FABRIC DEFECT DETECTION USING ONE-CLASS CLASSIFIER

V.K.N.Madhusanka

199476R

Degree of Master of Science in Artificial Intelligence

Department of Computational Mathematics,
Faculty of Information Technology

University of Moratuwa

Sri Lanka

March 2022

FABRIC DEFECT DETECTION USING ONE-CLASS CLASSIFIER

V.K.N.Madhusanka

199476R

Thesis submitted in partial fulfilment of the requirements for the degree of Master of Science in Artificial Intelligence

Department of Computational Mathematics,
Faculty of Information Technology

University of Moratuwa

Sri Lanka

March 2022

DECLARATION

I declare that this dissertation doesn't include, without acknowledgment, any content submitted for a Diploma or a Degree at any institute, and that it does not contain, to the best of knowledge and believing, any material previously released or written by another person or myself, except where suitable citations are made in the text.

I also offer my consent for my dissertation to be photocopied and interlibrary loaned if it is accepted, and for the title and description to be made available to outside organizations.

Name if the student

V. K. N. Madhusanka Signature of Student

Date:

Supervised by

Dr. Subha Fernando Signature of the supervisor

Date:

Abstract

Textile is wide, very important in critical industry, because it provide lot prodcut to the human day to day life. As example cloths, wipes, transportation materials, wipes, hosuning materials etc. Then quality of the products are very important for their demand. Therefore defect identification during the production is very importat and then they can maintain better price for their production. Therefore fabric defect detection and identification is a very impotant part of the textile industry's quality control process. Currently, there are many manualinspection method to identify defects and, to enhance the efficiency, it is needed to repace manual inspectionmethod bby a automatic inspection method.

Machine vision is diversifying and expanding in defect detection using deep learning. Traditional systems like detecting and classifying defects using image segmentation, defect detection and image classification have some limitations like requiring a lot of defective data to the training process and needing pre-identification of defects in the datasets. However, it is very difficult to get a large amount of actual data with defects and real-time processes.

The one-class classifier is a classical machine learning problem that has received considerable attention recently for fabric defect detection. Tin this scenario, only non-defective class data are available in the training process and avoid the requirement of defective to train process. However, State of art models in deep neural networks with one-class classifiers is still unable to record higher accuracy.

This research proposes our approach, for identifying defective fabric using features of the non-defective fabric with higher accuracy. The implications of this research can be an initiative to such applications. That approach consists of a VGG-16 pre-trained framework and trainable network with a new Loss function for increase accuracy of defect detection.

DEDICATION

To my parents for their dedicated partnership in the success of my life

ACKNOWLEDGEMENTS

I'd want to express my thankfulness to Dr K.S.D. Fernando, who sparked my interest in research and provided guidance and advice during this research. Her advice was invaluable to my thesis, and I owe her a significant portion of its success to her. I would like to thank and appreciate the staff members of the Department of Computation Mathematics for their consideration and interest on my research work.

Finally, I'd like to thank all of my family members and friends for their unconditional support during this long endeavour and through all my life. Though they may not realize it, they have had a huge impact on me and I would not have grown to where I am now without them.

TABLE OF CONTENTS

DECLARATION 3

DEDICATION 5

ACKNOWLEDGEMENTS 6

Chapter 1 INTRODUCTION 13

1.1 Prolegomena 13

1.2 Background and Motivation 13

1.3 Aim and Objectives 15

Aim 15

Objectives 15

1.4 Problem in Brief 15

1.5 Proposed Solution 15

1.6 Resource Requirements 16

1.7 Outline 16

1.8 Summary 16

2.1 Introduction 17

2.2 Artificial Neural network 17

2.2.1 Batch normalization 19

2.2.2 Dropout 19

2.3 Convolutional Neural Network 19

2.3.1 VGGNet 21

2.3.1.1 VGG 16 architecture 22

2.3.1.2 Transfer Learning 23

2.3.1.3 Complexity and challenges 23

2.4 Wavelet transformation 23

2.4.1 Discrete Wavelet Transform 24

2.5 Support Vector Machine 24

2.6 One-class Support Vector Machine (OCSVM) 26

2.6.1 Hinge loss with SVM 28

2.5.2 OCSVM with Hinge loss 29

Chapter 3 LITERATURE REVIEW 30

3.1 Introduction 30

3.5 Image-based defect detection with one-class classifier 30

3.4 One-class classifier based anomaly detection using 33

3.6 Challenges in OCSVM based defect detection 38

3.7 Summary 39

Chapter 4 APPROACH FOR FABRIC DEFECT DETECTION 40

4.1 Introduction 40
4.2 Input 40
4.3 Output 40
4.4 Process 41
4.5 Users 41
4.6 Features 41
4.7 Summary 41

Chapter 5 DESIGN OF FABRIC DEFECT DETECTION 42

5.1 Introduction 42
5.2 Input Images 42
5.3 Feature extraction network 42
5.4 Fully connected network 43
5.5 Gradient ramp loss function 43
Hinge loss function with one class classifier loss function. 43
Ramp loss function 44
Gradient ramp loss function with one-class classifier. 44
5.5 Summary 45

Chapter 6 IMPLEMENTATION 47

6.1 Introduction 47
6.2 Datasets 47
6.3 Feature Extraction 47
6.4 Fully connected network 48
6.5 Gradient ramp loss function 49
6.6 Training 50
6.7 Summary 50

Chapter 7 EVALUATION 51

7.1 Introduction 51
7.2 Evaluations 51
7.2.1 OCSVM 51
7.2.2 Ramp-OSVM 52
7.2.3 Robust-OCSVM 53
7.2.4 Grad-ramp OCSVM 54
7.2.5 Method comparison 55
7.3 Summary 56

Chapter 8 CONCLUSION & FURTHER WORK 57

8.1 Introduction 57
8.2 Conclusion 57
8.2.1 Achievement of Project Objectives 57
8.2.2 Overall Conclusion 58
8.3 Limitations and Further Works 58
8.4 Summary 59
Reference 60

List of figures

Figure 2.1: Leyers in Artificial Neural Network 18

Figure 2.2: Structure of Artificial Neural Network 18

Figure 2.3: CNN Architecture 20

Figure 2.4: Process of Convolutional layer 20

Figure 2.5: VGG 16 Network Architecture 23

Figure 2.6: Optimal Hyperplane of OCSVM 24

Figure 2.7: Support vectors 25

Figure 2.8: One-class Support Vector Machine 27

Figure 2.9: Hinge Loss Function 29

Figure 4.1: Basic Architecture of Novel Approach 40

Figure 5.1: Hinge Loss Function 43

Figure 5.2: Ramp loss Function 44

Figure 5.3: New gradient-Ramp loss function 45

Figure 6.1: Feature Extractor and Classifier in VGG 16 Network 48

Figure 6.2: Operations of Loss Function. 50

List of Tables

Table 7.1: OCSVM results 52

Table 7.2 Ramp-OCSVM results 52

Table 7.3 Robust-OCSVM results 53

Table 7.4 Grad-Ramp-OCSVM results with 0.5 ramp limit 54

Table 7.5 Grad-Ramp-OCSVM results with 0.1 ramp limit 55

Table 7.6 Grad-Ramp-OCSVM results with 0.05 ramp limit 55

Abbreviations

ANN	Artificial neural network
CNN	Convolutional neural network
SVM	Support Vector Machine
OCSVM	One-clas Support Vector Machine
VGG	Visual Geometry Group
DCNN	Deep Convolutional neural network
GPU	Graphics Processing Unit
VGGNet	Visual Geometry Group network
PB-OCSVM	Pinball - One-class Support Vector Machine
RLOCSVM	Ramp Loss One-clas Support Vector Machine
ROCSVM-RHHQ	Robust One-clas Support Vector Machine based on rescaled hinge loss function
CCP	concave -Convex Procedure
SVDD	Support Vector Data Description
SAD	semi-supervised anomaly detection
UCL	Upper control limit
LCL	Lower control limit
AUC	Area Under the Curve
OCSTM	One-class Support Tensor Machine

Chapter 1 INTRODUCTION

1.1 Prolegomena

Fabric defect detection is a challenging and important task in textile industry. Because defects in the products are directly affect to the quality of the product and that reduce the price and demand of the products.

Still there are many companies which are use man power to identify defects in the fabrics. But efficiency of the human is very low and accuracy is defend on time, background and fatigue of the human. Then machine based defect detection method as introduced to the industry. In lot of projects, main technology was image processing and computer vision. Those projects have addressed the mentioned limitations, but does't address the problem to optimal stage.

Our research proposes a practical method to detect defects of fabrics. Collecting defective data is a difficult task, because, an error may occur after thousands of meters of production. But existing methods need defective data for development. Our research aims to avoid the use of defective data for development.

This chapter gives an overall idea about the research we have done in the subject area of swarm robots in excavation. In doing so, we have structured the introduction chapter under several subheadings, aims and objectives, background and motivation, problem definition and our solution.

1.2 Background and Motivation

The defect detection and identification of fabric is a significant task of the quality control process in the textile manufacturing sector. Because clothing is one of humanity's most fundamental needs, and the textile industry's history dates back to the dawn of civilization. Fabric is the key element of human clothes and that fabric is utilized in a variety of industrial applications. In the textile business, sophisticated machinery is employed to make this fabric, and defects are discovered during the inspection phase. The price of fabric brought to market is directly dependent on

quality, it means depends on the number of co-occurrences of defects, and the price rises as the number of defects reduces.

Traditionally, manual human efforts were used to detect detection in the fabric production process. The manual fabric flaw identification procedure has several disadvantages including a lack of concentration, human tiredness, and time wastage. Then researchers have involved that area and proposed many approaches to that purpose. The vision-based approach has performed efficiently and effectively for quality control tasks. Machine vision applications can solve the limitations and downsides stated above ([1], [2], [3], [4]). To solve these constraints, many vision-based various applications have been presented in various research papers during the previous two decades.

Deep Convolutional Neural Networks (DCNNs)-based approaches have shown tremendous performance improvement for detection and recognition challenges over the last five years. Rasheed et al[1] review paper describes a lot of methods that are used for image processing techniques and Bergmann et al[2] show methods that are used for anomaly detection techniques with Mvtec dataset launch. From all of these methods, CNN has proven to be superior than other methods in terms of image classification tasks. CNN have the ability to extract features of the input image using minimal computation and higher effectively. Zhang et al. [3], perera and patel[4] have proposed a CNN based classification system for defect detection. One class classifier is an immersing concept in fabric defect detection. Ruff et al[5], perera and patel [4] proposed systems as one class classifier for defect detection. Weninger et al [6] proposed a different solution for defect detection. It tracks all the warp weft yarn to find defects by variations of yarn distribution. That system does not need prior knowledge to detect defects. There are many projects related to fabric defect detection with various technologies and advantages and disadvantages are at different levels.

1.3 Aim and Objectives

Aim

The aim of the research is to come up with an efficient algorithm that can detect defects of fabric with a one-class classifier.

Objectives

- To critically review existing approaches of fabric defect detection in the textile industry.
- To depth study on Deep learning technologies and image processing technologies that can be used in defect detection
- To design and develop an algorithm to achieve the final task.
- To benchmark the findings against the other research.

1.4 Problem in Brief

Fabric defect detection has to be developed to achieve effective accuracy for the industry. Past few decades, there were few concepts for fabric defect detection and those have achieved breakthrough results.

One class classifier with CNN is considered as a leading and effective method for the textile industry because that concept doesn't require defective fabric data for development. Defective data collection is very difficult in the textile industry because defects occurred during production very rarely and there are a lot of variations of defects. With those, all factors, the One-class classifier is a better solution for that field and State of art models in the deep neural networks are still unable to record higher accuracy for a given dataset when they are used as a one-class classifier.

1.5 Proposed Solution

Fabric defect detection and anomaly detection concepts have been developed with many new concepts to achieve higher accuracy. In this research, a deep neural network based one-class classifier is proposed to achieve higher accuracy for fabric defect detection compared to existing methods.

1.6 Resource Requirements

Since this is a deep learning-based approach, a computer with GPU capabilities was required for the successful completion of the project. For the training process, the model needs fabric data sets with the same light condition and same angles as well as for the testing, the model needs both defective and non-defective data with the same conditions.

1.7 Outline

Following is an outline of the rest of the thesis. In chapter 2, the basic techniques applied in this project is described. The next chapter is dedicated to reviewing existing research efforts in the field of fabric defect detection, highlighting the advantages and disadvantages of each strategy, as well as providing some background on the problem being addressed in this research. You may see an overview of the solution that I offer here in the Approach chapter. The design chapter describes the high-level process proposed method. The Implementation chapter explains how the design is implemented from beginning to end. The results are given out in the Evaluation chapter, which serves as an introduction to the final chapter, Conclusion, which focuses on the discussion of our work and future progress.

1.8 Summary

You may see an overview of the approach that I offer here in the Approach chapter. You'll learn about design in the Design chapter. This chapter introduces the study methods as well as the motivation for choosing this particular method. It also includes a summary of the project's goals and some background information about the problem that will be addressed in this study. The proposed remedy as well as the resources needed to conduct the research have already been mentioned. Finally, the chapter discusses the design of the following chapters.

Chapter 2 TECHNOLOGY - FABRIC DEFECT DETECTION

2.1 Introduction

In chapter 2, presented a critical review of literature on the usage of fabric defect detection and research problem as well as how to solve those problems. We have identified the requirements for research to create a fabric defect detection method to overcome this problem. This chapter deeply discusses the technologies and models that are used in this approach.

2.2 Artificial Neural network

Artificial intelligence is experiencing tremendous growth in bridging the gap between human and machine capabilities. Researchers and enthusiasts work in different parts of the field to do amazing things. The agenda in Artificial intelligence is to allow machines to see and understand the world like humans, to use knowledge for many purposes like NLP, classifications, media entertainment and referral systems etc

Artificial neural networks (ANN) are subsets of machine learning and it is the heart of deep learning. The structure and name of ANN are inspired by the human brain, imitating how biological neurons signal each other. Figure 2.1 shows the basic layer architecture of a neural network. Each node of layers connects to another node and the connection has weight and threshold.

