage are fed as the inputs to the classifier whose labels are already known, techniques include neural networks, k-nearest neighbor and support vector machine [18]. In unsupervised learning the labels are not known, classifiers include k-means and self-organizing maps [15].

Figure 2.4 shows a typical pattern recognition system with all the components mentioned in the preceding sections.

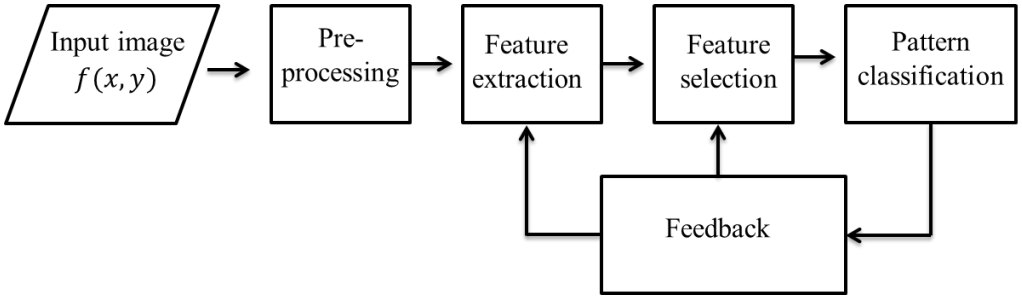


Figure 2.4. *Pattern recognition system*

3. UNITARY CLASSIFICATION

Unitary classifiers are a category of semi supervised classifiers where the learning uses only samples from one class. There exists three broad categories of unitary classifiers viz., (i) density-based classifiers (ii) boundary based classifiers (iii) reconstruction-based classifiers [3].

The density-based classifiers approach the classification problem by estimating a probability density function from a set of given data which form the training set as well, the classification is obtained by applying thresholding. This method generally requires many training samples and performs very well when a good density estimation model is used. The boundary based classifiers on the other hand try to obtain an optimal boundary around the target class, this method needs less number of training samples compared to the density estimation methods however they are dependent on the separation between objects and tend to be sensitive to the relative distance between features [3].

In the following section important features of unitary classifiers are explained followed by a survey on the three categories of unitary classifiers' methods including their application to classification.

3.1. Density Estimation

For most algorithms to work properly in signal processing applications a probability density is needed. Since in reality these probability densities are not available an estimation is needed, a density based classifier therefore works on the basis of probability density estimation [30].

A probability density function $f(y)$ of a p dimensional data y is a smooth continuous function satisfying the positivity shown in equation 3.1[30]

$$f(y) \geq 0, \qquad \int_{R^p} f(y)\, dy = 1 \tag{3.1}$$

Starting with a set of p −dimensional observed data $\{y_n,\ n = 1, \dots, N\}$, the goal of density estimation is to find an estimated function $\hat{f}$ which "best" approximates the true probability density function f [30].

There exist two types of density estimation viz., parametric, and non-parametric. Parametric techniques tend to make an assumption that the data are drawn from one of the known parametric family of distributions [31], for instance given a parametric density family $f(\cdot|\theta)$, such as the two parameter normal distribution family $N(\mu, \sigma^2)$ where $\theta = (\mu, \sigma^2)$, the emphasis is on obtaining the best estimator $\hat{\theta}$ of θ [32]. Non-parametric techniques on the other hand make less rigid assumptions about the distribution [31], the emphasis is directly on obtaining a good estimate $\hat{f}(\cdot)$ of the entire density function $f(\cdot)$ [32]. The most used density estimation techniques include the kernel density estimator and the nearest neighbor density estimator [33].

Kernel Density Estimation

$$\hat{f}(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i) \tag{3.2}$$

Where $K_h(x) = K(x/h)/h$

Nearest Neighbor Density Estimation

The nearest neighbor method is widely used in the fields of pattern recognition and nonparametric discriminant analysis [31].

For each sample x the distances are defined as

$$d_1(x) \leq d_2(x) \leq \cdots \leq d_n(x) \tag{3.3}$$

to be the distances, arranged in ascending order, from x to the points of the sample.

The kth nearest neighbor density estimate is then defined by

$$\hat{f}(x) = \frac{k}{2nd_k(x)} \tag{3.4}$$

The kth nearest neighbor estimate is defined by

$$\hat{f}(x) = \frac{1}{nd_k(x)}\sum_{i=1}^{n} K\left(\frac{t - X_i}{d_k(x)}\right) \tag{3.5}$$

Where $\hat{f}(x)$ is the kernel estimate evaluated at x with window width $d_k(x)$. The choice of integer k governs the smoothing.

For the nearest neighbor density estimator in d dimensional space, let $r_k(x)$ be the Euclidean distance from x to the kth nearest data point, and let $V_k(x)$ be the (d -dimensional) volume of the d -dimensional sphere of radius $r_k(x)$; thus $V_k(x) = c_d r_k(x)^{\,d}$, where c_d is the volume of the unit sphere in d dimensions (so that $c_1 = 2$, $c_1 = \pi$, $c_3 = 4\pi/3$, etc.).

The nearest neighbor density estimate is then defined by

$$\hat{f}(x) = \frac{k/n}{V_k(x)} = \frac{k/n}{c_d r_k(x)^{\,d}} \tag{3.6}$$

In principle the nearest neighbor estimator is just a near neighbor estimator and gives the probability that a new input x will fall within (X1 and X2) using the specified k th neighbor.

The following section highlights examples of density based unitary classifiers.

3.1.1. Gaussian Model

The probability distribution for a d-dimensional object x is given by:

$$\hat{f}(x;\, \mu, \sum) = \frac{1}{(2\pi)^{d/2}|\sum|^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu)^T \sum^{-1}(x-\mu)\right\} \tag{3.7}$$

The classifier is defined as

$$h(x) = \begin{cases} target & if \quad \hat{f}(x) \leq \theta \\ outlier & if \quad \hat{f}(x) > \theta \end{cases} \tag{3.8}$$

Where μ is the mean and $\sum$ is the covariance matrix and represent sample estimates [34]. The method is very simple and it imposes a strict uni-modal and convex density model on the data [2].

3.1.2. Gaussian Mixture Models

The target class is modeled as / the unconditional probability density is estimated as the linear combinations of K Gaussian kernels [35].

$$\hat{f}(x) = \sum_{i=1}^{K} P_i \, p(x|i) \tag{3.9}$$

Where $p(x|i)$ is given by:

$$\frac{1}{(2\pi)^{d/2}\left|\sum_i^{-1}\right|^{1/2}}\ exp\left(-\frac{1}{2}(x-\mu_i)^T\sum_i^{-1}(x-\mu_i)\right) \quad (3.10)$$

P_i is the prior for the ith kernel, d is the dimension of the feature vector, μ_i is the mean vector (or center) for the ith kernel and $\sum_i$ its covariance matrix. These parameters are optimized by the expectation-maximization (EM) algorithm [34].

While the classifier is represented as

$$h(x) = \begin{cases} target & if \quad \hat{f}(x) \geq \theta \\ outlier & if \quad \hat{f}(x) < \theta \end{cases} \quad (3.11)$$

3.1.3. Parzen Density Estimator

In Parzen density estimation, the estimated density is mostly a mixture of a Gaussian kernel around each of the training objects. One of its characteristics is the large number of training samples required [3].

$$\hat{f}(x) = \sum_{i=1}^{K} exp\left(-(x-x_i)^T h^{-2}(x-x_i)\right) \quad (3.12)$$

Training a Parzen density involves the specification of the optimal kernel width h [34]. The free parameter h is optimized by maximizing the likelihood on the training data using leave one-out. The classifier becomes like equation 3. This method often performs well but it requires a reasonable training set. The method requires fewer assumptions than the Gaussian mixture model [35].

3.2. Boundary Methods

Unlike the density estimation methods, boundary methods fit an optimized boundary around a data set without estimating the whole density making the boundary methods more appropriate when there are few instances of the target data set available [2]. A major drawback to the boundary methods is that the distance between objects should be adequately defined [2]. Some of the boundary methods are highlighted in this section.

3.2.1. K-centers

Given a dataset containing the target objects, the k-center method puts ball with equal radii across the data. The method then finds the minimum distance between a target object x_i and the k centers. This results into i minimum distances. The algorithm then minimizes the maximum distance among the i minimum distances [2].

$$\varepsilon_{k-center} = \max_i \left(\min_k \|x_i - \mu_k\|^2 \right) \tag{3.13}$$

3.2.2. Nearest Neighbor Method

The method is based on local densities of (1) the test object and (2) the first nearest neighbor [2]. The approach involves fitting a cell around a test object and growing it till it captures k objects from the target set. The local density of the test object is given as

$$p_{NN}(z) = \frac{k/N}{V_k\left(\left\|z - NN_k^{tr}(z)\right\|\right)} \tag{3.14}$$

Where $NN_k^{tr}(z)$ is the k nearest neighbor of z in the training set and V_k is the volume of the cell containing the object.

A test object z is accepted when its local density is larger or equal to the local density of its first neighbor [2].

3.2.3. Support Vector Data Description

This method is based on the support vector classifier and entails obtaining a boundary directly around the target data set; a task which is achieved by fitting a hyper sphere around the target set and reducing the volume of the hyper sphere to achieve the required target rejection rate [2]. The sphere is defined by its center a and radius R. The structural error is computed in such a way that the hyper sphere includes all the target data set.

$$\varepsilon_{struct}(R, a) = R^2 \tag{3.15}$$

This structural error is minimized by the constraint

$$\|x_i - a\|^2 \leq R^2, \forall i \tag{3.16}$$

Having all the target objects however does not make the method robust and a portion of the target data set must be rejected implying that the empirical error does not have to be zero. The error is now characterized by a both a structural and an empirical error contribution. The minimization problem becomes

$$\varepsilon(R, a, \boldsymbol{\varepsilon}) = R^2 + C \sum_i \varepsilon_i \tag{3.17}$$

The new error is now minimized by the constraint below which requires that almost all objects are within the sphere but not all as was the case with the zero rejection rate of the target class.

$$\|x_i - a\|^2 \leq R^2 + \varepsilon_i, \quad \varepsilon_i \geq 0 \quad \forall i \tag{3.18}$$

Parameter C gives the tradeoff between the volume of the classifier and the errors. The free parameters that need to be optimized are R, a, ε [2].

3.3. Reconstruction Methods

The reconstruction methods approach to unitary classification are inspired from data modeling methods [2]. The common theme among these methods is that given a set of data in this case the target data set, the reconstruction methods transforms the data to another domain of representation. This introduces a reconstruction error between the original target object and the transformed target object.

