SELF-REPORTING SYSTEM FOR INCIDENTS DETECTION IN AUTOMATED TELLER MACHINE (ATM) USING MACHINE LEARNING TECHNIQUES

BY

IVY NKRUMAH PAYNE

(10701873)

THIS THESIS IS SUBMITTED TO THE UNIVERSITY OF GHANA, LEGON, IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE AWARD OF MPHIL COMPUTER ENGINEERING DEGREE

DEPARTMENT OF COMPUTER ENGINEERING

SCHOOL OF ENGINEERING SCIENCES

UNIVERSITY OF GHANA, LEGON

SEPTEMBER 2021

DECLARATION

I, Ivy Nkrumah Payne, author of this thesis, hereby declare that the work presented in this thesis, **Self-Reporting System for Incidents Detection In Automated Teller Machine (ATM) Using Deep Learning Techniques**, is my work, produced from research undertaken under supervision in the Department of Computer Engineering, School of Engineering Sciences, University of Ghana, Legon from September 2019 to July 2021. This work has never been presented either in whole or in part for any other degree in this University or elsewhere.

October 11, 2022

……. …….

Ivy Nkrumah Payne Date

(Student)

October 11, 2022

……. …….

Prof. Robert Adjetey Sowah Date

(Principal Supervisor)

DEDICATION

This work is dedicated to YAHWEH, GOD ALMIGHTY, and the memory of my late mother, Faustina Serwah Agyarkoh.

ACKNOWLEDGEMENT

First of all, thanks and exaltation go to God Almighty for granting the strength and wisdom throughout this academic journey. I wish to extend my appreciation to my supervisor, Dr. Robert Adjetey Sowah, for all the ideas, guidance, and encouragement toward the successful completion of this research. I also wish to thank all Department of Computer Engineering members, especially Dr. Wiafe Owusu-Banahene, Dr. Godfrey A. Mills, Dr. Nii Longdon Sowah, and the entire department graduate committee for their input in this work.

ABSTRACT

Automated Teller Machines (ATMs) have increased over the past decade due to their advantages in the banking sector. ATMs provide convenience to customers, optimizes banking operations, and minimizes transaction cost. However, undesirable security incidents such as tempering, skimming, physical attacks, robbery, and transaction reversal fraud may occur on ATM systems and negatively affect the user experience and banking institutions. ATM incidents occur either by system defect or through a deliberate act of physical attack by an intruder. In most security incidents, financial losses are imminent, and the customers' confidence in banking reduces. Developing a Self-Reporting System for ATM Incident Detection (SRSAID) is needed to avert the threats posed by security incidents on ATM systems. This research uses a machine-learning approach to solve this problem. Regional Convolutional Neural Network (R-CNN) and Support Vector Machine (SVM) algorithms are used to develop a detection model that detects occurrences of security incidents on an ATM system. Datasets used in the machine learning model development were obtained from NCR Ghana and the online repository. Experimental results showed that two CNN architecture models, ALEXNET and ssdlite_mobilenet_V2, obtained an accuracy score of 80% and 96%, respectively. SVM classifiers were developed using the linear, polynomial, and radial basis kernels, getting accuracy scores of 70.6%, 72.56%, and 81.21%, respectively. The initial results necessitated hyperparameter optimization to improve the performance of the classifiers. This resulted in improved accuracy scores of 76%, 77%, and 86% for linear, polynomial, and radial basis kernels, respectively, for the SVM models. The machine learning model was later deployed on a Raspberry Pi system which connected to a web application that provided a graphical user interface for user interactivity and viewing of reports.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ATM	Automated Teller Machine
CNN	Convolutional Neural Network
FN	False Negative
FP	False Positive
GPS	Global Positioning System
GSM	Global System for Mobile Communication
MCC	Matthews Correlation Coefficient
R-CNN	Regional Convolutional Neural Network
SRSAID	Self-Reporting System for ATM Incident Detection
SVM	Support Vector Machines
TN	True Negative
TP	True Positive

CHAPTER ONE

INTRODUCTION

1.0 Introduction

Technology has brought about many improvements in human livelihood, and one of the few is the usage of Automated Teller Machines (ATMs) for financial institutions. ATM is an electronic banking outlet that allows customers to complete basic transactions without the aid of a branch representative or teller. An ATM provides clients of a Bank with 24Hours deposit and withdrawal services [1][2]. Cash dispenser machines or ATMs are installed in Banks and other strategic locations for the convenience of the customer. As we move towards social distancing and a cashless society, ATMs will continue to play a vital role in paperless financial transactions. All these show the importance of ATMs in financial transactions; however, security and other technical system defects remain a key challenge for Banks and their ATMs [2].