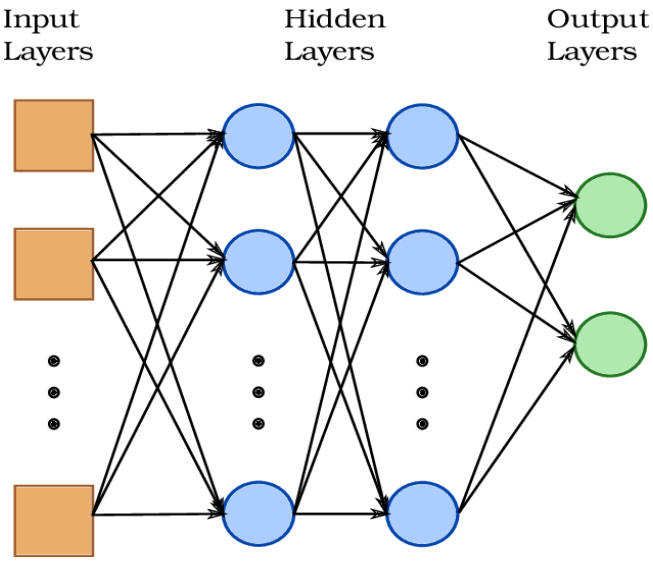


Figure 2.1 Leyers in Artificial Neural Network

The artificial neural network can be better represented as a weighted directed graph of the nodes. The relationship between neuron output and neuron input can be seen as the weight directed edges. The input signal pattern and image are received in vector form from an external source to the artificial neural network. These inputs are mathematically assigned by the numbers for each input. Then, each input is multiplied by its corresponding weight. All weight inputs are summarized in the computer unit.

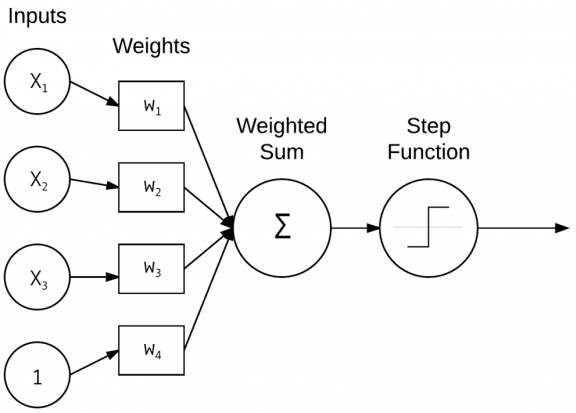


Figure 2.2 Structure of Artificial Neural Network

The activation function is the set of exchange functions used to obtain the desired output. There are different types of activation functions, but they are linear or non-linear function sets. Some of the most commonly used activation kits are binary, linear, and tan hyperbolic sigmoidal activation functions.

Neural network learning and accuracy improvement are directly based on training data. Once the learning process completes the fine-tuning of the neural network, it is a powerful tool in artificial intelligence that can classify and cluster data.

2.2.1 Batch normalization

ANN use batch normalization through a separate network layer that helps each layer of the network to learn more independently. It's used to keep the output from the previous layers more normal. The normalization helps to activations scales the layer.After doing normalization, the learning process becomes more efficient and that process can also be used to avoid model overfitting. Furthermore, that layer is used to standardize the inputs or outputs of the payers.

The batch normalization layer can be used in different points in the neural network, but typically use that layer after defining the sequential model and after the convolutional layer.

2.2.2 Dropout

To avoid the overfitting problem in the neural networks, the dropout technique can be used. In the dropout process, it ignores units randomly in the neural network during the training process. That means particular forward and backward pass does not these ignoring units.

2.3 Convolutional Neural Network

The CNN is a subclass of deep neural networks. CNN is typically used to analyze visual representations. CNN can input images and assign importance (learnable biases and weights) to the various objects in the image. Then the system is able to differentiate images from others. The requirement of pre-processing to images is

much lower in CNN compared to other classification techniques. While basic approaches require hand-engineering of filters, with sufficient training, CNNs can learn these characteristics. Figure 2.3 shows the basic architecture of CNN.

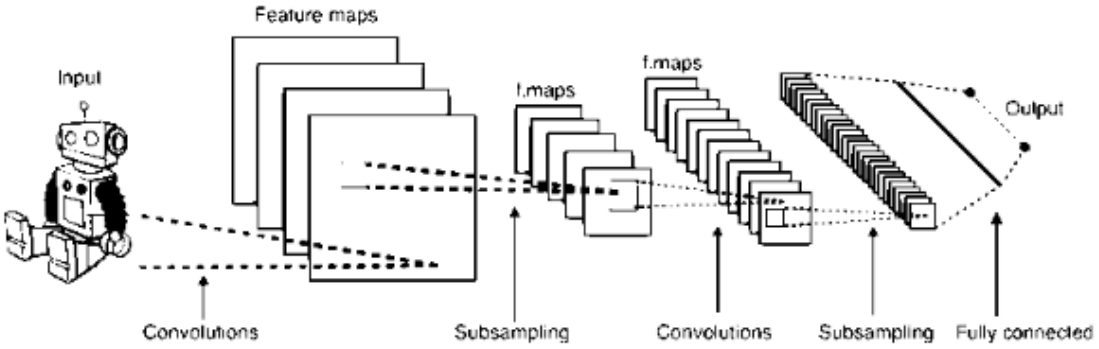


Figure 2.3 CNN Architecture

when we consider the neural network, we mainly focus on matrix multiplication, but in CNN it is not so. CNN uses Convolution technology for model development. Figure 2.4 shows the convolution operation. It uses a kernel matrix and applies to the image to get the convolved feature. Then that convolved features pass to the next layer.

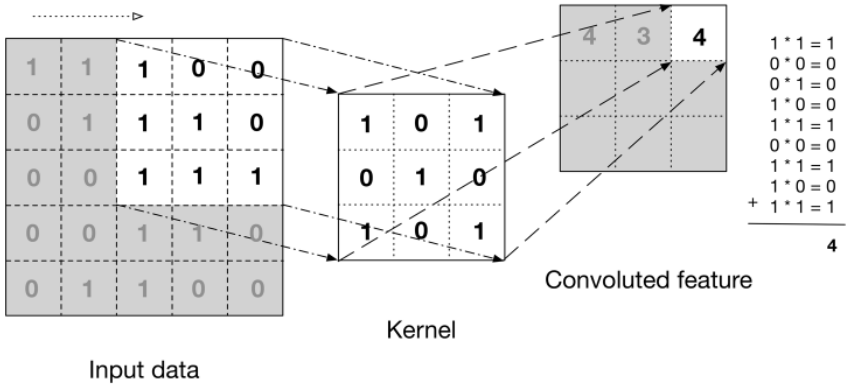


Figure 2.4 Process of Convolutional layer

CNN is made up of multiple layers of artificial neurons. Each layer creates numerous activation functions that transfer the image to the next layer when you insert image

into CNN. The first few layers of the network extract basic features of the image such as diagonal and horizontal edges. This output passes into the middle layers of the network and those layers detect more complex features such as combinational edges and corners. When it is needs to extract more complex features of images, complex and deeper network can be used.

A pooling layer is a another layer after the convolutional layer in CNN. The pooling layer is used to reduce the spatial size of convolutional features. According to that, this layer can reduce the computational power requirement by reducing the feature dimension.

There are two pooling methods as Average pooling and maximum pooling. Finally pooling layer output summarized version of the feature vector and pooling helps to keep feature representation approximately invariant to small translation in input. That means, if there is a small translation in the input image, pooled output values does not change.

While the pooling layer eliminates a lot of information, it does have several benefits for output. As an example, minimize overfitting, increase efficiency and decrease complexity.

At the end of the pooling layer, features send into the fully connected layer. This layer is used to classify images based on feature vectors. The convolutional and pooling layers generally use ReLu functions and fully connected layers generally use a softmax function.

This smart architecture has accelerated the development of Machine Vision tasks, and most of the current Artificial intelligence developments can be traced back to CNN. The CNN has been employed for anomaly detection because of the advanced layout behind it.

2.3.1 VGGNet

Visual Geometry Group (VGG) is a CNN architecture, often known as VGGNet. VGG was developed to enhance the model's performance by increasing the depth of

such CNNs. The term "deep" refers to the amount of layers in VGG-16 or VGG-19, which each have 16 or 19 convolutional layers.

The VGG architecture provides a framework for object recognition models. In addition to ImageNet, the VGGNet, which was formed as a deep neural network, performs baselines on a variety of projects and datasets. Furthermore, it is still one of the most widely used image recognition architectures recently.

The VGG16 model achieves about 92.7% test accuracy for the ImageNet dataset. ImageNet is a collection with almost 14 million photos categorized into over 1000 categories. It was also one of the most popular models submitted to the 2014 ILSVRC. VGG performed better than AlexNet due to a modification of kernel operations. Nvidia Titan Black GPUs were used to the training process of VGG. The VGG 19 model is also the same as the VGG 16 having 19 layers. That means VGG 19 model contains three additional convolutional layers than VGG 16 model.

2.3.1.1 VGG 16 architecture

The VGG 16 architecture is consists of very small convolutional filters and there are 13 convolutional layers and 3 fully connected layers. The network takes images with the size of 224x244.

VGG's convolutional layers are using the smallest receptive field (3x3), that to captures up/down and left/right actions. moreover, there are 1x1 convolution filters that perform a linear transformation. There is a ReLU unit, which is an AlexNet huge invention that cuts training time in half. The Relu is a rectified linear d]fucntion that gives input as the output if positive, otherwise, the output is zero. All the hidden layers of the VGG network model use the ReLU function as the activation function. Figure 2.5 shows the VGG 16 network architecture.

VGG 16 architecture has around 138 million parameters and this is a pretty extensive network. According to the modern standard, VGG 16 is a huge network.

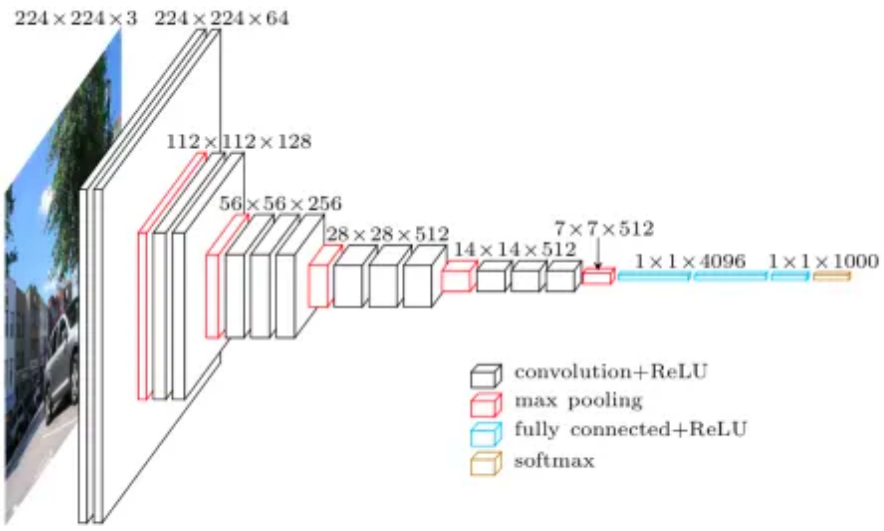


Figure 2.5: VGG 16 Network Architecture

2.3.1.2 Transfer Learning

Transfer Learning is the concept of beginning a new project with just a model that has already been trained on a machine learning project. This concept is used in different deep learning fields such as natural language processing, image classification and gaming processes. It is useful and easy to adapt a trained model to a new project.

2.3.1.3 Complexity and challenges

Every step of the convolutional layer doubles the number of filters that can be used and this is a major principle used to design VGG 16 architecture. The main disadvantage of VGG 16 is, it is a huge network, which means that it takes more time for the training process. Because of the number of fully connected layers and depth of the network, VGG 16 network is more than 533MB.

2.4 Wavelet transformation

The wavelet transform is a mathematical concept that is used in signal processing. The wavelet transform is also the same as the Fourier transform, but In the fourier

transform, it breakdown the input signal into sines signal and cosines signals. In wavelet transform, employs functions localized in both fourier space and localized.

2.4.1 Discrete Wavelet Transform

The discrete wavelet transform (DWT) is also a wavelet transform implementation that uses a limited set of wavelet scales and translations that satisfy specific principles. The advantage of DWT over Fourier transform is, DW can capture both location and frequency information. The Haar wavelet is the first DWT method and it represents an input by a list of 2^n. there are few other DWT methods named, Daubechies wavelet, dual-tree complex wavelet etc.

2.5 Support Vector Machine

The Support Vector Machine (SVM) is a machine learning technique that can solve supervised regression and classification task. This method is highly preferred by many as it performs with higher accuracy with lower computational power. However, SVM is mostly used to solve classification problems.

The main goal of SVM is to determine the best hyperplane in an n-D space for classifying data points. Figure 2.6 shows the optimal hyperplane of SVM.

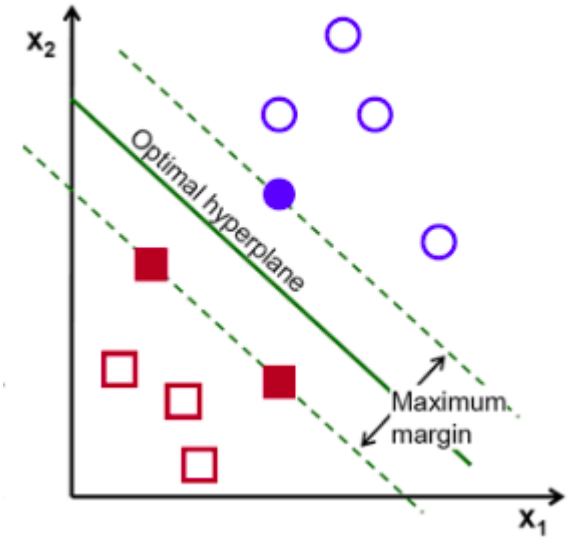


Figure 2.6: Optimal Hyperplane of OCSVM

There are lot of hyperplanes to separate two classes of data, but there is a one optimal hypeplant ot sepeare datapomts with maximum margin. The goal is to find that optimal hyperplane and that hyperplane helps to classify upcoming datapoints.

Hyperplane is the decision boundary to classification process. During the classification, data points are falling into a side of hyperplane. Then a class label can be assign to the each data point. Hyperplane can have different dimensions and that is depend on dimension of data points.

In SVM, it consider few data points as special, that are closer to the hyperplane. That data points are called support vectors. Figure 2.7 shows the support vectors in SVM and, that support vectors are used to maximize the margin to find best hyperplane.