3.3.1. K-means

The k-means approach assumes that the data is clustered. Cluster points are placed within the data set and the minimum distance from a data point x_i to its nearest cluster point k is computed for all i data points. The average of the minimum distances is minimized [2].

$$\varepsilon_{k-means} = \sum_i \left(\min_k \|x_i - \mu_k\|^2 \right) \tag{3.19}$$

For a test object z the distance to the target set is

$$d_{k-means}(z) = \min_k \|z - \mu_k\|^2 \quad (3.20)$$

3.3.2. Self-Organizing Map (SOM)

The SOM approach to unitary classification is inspired from the conventional unsupervised clustering functionality of the SOM using artificial neural networks. Neurons are introduced in the feature space and the system is trained until convergence. There are now minimum distances of each target object to a cluster point just like in the previous technique of k-means. For a test object z the distance to the target set is

$$d_{som}(z) = \min_k \|z - \mu_k\|^2 \quad (3.21)$$

3.3.3. Principal Component Analysis

This unitary method of classification uses the transformation functionality of the PCA to transform a target object from the initial domain of the feature space to a new domain using by computing the Eigen values from the target data covariance matrix, the Eigen vectors thus formed with the largest Eigen values form the principal components. The reconstruction error for a test object is calculated as the error in the transformation of a test object from the original feature space domain to the new domain [2].

$$d_{PCA}\ (z) = \|z - (W(W^TW)^{-1}W^T)z\|^2 \quad = \|z - (WW^T)z\|^2 \quad (3.22)$$

3.3.4. Auto-Encoders

These are neural networks with a single hidden layer with a large number of hidden units h_{auto} which learn the representation of the data [2]. The data is input at the input layer and an approximation of the input is obtained at the output layer. The associated error is minimized during training.

$$d_{auto}(z) = \|f_{auto}(z{:}\,W) - z\|^2 \quad (3.23)$$

It is assumed that the target objects will have a smaller reconstruction error than the outlier objects.

3.4. Unitary Versus Binary Classification

The study conducted by Bellinger et al. [36] highlighted the circumstances under which unitary classifier could be preferred to the binary classifier. They observed that the binary classifiers perform better when there is a relative balance between class samples [36]. When there is a high level of imbalance a common situation in defect detection systems the performance of the binary classifiers was found to drop as the imbalance grows.

In their study Bellinger et al. [36] illustrated how an imbalance in one of the classes creates a bias in binary classifiers towards the majority class since they use prior probabilities. In unitary classification the prior probabilities are ignored since there is only one class, the point of interest then becomes estimating a PDF of the target class and a new sample will be classified based on a threshold set on the probability. This was illustrated in equation 2.19. The fact that only one class is used in estimating the PDF of the model eliminates any bias towards a certain class, this property makes the unitary classifier a better classifier than the binary classifier in cases of imbalanced data. An empirical verification of this analysis was done by a combination of different classifiers on various dataset and establishing the point where the binary classifiers fail for a specific problem.

There is no blanket definition of imbalance where binary classifiers start to fail as this is dependent on the dataset and the classifier. Some binary classifiers can still perform well when there is a high level of imbalance even to levels above the occ. Therefore the choice of which classifier to use when is greatly dependent on the problem at hand [36].

In another study the choice of unitary based classifier in textile defect detection application by [7] was attributed albeit theoretically, to the fact that binary classifiers' performance cannot be guaranteed when the defect class shows a wide unpredictable spectrum of possible defects [7].

4. SYSTEM DESIGN

4.1. Data Set

The Tilda dataset for textile textures [4] was used which comprises of 8 textures with each texture having 7 defect types with one reference class which is free of defect. For texture analysis we excluded the figurative textiles as was done in [7], the choice of the data set was appropriate since textile fabric images have complex textures and the defects occupy a wide spectrum (performance of an algorithm trained by one defect type cannot be guaranteed to detect other defects). In this study one texture pattern from the C1-R1 Tilda dataset with five different textural defect types was selected. The defect types include defect-free (E0) Figure 4.1, holes (E1) Figure 4.2, oil spots (E2) Figure 4.3, thread errors (E3) Figure 4.4, objects on the surface (E4) Figure 4.5 and camera defects (E7) Figure 4.6. All feature sets extracted by these methods are normalized to 0 to 1 along each dimension. The experiments are conducted in Matlab R2015b, running on a Toshiba Satellite C850-B699 Laptop with an Intel Pentium Dual CPU B960 at 2.20 GHz and 2 GB memory.

Figure 4.1. *Non-defective image*

Figure 4.2. *Medium-sized hole in the lower half of the picture*

Figure 4.3. *Small spot in the lower right corner*

Figure 4.4. *Thread defects in the center of the picture*

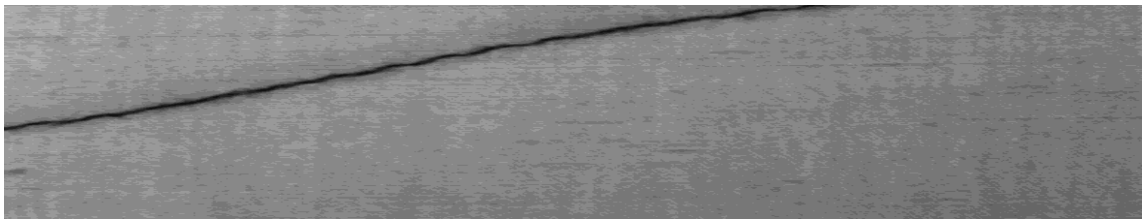


Figure 4.5. *Dark thread on the fabric in the upper left corner of the picture*

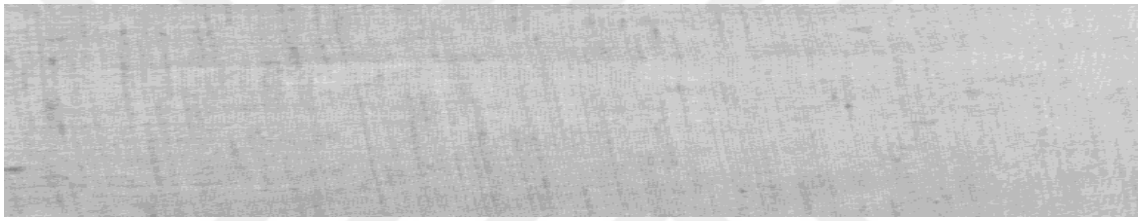


Figure 4.6. *Camera error*

4.2. Image Pre-processing

This section puts into context the nature of the defects from the Tilda set in an effort towards characterizing the datasets defects. We used the ratio of the defect area to the normal non-defective area of the 768x512 sized Tilda images as our metric for the characterization.

At this stage we sought to find the appropriate preprocessing mechanisms without losing or distorting information from the input images. The first step was to segment the 768x512 images into sub-windows or patches. This step did not include any transformation on the images rather preparing the images for the feature extraction stage. the choice of the appropriate sub-window size was adopted from the method by Sezer et al. [7] where they observed that when many sub-windows are randomly selected the statistical information about the texture becomes rotation and translation invariant. They calculated the optimal sub-window size by picking random sub-windows from the normal image and picking another sub-window from a rotated and translated versions and checking the statistical difference using the mean square error (MSE) between the two sets. The mean of these MSE's from one sub-window was

taken and repeated for other sub-window sizes. The scale invariance was not considered since the normal class (e0) and the defective classes (e1, e2, e3, e4) of the Tilda dataset are obtained from a fixed camera. For the resized Tilda images of size 384x256 they found the optimal sub-window size to maintain the statistical information of the texture in the images to be 32x32. Extrapolating this finding to our case of 768x512 we adopted a 64x64 window size.

We used a sliding overlapping window approach as opposed to a non-overlapping window to obtain as many samples from one image as possible again adopted from the method in [7]. For the purposes of generating many samples from a single 768x512 image, we used a 128x128 overlapping sub-window with a step size of 64 to generate 1925 patches from 25 images with the other 25 kept as the testing set (each defect type has 50 images). Through visual inspection we removed parts which were clearly showing defects despite the fact that they came from non-defective (e0) images as shown in Figure 4.7, this is done because outliers in training data will degrade the performance of the classifier [5]. This method of using a human operator to label patches as defective and non-defective was also used in the study of L Bissi et al. [37].

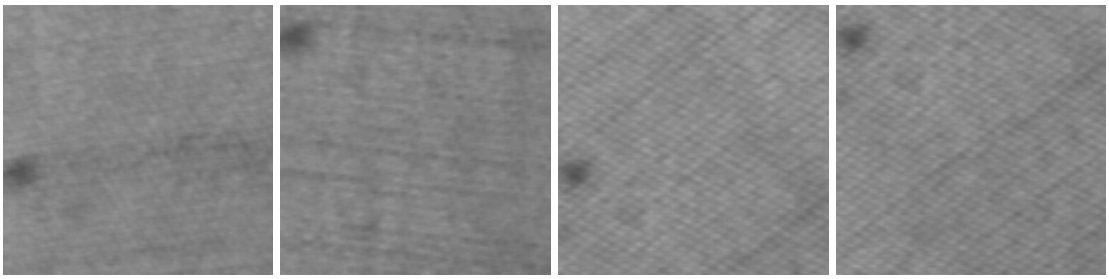


Figure 4.7. *Defective patches from e0 (defect free) images*

A total of 1875 cleaned images of 128x128 size were left which were then subdivided into 64x64 sized windows by a 64x64 overlapping window of 32 step size yielding a total of 16875 of 64x64 pixel size.

For the training stage outlier samples, we isolated the defective patches from 25 (e1, e2, e3, e4, e7) images through visual inspection and labeled them into class defects based on visual variation, since one defect type could have more than one visually unique defect. We extended the initial five defect classes into 10 defect classes as shown in Figure 4.8 crack defects (a), hole defects (b), oil patch defects (c), thread

defect (d), ball thread defect (e), dark thread defect (f), light threads (g), twisted thread defect (h), white thread (i), camera error defect (j).

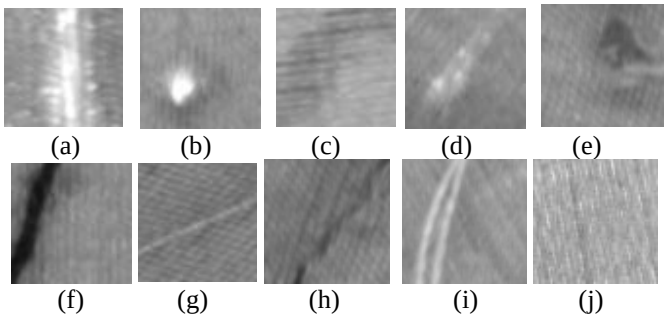


Figure 4.8. *Sample patches showing different defect types*

4.3. Data Splitting

Using the ovelapping window approach, the least number of defective patches from the 10 defect types was 40 making a total of 400 defective images (outliers). The number of defective images had to be the same to eliminate the effect of imbalance data set during training. With 400 defective images at hand the study by Bellinger et al. [36] established that exceeding a ratio of 1:2.5 for outlier to target samples affects the performance of binary classifiers due to imblance data set. We further computed the average ratio of the datasets which conform to the 1:2.5 ratio and obtained a ratio of 1:1.5. Using this ratio, 600 randomly selected images (400x1.5) were taken as the targets for training set with another 600 as the targets for the validation set.

The test set was taken from the remaining 25 768x512 images from (e0, e1, e2, e3, e4). The e0 was subdivided into 600 128x128 patches. The defective images (e1, e2, e3, e4) were also sub-dived into 128x128 sized patches but only the patches with defects used (285 patches). Table 4.1 shows the result of the data splitting process.