In security management, Banks deploy security personnel to monitor their ATMs in addition to other technologies such as Close Circuit Television (CCTV) cameras. Monitoring ATMs can become tedious due to the complexity and multiplicity of ATM devices located in and off the Bank's premises. Some of the monitoring activities performed by the security personnel include the following[3]:

- Monitoring security incidents and the operational status of ATMs.
- Monitoring system defects incidents.
- Protection from attempted ATM tampering, theft, and vandalism.

- Dealing with perpetrators [4].

While some form of security is incorporated in ATM installations, these security mechanisms are often subjected to several escalating security incidents. These security incidents can be categorized as follows;

- Card skimming
- Card trapping
- Transaction Reversal Fraud
- Cash trapping
- Physical attack
- Robbery
- Jackpotting
- Logical attacks
- Occlusion.

Banks and other financial institutions work with third-party companies like NCR Corporation for their ATM installations and related services. NCR Corporation is a leading tech company that provides digital solutions to companies, hospitals, and financial institutions. Its subsidiary company, NCR Ghana, is a significant industry player in the financial sector in Ghana. Their core services include installing, configuring, and troubleshooting ATM devices for the banks they work with. The problems associated with ATM systems, especially on the technical side, become a shared responsibility between financial institutions and the ATM Service Companies like NCR Ghana [5]. Some of the technical problems associated with ATMs include the following:

- Dispenser Problem

- Non-remedial call
- Card reader problem
- Pick drive mechanism

To mitigate problems associated with ATMs, there is a need for a distributed solution that integrates system defect solutions with security incident solutions using a machine learning approach. This solution offers an advantage over the physical monitoring of ATMs; it also serves as a proactive mechanism for system defects and security incidents.

1.1 Problem Statement

Research on the ATM system, its management, and related challenges in the banking sector has been covered in many forms. While some are large –scale, few are done on electronic banking systems and networks such as ATMs [6]. The proliferation of ATMs in the country has aided business; it has also been a breeding ground for fraudsters. Security incidents in ATMs lead to financial losses for the banks. According to the Bank of Ghana (BoG) 2017 Annual Report, banks lost over GH₵ 30 million, a significant chunk of this due to ATM fraud [7].

Recently, most ATMs have in-built cameras that capture activities around them. While these cameras serve as security mechanisms, they do so in a passive state. For instance, the cameras can record footage of security incidents such as skimming and tampering on ATMs; the security camera cannot prevent fraudsters from perpetrating the act. This security mechanism (CCTV cameras in ATMs) has proven inefficient due to its inherent limitations, preventing ATM incidents in an active state or environment. On the other hand, technical defect incidents also share the challenge with ATM security incidents. This is because ATMs with technical

problems can be left unattended for months. This is due to the lack of a self-reporting mechanism in ATMs.

To ensure the availability, reliability, and security of ATM services, answers should be provided to the following fundamental questions:

- What would be the reaction of the ATM user after reading a message from the screen that the machine is temporarily down?
- How do we automatically detect ATM problems or incidents at the onset and make an intelligent decision on how to respond to them to bring perpetrators on board, improve profitability, reduce operational support costs, and deliver an amazing customer experience?

ATM security incidents were reported at approximately GhC1.7million in the 2017 Bank of Ghana Annual Report. Again, Figure 1.1 shows the distribution of the banking sector fraud type and losses, most of which are cybercrime, in which ATMs fall under this category.

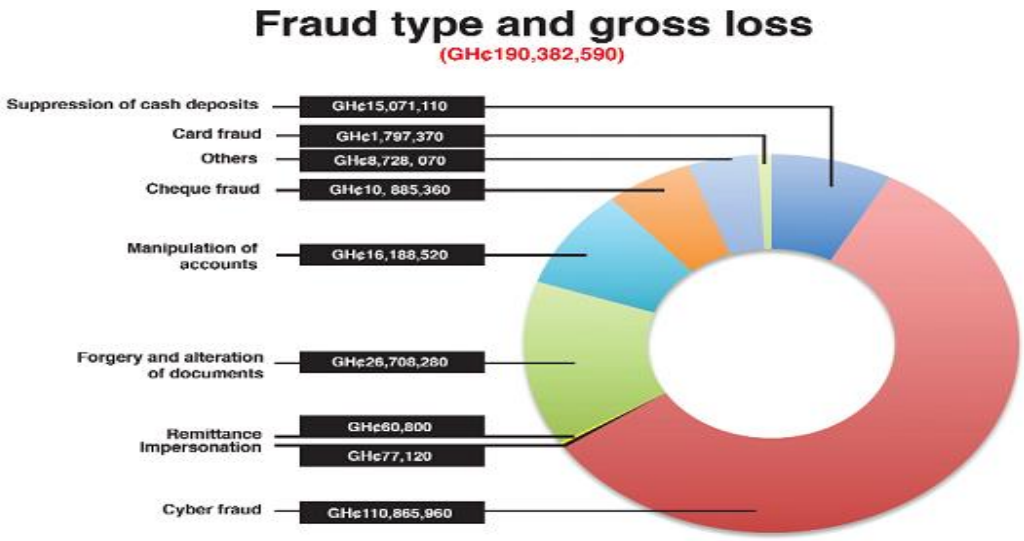


Figure 1. 1 Fraud type and Gross loss in Ghana [7]

ATM faults reported 3970 cases of the Dispenser problem according to NCR, Ghana, 2018 ATM incidents report.