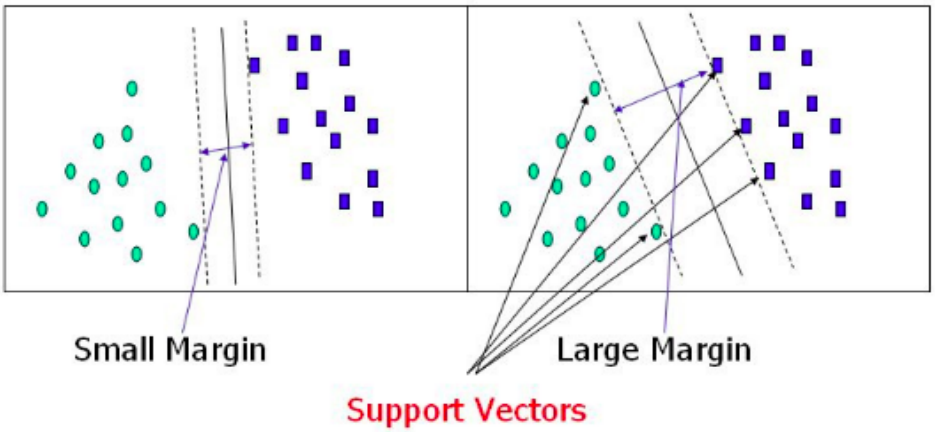


Figure 2.7: Support vectors

Cost function

The SVM technology's goal is to maximize the gap from data points to the hyperplane. Hinge loss function support to marginal optimization.

$$c(x, y, f(x)) = \begin{cases} 0, & \text{if } y * f(x) \geq 1 \\ 1 - y * f(x), & \text{else} \end{cases}$$

$$c(x, y, f(x)) = , \quad c(x, y, f(x)) = (1 - y * f(c))_+$$

If model predic class correctly, cost value is zero. Othervise it calculate a loss value. To balance the margin optimization and loss, a regularization parameter is added to the cost function. The equation looks like below after adding the regularization argument to the cost function.

$$min_w \lambda \, ||w||^2 \; + \; \sum_{i=1}^{n} (1 - y_i < x_i, w >_+$$

The gradient of the loss function can be found by taking partial derivatives with respect to weight. Using that gradient, weight can be updated.

$$\frac{\delta}{\delta w_k} \lambda ||w||^2 = 2\lambda w_k$$

$$\frac{\delta}{\delta w_k} (1 - y_i < x_i, w >)_+ = \begin{cases} 0, & \text{if } y_i \langle x_i, w \rangle \geq 1 \\ -y_i x_{ik}, & \text{else} \end{cases}$$

When there is no misclassification in the output of SVM, the model predicts correctly the class of all the data. Then gradient should be updated using regularization parameters.

$$\omega = \omega - \alpha \cdot (2\lambda\omega)$$

When there are misclassifications, the SVM model makes mistakes in the class prediction of the data points. Then loss along with the regularization parameters used to perform gradient update.

$$\omega = \omega - \alpha \cdot (y_i \cdot x_i - 2\lambda\omega)$$

2.6 One-class Support Vector Machine (OCSVM)

One-class Support Vector Machines is a SVM based technique for outlier or anomaly detection. OCSVM consider only one class, therefore OCSVM is an unsupervised learning technique. Therefore OCSVM is especially used for tasks whose non-target class is either ill-defined or totally absent. As shown in Figure 2.8 shows that

OCSVM works in a similar way to Support Vector Machines, except that during the training phase, OCSVM seeks to create a hyperplane to differentiate outliers from normal data.

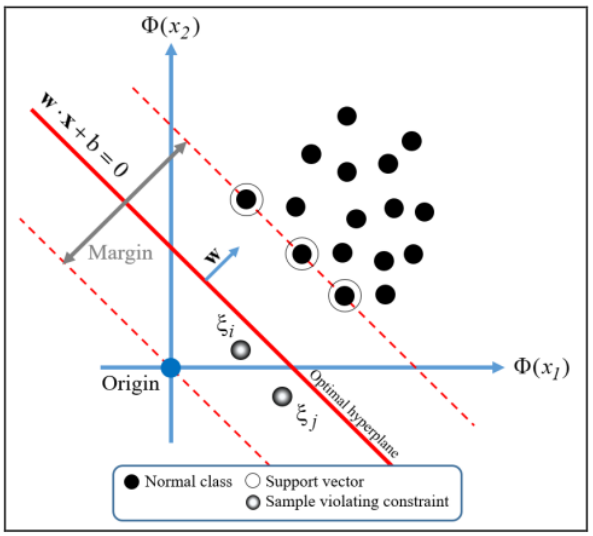


Figure 2.8: One-class Support Vector Machine

The first step in the training process is to use a kernel function to transform input data and translate that data into high-dimensional space, also known as feature space. The algorithm then selects the optimal separating hyper-plane from the training data by maximising the margin. The following is the objective function:

$$minimize\ f(w) = \frac{||w||^2}{2} + \frac{1}{vn}\sum_{i=1}^{n}\xi_i - b$$

$$subject\ to: (w \cdot \phi(x_i)) \geq b - \xi_i,\ \xi_i \geq 0, \forall i = 1,\ldots,n$$

ν represents the regularization coefficients in which the parameters controlled the crossing of data or proportion to outliers. The range of v is from 0 to 1.ξ i represent the slack variable and all the slack variables have positive values. The optimization of the loss function is done usually dual form.

$$minimize: -\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j K(x_i, x_j)$$

$$subject\ to: \sum_{i=1}^{n}\alpha_i = 1,\ 0 \leq \alpha_i \leq \frac{1}{vn}$$

$K(x_i, x_j) = \phi(x_i \tau).\phi(x_j)$ is called kernel function or nonlinear projection function. αi represents the Lagrange multiplier and α = [α1, α2,...,αN]T. Weight vector, w can be represented as

$$w = \sum_{j=1}^{N} \alpha_i \phi(x_i)$$

Any x_i whose corresponding Lagrange multiplier satisfies $0 < \alpha_i < \frac{1}{\nu N}$ can be used to compute the margin parameter, ρ.

$$\rho = \sum_{j=1}^{N} \alpha_j K(x_j, x_i)$$

At the end, decision function with the kernel can be derived as follows,

$$f(x) = \sum_{i=1}^{N} \alpha_j K(x_j, x_i) - \rho$$

Finally, class labels for the inputs is determined by,

$$\widehat{y} = sign(f(x))$$

Here, sign denotes the sign function.

2.6.1 Hinge loss with SVM

The hinge loss/error function is a common loss function used in support vector machines for binary classification. Furthermore hinge loss can also be extended to multi-class classification, while it can also be used in neural networks. As shown in Figure 2.9, Hinge loss always gives a non-negative loss value. It is always giving value which zero and positive.

The hinge loss can be defined as follows,

$$l(y) = max(0, 1 - t \cdot y)$$

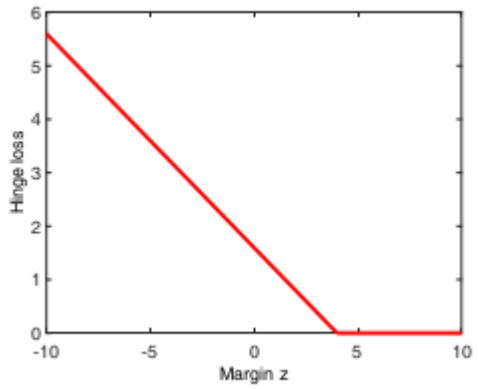


Figure 2.9: Hinge Loss Function

Here, $t = \{-1, 1\}$ as the labels and if the labels in dataset is $\{0, 1\}$, then labels should be map in to $\{-1, 1\}$. y represents the output predicted from the classifier. In the SVM, $y = w \cdot x + b$, where b and w are the parameters of the hyperplane.

2.5.2 OCSVM with Hinge loss

The original optimization problem of OCSVM can be rewritten as follows.

$$min_{w,\rho} j(w, \rho) = \frac{1}{2}||w||^2 - \rho + \frac{1}{vN}\sum_{i=1}^{N} H_{hinge}(z_i)$$

Here, hinge loss is a type of cost function that calculates the cost based on a margin or distance from the classification boundary. Here, Hinge $Z_i = max\{0, \rho - z_i\}$ is the hinge loss function definition with $Z_i = w_t \cdot \phi(x_i)$ and it shown in figure 2.9. Bigger loss values can be made by outliers and that can redice anti outlier ability of OCSVM.

Chapter 3 LITERATURE REVIEW

3.1 Introduction

An introduction to the broader project was given in the previous chapter, emphasizing the importance of fabric defect detection using a one-class classifier. This chapter gives a critical review of the literature on Fabric defect detection. This chapter is structured the literature review under several subheadings, namely, image-based defect detection with one-class classifier, the one-class classifier based anomaly detection, Challenges in OCSVM based defect detection and problem definition.

3.5 Image-based defect detection with one-class classifier

Image-based defect defection with one-class is a developing field recently because images can reveal a lot of details of the thing. But that derails may not be a boundary. There may be an infinite variation of the details. But there may be a few variations as expected and all of the others are unexpected. Then it is difficult to define all the unexpected details at the beginning. But expected details can be defined. With that discussion, when we come into image-based defect detection, it is easy to find positive class images, but difficult to negative class images with all the variation, because of infinite variations of negative class. In addition to that challenge, in the industry, difficult to collect negative class datasets due to the lower frequency of defects during production.

People have tried to sort out that problem with many concepts. The main factor of a lot of approaches is, considering defects as an anomaly. Then they treat defects as anomalies and correct images are target class data. At that time One-class classifier came into development. There are many approaches to solve that problem and they have achieved significant achievements with the solving problem.

A combined approach of OCSVM and CNN has been introduced by Zhang et al.,2017 [3] to reduce the negative dataset requirement of training the neural network. This approach has been applied to an Electronic component image dataset. The

network architecture of this approach is the same as the CNN two-class classifier except that CNN two-class classifier has an additional two-node output layer that is fully connected to the previous layer. The new approach has introduced a new contrastive loss function to one class classifier while CNN two-class classifier has the softmax loss function. Their aim was to put feature maps of defective data that are given from CNN into a high-dimensional hypersphere while defective features are far away from the hypersphere. The radius of the hypersphere is set according to the accuracy of the model. A Larger hypersphere radius allowed to rough defect detection while a lower radius for slight defect out. They have compared their approach with basic CNN. The datasets they have selected have four ratios of non-defective ample to defective samples as 4:1, 10:1, 20:1 and 40:1. Their model maintains better performance than the basic CNN model for all ratios.

They have concluded that their approach is suitable for the case with the small number of the negative class datasets, but they haven't been able to avoid the negative class. Then this approach does not solve the problem completely. To solve the problem, the system has to treat negative class data and use positive class for system generation.

Xing and Ji, 2018 [7], Tian et al., 2018 [8], Sonbhadra et al., 2021 [9] and Razzak and Khan, 2020[10] were trying to solve the problem in a different way. They have introduced a new loss function instead of hinge loss in OCSVM because the loss function has to calculate loss during the training process, then it has to face unexpected variation in anomalies. When SVM use those datasets with default hinge loss, it causes an inappropriate shift in the decision hyperplane of the SVM classifier towards the outlier because of the convexity of the hinge loss function. New loss functions that were introduced with these papers have more advantages than hinge loss. Let's see it.

Sonbhadra et al., 2021 [9] introduce Pinball one-class SVM with a more generalized new loss function for anomaly detection, because of the high sensitivity of hinge loss to the noise. PB-OCSVM reduce the sensitivity of loss function to the noise. But the complexity of pinball OCSVM and conventional OCSVM is the same. Then new loss

function does not affect computational power. The experiment is implemented with a COVID-19 dataset for disease diagnosis and results show that PB-OCSVM shows better performance than deep learning models and conventional OCSVM. For the generalization test and robustness test, PB-OCSVM shows significantly better performance than conventional OCSVM. But when considering our main problem, PB-OCSVM does not sort out it. It reduces only the sensitivity of hinge loss. Convexity and bounding property are the main concern in our problem to solve it.

Xing and Ji, 2018 [7] Introduce a Robust one-class support vector machine to rescale hinge the loss. For that purpose, they introduced a new loss function called rescaled hinge loss function. It is a monotonic, non-convex and bounded loss function and to solve the corresponding optimization problem with robust OCSVM, they have utilized half-quadratic optimization techniques. This approach in our path to remove weakness of default loss function in OCSVM, because hinge loss is unbounded. Therefore it can cause larger loss which may reduce generalization of the performance and anti-outlier ability of the OCSVM, but rescale hinge loss solve that problem.

The performance of robust OCSVM has been compared with default OCSVM, weight-OCSVM and ramp-OCSVM. They have used many images datasets to test their concept such as cancer, Breast Cancer, Blood Transfusion etc.

The result of ROCSVM-RHHQ shows slightly different statistical performance than the other three methods and ROCSVM-RHHQ shows superior generalization ability than others. When reducing the influence of outlier of all datasets, ROCSVM-RHHQ shows better performance than other methods. That means the robustness of ROCSVM-RHHQ is higher than other methods. With these results, we can use this method to testing of our application and compare it with our new approach. Because robust-OCSVM have all the properties that we want and there may be performance variation.

All the approaches under Image-based defect detection with one-class classifier subtopic try to detect defects and doesn't try to identify defects. The main target was to avoid negative class data from training. That is the nature of the anomaly detection

concept. That concept gets all the defective or negative class data as an anomaly and doesn't try to classify it. The papers are discussed above are used directly in the image-based datasets. But there may be many other new anomaly detection concepts, but not applied in image-based anomaly detection. Therefore, let's learn about existing anomaly detection techniques, but not applied in image-based datasets.

3.4 One-class classifier based anomaly detection using

OCSVM act the main role in the anomaly detection field. But basic OCSVM does not perform perfectly in all the fields, because there are many variations in anomalies in each field. Therefore people have introduced many approaches to optimize OCSVM.

Tian et al., 2018 [8] implement the ramp loss function replacing the hinge loss function in OCSVM. Ramp loss OCSVM is developed as space and robust method for anomaly detection. In this approach, the ramp loss function is broken down into the sum of concave function and convex hinge loss function. The concave -Convex Procedure (CCP) is a powerful method to solve non-differential, non-convex optimization problems because CCP is simple in tuning, solve a sequential problem in each iteration and it requires lower computational cost. Experiments have been done with artificial datasets with two dimensional, some UCI repository datasets, NSL-KDD datasets and UNSW-NB15 network datasets. According to the result, Ramp-OCSVM shows better robustness and more generalization power with higher anomaly detection accuracy than other methods. Because of that result, it is good to compare this ramp-OCSVM with the novel approach by applying it to our application.