Table 4.1. *Data splitting results*

Set	Normal	Defective	Size
Training	600	0	64x64
Validation	600	400	64x64
Test	600	285	128x128

4.4. Feature Extraction

The algorithm we are proposing for the feature extraction had the following parameters to be determined for the two feature extraction schemes:

(i) Number of levels for the GLCM

(ii) LBP neighborhood

We extracted the training features from the non-defective images since in unitary classifier design only one type of class is available. For the GLCM parameters we used the most used parameters from literature viz., distance of 1 and a degree of 0, we further resorted to finding the appropriate dimension for the GLCM through trial and error. The dimension represents the number of gray scale levels for the GLCM feature extraction and the larger this value the more textural features are extracted at the cost of increasing computation complexity, a balance therefore must be reached at. On the LBP neighborhood we adopted the default Matlab neighborhood number of 8 and radius 1.

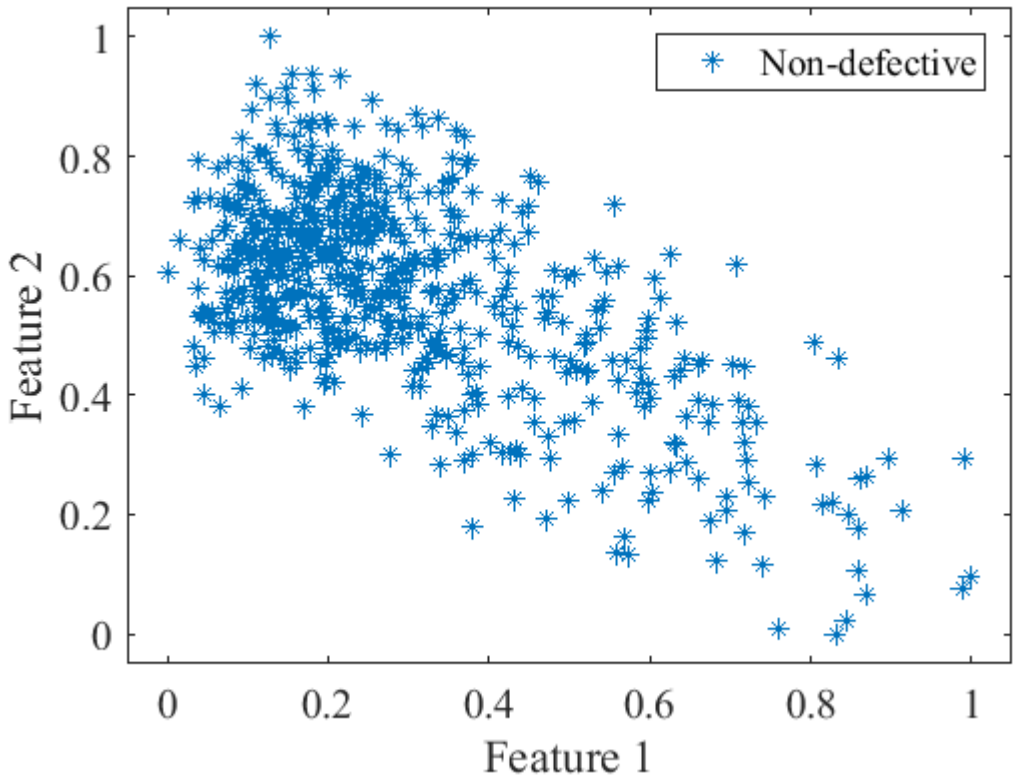


Figure 4.9. *Non defective features for training*

Figure 4.9 is the 2D representation of the extracted features from the normal non-defective samples.

4.5. Data Normalization

The data normalization is important when combining two different families of features to transform the individual features into a common domain to avoid comparing "apples and oranges". We assumed that the validation set is a true representation of the entire feature space. The max and min scheme was selected due to its robustness to outliers.

$$\bar{x} = \frac{x - \min(F_x)}{\max(F_x) - \min(F_x)} \tag{4.1}$$

Where x is a non-normalized feature value, $\bar{x}$ is the normalized value for x, F_x represents the function that generates x, and $\min(F_x)$ and $\max(F_x)$ denote the minimum and maximum values respectively for all possible values of x.

4.6. Feature Selection

In this stage we used the dimensionality reduction methods in [34]. We also utilized the validation data set since the training set contains only samples from one class and the defective classes are needed in the wrapper feature selection method from PRTools 5.0 a Matlab toolbox for pattern recognition [38] . Figure 4.10 and Figure 4.11 below shows the feature quality of the normalized data.

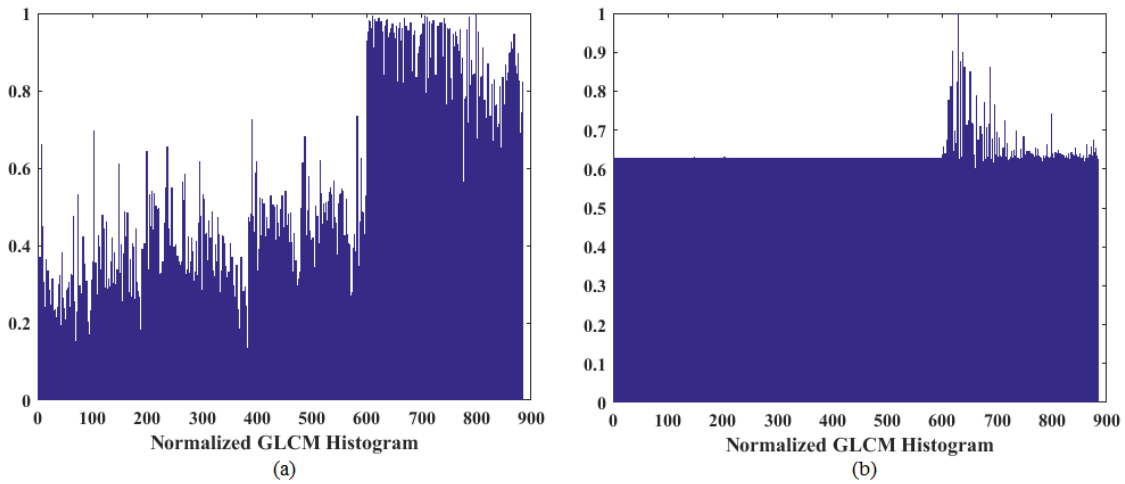


Figure 4.10. *GLCM features (a) correlation and (b) cluster shade components*

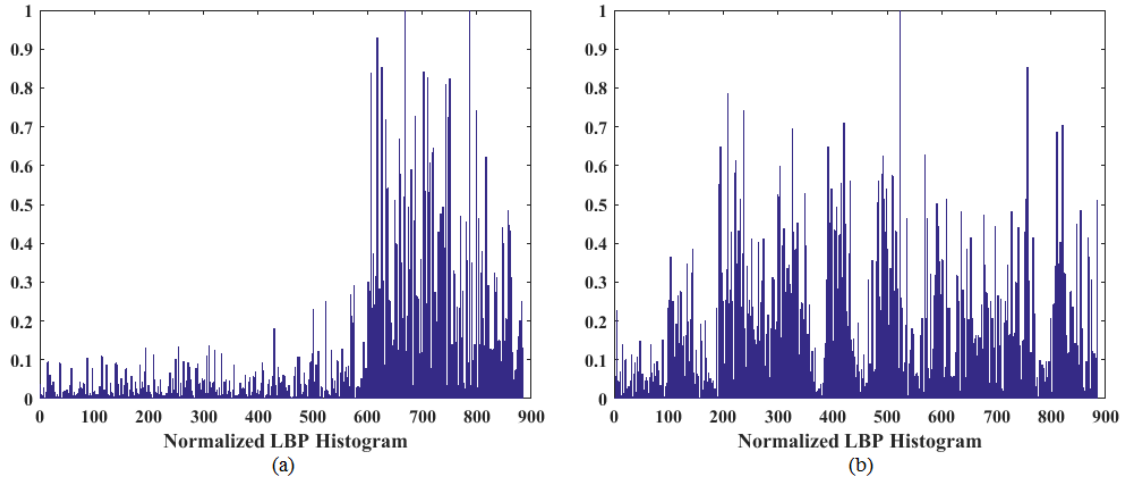


Figure 4.11. *LBP square error (a) ninth and (b) sixth components*

The first row represents the GLCM features while the second row is the LBP features. The first column shows the best performing features from each feature type while the second column is a representation of the worst performing features. There was a total of 23 features from both the GLCM and the LBP with 13 being from GLCM out of which 10 unique and most representative features were selected.

The figures 4.12 – 4.16 below show the comparison between different non-defective and the defective features. Different defects occupy different regions in the feature space.