ATM security and system defect incidents that cause financial loss and temporarily put the machine down can be prevented by automatically self-reporting these incidents to the Banks, ATM service providers, and security officials would efficiently enhance the handling of incidents.

The proposed self-reporting system for ATM incidents detection uses Machine Learning techniques for image classification of attempted instances of ATM security incidents. This technique is extended to the ATM system defects aspect of the problem. This method uses internet of things (IoT) components such as Global System for Mobile Communications (GSM) Alerts embedded in a software application that interacts with a Global Positioning System

(GPS) system, which points to the location of a detected incident using the Google Maps Application Programming Interface (API).

1.2 Objectives of Study

This study aims to design and implement a self-reporting security and system defect application for ATMs in Ghana.

The specific objectives of this study are as follows:

- Design and develop a security incident detection model for ATMs based on Convolutional Neural Networks (CNN).
- Design and develop a defective incident detection model based on Support Vector Machines (SVMs).
- Develop along with GSM alerts and indications on Google Maps for locating ATM incidents.
- Develop and deploy a user interface (UI) that provides access to monitoring all ATMs and system resources on the ATM network.
- Develop an integrated system for incident and defect detection modules with microcontroller-based hardware for overall proof of concept.
- Test the proposed model on a real-world ATM terminal on a Bank's ATM and inject a range of common incidents to evaluate efficacy.

1.4 Justification of Study

As we move towards a cashless economy, ATMs and Information Communication Technology (ICT) related technologies will play a vital role in banking activities. ATM security and its

defect management continue to become a significant concern for Banks, ATM Service Providers, and other third-party stakeholders like state security, the Bank of Ghana, and the general public. The financial losses due to ATM fraud associated with tampering have been enormous, and there is an urgent need for improvements to mitigate such occurrences.

While contributing to the scientific community, this research also addresses a real-world banking sector problem for decades. This research is not intended to phase out the work of ATM technicians; however, it seeks to augment and improve their services in a timely and efficient manner.

1.5 Research Limitation

The research takes into account two main aspects of ATM problems; security incidents and system defect incidents. While the security incidents are reported in real-time through the issuance of SMS alerts to mobile devices of the banks and ATM service providers, a dashboard interface also receives such reports. The system defect problems, on the other hand, are only reported on the dashboard. This implies a worker from the bank needs to monitor the system for periodic updates.

1.6 Thesis Outline

This study exhibited in this thesis gives details of Machine Learning techniques for ATMs to detect incidents and is organized into six chapters.

The remaining chapters continue as follows:

Chapter two presents a literature review of the current research on ATM incidents; Chapter two provides a literature review of the recent research on ATM incidents, Machine Learning

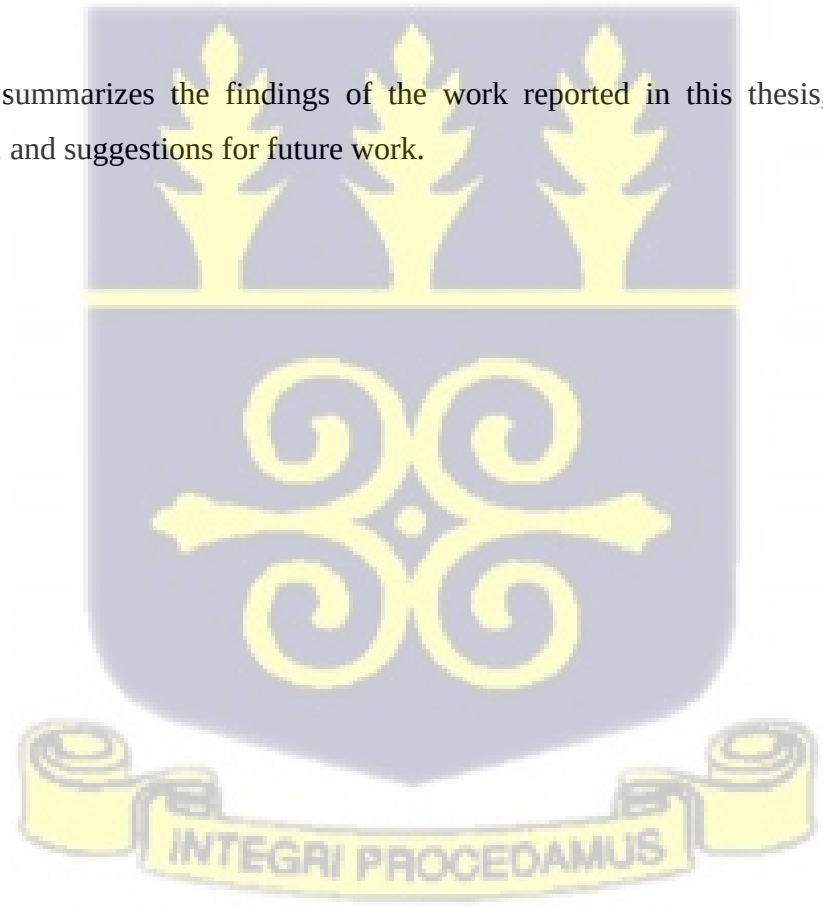
techniques and algorithms, fine-tuning techniques, and their applications to detect ATM incidents.