Pantazi et al., 2019 [11] proposed a system to plant disease identification. At first, they used the local binary pattern for the image segmentation process. Then classification part is done by a one-class classifier known as Support Vector Data Description (SVDD) to classify the sample into outlier or target classes. Then again they have used a one-class support vector machine for disease classification. Finally used Nearest Support Vector Strategy to label the disease. The accuracy of the

proposed system is 95% even for 44-46 types of diseases. According to the result, the system work with considerable accuracy. When it is compared with our requirement, the dataset pre-processing part of that approach is not compatible with our application. Because pre-process is directly depend on the nature dataset. Then the user has to analyse, a suitable pre-process model for the dataset.,

Ruff et al.,2020 [12] introduce a new method to deep semi-supervised anomaly detection called Deep SAD. Their aim was to reduce manual feature engineering tasks for the high dimensional dataset and larger datasets. This method uses the same loss term in deep SVDD for unlabeled data and introduces a new loss term for labelled data that is weighted to control the balance between labelled and unlabeled data. Here unlabeled data is got as normal class data. Deep SAD have performed well when there is a more diverse training dataset. Supervised methods sometimes catch up as a suspect, when the labelled sample is not sufficient. When considering about anomaly detection part, they have not done any improvement and have used the existing SVDD method and they have modified the loss function for each sample set. Deep SAD is practically beneficial for more complex data and when it is considered with our approach view, there is not a huge dataset complexity problem and the anomaly detection concept is not suitable for our application.

Lamrini et al. [13] propose a method for automatic model selection of OCSVM. Because it is highly cost to constitute labelled data and have to identify anomalous, novel observations of data, it is difficult to analyse behaviours of past data. Those are the main challenges in model selection. For that process, the kernel parameter and regulation parameter of OCSVM is taken as learning configurations. Then run OCSVM for several configurations and evaluate the similarity of KPI signals for the select best configuration. They have applied this concept to detect anomalies in real network traffic and that have given satisfying results. However, finally, they get the best OCSVM model for a particular dataset. But OCSVM does not perform better in our application. So that concept doesn't affect our approach.

EB-OCSVM is a classification method with the extended boundary of OCSVM to reduce the error of noise of the data which was introduced by Zhang et al.,2019[14].

The system is tested in industrial multivariate time series anomaly detection from a simulated Tennessee Eastman process (TEP) with many cyber attacks. They have introduced an upper control limit (UCL) and lower control limit (LCL) to the normal dataset by classifying high noise data OCSVM. That UCL and LCL get as maximum and minimum values of the normal dataset because industrial control systems have high noise. Then extend the control limit of abnormal boundaries of EB-OCSVM. That new approach can reduce false-positive of the strong noisy datasets. EB-OCVM system builds the model from trained only with target class dataset. Then model classifies test data as either normal or attacked based on the boundaries of the classifier. EB-OCSVM concept doesn't affect our application. Because fabric dataset image doesn't face to high noise issue and this approach uses existing OCSVM for development. As we discussed before, basic OCSVM doesn't perform well in our application.

Kittidachanan et al.,2020 [15, p.] have introduced a method called GC-OCSVM to estimate the hyperparameters of OCSVM. Because it is difficult for researchers to find the parameter settings of the most effective SVM model. Normally, parameter settings are tested on the dataset. But in this approach, grid search can find the best parameter setting of OCSVM. For that process, they used the maximum of the area under the curve (AUC). The new approach is compared with the Isolation forest algorithm and this approach shows higher performance. When suitable parameters are chosen, the OCSVM model is a robust classifier in anomaly detection. This approach is also the same as the Lamrini et al. [13] that discussed above. So that approach is doesn't perform better in our approach.

Xing and Li, 2020 [16] introduced least-squares OCSVM to describe the similarity between the raining training dataset and new datasets. LS-OCSVM tries to find an optimal hyperplane for feature space that reflects the similarity of training dataset and sample data by analysing the distance between hyperplane and samples. They have introduced a more robust method called robust least square OCSVM (RLS-OCSVM) by correntropy loss function replacing the loss function of LS-OCSVM. As the result, RLS-OCSVM is more robust than LS-OCSVM. That robust least square OCSVM loss function concept is the same as the [7] concept.

Both are used rescaled hinge loss function. The performance of that method is compared with RL-ROCSVM, ROCSVM, R-SVDD and SD-SVDD. RLS-OCSVM has shown better robustness and generalization ability. That robust OCSVM concept has given the better performance and that concept will be used in our approach for comparison.

Razzak and Khan, 2020[10] introduce OCSTM with a new function to avoid the negative effects of the outlier of the dataset on OCSTM performance. The new loss function is also followed the rescaled hinge loss that is used in [7], [16]. That loss function reduces the effect of samples that are far away from labels, on the performance of the model. The OCSTM-BH method has been compared with OCSVM, R1STM, LOCSTM and deep one-class classification methods. They have done experiments with UCI machine learning datasets that are Breast Cancer, USPS, SONAR, MNIST, Daily Human Activity, Human activity datasets etc. The experiment results show that OCSTM-BH outperforms other methods for real-world datasets with the presence of outliers. As it is discussed above, that loss function will be used in our approach for the comparison part.

In one-class learning, boundaries are very critical. Accuracy and accuracy levels directly depend on the boundary. Wand and Cherian [17] have proposed a method to optimize tasks that are, minimize the distance of two origins of two subspaces and maximize the margin size between two hyperplanes. To achieve that goal, they have used the basic variant of one-class discriminative space called BODS to classify data to positive half-apace and negative half-space with minimizing the distance between hyperplanes. The Dash-Cam-Pose dataset has been used for testing purposes and that model has shown better performance and accuracy than other methods such as SVDD, OCSVM, LS-OCSVM etc. This concept does not affect to boundaries of the loss function, then this method does not address the overfitting problem in our application. Therefore that concept doesn't applicable in our application.

OCSVM based Fault detection system for reactive ion etching system propose by Tomas Sarmiento et [18] to avoid defective data in the training phase. They have applied the basic OCSVM in the reactive ion etching technique. That technique is a

combination of a chemical reaction and physical ion bombardment. They have achieved 100% accuracy with zero faults. Finally, they say that SVM can be used as an alternative to neural networks. But as we discussed above, OCSVM is not good at our application. Therefore that approach not align with our requirements.

Gu et al., [19] have proposed a method called the cost-sensitive hinge loss support vector machine (CSHL-SVM) model to get better generalization accuracy. In that technique, they consider false positive and false negative classification samples separately for hinge loss. Then there are two losses parts in hingeless by multiplying two constants called beta and lambda. Their approach has shown the result that, it is better than traditional cost-sensitive SVM. That new approach can be used in new samples, no need to re-train, it can update without training from scratch. This model has been tested in various samples and it has shown better results than the batch algorithm of CSHL-SVM. When considering our application, modification of OCSVM in this approach doesn't support our requirement. This does not provide a solution for the overfitting problem in our application.

The IoT field also requires anomaly detection concepts to detect unusual behaviour of various tasks. Razzak et al[20] have proposed a method to detect unusual patterns of IoT high dimensional data. They have also proposed a robust OCSVM concept that we have discussed in projects [7], [16]. To improve the robustness of the project, they have used a randomized feature learning concept and that concept can improve robustness and reduce the training time of the system. This concept has been tested with 10 datasets and it is perform better than SVDD, Autoencoder and OCSVM.

As we discussed earlier, we hope to use that loss function modification for comparison with our approach and their feature extraction concept cannot be used with our approach due to different types of data.

To increase the robustness of SVM, robust SVM based on rescaled hinge loss function (RSVM-RHHQ) is developed. As a version upgrade of RSVM-RHHQ, Singla et al[21] did research that focus on enhancing the sparsity. In the new approach, it adds correntropy to the alpha hinge loss function and it gives a better

loss function than rescaled hinge loss function. And also, they have used the primal-dual proximal method to remove the non-smooth and non-convex problems of the loss function. That system has been tested with various publicly available datasets and it has shown outperforms in sparseness, robustness and accuracy than existing methods. This concept directly cannot be used in our application due to SVM. Our target is one-class classifier based anomaly detection in an image dataset, but this concept is important to our approach to enhance performance.

3.6 Challenges in OCSVM based defect detection

The main concern of the one-class SVM is to avoid the requirement of the negative class of data for the training process. Few researchers have introduced an approach to minimize the negative class data required for the training process. Zhang et al.,2017 [3] approach have the ability to reduce negative class size to 40:1. The requirement is to remove negative class data completely from the training process.

The feature extraction process is a very important part of the whole project. It should be more generalized and effective for all processes. An image-based anomaly detection system doesn't have both positive and negative samples for the training process. Therefore, training a feature extraction model is an impossible task in this approach. As an alternative, a pre-trained feature extraction model can be used for the training and testing stage, but there is no 100% guarantee for features extracted from the model. A lot of researchers have used Alexnet ([22], [23]), VGG ([22], [4], [23]) and ResNet ([9]) framework for feature extraction method and few researchers have introduced their own framework ([3], [24]).

The loss function is the main factor that can change the whole performance. Loss function can control the sensitivity of defects for the training process. Then novel defects may not obey the loss function. Therefore trained model should have a suitable loss function for the relevant dataset. Otherwise, models may show poor performance. There are few researchers ([7], [8], [9]) that have introduced new loss functions for relevant datasets for higher performance. Thus, there are the main challenges when performing a one-class classifier for anomaly detection and there are a lot of approaches to handle those challenges with better performance.

3.7 Summary

In this chapter, a critical review of fabric defect detection methods and anomaly detection methods highlighting the applications of each approach with evaluations and limitations are presented. In the next chapter, the overall research approach will be discussed.

Chapter 4 APPROACH FOR FABRIC DEFECT DETECTION

4.1 Introduction

According to the theory described in chapter 2, the fabric defect detection approach using a one-class classifier is proposed as shown in figure 4.1.

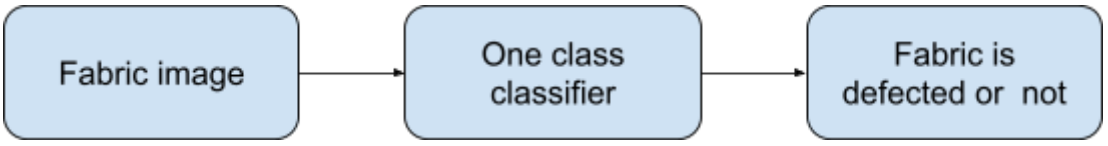


Figure 4.1: Basic Architecture of Novel Approach

The rest of the chapter describes the input, output and process along with the users and features of the system.

The following hypothesis is established for the project using this approach.

Hypothesis - The hypothesis of this project is that one-class deep neural networks can detect defects in fabrics with higher accuracy compared to existing methods.

4.2 Input

The inputs to the system are fabric datasets. To get higher performance and accuracy, each image in a dataset should be captured in the same light condition ad background, because fabric defects are very critical and there are many variations in the defects. For the development of this research, MVTech datasets and the UAMDD face datasets are used. The MVTech carpet dataset is a fabric dataset and other datasets are used for more generalization of the new approach. Other datasets in MVTech are transistor, cable, capsule, Hazelnut, leather and tile.

4.3 Output

The output of this system is a label for each fabric image. In the training process, the system creates a well trained one-class classifier. That model can be used in the testing process, the model predicts a label for each input image as it has defected fabric or non- defected fabric. That label indicate that the impute image is defeated or not.

4.4 Process

The new system get fabric images as inputs and extract features of each image using a pre-trained CNN model and create a feature vector for each image. That feature vector is used for the identification of the defects. For that purpose, the feature vector is again sent into a fully connected neural network which has a transformation layer to transform the feature vector and has a new loss function that is more suitable for the training process of the system. Then system creates well trained one-class classifier model and that model can be used to predict a fabric image as defective or non-defective.

4.5 Users

The proposed solution is particularly focused on the textile industry. In a lot of factories, they use manpower to detect defects and the problem is the efficiency of humans is not enough for that purpose. Because of that factories have to spend a lot of money on this task otherwise they will lose their production demand.

4.6 Features

The main feature of that system is, it requires only non-defective fabric sample images for development. Then users can develop the system without knowing about the defects. Then the user can start the system to defect detect along with the start of the production. This system can do real-time defect detection with higher efficiency. Then users can use that system in the production line for real-time defect detection and using that system, they can take decisions on the defects as soon as the defect arises.

4.7 Summary

The current chapter, which summarizes the project's components, might be considered the essence of the overall project. The input, process, and output generated by the process are described here, together with other project elements such as project users and specific features which are stretched out in the implementation. The overall design of the process discussed in this chapter is expanded in the next chapter.

Chapter 5 DESIGN OF FABRIC DEFECT DETECTION

5.1 Introduction

This chapter expands the system process that was explained in the previous chapter as the system by separating it into sub-models. The design diagram will be used to explain the fabric defect detection system.

5.2 Input Images

The new approach uses the convolutional neural network for feature extraction. As discussed in chapter 2, CNN can do preprocessing past. Therefore the system can work with colour images and the system can handle image size. To get higher accuracy from the system, users have to use images with the same light condition and same angle images. Because defects of fabric is critical. Resolution is also important and resolution higher than 224x224 is important for feature extraction network.

5.3 Feature extraction network

In the practical scenario, most of the issues may be generalized, allowing our model to be used for a variety of activities. We'd like to be able to tune our model to the use case, just like the adjustable wrench for all fabric types. Because of that, transfer learning is used for feature extraction in this approach. For that purpose, it uses a model that has previously been trained on one problem to solve a problem that is similar to the new approach. Then feature extraction model in this solution can be used for all the fabric defect detection without training the feature extraction model for each fabric. The advantage is the system time requirement for the training process is lower and the system will more generalize for fabric defect detection.

The new approach uses VGG 16 for feature extraction. Because it has performed better than other models in fabric-related areas. Then VGG 16 model give a feature vector for each input image. Then feature vector sends into a fully connected layer to classify the image as defective or non-defective.

5.4 Fully connected network

The fully connected layer is used for the classification of feature vectors as defective or non-defective. At the beginning of the neural network, it uses wavelet transformation to transfer feature vectors for more accuracy. Then there are a few fully connected layers with batch normalization and dropout to increase the accuracy and efficiency of the system. At the end of the network, the new approach uses a new loss function for training the fully connected network.

5.5 Gradient ramp loss function

The hinge loss function is the basic loss function used in a one-class support vector machine. But there are a few problems as described in chapter 2. Then new loss function introduces for the fabric defect detection task. That loss function uses the ramp loss function concept too.

Hinge loss function with one class classifier loss function.