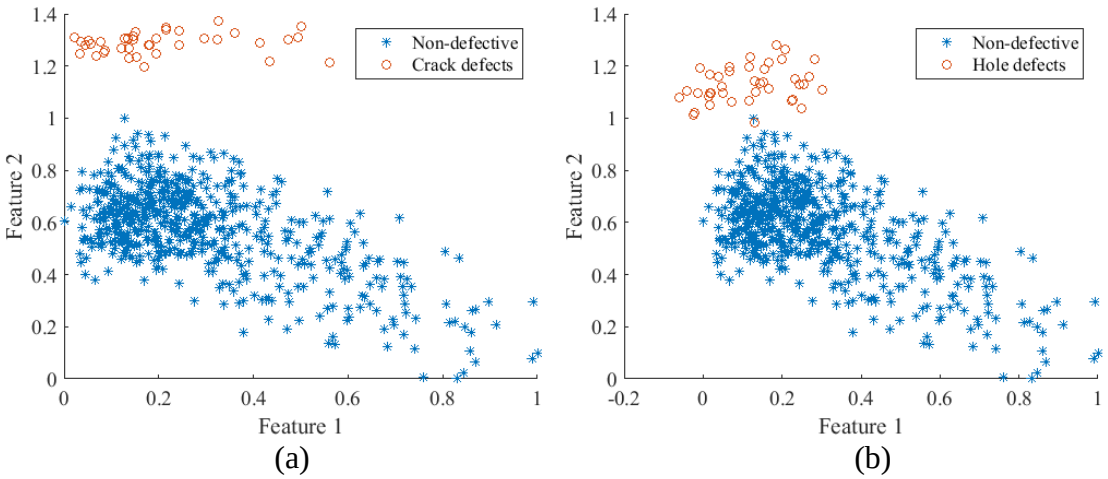


Figure 4.12. *Defect comparison (a) crack (b) hole defects*

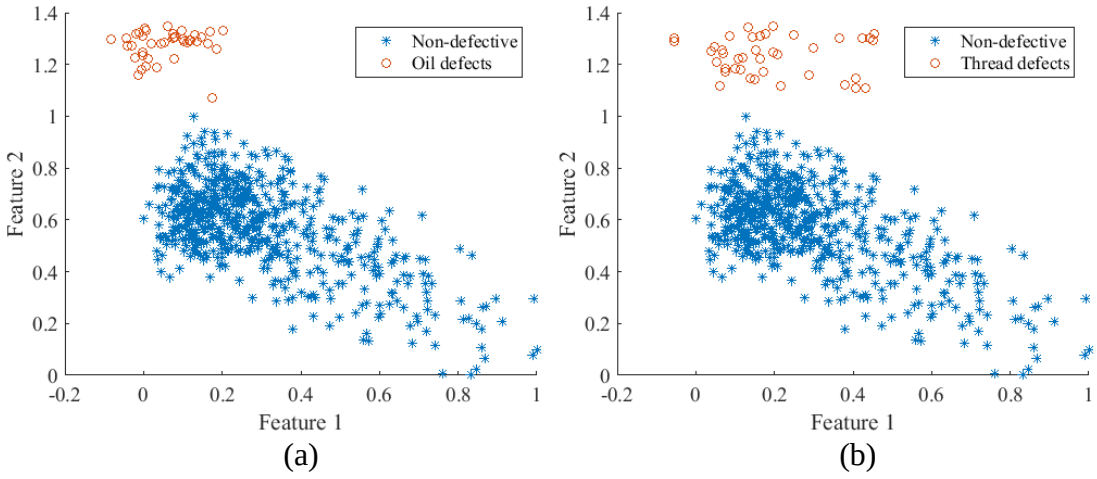


Figure 4.13. *Defect comparison (a) oil patch and (b) thread defects*

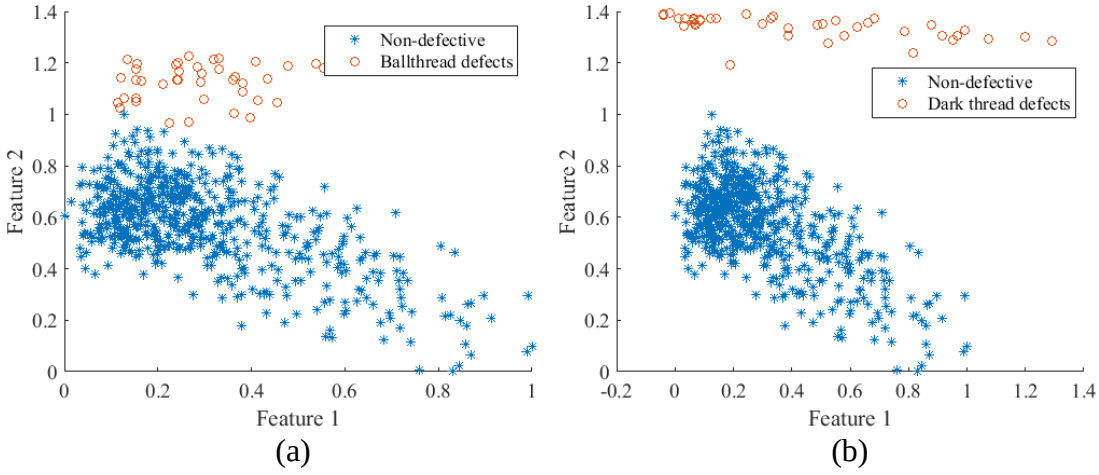


Figure 4.14. *Defect comparison (a) ball thread and (b) dark thread defect*

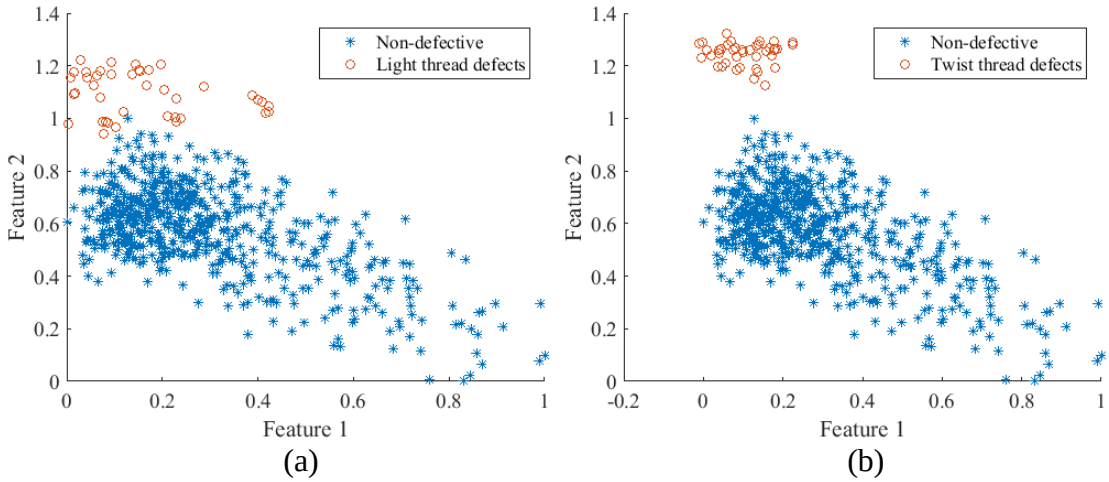


Figure 4.15. *Defect comparison (a) light thread and (b) twisted thread*

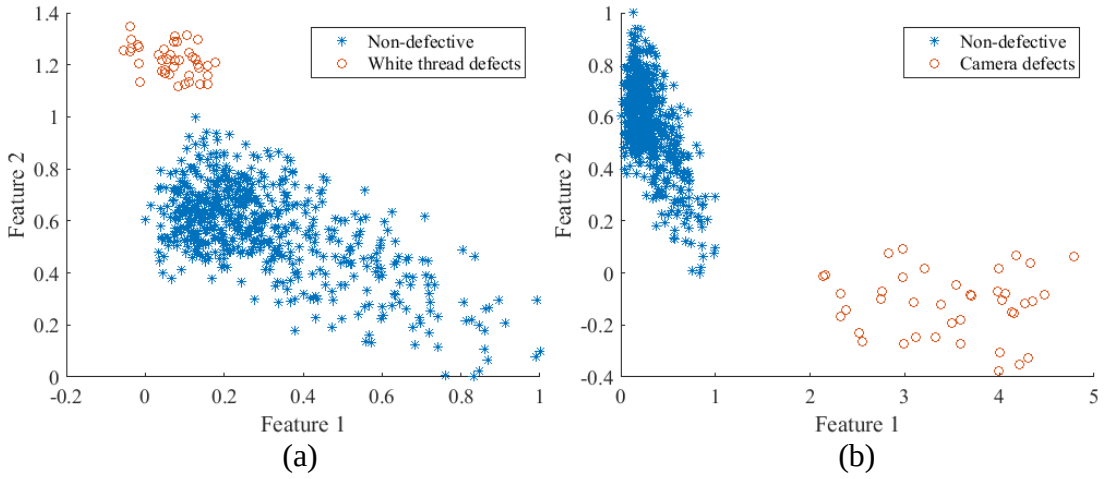


Figure 4.16. *Defect comparison (a) white thread and (b) camera defect*

4.7. Model Validation

We selected the common unitary classifiers from the three unitary classifier categories highlighted in chapter 3. We used Gauss, MoG (Mixture of Gaussian), KNN (K-Nearest Neighbor), K-means, PCA (Principle Component Analysis) and AE (Auto Encoder) from the dd tools 2.0.0 (a Matlab toolbox for data description, outlier and novelty detection) [34]. In all the unitary classifiers, there are two parameters that are set by the user. One of these two parameters referred to in this thesis as parameter 1 is common to all the unitary classifier and represents the target rejection rate which is the percentage of normal samples to be treated as outliers (the fraction false negative). The second parameter which we refer to as parameter 2 is dependent on the specific unitary classifier, the actual parameter identity for each unitary classifier under review is given in Table 4.2 except for the nearest neighbor classifier which does not have a second user defined parameter in this context [34]. These two parameters greatly influence the performance of the unitary classifiers and need to be optimized through a validation process.

Table 4.2. *Model parameters*

	Parameter 1	Parameter 2
Gauss	Rejection rate	Number of clusters
MoG	Rejection rate	Number of clusters
KNN	Rejection rate	NA
K-means	Rejection rate	Number of clusters
PCA	Rejection rate	Number of PCA components
AE	Rejection rate	Number of hidden units

4.7.1. Model Validation Metrics

A brief description of some of the metrics used in this section is given below. As a first step the basic measures from which other performance metrics can be extracted can be presented as elements of what is called a confusion matrix Table 4.3.

Table 4.3. *Confusion matrix*

	Actual Yes	**Actual No**
Predicted Yes	TP	FP
Predicted No	FN	TN

FN, FP, TN, and TP represent false negative, false positive, true negative, and true positive, respectively [39], which form the basis for the following metrics.

$$accuracy = \frac{TN + TP}{TN + TP + FN + FP} \times 100 \quad (4.2)$$

Accuracy represents a measure of the effectiveness of a classifier with respect to both the negative and positive samples.

$$sensitivity, recall = \frac{TP}{TP + FN} \times 100 \quad (4.3)$$

Sensitivity gives the ratio of true positives versus all the positives; it tells us what proportion of actual positives is correctly classified.

$$precision = \frac{TP}{TP + FP} \times 100 \quad (4.4)$$

Precision is basically a measure of the system's ability to make correct prediction for the positive class. It tells us what proportion of the positive predictions is indeed positive.

$$specificity = \frac{TN}{TN + FP} \times 100 \quad (4.5)$$

$$F_1 score = \frac{2\,P.R}{P + R} \quad (4.6)$$

Having determined the metrics that we shall use, the process of finding optimal parameters was as described below.

4.7.2. Optimal Target Rejection Rate (Parameter 1)

Since we only had the positive samples for training we used artificial outliers derived from the positive training samples [2] using the methods described in [38]. Figure 4.17 shows outliers generated from the normal samples using the PCA (a), sphere (b) and the box distribution methods (c). In this study we used all the three above mentioned methods to benefit from all their unique properties.