The following equation shows the OCSVM loss function with hinge loss and

$$min_{w,\rho} \frac{1}{2}||w||_2^2 + \frac{1}{vl} \sum_{i=1}^{l} H_\rho(g(x_i)) - \rho$$

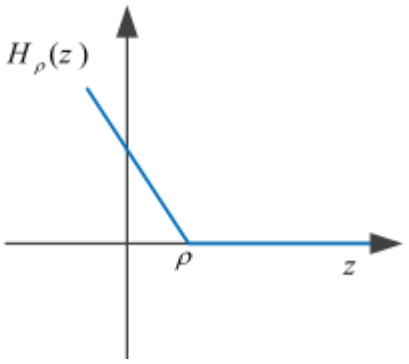


Figure 5.1: Hinge Loss Function

Ramp loss function

The Ramp loss function limit the effects of misclassification of the sample point to the new model weight updates. The following equation shows the OCSVM loss function with ramp loss and Figure 5.2 show the Ramp loss.

$$R_{p,s}(z) = \begin{cases} 0, & z \geq \rho \\ \rho - z, & \rho - s < z < \rho \\ s, & z \leq \rho - s \end{cases}$$

$$min_{w,\rho,s} \frac{1}{2}||w||_2^2 + \frac{1}{vl}\sum_{i=1}^{l} R_{\rho,s}(g(x_i)) - \rho$$

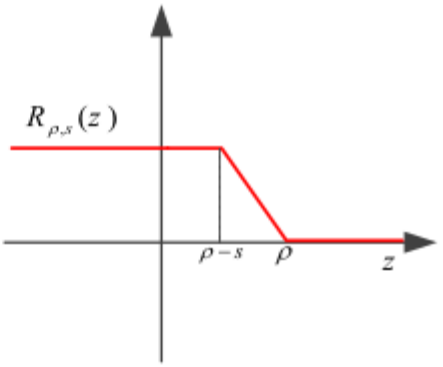


Figure 5.2: Ramp loss Function

Gradient ramp loss function with one-class classifier.

This loss function is used to replace hinge loss from the one-class classifier. The new loss function uses the property of ramp loss function and the new loss function introduce the gradient changing the property to get the handle of the loss function to the user. Then users have more controllability of the loss function and then the loss function can be changed as the controlling effects of defected to the training process.

When considering the loss values of miss classifications, it depends on the size of the defect that occurred in the fabric. When fabric always shows smaller defects, it may be not sufficient for the training process of the model. When the defect is larger, then the loss is larger. Then the loss value of the loss function should be fine-tuned with the defect otherwise system doesn't train perfectly according to the defects. Because

of that new loss function can be used to control the effect of defects on the training process. That new Los function is bounded, monotonic and non-convex.

The following equation is the loss function of novel gradient ramp OCSVM and it shows that hinge loss function is multiplied by gradient ratio to take controllability of the loss function. After that, that value applies to the ramp property.

$$\alpha = Gradient\ Ratio$$

$$min_{w,\rho,s} \frac{1}{2}||w||_2^2 + \frac{1}{vl}\sum_{i=1}^{l} R_{\rho,s}(\alpha \cdot g(x_i)) - \rho$$

According to the figure 5.3, it shows α value can change the gradient of the loss function and the ramp loss concept can change the upper boundary of the loss function. Then the user has the controllability of the loss function completely. After training the system with this model, we can get the final one-class classifier to test with sample data.

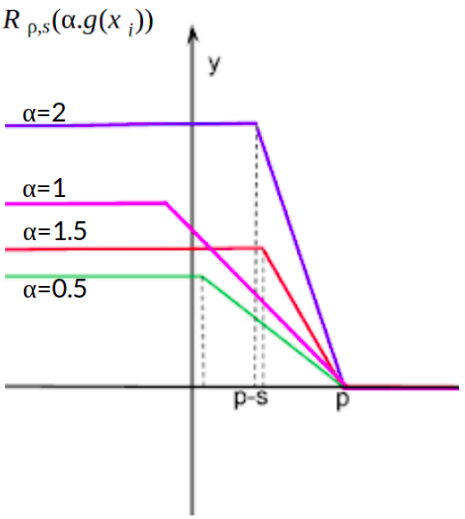


Figure 5.3: New gradient-Ramp loss function

5.5 Summary

This chapter gives an overview of the project's overall design elements. It is prolonged under several major topics of the system namely feature extraction network, fully connected network and gradient ramp loss function. Each module and

its sub-modules are described in detail in terms of the tasks they fulfil. The next chapter discusses in detail how each of these design elements is implemented and realized.

Chapter 6 IMPLEMENTATION

6.1 Introduction

The designs from the previous chapter are put into action in this chapter. Each module mentioned in the Design chapter is explored in the following elements, which include but are not limited to Software, Hardware, Algorithms, Flow Charts, and Code. Any special hardware components were not used for this project except the PC.

6.2 Datasets

Data is a deciding factor in the success of every Deep Learning-based project, the same as it is in any other. In this project, the main focus is on fabric defect detection. Hence it requires a fabric dataset.

MVTec Software GmbH is a leading company that manufacturer of software for machine vision. They have created the MVTec AD dataset [2] for benchmarking anomaly detection methods for industrial inspection in 2019. That dataset has a carpet dataset and it is suitable for a new approach. For more generalization process of the new approach, system train with few other classes in MVTec named as Cable, Transistor, Capsule, Hazelnut, leather, Tile and use UMDAA
face dataset.

Datasets:

MVTec AD - https://www.mvtec.com/company/research/datasets/mvtec-ad
UMDAA - https://umdaa02.github.io

6.3 Feature Extraction

The feature extraction process of the system uses the VGG 16 model. Keras API provides the facility of VGG 16. For the feature extraction, the system performs transfer learning with VGG 16. Figure 6.1 shows the feature extractor and classifier of VGG 16. For transfer learning purposes, the system remove the classification layer that was trained on the ImageNet dataset. Then set the model as non-trainable,

because it was well trained. Then VGG 16 give the output of the previous layer to the outside as the output of the model.

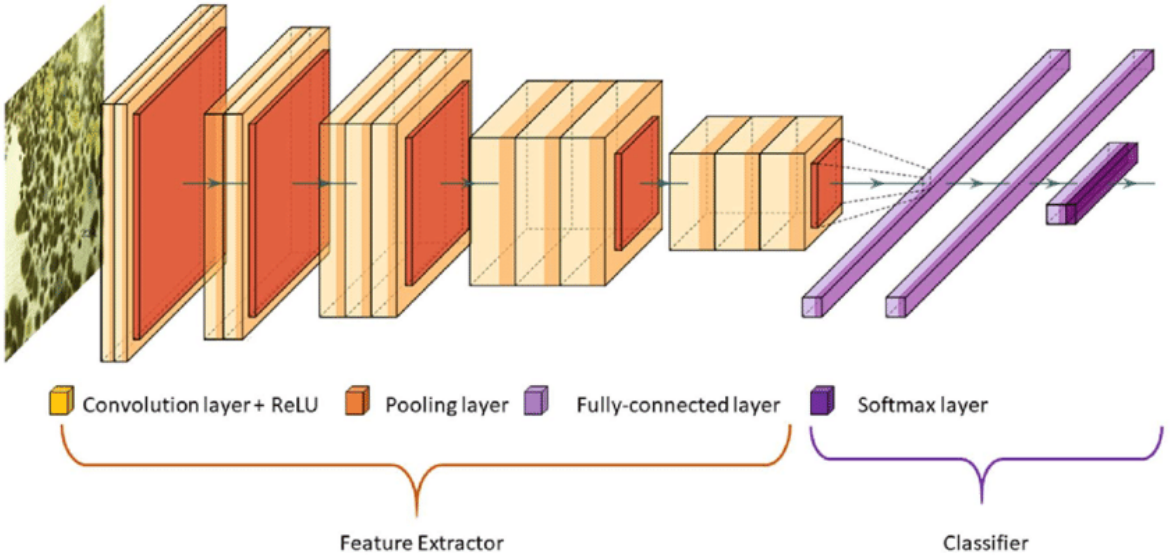


Figure 6.1: Feature Extractor and Classifier in VGG 16 Network

VGG 16 model output a feature vector with the size of 4096 for each input image. Then reshape each feature vector as 1x1x4096 to input to the fully connected network.

6.4 Fully connected network

The fully connected network is created using Keras API. It consists of 10 layers and at the start of the network. There is a wavelet transformation layer to transform the input features that came out from VGG 16 model.

The system uses discrete wavelet transformation with haar wavelet. Python has the facility of wavelet transformation through TensorFlow. For the get higher accuracy in fabric defect detection, the system had to use a custom wavelet. Then custom wavelet is created with a custom decomposition filter and reconstruction filter and it performs better than the default haar wavelet. Those filter values choose to like,

lowpass decomposition filter = [0.9, 0.9]

highpass decomposition filter = [-0.9, 0.9]

Lowpass reconstruction filter = [0.9, 0.9]

highpass reconstruction filter= [0.9, -0.9]

Generally, those values are replaced by $1/\sqrt{2}$. But 0.9 perform better than the default value for fabric data samples.

The output of the haar wavelet sends into the fully connected layer. During the fully connected layer, the system had to use batch normalization to normalize the hidden representations of the hidden layers. The dropout layer is also used inside the network to prevent over fittings. Dropout is placed in the middle area of the network to prevent overfitting without doing harm to the network performance. In this approach, it is used 0.5 is the dropout value.

Optimization of the model is done using adam optimizer. The learning rate of the network is very important for better accuracy and efficiency. Because of that 1e-5 learning rate is used for the training process of fabric data.

6.5 Gradient ramp loss function

Tensorflow Keras give the facility to create customized loss function for compilation. In the loss function, at first, it creates a hinge loss function that uses in basic OCSVM. In the hinge loss, it gives if the actual value is positive, get it as output, otherwise, it gives zero as output. Then the output is multiplied by the gradient factor. As an example, gradient values that are used by the system is 2, 1, 0.5, 0.02, 0.05 etc. Then output value is filtered by the ramp limits. As example, ramp limit values taken by the system are 0.5, 0.5, 0.01, 0.05 etc. Especially, in this practice, reduce the effect of weight values of the network and use the loss of miss classification. Figure 6.2 shows the operations inside the loss function.

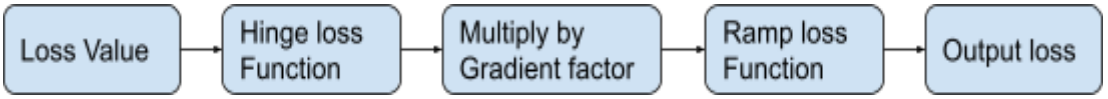


Figure 6.2: Operations of Loss Function.

6.6 Training

The training was conducted on a computer that has the following requirements.

Ram - 16 GB
CPU - Intel® Core™ i7-8750H CPU @ 2.20GHz × 12

The training process of the fully connected layer is done under the following configurations.

Batch size - 2
Epochs - 20
Dataset -
Learning rate - 0.00001

6.7 Summary

The current method classifies defective and non-defective fabrics images against the state of the art method. The experimental design is used to explain this complete evaluation, which starts with the approach and ends with the results. The results will support the overall conclusion, which will be discussed in the following chapter.

Chapter 7 EVALUATION

7.1 Introduction

The main goal of this chapter is to compare the novel one-class based fabric defect detection techniques presented in this study to state-of-the-art one-class based defect detection techniques as well as the widely utilized anomaly detection techniques. For the comparison purpose, not only limited to fabric datasets, it used other images based anomaly datasets for the generalization purpose of the novel technique. The following section provides a brief overview of different indices. For more information, methods are evaluated over image accuracies, train accuracy, training time is also evaluated.

In this chapter, the evaluation methodologies, experimental design, and evaluation results are presented.

7.2 Evaluations

In the evaluation process, three other approaches named OCSVM, Ramp-OCSVM, Robust-OCSVM along with the novel approach are evaluated. For the evaluation process, few datasets are used and the novel approach mainly focuses on a fabric dataset named MVTec carpet dataset. Other datasets are MVTech (cable, hazelnut, leather, tile, transistor) and UMDAA face dataset.

7.2.1 OCSVM

Basic OCSVM with hinge loss function is deployed with all the datasets and accuracy and time consumption is reported. But OCSVM shows an overfitting problem. During the training process, the training accuracy of the system is very high, But validation accuracy is very low. Because that loss functions is not suitable for this application. The following table shows the result.

Table 7.1: OCSVM results

Dataset	Training Accuracy	Validation Accuracy	Training loss	Validation loss
MVTech Carpet	1	0.6279	1.9926	121.2064
MVTech cable	1	0.6125	2.0625	155.2737
MVTech hazelnut	1	0.6679	2.0606	71.6962
MVTech leather	1	0.5185	2.0414	47.3693
MVTech tile	1	0.9057	2.0606	15.2445
MVTech transistor	1	0.8143	1.9871	20.7713
UMDAA face	0.7969	0.4629	16.4968	24.1011

7.2.2 Ramp-OSVM

As discussed in chapter 3, Ramp-OCSVM limits the normal OCSVM using ramp limit. Therefore, Ramp-OCSVM has been evaluated all the datasets with different ramp limits.

Table 7.2 Ramp-OCSVM results

	Accuracy at Ramp limit			
dataset	0.5	0.1	0.05	0.01
MVTech Carpet	0.6919	0.6303	0.6723	0.4230
MVTech cable	0.6667	0.6785	0.6585	0.5862
MVTeccapsule	0.5623	0.6423	0.6455	0.5625
MVTech hazelnut	0.7955	0.7438	0.7456	0.7085
MVTech leather	0.7460	0.673	0.6852	0.6145
MVTech tile	0.6871	0.6667	0.6695	0.5963
MVTech transistor	0.7819	0.5720	0.6523	0.6245
UMDAA face	0.7562	0.7456	0.7012	0.7152

7.2.3 Robust-OCSVM

Robust OCSVM introduce beta multiplier for OCSVM hinge liss for applying monotonic, non-convex and bounded property. Beta value is generated using alpha value, therefore, evaluation of Robust OCSVM have done with the variation of alpha.

Table 7.3 Robust-OCSVM results

dataset	Accuracy
MVTech Carpet	0.2129
MVTech cable	0.6837
MVTech capsule	0.7284
MVTech hazelnut	0.7146
MVTech leather	0.9143
MVTech tile	0.8776
MVTech transistor	0.8971
UMDAA face	0.5793

7.2.4 Grad-ramp OCSVM

Novel approach is an extension of ramp OCSVM with gradient property. As discussed in chapter 4, the new approach has two controllability of loss functions as gradient ratio and ramp limit. Then the evaluation process was done for different values of gradient and ramp parameters.

Table 7.4 Grad-Ramp-OCSVM results with 0.5 ramp limit

	Accuracy at Ramp = 0.5			
dataset	Gradient = 0.5	Gradient = 0.1	Gradient= 0.05	Gradient = 0.02
MVTech Carpet	0.6919	0.7311	0.7227	0.7283
MVTech cable	0.7015	0.7236	0.7014	0.7045
MVTech hazelnut	0.7542	0.7615	0.7588	0.7489
MVTech leather	0.7823	0.7951	0.7944	0.7645
MVTech tile	0.7625	0.7899	0.7755	0.7741
MVTech transistor	0.7745	0.7455	0.7541	0.7469
UMDAA face	0.8015	0.8044	0.7999	0.7688

Table 7.5 Grad-Ramp-OCSVM results with 0.1 ramp limit

	Accuracy at Ramp = 0.1			
dataset	Gradient = 0.5	Gradient = 0.1	Gradient= 0.05	Gradient = 0.02
MVTech Carpet	0.6527	0.7877	0.7073	0.7288
MVTech cable	0.7125	0.7952	0.6955	0.6589
MVTech hazelnut	0.6954	0.7758	0.7256	0.7365
MVTech leather	0.7122	0.8015	0.7541	0.7844
MVTech tile	0.6952	0.7256	0.7088	0.7799
MVTech transistor	0.6799	0.6988	0.7259	0.7589
UMDAA face	0.8011	0.7785	0.7584	0.8055

Table 7.6 Grad-Ramp-OCSVM results with 0.05 ramp limit

	Accuracy at Ramp = 0.05			
dataset	Gradient = 0.5	Gradient = 0.1	Gradient= 0.05	Gradient = 0.02
MVTech Carpet	0.7901	0.7927	0.7899	0.5714
MVTech cable	0.8122	0.8255	0.8014	0.6658
MVTech hazelnut	0.8202	0.8145	0.8094	0.7125
MVTech leather	0.7985	0.800	0.7825	0.7012
MVTech tile	0.7655	0.7925	0.7698	0.6689
MVTech transistor	0.7755	0.7723	0.7568	0.7356
UMDAA face	0.8258	0.7982	0.7952	0.7685

7.2.5 Method comparison

According to result of OCSVM, Table 7.1 shows overfitting problem. Loss during the training process also very high. OCSVM loss function doesn't able to handle overfitting problem from of training process. Robust-OCSVM also tend to overfitting

problem. Training accuracy is high. But validation accuracy and test accuracy is very low. Table 7.3 show result without overfitting. But accuracy is low compared to ramp and Grad-ramp-OCSVM approaches.

By comparing ramp-OCSVM and novel approach called Grad-Ramp-OCSVM,novel method can change the output of ramp-OCSVM by changing gradient of loss function. Using that property, as shown in Table 7.4, 7.5 and 7.6 , shows higher accuracy than other methods. By considering ramp values, tables shows, ramp value make a huge effect on accuracy value in model.

7.3 Summary

The novel method of fabric defect detection is evaluated against to existing image-based anomaly detection methods. The results of each approach are shown in tables for each approach and according to the comparison, the new approach shows better performance than the existing approach in the fabric defect detection field. These evaluated results and comparisons will support the overall conclusion in the next chapter.

Chapter 8 CONCLUSION & FURTHER WORK

8.1 Introduction

This research introduced a novel integration of a one-class classifier approach for anomaly detection. That new approach has been deployed on image-based defect detection applications. Moreover, several one-class classifiers are previously introduced for anomaly detection were also brought up. This research was done highlighting fabric defect detection as anomaly detection and a few other image-based defect detection datasets are used for more generalization purpose of the approach. The previous chapter presented the evaluation of the new approach and with the help of the information supplied in the evaluation chapter, this chapter will conclude the dissertation by highlighting the achievement of the objectives, derived conclusions, and future work to be done in this field of research.

8.2 Conclusion

8.2.1 Achievement of Project Objectives

The overall project approach was based on attaining the objective set forth in the introduction chapter. All of the objectives were achieved as planned by following the proper project approach.

Following a comprehensive and thorough literature review, various weaknesses of current approaches were found, as well as attempts to overcome these problems. By spotting current methods in fabric defect detection, these methods tend to anomaly detection fields. As is mentioned in chapter 3, defects in fabrics are unpredictable. Therefore it is easy to approach that problem considering it as an anomaly detection approach.

According to the second objective of the research, image-based defect detection and anomaly detention technologies are studied and integrated to achieve fabric defect detection using the one-class classifier. Several methods that are performed well are chosen for development in the new approach integrated them to get higher performance.

As indicated in chapter 7 implementations, all methods were evaluated on each dataset and several standard methods are used for that purpose. Furthermore, each model's training time and parameter analysis were carried out, providing more information about this new approach.

According to the last objective of the research, novel approach results have higher accuracy and it is more appropriate for fabric defect detection application.

8.2.2 Overall Conclusion

The fabric defect detection concept is a difficult task to perform as a multiclass classifier, because of the variation of the defect. Therefore there are many approaches to perform that task as a one-class classifier as taking the project as anomaly detection. Existing approaches haven't been generated a higher accurate approach along with avoiding the defective data image requirement.

Anomaly detection with OCSVM is the basic method and there are many variations of OCSVM for anomaly detection. Those methods have higher accuracy for relevant approach data, but not in the image-based datasets.

The new approach shows higher accuracy than the existing one-class classifier methods in the image-based defeat detection Applications. For fabric defect detection, bounded and non-convex loss function is needed due to the nature of the defects. The new approach provides that facility with good controllability of loss function.

8.3 Limitations and Further Works

The feature extraction model of the dataset that was used is a pre-trained VGG 16 model. When training the VGG 16 they have used the ImageNet dataset. Therefore VGG 16 may not perform higher efficiency in the fabric defect detection field. It will be more accurate and efficient if use a model feature extraction that is trained using a fabric image dataset.

The gradient factor directly affects the loss value of miss-classifications, but at the beginning of the model generation, the user has to define the gradient value. For that purpose, the user has to analyse the loss values of each dataset and define a value.

8.4 Summary

This chapter concludes the thesis by describing the extent to which the objectives were achieved, the overall conclusion, as well as the limitations and future work that has to be done. A few side tasks were identified in order to further study the fabric defect detection topic and improve the existing method more efficient.

Reference

[1] A. Rasheed *et al.*, “Fabric Defect Detection Using Computer Vision Techniques: A Comprehensive Review,” *Math. Probl. Eng.*, vol. 2020, pp. 1–24, Nov. 2020, doi: 10.1155/2020/8189403.

[2] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger, “MVTec AD — A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, Jun. 2019, pp. 9584–9592. doi: 10.1109/CVPR.2019.00982.

[3] M. Zhang, J. Wu, H. Lin, P. Yuan, and Y. Song, “The Application of One-Class Classifier Based on CNN in Image Defect Detection,” *Procedia Comput. Sci.*, vol. 114, pp. 341–348, 2017, doi: 10.1016/j.procs.2017.09.040.

[4] P. Perera and V. M. Patel, “Learning Deep Features for One-Class Classification,” *IEEE Trans. Image Process.*, vol. 28, no. 11, pp. 5450–5463, Nov. 2019, doi: 10.1109/TIP.2019.2917862.

[5] L. Ruff *et al.*, “Deep One-Class Classification,” p. 10.

[6] L. Weninger, M. Kopaczka, and D. Merhof, “Defect detection in plain weave fabrics by yarn tracking and fully convolutional networks,” in *2018 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, Houston, TX, USA, May 2018, pp. 1–6. doi: 10.1109/I2MTC.2018.8409546.

[7] H.-J. Xing and M. Ji, “Robust one-class support vector machine with rescaled hinge loss function,” *Pattern Recognit.*, vol. 84, pp. 152–164, Dec. 2018, doi: 10.1016/j.patcog.2018.07.015.

[8] Y. Tian, M. Mirzabagheri, S. M. H. Bamakan, H. Wang, and Q. Qu, “Ramp loss one-class support vector machine; A robust and effective approach to anomaly detection problems,” *Neurocomputing*, vol. 310, pp. 223–235, Oct. 2018, doi: 10.1016/j.neucom.2018.05.027.

[9] S. K. Sonbhadra, S. Agarwal, and P. Nagabhushan, “Pinball-OCSVM for early-stage COVID-19 diagnosis with limited posteroanterior chest X-ray images,” *ArXiv201008115 Cs Eess*, Jun. 2021, Accessed: Jul. 26, 2021. [Online]. Available: http://arxiv.org/abs/2010.08115

[10] I. Razzak and T. M. Khan, “One-Class Support Tensor Machines with Bounded Hinge Loss Function for Anomaly Detection,” in *2020 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2020, pp. 1–8. doi: 10.1109/IJCNN48605.2020.9207429.

[11] X. E. Pantazi, D. Moshou, and A. A. Tamouridou, “Automated leaf disease detection in different crop species through image features analysis and One Class Classifiers,” *Comput. Electron. Agric.*, vol. 156, pp. 96–104, Jan. 2019, doi: 10.1016/j.compag.2018.11.005.

[12] L. Ruff *et al.*, “Deep Semi-Supervised Anomaly Detection,” *ArXiv190602694 Cs Stat*, Feb. 2020, Accessed: Jul. 11, 2021. [Online]. Available: http://arxiv.org/abs/1906.02694

[13] B. Lamrini, A. Gjini, S. Daudin, F. Armando, P. Pratmarty, and L. Travé-Massuyès, “Anomaly Detection Using Similarity-based One-Class SVM for Network Traffic Characterization,” p. 8.

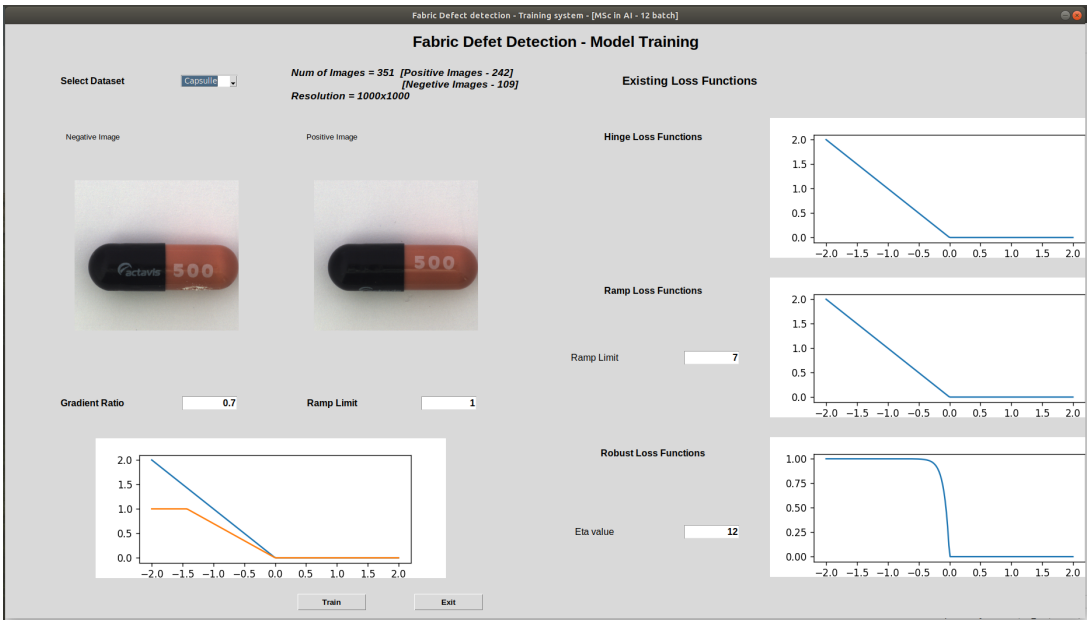
[14] R. B. Zhang, L. H. Xia, and Y. Lu, “Anomaly Detection of ICS based on EB-OCSVM,” *J. Phys. Conf. Ser.*, vol. 1267, p. 012054, Jul. 2019, doi: 10.1088/1742-6596/1267/1/012054.

[15] K. Kittidachanan, W. Minsan, D. Pornnopparath, and P. Taninpong, “Anomaly

Detection based on GS-OCSVM Classification," in *2020 12th International Conference on Knowledge and Smart Technology (KST)*, Pattaya, Chonburi, Thailand, Jan. 2020, pp. 64–69. doi: 10.1109/KST48564.2020.9059326.
[16] H.-J. Xing and L.-F. Li, "Robust least squares one-class support vector machine," *Pattern Recognit. Lett.*, vol. 138, pp. 571–578, Oct. 2020, doi: 10.1016/j.patrec.2020.09.005.
[17] J. Wang and A. Cherian, "GODS: Generalized One-Class Discriminative Subspaces for Anomaly Detection," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea (South), Oct. 2019, pp. 8200–8210. doi: 10.1109/ICCV.2019.00829.
[18] Tomas Sarmiento, Sang J. Hong, and Gary S. May, "Fault detection in reactive ion etching systems using one-class support vector machines," in *IEEE/SEMI Conference and Workshop on Advanced Semiconductor Manufacturing 2005.*, Munich, Germany, 2005, pp. 139–142. doi: 10.1109/ASMC.2005.1438783.
[19] B. Gu, X. Quan, Y. Gu, V. S. Sheng, and G. Zheng, "Chunk incremental learning for cost-sensitive hinge loss support vector machine," *Pattern Recognit.*, vol. 83, pp. 196–208, Nov. 2018, doi: 10.1016/j.patcog.2018.05.023.
[20] I. Razzak, K. Zafar, M. Imran, and G. Xu, "Randomized nonlinear one-class support vector machines with bounded loss function to detect of outliers for large scale IoT data," *Future Gener. Comput. Syst.*, vol. 112, pp. 715–723, Nov. 2020, doi: 10.1016/j.future.2020.05.045.
[21] M. Singla, D. Ghosh, and K. K. Shukla, "Improved Sparsity of Support Vector Machine with Robustness Towards Label Noise Based on Rescaled $$\alpha $$-Hinge Loss with Non-smooth Regularizer," *Neural Process. Lett.*, vol. 52, no. 3, pp. 2211–2239, Dec. 2020, doi: 10.1007/s11063-020-10346-0.
[22] X. Jun, J. Wang, J. Zhou, S. Meng, R. Pan, and W. Gao, "Fabric defect detection based on a deep convolutional neural network using a two-stage strategy," *Text. Res. J.*, vol. 91, no. 1–2, pp. 130–142, Jan. 2021, doi: 10.1177/0040517520935984.
[23] P. Oza and V. M. Patel, "One-Class Convolutional Neural Network," *IEEE Signal Process. Lett.*, vol. 26, no. 2, pp. 277–281, Feb. 2019, doi: 10.1109/LSP.2018.2889273.
[24] T. Wang, Y. Chen, M. Qiao, and H. Snoussi, "A fast and robust convolutional neural network-based defect detection model in product quality control," *Int. J. Adv. Manuf. Technol.*, vol. 94, no. 9–12, pp. 3465–3471, Feb. 2018, doi: 10.1007/s00170-017-0882-0.