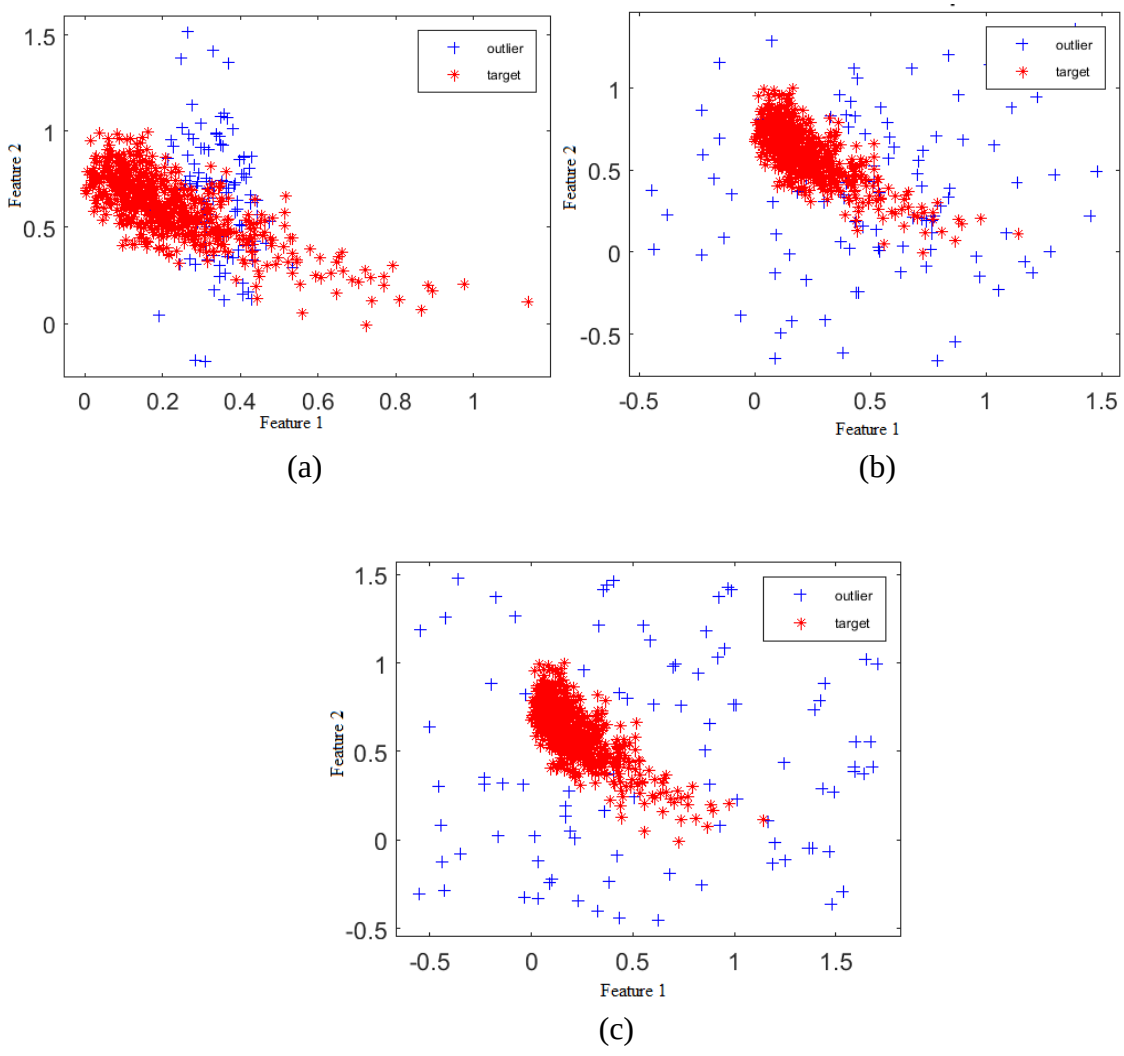


Figure 4.17. *Artificial outliers generated by (a) PCA (b) sphere and (c) box distribution methods*

To determine the best rejection rate, we held the second parameter for each classifier at the default value and tested the rejection rates from 0 to 1 with 0.01 step increment. For clarity we show the plot from 0 to 0.1 as this was the range where the optimal parameters occurred.

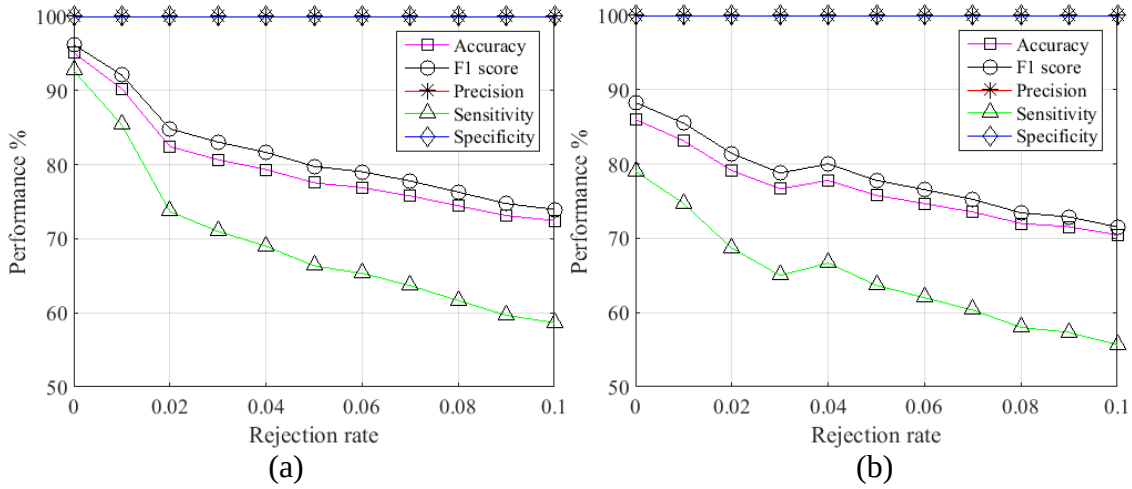


Figure 4.18. *Performance vs rejection rate (a) Gauss (b) MoG*

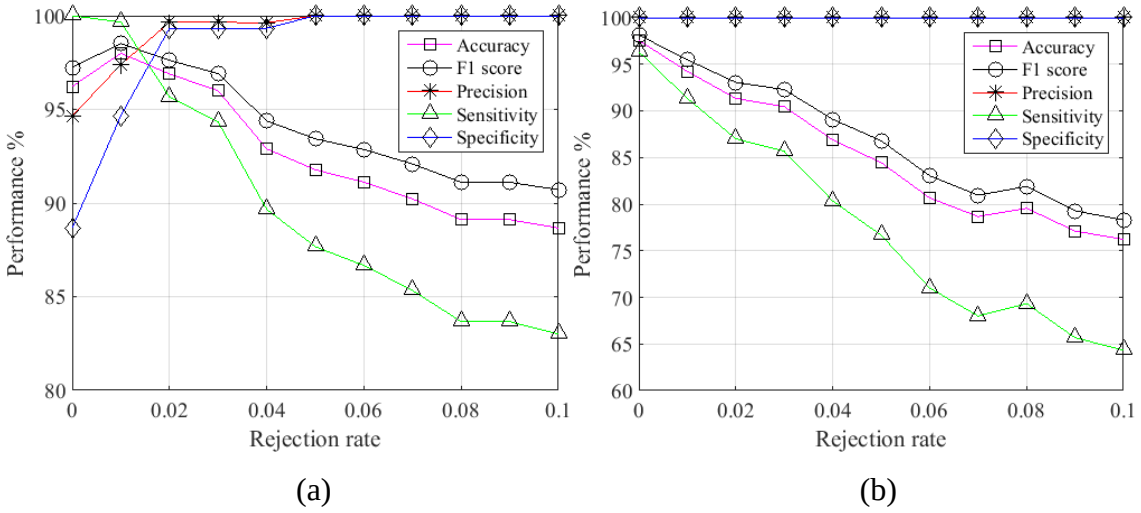


Figure 4.19. *Performance vs rejection rate (a) KNN (b) K-means*

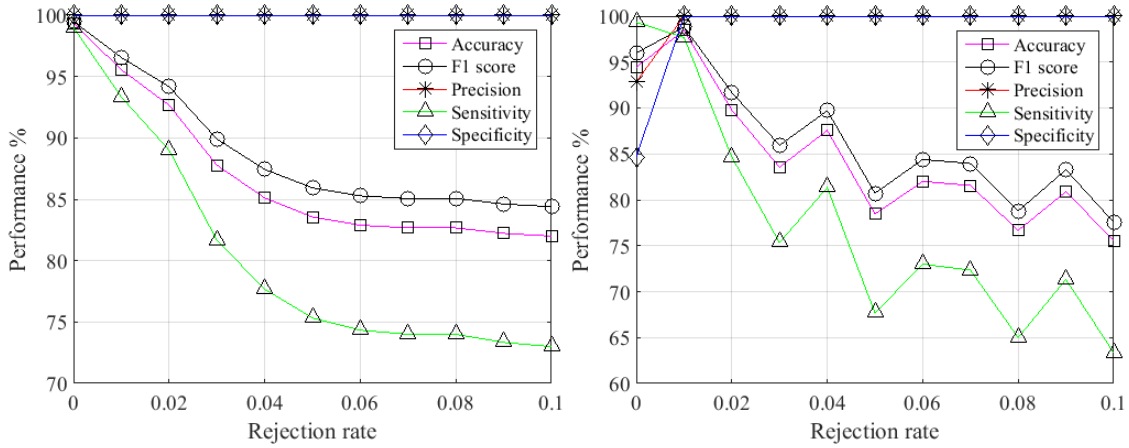


Figure 4.20. *Performance vs rejection rate (a) PCA (b) Auto Encoder*

Results of this optimization are presented in the third column of Table 4.4 which correspond to the values of rejection rates where the accuracy scores were highest.

4.7.3. Parameter 2 Optimization

For the second parameters for each classifier where applicable, we held the rejection rate value computed from the previous section constant and tested the classifiers with values ranging from 1-10 for the second parameter as shown in figures 4.21 – 4.22.

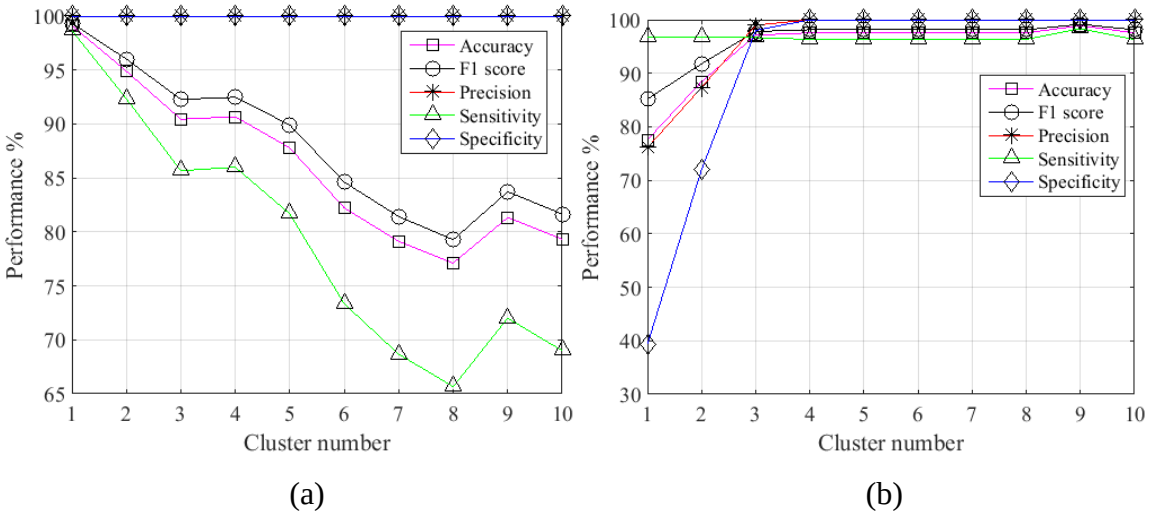
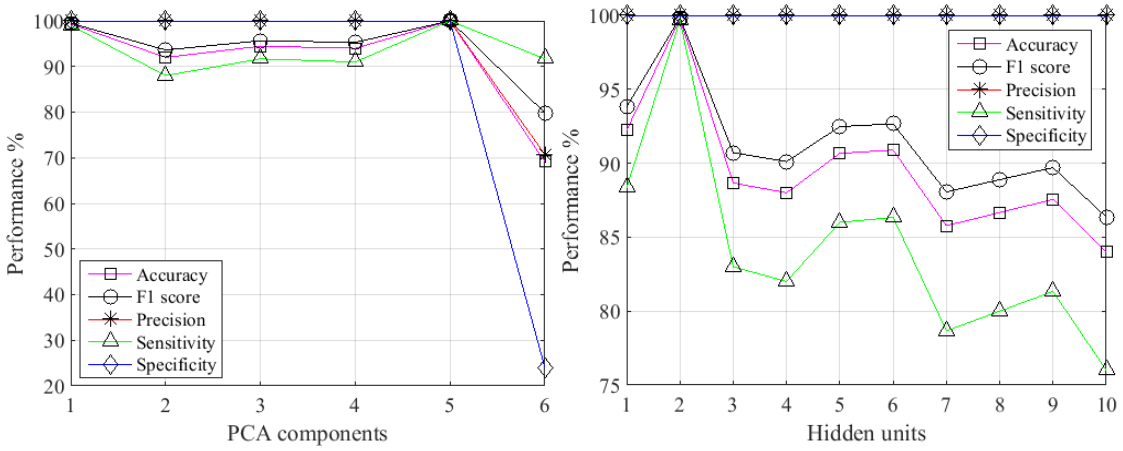


Figure 4.21. *Performance plot for (a) MoG (b) K-means*



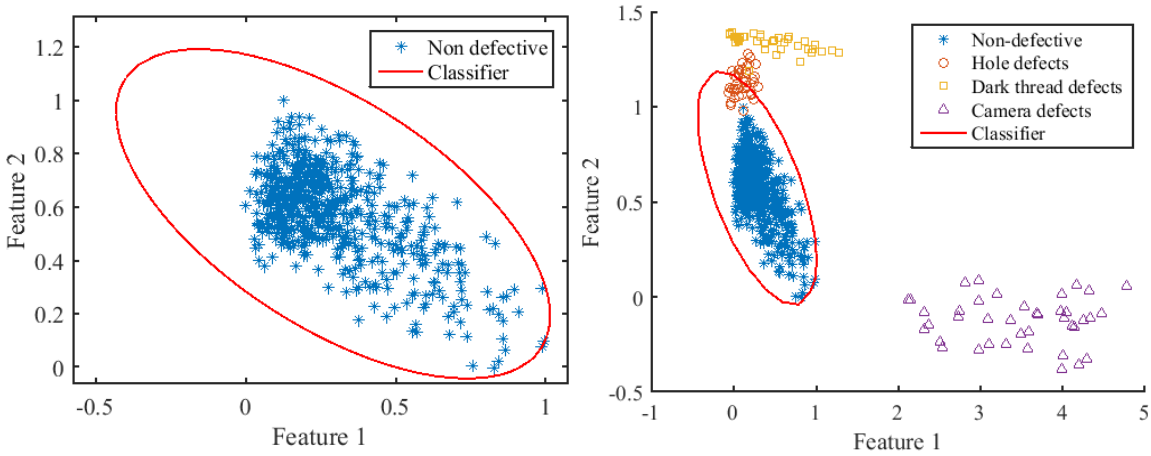
(a) (b)

Figure 4.22. *Performance plot for (a) PCA (b) Auto Encoder*

The values which gave the highest accuracy score were taken as the second parameters. The values are as tabulated in the fifth column of Table 4.4. The number of clusters for the Gauss unitary classifier is 1 by default since it has only one cluster by design.

Table 4.4. *Model parameters*

	Parameter 1	Value	Parameter 2	Quantity
Gauss	Rejection rate	0	Clusters	1
MoG	Rejection rate	0	Clusters	1
KNN	Rejection rate	0.01	NA	-
K-means	Rejection rate	0	Clusters	3
PCA	Rejection rate	0	PCA components	1
AE	Rejection rate	0.01	Hidden units	2



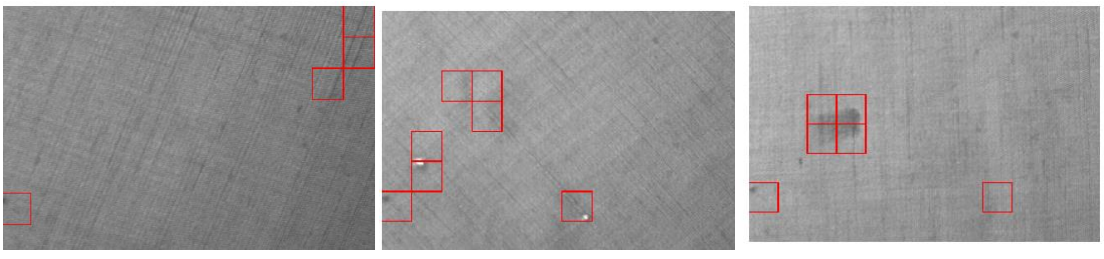
(a) (b)

Figure 4.23. *Trained unitary classifier (a) classification result (b)*

Figure 4.23 (a) above shows a trained classifier using only defect free samples and (b) the classification result for three different defect types. The non-defective samples that are found outside the ellipse form the false negatives. The dark thread, hole defects and camera defect samples that are found outside the ellipse represent the true negatives while the non-defective samples inside the ellipse represent the true positives. The hole defects found inside the ellipse represents the false positives.

The following figures give a visual illustration of the classification process for different scenarios.

4.7.4. False Negatives



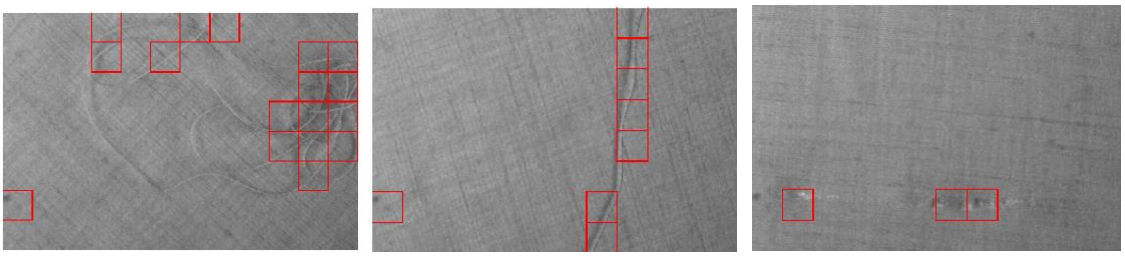
(a) (b) (c)

Figure 4.24. *False negatives in (a) normal image (b) hole defects (c) oil patches*

The results from the training stage indicate that even the image samples from the Tilda dataset's folder of normal samples contain some patches which are defective as shown in (a). This is one of the advantages of the unitary classifier since in a conventional binary classification environment these defective patches from normal

samples would be labeled normal and lead to false positives. In a unitary classifier the degree of acceptance for these defective patches is known as the target rejection rate; a large rejection rate will lead to more false negatives if the normal samples contain a lot of such patches leading to a reduced performance score albeit falsely.

4.7.5. False Positives



(a) (b) (c)

Figure 4.25. *False positives in (a) Ball thread (b) Light thread (c) Thread defect*

The areas of the thread not enclosed by the red boxes provides a manifestation of the false positive, these formed the majority of the false positive cases.

4.7.6. True Positive and True Negative

All the areas not boxed with no visible defects in them represent the true posistives while all boxed areas with visible defects in them represent the true negatives.

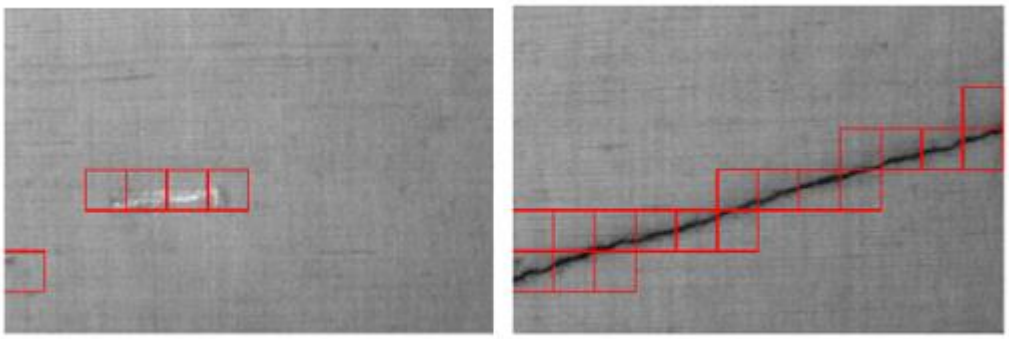


Figure 4.26. *Results of true negative (a) crack defect (b) dark thread defect*

Overall, the proposed solution is as shown in Figure 4.27.

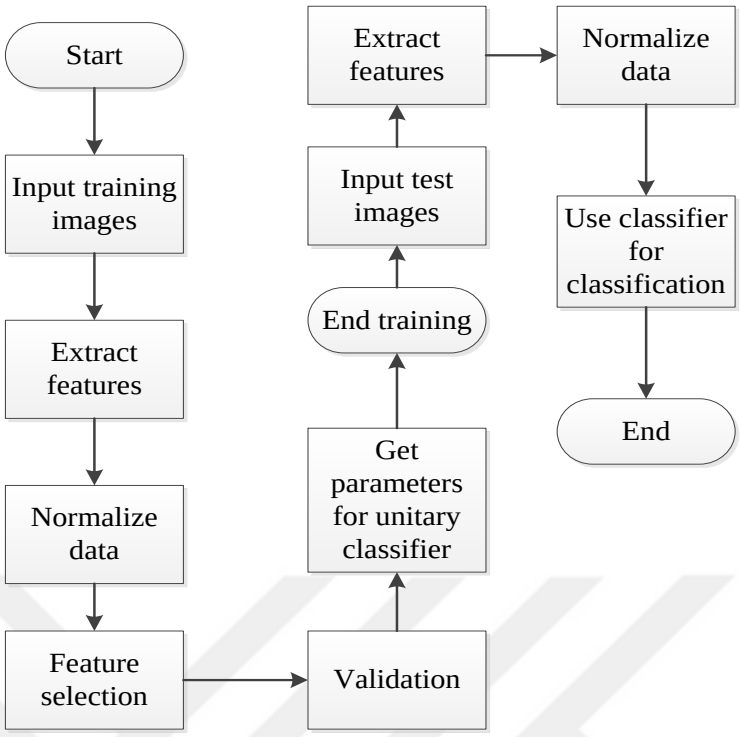


Figure 4.27. *Proposed system flowchart*

5. PERFORMANCE EVALUATION

5.1. Unitary Classifiers' Performance

Table 5.1. *Unitary classifier performance*

	Gauss	MoG	KNN	K-means	PCA	AE
Precision	98.39	98.77	81.82	99.50	99.49	92.49
Sensitivity	99.33	99.50	90.67	99.83	99.83	96.83
Specificity	85.96	84.56	88.42	69.47	69.12	82.11
Accuracy	95.03	94.69	89.94	90.06	89.94	92.09
F1 score	98.86	99.13	86.02	99.67	99.66	94.61

From Table 5.1 above, it can be shown that the Gaussian model at 0 rejection rate and 1 kernel had the highest accuracy score of 95.03 due to its better generalizing ability. The KNN and the AE which had a larger target rejection rate (0.01%) had the least precision (81.82% - 92.9%) while the remaining classifiers with smaller rejection rates (0%) had the highest precision scores (98.39% – 99.50%) implying an inverse relationship between the target rejection and precision.

5.2. Comparison with Binary Classifiers

To mimic the real-life defect detection experience, we trained all the classifiers binary and unitary with the normal samples and one defect type at a time with the last training batch containing all the defects. This is a deviation from previous studies where the focus had been on data imbalance [36]. For clarity purposes the comparison with binary classifier will only take into consideration the defects which occupy significantly different areas. We used NNB (neural network binary), RFB (Random forest binary), DTB (decision tree binary), KNNB (knearest neighbor binary), and SVMB (support vector machine) for binary classifiers. To avoid being confused with the unitary versions, the binary classifiers the letter 'B' has been appended at the end to indicate that the classifier is binary.