Appendix

Appendic I: User Interface



Appendic II: Main programe to train the model

```
import getopt
import math
import os
import pickle
import sys
import cv2
import pywt
from datetime import datetime
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.layers.experimental import RandomFourierFeatures
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.applications.vgg16 import preprocess_input
from keras.applications.vgg16 import decode_predictions
from keras.applications.vgg16 import VGG16
from keras.models import Model
import joblib
import dill
import weakref
from wavetf import WaveTFFactory
import tensorflow_wavelets.Layers.DWT as DWT
import tensorflow_wavelets.Layers.DTCWT as DTCWT
import tensorflow_wavelets.Layers.DMWT as DMWT
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
```

```
from sklearn.metrics import cohen_kappa_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import confusion_matrix
import numpy as np
import tensorflow as tf
import tensorflow.keras.backend as K
import matplotlib.pyplot as plt

dataset = ''
gradient = 0.0
ramp = 0.0

def history_(history):
   print(history.history.keys())
   plt.plot(history.history['accuracy'])
   plt.plot(history.history['val_accuracy'])
   plt.title('model accuracy')
   plt.ylabel('accuracy')
   plt.xlabel('epoch')
   plt.legend(['train', 'test'], loc='upper left')
   plt.show()
   plt.plot(history.history['loss'])
   plt.plot(history.history['val_loss'])
   plt.title('model loss')
   plt.ylabel('loss')
   plt.xlabel('epoch')
   plt.legend(['train', 'test'], loc='upper left')
   plt.show()

# LOSS FUNCTION 1 - SVM Loss
def svm_loss(layer):
   weights = layer.weights[0]
   weights_tf = tf.convert_to_tensor(weights)

   def categorical_hinge_loss(y_true, y_pred):
       pos = K.sum(y_true * y_pred, axis=-1)
       neg = K.max((1.0 - y_true) * y_pred, axis=-1)
       hinge_loss = K.mean(K.maximum(0.0, neg - pos + 1), axis=-1)
       regularization_loss = 0.00001 * (tf.reduce_sum(tf.square(weights_tf)))
       return regularization_loss + hinge_loss
   return categorical_hinge_loss

# LOSS FUNCTION 2 - Ramp Loss
def svm_ramp_loss(layer):
   weights = layer.weights[0]
   weights_tf = tf.convert_to_tensor(weights)

   def categorical_hinge_loss(y_true, y_pred):
       pos = K.sum(y_true * y_pred, axis=-1)
       neg = K.max((1.0 - y_true) * y_pred, axis=-1)
       hinge_loss = K.mean(K.maximum(0.0, neg - pos + 1), axis=-1)
       hinge_loss_ramp = K.minimum(0.05,hinge_loss)
       regularization_loss = 0.00001 * (tf.reduce_sum(tf.square(weights_tf)))
       return regularization_loss +  hinge_loss_ramp
   return categorical_hinge_loss

# LOSS FUNCTION 3 - Robust Loss
def svm_robust_loss(layer):
   weights = layer.weights[0]
   weights_tf = tf.convert_to_tensor(weights)
```

```
    def categorical_hinge_loss(y_true, y_pred):
        pos = K.sum(y_true * y_pred, axis=-1)
        neg = K.max((1.0 - y_true) * y_pred, axis=-1)
        hinge_loss = K.mean(K.maximum(0.0, neg - pos + 1), axis=-1)
        eta = 0.05
        beta = 1/(1 - K.exp(-1 * eta))
        hinge_loss_robust = beta*(1 - K.exp(-1*eta*hinge_loss))
        regularization_loss = 0.00001 * (tf.reduce_sum(tf.square(weights_tf)))
        return regularization_loss +  hinge_loss_robust
    return categorical_hinge_loss

# LOSS FUNCTION 4 -  Robust + Ramp Loss
def svm_robust_with_ramp_loss(layer):
    weights = layer.weights[0]
    weights_tf = tf.convert_to_tensor(weights)

    def categorical_hinge_loss(y_true, y_pred):
        pos = K.sum(y_true * y_pred, axis=-1)
        neg = K.max((1.0 - y_true) * y_pred, axis=-1)
        hinge_loss = K.mean(K.maximum(0.0, neg - pos + 1), axis=-1)
        eta = 0.005
        ramp_limit = 8.0
        beta = 1/(1 - K.exp(-1 * eta))
        hinge_loss_robust = beta*(1 - K.exp(-1*eta*hinge_loss))
        hinge_loss_robust_with_ramp = K.minimum(ramp_limit, hinge_loss_robust)
        regularization_loss = 0.00001 * (tf.reduce_sum(tf.square(weights_tf)))
        return regularization_loss +  hinge_loss_robust_with_ramp
    return categorical_hinge_loss

# LOSS FUNCTION 5 - Upper Ramp Loss
def svm_upper_ramp_loss(layer):
    weights = layer.weights[0]
    weights_tf = tf.convert_to_tensor(weights)

    def categorical_hinge_loss(y_true, y_pred):
        pos = K.sum(y_true * y_pred, axis=-1)
        neg = K.max((1.0 - y_true) * y_pred, axis=-1)
        hinge_loss = K.mean(K.maximum(0.0, neg - pos + 1), axis=-1)
        hinge_loss_ramp = K.minimum(0.05,hinge_loss)
        if hinge_loss_ramp >0:
            hinge_loss_ramp = hinge_loss_ramp + 0.005
        regularization_loss = 0.00001 * (tf.reduce_sum(tf.square(weights_tf)))
        return regularization_loss +  hinge_loss_ramp
    return categorical_hinge_loss

# LOSS FUNCTION 6 - gradient Ramp Loss
def svm_gradient_ramp_loss(layer):
    weights = layer.weights[0]
    weights_tf = tf.convert_to_tensor(weights)

    def categorical_hinge_loss(y_true, y_pred):
        pos = K.sum(y_true * y_pred, axis=-1)
        neg = K.max((1.0 - y_true) * y_pred, axis=-1)
        grad_ratio = gradient #0.02
        print("grad ratio",grad_ratio)
        print("ramp",ramp)
        hinge_loss = grad_ratio * K.mean(K.maximum(0.0, neg - pos + 1), axis=-1)
        hinge_loss_ramp = K.minimum(ramp,hinge_loss)
        regularization_loss = 0.00001 * (tf.reduce_sum(tf.square(weights_tf)))
```

```
        return regularization_loss +   hinge_loss_ramp
    return categorical_hinge_loss

def wavelet_activation(x):
    cwtmatr, freqs = pywt.dwt(x,'db1')
    return cwtmatr

# create SVM model
def model_():
    model = keras.Sequential(
        [
            keras.Input(shape=(1,1,4096)),
            DWT.DWT(name = 'haar',concat=1),
            layers.Flatten(),
            layers.Dense(units=4096),
            layers.Dense(units=8192),
            layers.Dense(units=8192),
            layers.BatchNormalization(),
            layers.Dense(units=4096),
            layers.BatchNormalization(),
            layers.Dropout(0.5),
            layers.Dense(units=2048),
            layers.Dense(units=512),
            layers.Dense(units=128),

            layers.Dense(units=32),
            layers.Dense(units=2, name='svm'),
        ]
    )
    print(model.summary())
    optimizer = keras.optimizers.Adam(learning_rate=1e-5)
    loss = svm_gradient_ramp_loss(model.get_layer('svm'))
    print("svm_gradient_ramp_loss")

    model.compile(
        optimizer=optimizer,
        loss=loss,
        metrics=['accuracy']
    )
    return model

# Extract features from VGG16
def feature_extract(images_processed_):
    feature_list = []
    # load model
    model = VGG16()
    # remove the output layer
    model = Model(inputs=model.inputs, outputs=model.layers[-2].output)
    # get extracted features
    for image_prop in images_processed_:
        features = model.predict(image_prop)
        feature_list.append(np.array(features[0]).reshape(1,1,4096))

    return np.array(feature_list)

# pre process image for VGG16
def preprocess_(image):
    # convert the image pixels to a numpy array
    image = img_to_array(image)
    # reshape data for the model
```

```
    image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
    # prepare the image for the VGG model
    image = preprocess_input(image)
    return image

# Read images from train and test directory
def get_data(data_dir,labels):
    data = []
    for label in labels:
        path = os.path.join(data_dir, label)
        class_num = labels.index(label)
        for img in os.listdir(path):
            try:
                data.append([load_img(os.path.join(path, img), target_size=(224, 224)),
class_num])
            except Exception as e:
                print(e)
    return data

def load_data(train_path,test_path,labels):
    train = get_data(train_path,labels)
    test = get_data(test_path,labels)
    x_train_ = []
    y_train_ = []
    x_test_ = []
    y_test_ = []
    for feature, label in train:
        x_train_.append(preprocess_(feature))
        y_train_.append(label)

    for feature, label in test:
        x_test_.append(preprocess_(feature))
        y_test_.append(label)

    y_train_ = np.array(y_train_)
    y_test_ = np.array(y_test_)
    num_classes = 2
    y_test_ = tf.keras.utils.to_categorical(y_test_, num_classes)
    y_train_ = tf.keras.utils.to_categorical(y_train_, num_classes)
    return x_train_, x_test_, y_train_, y_test_

def main(argv):
    global dataset,gradient,ramp
    try:
        opts, args = getopt.getopt(argv, "hd:g:r:", ["dataset=", "gradient=", "ramp="])
    except getopt.GetoptError:
        print("Error")
        print('test.py -d <dataset> -g <gradient> -r <ramp limit>')
        sys.exit(2)
    for opt, arg in opts:
        if opt == '-h':
            print('test.py -d <dataset> -g <gradient> -r <ramp limit>')
            sys.exit()
        elif opt in ("-d", "--dataset"):
            dataset = arg
        elif opt in ("-g", "--gradient"):
            gradient = float(arg)
        elif opt in ("-r", "--ramp"):
            ramp = float(arg)
```

```
    if dataset == "Carpet":
        # carpet
        train_path_ = '/home/nuwan/Applications/AI_cource/research/data/carpet/data/train'
        test_path_ = '/home/nuwan/Applications/AI_cource/research/data/carpet/data/test'
        labels = ['defect', 'nondefect']
        print("carpet")
    elif dataset == "UMDAA":
        # # UMDAA
        train_path_ =
'/home/nuwan/Applications/AI_cource/research/data/umdaa02-fd-20211017T053249Z-001/new_data/
train'
        test_path_ =
'/home/nuwan/Applications/AI_cource/research/data/umdaa02-fd-20211017T053249Z-001/new_data/
test'
        labels = ['face', 'nonface']
        print("UMDAA")
    elif dataset == "Capsulle":
        # capsulle
        train_path_ = '/home/nuwan/Applications/AI_cource/research/data/capsule/data/train'
        test_path_ = '/home/nuwan/Applications/AI_cource/research/data/capsule/data/test'
        labels = ['defect', 'nondefect']
        print("capsulle")
    elif dataset == "Cable":
        # cable
        train_path_ = '/home/nuwan/Applications/AI_cource/research/data/cable/data/train'
        test_path_ = '/home/nuwan/Applications/AI_cource/research/data/cable/data/test'
        labels = ['defect', 'nondefect']
        print("cable")
    elif dataset == "Hazelnut":
        # hazelnut
        train_path_ = '/home/nuwan/Applications/AI_cource/research/data/hazelnut/data/train'
        test_path_ = '/home/nuwan/Applications/AI_cource/research/data/hazelnut/data/test'
        labels = ['defect', 'nondefect']
        print("hazelnut")
    elif dataset == "Leather":
        # leather
        train_path_ = '/home/nuwan/Applications/AI_cource/research/data/leather/data/train'
        test_path_ = '/home/nuwan/Applications/AI_cource/research/data/leather/data/test'
        labels = ['defect', 'nondefect']
        print("leather")
    elif dataset == "Tile":
        # tile
        train_path_ = '/home/nuwan/Applications/AI_cource/research/data/tile/data/train'
        test_path_ = '/home/nuwan/Applications/AI_cource/research/data/tile/data/test'
        labels = ['defect', 'nondefect']
        print("tile")

    elif dataset == "Transistor":
        # transistor
        train_path_ =
'/home/nuwan/Applications/AI_cource/research/data/transistor/data/train'
        test_path_ = '/home/nuwan/Applications/AI_cource/research/data/transistor/data/test'
        labels = ['defect', 'nondefect']
        print("transistor")
    else:
        print("ERROR - datset name input is not correct")
        exit()
    x_train, x_test, y_train, y_test = load_data(train_path_,test_path_,labels)
    x_train = feature_extract(x_train)
    x_test = feature_extract(x_test)
```

```
    model = model_()
    history = model.fit(x_train, y_train, epochs=20,
batch_size=20,validation_data=(x_test,y_test))
    date = datetime.now().strftime("%Y_%m_%d-%I:%M:%S_%p")
    filename = f'models/Trained_model_'+dataset+'.sav'

    model.save_weights("ckpt")

    yhat_probs = model.predict(x_test, verbose=0)
    yhat_classes = np.argmax(yhat_probs, axis=1)
    y_test = y_test[:, 1]
    accuracy = accuracy_score(y_test, yhat_classes)
    print('Accuracy: %f' % accuracy)
    precision = precision_score(y_test, yhat_classes)
    print('Precision: %f' % precision)
    recall = recall_score(y_test, yhat_classes)
    print('Recall: %f' % recall)
    f1 = f1_score(y_test, yhat_classes)
    print('F1 score: %f' % f1)
    kappa = cohen_kappa_score(y_test, yhat_classes)
    print('Cohens kappa: %f' % kappa)
    matrix = confusion_matrix(y_test, yhat_classes)
    print(matrix)

if __name__ == "__main__":
    main(sys.argv[1:])
```

Appendic III: User Interface Program

```
import math
import os
from tkinter import *
from tkinter import ttk
from PIL import ImageTk,Image
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import (FigureCanvasTkAgg,NavigationToolbar2Tk)
import numpy as np
dataset = "tile"
r = 0.5
g = 0.85

command = "python3 main_script.py -d "+ dataset + " -g " + str(g) + " -r " + str(r)
print(command)

def print_cbox(event):
    country = event.widget.get()
    print(country)

def cbox():
    print(datasetchoosen.current(), datasetchoosen.get())
    print("grad ",e_grad.get())
    print("ramp ",e_ramp.get())
    os.system("python3 main_script.py -d "+str(datasetchoosen.get())+" -g
"+str(e_grad.get())+" -r "+str(e_ramp.get()))
```

```
    print("python3 main_script.py -d "+str(datasetchoosen.get())+" -g "+str(e_grad.get())+"
-r "+str(e_ramp.get()))
def load_images():
    global
img,Carpet_img_n,Carpet_img_d,UMDAA_img_d,UMDAA_img_n,capsulle_img_d,capsulle_img_n,cable_i
mg_d,cable_img_n
    global
hazelnut_img_d,hazelnut_img_n,leather_img_d,leather_img_n,tile_img_d,tile_img_n,transistor_
img_d,transistor_img_n

    img = PhotoImage(file="images/processed/Adefault.png")

    Carpet_img_d = PhotoImage(file="images/processed/Acarpet_d.png")
    Carpet_img_n = PhotoImage(file="images/processed/Acarpet_n.png")

    UMDAA_img_d = PhotoImage(file="images/processed/uaamd_d.png")
    UMDAA_img_n = PhotoImage(file="images/processed/uaamd_n.png")

    capsulle_img_d = PhotoImage(file="images/processed/Acapsule_d.png")
    capsulle_img_n = PhotoImage(file="images/processed/Acapsule_n.png")

    cable_img_d = PhotoImage(file="images/processed/Acable_d.png")
    cable_img_n = PhotoImage(file="images/processed/Acable_n.png")

    hazelnut_img_d = PhotoImage(file="images/processed/Ahazelnut_d.png")
    hazelnut_img_n = PhotoImage(file="images/processed/Ahazelnut_n.png")

    leather_img_d = PhotoImage(file="images/processed/Aleather_d.png")
    leather_img_n = PhotoImage(file="images/processed/Aleather_n.png")

    tile_img_d = PhotoImage(file="images/processed/Atile_d.png")
    tile_img_n = PhotoImage(file="images/processed/Atile_n.png")

    transistor_img_d = PhotoImage(file="images/processed/Atransistor_d.png")
    transistor_img_n = PhotoImage(file="images/processed/Atransistor_n.png")

def TextBoxUpdate(event):
    print(datasetchoosen.current(), datasetchoosen.get())
    if datasetchoosen.get() == ' Carpet':
        l_dataset_details.config(text="Num of Images = 400  [Positive Images - 308]\n\t\t
[Negetive Images - 92]\nResolution = 1024x1024")
        c_defect_image.itemconfig(c_d_container,image = Carpet_img_d)
        c_nondefect_image.itemconfig(c_n_container,image = Carpet_img_n)
    elif datasetchoosen.get() == ' UMDAA':
        l_dataset_details.config(text="Num of Images = 32135  [Positive Images -
23196]\n\t\t      [Negetive Images - 8939]\nResolution = 960x1280")
        c_defect_image.itemconfig(c_d_container,image = UMDAA_img_d)
        c_nondefect_image.itemconfig(c_n_container,image = UMDAA_img_n)
    elif datasetchoosen.get() == ' Capsulle':
        l_dataset_details.config(text="Num of Images = 351  [Positive Images - 242]\n\t\t
[Negetive Images - 109]\nResolution = 1000x1000")
        c_defect_image.itemconfig(c_d_container,image = capsulle_img_d)
        c_nondefect_image.itemconfig(c_n_container,image = capsulle_img_n)
    elif datasetchoosen.get() == ' Cable':
        l_dataset_details.config(text="Num of Images = 374  [Positive Images - 282]\n\t\t
[Negetive Images - 92]\nResolution = 1024x1024")
        c_defect_image.itemconfig(c_d_container,image = cable_img_d)
        c_nondefect_image.itemconfig(c_n_container,image = cable_img_n)
    elif datasetchoosen.get() == ' Hazelnut':
        l_dataset_details.config(text="Num of Images = 401  [Positive Images - 331]\n\t\t
```

```
[Negetive Images - 70]\nResolution = 1024x1024")
        c_defect_image.itemconfig(c_d_container,image = hazelnut_img_d)
        c_nondefect_image.itemconfig(c_n_container,image = hazelnut_img_n)
    elif datasetchoosen.get() == ' Leather':
        l_dataset_details.config(text="Num of Images = 378  [Positive Images - 286]\n\t\t
[Negetive Images - 92]\nResolution = 1024x1024")
        c_defect_image.itemconfig(c_d_container,image = leather_img_d)
        c_nondefect_image.itemconfig(c_n_container,image = leather_img_n)
    elif datasetchoosen.get() == ' Tile':
        l_dataset_details.config(text="Num of Images = 347  [Positive Images - 263]\n\t\t
[Negetive Images - 84]\nResolution = 840x840")
        c_defect_image.itemconfig(c_d_container,image = tile_img_d)
        c_nondefect_image.itemconfig(c_n_container,image = tile_img_n)
    elif datasetchoosen.get() == ' Transistor':
        l_dataset_details.config(text="Num of Images = 313  [Positive Images - 273]\n\t\t
[Negetive Images - 40]\nResolution = 1024x1024")
        c_defect_image.itemconfig(c_d_container,image = transistor_img_d)
        c_nondefect_image.itemconfig(c_n_container,image = transistor_img_n)

def exit():
    master.quit()

def change_graph_grad_sv(sv):
    print(sv.get())
    if sv.get():
        grad_val = float(sv.get())
    else:
        grad_val = 0

    if e_ramp.get():
        ramp_val = float(e_ramp.get())
    else:
        ramp_val = 0
    print("ramp val = ",ramp_val)
    # z = [(i ** 2) * 1.1 for i in range(101)]
    inner = [-1*i*grad_val if i < 0 else 0 for i in x]
    z = [min(i,ramp_val) for i in inner]

    graph_plot1.clear()
    graph_plot1.plot(x, y)
    graph_plot1.plot(x,z)
    c_graph.draw()

def change_graph_ramp_sv(sv):
    print(sv.get())
    if sv.get():
        grad_val = float(e_grad.get())
    else:
        grad_val = 0
    if e_ramp.get():
        ramp_val = float(sv.get())
    else:
        ramp_val = 0
    print("ramp val = ",ramp_val)
    # z = [(i ** 2) * 1.1 for i in range(101)]
    inner = [-1*i*grad_val if i < 0 else 0 for i in x]
    z = [min(i,ramp_val) for i in inner]

    graph_plot1.clear()
    graph_plot1.plot(x, y)
```

```
    graph_plot1.plot(x,z)
    c_graph.draw()

def e_robust_val_graph_eta_sv(sv):
    print(sv.get())
    if sv.get():
        eta_val = float(sv.get())
    else:
        eta_val = 0

    beta_val = 1 / (1 - math.exp(-1 * eta_val))

    y = [beta_val * (1 - math.exp(-1 * eta_val * (-1*i))) if i < 0 else 0 for i in x]

    graph_plot1_robust.clear()
    graph_plot1_robust.plot(x, y)

    c_graph_robust.draw()

def e_ramp_val_graph_sv(sv):
    print(sv.get())
    if sv.get():
        ramp_val = float(e_ramp_graph.get())
    else:
        ramp_val = 0

    graph_plot1_Ramp.clear()
    x = np.linspace(-2, 2, 200)
    y = [min(ramp_val,-i) if i < 0 else 0 for i in x]

    graph_plot1_Ramp.plot(x, y)

    c_graph_Ramp.draw()

master = Tk()
master.title("Fabric Defect detection - Training system - [MSc in AI - 12 batch]")
master.geometry("1850x1020")
master.grid_columnconfigure(0, minsize=50)
master.grid_columnconfigure(1, minsize=200)
master.grid_columnconfigure(2, minsize=200)
master.grid_columnconfigure(3, minsize=10)
master.grid_columnconfigure(4, minsize=200)
master.grid_columnconfigure(5, minsize=200)
master.grid_columnconfigure(6, minsize=50)

master.grid_columnconfigure(7, minsize=200)
master.grid_columnconfigure(8, minsize=200)
master.grid_columnconfigure(9, minsize=90)
master.grid_columnconfigure(10, minsize=200)
master.grid_columnconfigure(11, minsize=200)
master.grid_columnconfigure(12, minsize=50)

master.rowconfigure((0,1, 2, 3, 4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21), weight=2)
master.rowconfigure(4, weight=1)
master.rowconfigure(3, weight=3)
master.resizable(width=False, height=False)

l_title = Label(master,text="Fabric Defet Detection - Model Training" ,font="arial 20
bold")
```

```
l_Dataset = Label(master,text="Select Dataset" ,font="arial 12 bold")

l_dataset_details = Label(master,text="             \n           ",font="arial 13 italic
bold",justify=LEFT)
n = StringVar()
datasetchoosen = ttk.Combobox(master, width=10,
                              textvariable=n, validatecommand=cbox)

# Adding combobox drop down list
datasetchoosen['values'] = (' Carpet',
                            ' UMDAA',
                            ' Capsulle',
                            ' Cable',
                            ' Hazelnut',
                            ' Leather',
                            ' Tile',
                            ' Transistor')

datasetchoosen.bind("<<ComboboxSelected>>", TextBoxUpdate)
# monthchoosen.current(0)

l_positive_image = Label(master,text="Negative Image" ,font="arial 10",justify=LEFT)
l_negative_image = Label(master,text="Positive Image" ,font="arial 10",justify=LEFT)

c_defect_image = Canvas(master, width = 300, height = 300)
c_nondefect_image = Canvas(master, width = 300, height = 300)
load_images()
c_d_container = c_defect_image.create_image(20, 20, anchor=NW, image=img)
c_n_container = c_nondefect_image.create_image(20, 20, anchor=NW, image=img)

l_Grad_Ratio = Label(master,text="Gradient Ratio" ,font="arial 12 bold")
l_Ramp_Limit = Label(master,text="Ramp Limit" ,font="arial 12 bold")

e_grad_val = StringVar()
e_grad_val.trace("w", lambda name, index, mode, sv=e_grad_val: change_graph_grad_sv(sv))
e_grad = Entry(master, width=10, font=("arial 12 bold"), justify=RIGHT,
textvariable=e_grad_val)

e_ramp_val = StringVar()
e_ramp_val.trace("w", lambda name, index, mode, sv=e_ramp_val: change_graph_ramp_sv(sv))
e_ramp = Entry(master, width=10, font=("arial 12 bold"), justify=RIGHT,
textvariable=e_ramp_val)

graph_fig = Figure(figsize=(5, 2),dpi=120)
graph_plot1 = graph_fig.add_subplot(111)
# graph_plot2 = graph_fig.add_subplot(111)

c_graph = FigureCanvasTkAgg(graph_fig, master=master)

x = np.linspace(-2,2,200)
y = [-i if i<0 else 0 for i in x]
graph_plot1.plot(x,y)

c_graph.draw()

b_train = Button(master, text="Train", width=13,command=cbox, font="arial 10 bold")
b_exit = Button(master, text="Exit", width=13,command=exit, font="arial 10 bold")

l_title.grid(row=1, column=3, columnspan=7, pady=7)
l_Dataset.grid(row=3, column=1, pady=7)
```

```
l_dataset_details.grid(row=3, column=4,columnspan=3,rowspan=2, pady=7)
datasetchoosen.grid(row=3, column=2)

l_positive_image.grid(row=5, column=1, pady=7)
l_negative_image.grid(row=5, column=4, pady=7)

c_defect_image.grid(row=6, column=1, columnspan=2,rowspan=6,pady=7)
c_nondefect_image.grid(row=6, column=4,columnspan=2,rowspan=6, pady=7)

l_Grad_Ratio.grid(row=13, column=1, pady=7)
l_Ramp_Limit.grid(row=13, column=4, pady=7)

e_grad.grid(row=13, column=2, pady=7)
e_ramp.grid(row=13, column=5, pady=7)

c_graph.get_tk_widget().grid(row=15, column=1, columnspan=5,rowspan=5, pady=7)

b_train.grid(row=20, column=4, pady=7)
b_exit.grid(row=20, column=5, pady=7)

#-----------------Right Side-------------------------#
l_exist_loss = Label(master,text="Existing Loss Functions" ,font="arial 15 bold",)
l_hinge_loss = Label(master,text="Hinge Loss Functions" ,font="arial 12 bold",anchor='e')
l_Ramp_loss = Label(master,text="Ramp Loss Functions" ,font="arial 12 bold",anchor='w')
l_Robust_loss = Label(master,text="Robust Loss Functions" ,font="arial 12 bold",anchor='w')

graph_fig_hinge = Figure(figsize=(5, 2),dpi=120)
graph_plot1_hinge = graph_fig_hinge.add_subplot(111)
c_graph_hinge = FigureCanvasTkAgg(graph_fig_hinge, master=master)
x = np.linspace(-2,2,200)
y = [-i if i<0 else 0 for i in x]
graph_plot1_hinge.plot(x,y)
c_graph_hinge.draw()

# Ramp graph
l_ramp_graph = Label(master,text="Ramp Limit" ,font="arial 12")

e_ramp_val_graph = StringVar()
e_ramp_val_graph.trace("w", lambda name, index, mode, sv=e_ramp_val_graph:
e_ramp_val_graph_sv(sv))
e_ramp_graph = Entry(master, width=10, font=("arial 12 bold"), justify=RIGHT,
textvariable=e_ramp_val_graph)

graph_fig_Ramp = Figure(figsize=(5, 2),dpi=120)
graph_plot1_Ramp = graph_fig_Ramp.add_subplot(111)
c_graph_Ramp = FigureCanvasTkAgg(graph_fig_Ramp, master=master)
c_graph_Ramp.draw()

# Robust graph
l_robust_graph_eta = Label(master,text="Eta value" ,font="arial 12")

e_robust_val_graph_eta = StringVar()
e_robust_val_graph_eta.trace("w", lambda name, index, mode, sv=e_robust_val_graph_eta:
e_robust_val_graph_eta_sv(sv))
e_robust_graph_eta = Entry(master, width=10, font=("arial 12 bold"), justify=RIGHT,
textvariable=e_robust_val_graph_eta)

graph_fig_robust = Figure(figsize=(5, 2),dpi=120,)
```

```
graph_plot1_robust = graph_fig_robust.add_subplot(111)
c_graph_robust = FigureCanvasTkAgg(graph_fig_robust, master=master)

c_graph_robust.draw()

l_exist_loss.grid(row=3, column=7,columnspan=3, pady=7)
l_hinge_loss.grid(row=5, column=7,columnspan=2, pady=7)
l_Ramp_loss.grid(row=10, column=7,columnspan=2, pady=7)
l_Robust_loss.grid(row=15, column=7,columnspan=2, pady=7)

# hinge
c_graph_hinge.get_tk_widget().grid(row=5, column=9, columnspan=3,rowspan=4, pady=7)

#ramp
l_ramp_graph.grid(row=12, column=7, pady=7)
e_ramp_graph.grid(row=12, column=8, pady=7)
c_graph_Ramp.get_tk_widget().grid(row=10, column=9, columnspan=3,rowspan=4, pady=7)

#robust
l_robust_graph_eta.grid(row=17, column=7, pady=7)

e_robust_graph_eta.grid(row=17, column=8, pady=7)

c_graph_robust.get_tk_widget().grid(row=15, column=9, columnspan=3,rowspan=4, pady=7)

master.mainloop()
```