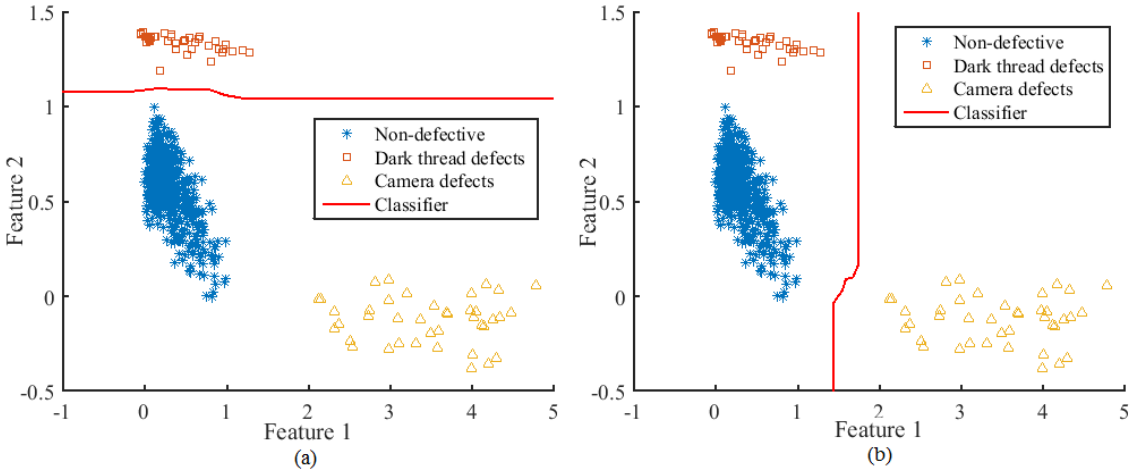


Figure 5.1. *Binary classifier (a) trained with dark thread defects (b) trained with camera defects*

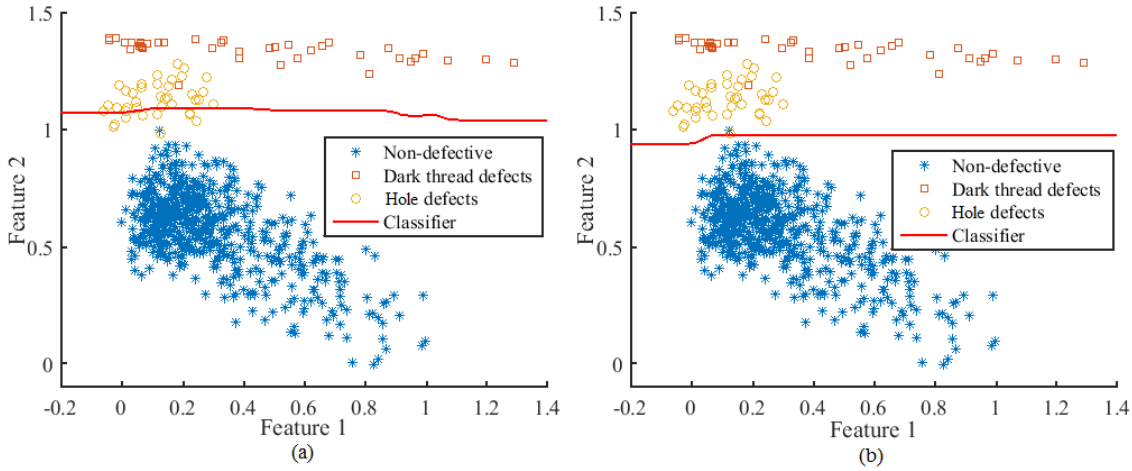


Figure 5.2. *Binary classifier (a) trained with dark thread defects (b) trained with hole defects*

The classifier from Figure 5.1 (a) was trained using the dark thread defects; in this case the features from the camera defects fall on the same side as the non-defective features. Similar results are observed when the classifier is trained with camera repositioning defect samples; Figure 5.1 (b), where both the non-defective and the dark thread defects fall on the same region and the classifier fails. On the hand when the classifier is trained with dark thread defects and tested with hole defects Figure 5.2 (a) performance drops slightly. The same is not true for a classifier trained with hole defects and tested on dark thread defects Figure 5.2 (b) where the classifier still performs well. The Random Forest binary classifier was used in the illustrations of Figure 5.1 and Figure 5.2.

Table 5.2. *Accuracy comparison for binary classifiers trained with different defect types*

Defect type	NNB	RFB	DTB	KNNB	SVMB
Crack	79.32	82.82	84.41	82.37	82.15
Hole	80.34	76.38	74.69	87.8	87.91
Oil patch	75.37	87.23	73.79	91.98	90.85
Threads	85.88	90.96	85.65	93.45	91.86
Ball thread	92.2	95.59	89.83	95.93	95.71
Dark thread	76.05	74.46	71.07	76.61	76.61
Light thread	87.01	87.68	86.89	94.01	94.8
Twisted thread	81.47	81.92	70.4	92.77	92.88
White thread	80.11	80.9	71.98	85.08	87.12
Camera error	71.98	67.91	68.81	68.02	72.77
All	92.77	90.51	93.9	95.37	96.84

Table 5.3. *Accuracy comparison for unitary classifiers trained with different defect types*

Defect type	Gauss	MoG	KNN	K-means	PCA	AE
Crack	95.03	94.69	89.94	90.06	89.94	92.43
Hole	95.03	94.69	89.94	90.06	89.94	92.32
Oil patch	95.03	94.69	89.94	90.06	89.94	92.66
Threads	95.03	94.69	89.94	90.06	89.94	92.32
Ball thread	95.03	94.69	89.94	90.06	89.94	88.36
Dark thread	95.03	94.69	89.94	90.06	89.94	92.99
Light thread	95.03	94.69	89.94	90.06	89.94	92.88
Twisted thread	95.03	94.69	89.94	90.06	89.94	92.66
White thread	95.03	94.69	89.94	90.06	89.94	93.79
Camera error	95.03	94.69	89.94	90.06	89.94	92.32
All	95.03	94.69	89.94	90.06	89.94	92.66
No defect	95.03	94.69	89.94	90.06	89.94	92.09

Table 5.2 shows the performance of binary classifier for the 10 different defect types, the last row of Table 5.2 shows the performance when all defect types are used.

Table 5.3 on the other hand shows the performance for unitary classifiers trained with the same defect types as the binary classifiers for the purposes of comparison (unitary classifiers do not need to be trained with defective samples).

According to the study by Bellinger et al. [36] on the factors influencing the choice between unitary and binary classifiers, a more appropriate metric for the performance is the accuracy which takes the into account the contribution of the performance for both defective samples (negative) and the non-defective samples (positives) as defined in equation 3.2. We therefore only consider this metric in our comparisons.

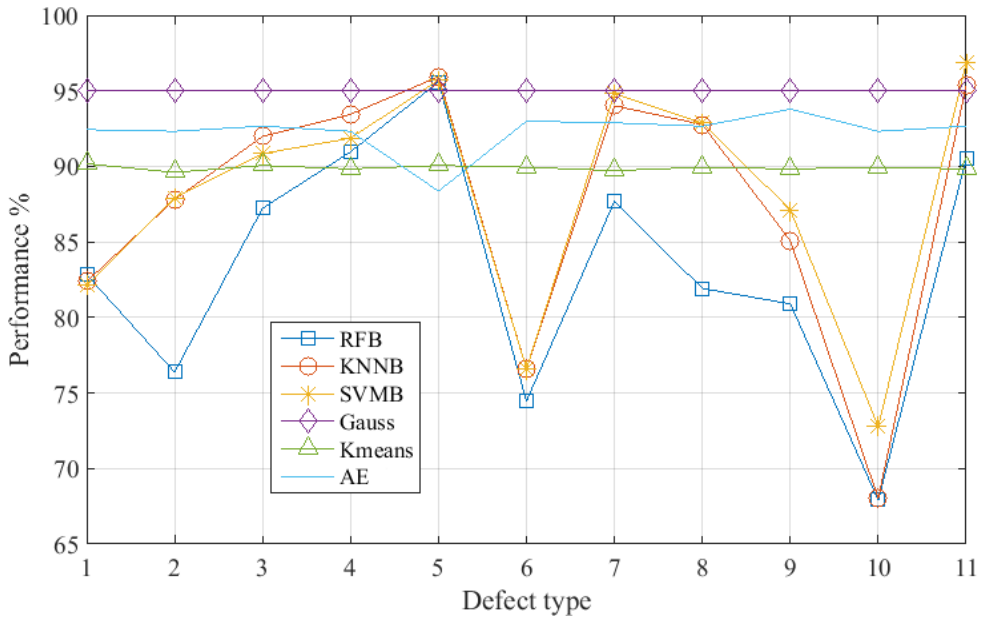


Figure 5.3. *Unitary vs binary classifier accuracy comparison*

Our experimental results show that all the binary classifiers trained with defect 10 (camera error) gave the worst accuracy score followed by defect 6 (dark thread) as shown in Figure 5.3. The binary classifiers generally showed inconsistencies when trained with different types of defects. Generally, the binary classifiers had better performance when trained with all defects than when trained with single defect types except for defect 5 where all the binary classifiers, performance was highest. The performance of the unitary classifiers was stable for all save for the auto-encoder which had some deviation at defect 5 but not to the magnitude of the binary classifiers

.

6. DISCUSSION & CONCLUSION

This study entailed making an empirical evaluation on the effect of data spectrum on performance of binary and unitary classifiers with respect to texture classification. We trained the unitary classifiers with only non-defective samples from the Tilda dataset and used artificial outliers generated from the non-defective samples for the validation stage. The texture classification process entailed detecting defects on textured textile images from the Tilda repository.

Binary classifiers differ from unitary ones not only in the nature of training samples but more importantly in the architecture of the classifiers themselves. It is in this unitary architecture that the favorable performance emanates from. While the binary classifier tries to look for an optimal boundary between classes, the unitary classifier looks for the optimal relationship of training samples to the classifier.

The quality of training features affects how the binary classifiers operate in the sense that when trained with all possible defect types, the performance of the classifier was poorer than when trained with a single more representative defect type. From this it can be deduced that more is not always better when it comes to training features but rather quality of the defect features. It was seen that some defect types, for instance thread defects produced better performance of the binary classifiers than other defects like the dark thread defects. This could be attributed to the separation distance from the target class being wide in some defects like the dark thread's defects hence providing more freedom for the separator position giving a higher probability of other defects falling on the same region as the positive class. On the other hand, some defect types like the thread defects have a narrower boundary from the positive class hence the probability of other defects lying on the negative side of the separator is higher. We call this the generalizing effect of the defect type. When the binary classifier was trained with this good generalizing defect type the performance was superior to the unitary classifier as well as binary classifiers trained with all the defects. The same also happens when the training outlier class occupies a different region in the feature space with respect to the testing outliers which also gave the greatest failure margin for the binary classifier.

The precision scores of the binary classifiers was better compared to the unitary classifiers since a balance has to be achieved in the unitary classifier in rejecting outliers and accepting targets, a tighter classifier will have a high specificity (rejecting more

negatives) at the expense of lowering the sensitivity (rejecting some positives). The binary classifier therefore works better when a sufficient defect distribution is available. In situations where the distribution of the defective samples is not adequately sampled the unitary classifier shows superior performance as the binary classifiers fail when they encounter defects which cannot be generalized by the defects that the binary classifier was already trained with.

7. FUTURE WORK

As a recommendation for future work we recommend the use of an online approach to the training stage to have a much more practical outlook to the defect detection problem.

We also recommend the computation of target rejection threshold from the distribution of the training data and not through trial and error using artificial outliers.

Since the Tilda dataset does not provide defect location labels, we recommend that defect masks be developed for more appropriate comparison with other methods from the literature.

In terms of application we recommend an extension of the unitary classifiers to traffic clustering and ROI determination in medical imaging compression applications.

REFERENCES

[1] R. O. Duda, P. E. (Peter E. Hart, and D. G. Stork, *Pattern classification*. .

[2] D. M. J. Tax, "One-class classification; Concept-learning in the absence of counter-examples," *Delft Univ. Technol.*, p. 202, 2001, doi: 10.1063/1.3605545.

[3] C. Gautam *et al.*, "Minimum variance-embedded deep kernel regularized least squares method for one-class classification and its applications to biomedical data," *Neural Networks*, vol. 123, pp. 191–216, Mar. 2020, doi: 10.1016/j.neunet.2019.12.001.

[4] T. B. E. Ilg and N. Mayer and T. Saikia, M. Keuper, A. Dosovitskiy, "Computer Vision Group, Freiburg," 2017. [Online]. Available: https://lmb.informatik.uni-freiburg.de/resources/datasets/tilda.en.html. [Accessed: 27-May-2020].

[5] S. Alam, S. K. Sonbhadra, S. Agarwal, and P. Nagabhushan, "One-class support vector classifiers: A survey," *Knowledge-Based Syst.*, vol. 196, p. 105754, 2020, doi: https://doi.org/10.1016/j.knosys.2020.105754.

[6] A. Kumar, "Computer-vision-based fabric defect detection: A survey," *IEEE Trans. Ind. Electron.*, vol. 55, no. 1, pp. 348–363, Jan. 2008, doi: 10.1109/TIE.1930.896476.

[7] O. G. Sezer, A. Ercil, and A. Ertuzun, "Using perceptual relation of regularity and anisotropy in the texture with independent component model for defect detection," *Pattern Recognit.*, vol. 40, no. 1, pp. 121–133, Jan. 2007, doi: 10.1016/j.patcog.2006.05.023.

[8] J. R. (Jim R. . Parker and K. Terzidis, *Algorithms for image processing and computer vision*. Wiley Pub, 2011.

[9] Y. Hu and C. X. Zhao, "Unsupervised texture classification by combining multi-scale features and K-means classifier," in *Proceedings of the 2009 Chinese Conference on Pattern Recognition, CCPR 2009, and the 1st CJK Joint Workshop on Pattern Recognition, CJKPR*, 2009, pp. 364–368, doi: 10.1109/CCPR.2009.5344087.

[10] M. A. García and D. Puig, "Supervised texture classification by integration of multiple texture methods and evaluation windows," *Image Vis. Comput.*, vol. 25,

no. 7, pp. 1091–1106, Jul. 2007, doi: 10.1016/j.imavis.2006.05.023.

[11] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004, doi: 10.1023/B:VISI.0000029664.99615.94.

[12] H. Bay, T. Tuytelaars, and L. Van Gool, “SURF: Speeded up robust features,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2006, vol. 3951 LNCS, pp. 404–417, doi: 10.1007/11744023_32.

[13] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, 2005, vol. I, pp. 886–893, doi: 10.1109/CVPR.2005.177.

[14] T. Ojala, M. Pietikäinen, and D. Harwood, “A comparative study of texture measures with classification based on feature distributions,” *Pattern Recognit.*, vol. 29, no. 1, pp. 51–59, 1996, doi: 10.1016/0031-3203(95)00067-4.

[15] A. K. Jain and F. Farrokhnia, “Unsupervised texture segmentation using Gabor filters,” in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 1990, pp. 14–19, doi: 10.1109/icsmc.1990.142050.

[16] R. M. Haralick, I. Dinstein, and K. Shanmugam, “Textural Features for Image Classification,” *IEEE Trans. Syst. Man Cybern.*, vol. SMC-3, no. 6, pp. 610–621, 1973, doi: 10.1109/TSMC.1973.4309314.

[17] C. [Di Ruberto] and L. Putzu, “Chapter 3 - A Feature Learning Framework for Histology Images Classification,” in *Emerging Trends in Applications and Infrastructures for Computational Biology, Bioinformatics, and Systems Biology*, Q. N. Tran and H. R. Arabnia, Eds. Boston: Morgan Kaufmann, 2016, pp. 37–47.

[18] A. Suruliandi and K. Ramar, “Local texture patterns -A univariate texture model for classification of images,” in *Proceedings of the 2008 16th International Conference on Advanced Computing and Communications, ADCOM 2008*, 2008, pp. 32–39, doi: 10.1109/ADCOM.2008.4760424.

[19] V. L. Nguyen, N. S. Vu, H. H. Phan, and P. H. Gosselin, “An integrated descriptor for texture classification,” in *Proceedings - International Conference on Pattern Recognition*, 2016, vol. 0, pp. 2006–2011, doi: 10.1109/ICPR.2016.7899931.

[20] C. C. Hung, M. Pham, S. Arasteh, B. C. Kuo, and T. Coleman, “Image texture classification using texture spectrum and local binary pattern,” in *International Geoscience and Remote Sensing Symposium (IGARSS)*, 2006, pp. 2750–2753, doi: 10.1109/IGARSS.2006.707.

[21] M. Ammar, S. Mahmoudi, and D. Stylianos, “A set of texture-based methods for breast cancer response prediction in neoadjuvant Chemotherapy Treatment,” in *Soft Computing Based Medical Image Analysis*, Elsevier Inc., 2018, pp. 137–147.

[22] F. S. Cohen, Z. Fan, and S. Attali, “Automated inspection of textile fabrics using textural models,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 8, pp. 803–808, Aug. 1991, doi: 10.1109/34.85670.

[23] M. S. Allili, N. Baaziz, and M. Mejri, “Texture modeling using contourlets and finite mixtures of generalized gaussian distributions and applications,” *IEEE Trans. Multimed.*, vol. 16, no. 3, pp. 772–784, 2014, doi: 10.1109/TMM.2014.2298832.

[24] P. García-Sevilla, “Analysis of irregularly shaped texture regions: A comparative study,” *Proc. - Int. Conf. Pattern Recognit.*, vol. 15, no. 3, pp. 1068–1071, 2000, doi: 10.1109/icpr.2000.903730.

[25] R. E. Bellman, *Dynamic Programming*. USA: Dover Publications, Inc., 2003.

[26] Z. M. Hira and D. F. Gillies, “A Review of Feature Selection and Feature Extraction Methods Applied on Microarray Data,” *Adv. Bioinformatics*, vol. 2015, p. 198363, 2015, doi: 10.1155/2015/198363.

[27] S. Khalid, T. Khalil, and S. Nasreen, “A survey of feature selection and feature extraction techniques in machine learning,” in *Proceedings of 2014 Science and Information Conference, SAI 2014*, 2014, pp. 372–378, doi: 10.1109/SAI.2014.6918213.

[28] F. Shih, *Image Processing and Pattern Recognition*. 2010.

[29] Y. Xu and R. Goodacre, "On Splitting Training and Validation Set: A Comparative Study of Cross-Validation, Bootstrap and Systematic Sampling for Estimating the Generalization Performance of Supervised Learning," *J. Anal. Test.*, vol. 2, no. 3, pp. 249–262, 2018, doi: 10.1007/s41664-018-0068-2.

[30] J. N. Hwang, S. R. Lay, and A. Lippman, "Nonparametric Multivariate Density Estimation: A Comparative Study," *IEEE Trans. Signal Process.*, vol. 42, no. 10, pp. 2795–2810, 1994, doi: 10.1109/78.324744.

[31] B. W. Silverman, *Density estimation for statistics and data analysis*. 2018.

[32] D. W. Scott, *Multivariate density estimation : theory, practice, and visualization*. Wiley, 1992.

[33] K. M. Ting, T. Washio, J. R. Wells, and F. T. Liu, "Density estimation based on mass," in *Proceedings - IEEE International Conference on Data Mining, ICDM*, 2011, pp. 715–724, doi: 10.1109/ICDM.2011.47.

[34] J. Tax, "Data description toolbox," pp. 1–19, 2003.

[35] L. Tarassenko, P. Hayton, N. Cerneaz, and M. Brady, "Novelty detection for the identification of masses in mammograms," in *IEE Conference Publication*, 1995, no. 409, pp. 442–447, doi: 10.1049/cp:19950597.

[36] C. Bellinger, S. Sharma, and N. Japkowicz, "One-class versus binary classification: Which and when?," in *Proceedings - 2012 11th International Conference on Machine Learning and Applications, ICMLA 2012*, 2012, vol. 2, pp. 102–106, doi: 10.1109/ICMLA.2012.212.

[37] L. Bissi, G. Baruffa, P. Placidi, E. Ricci, A. Scorzoni, and P. Valigi, "Automated defect detection in uniform and structured fabrics using Gabor filters and PCA," *J. Vis. Commun. Image Represent.*, vol. 24, no. 7, pp. 838–845, Oct. 2013, doi: 10.1016/j.jvcir.2013.05.011.

[38] "PRTools." [Online]. Available: http://prtools.tudelft.nl/. [Accessed: 05-Jul-2020].

[39] N. Japkowicz and M. Shah, “Evaluating Learning Algorithms: A Classification Perspective,” *undefined*, 2011.

CURRICULUM VITAE

ORCID ID: 0000-0002-5233-0365

Personal Info

Name	:	Khamis Salim BAMAMA
Place of birth	:	Mombasa, Kenya
Email	:	kbamash@gmail.com

Education

Sept 2018 – Aug 2020	:	Anadolu University MSc. Telecommunication Engineering
May 2009 – Jun 2014	:	Jomo Kenyatta University of Agriculture & Technology BSc. Telecommunication Engineering
Feb 2004 – Nov 2007	:	Allidina Visram High School Kenya Certificate of Secondary Education

Work Experience

Jul 2016 – Jul 2017	:	**Operations Manager**, Capcom Ltd
Nov 2015 – Jul 2016	:	**Engineer**, Capcom Ltd
Jan 2012 – Mar 2012	:	**Engineering Intern**, Sea Submarine Communication Ltd
Mar 2011 – May 2011	:	**Engineering Intern**, Kenya Civil Aviation Authority

Seminars, Awards & Honors

July 2020	:	Turkey's YTB High achiever award
Aug 2017	:	Turkish Government Graduate Scholarship
Jun 2011	:	CIC Insurance Group Leadership Seminar
May 2008	:	MEWA student of the year award KCSE 2007 2nd position
Jun 2007	:	Kenya National Mathematics Contest

Skills	:	Matlab, Python, C++, Video Surveillance Systems
Research Interests	:	Signal & Image Processing, Machine Learning
Languages	:	English, Swahili, Turkish