Chapter three discusses system requirements, specification analysis, blueprints, and the careful selection of development tools for the incident detection models.

Chapter four focuses on the development of the models and discussions on accuracy metrics. Results and findings are interpreted as well.

Chapter five presents the system simulation and experiment, implementation process, testing and results, and discussion of results. The outcome of the experimental processes will illustrate the capabilities of the proposed system to perform security incidents detection and security incidents categorization. Moreover, ATM system defects incident detection is discussed as well.

Chapter six summarizes the findings of the work reported in this thesis, challenges and observations, and suggestions for future work.



CHAPTER TWO

LITERATURE REVIEW

2.0 Introduction

This chapter reviews related work on ATM security and defect incidents. It also reviews work on several methodologies and proposed solutions within the scope of ATM fraud and technical defects. These solutions' algorithms and performance metrics are presented in a tabular format.

2.1 ATM Network Architecture

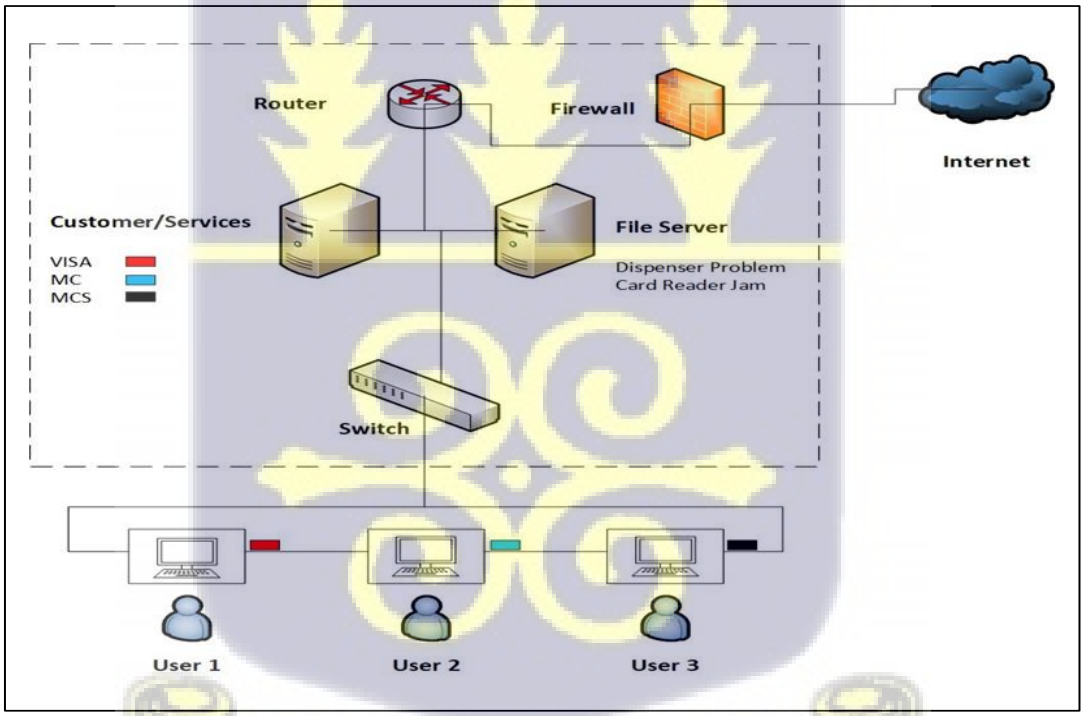


Figure 2. 1 ATM Network Diagram

Figure 2.1 shows the network diagram for an ATM system. In the diagram, the subcomponents of the network setup are listed below:

- ATM
- Switch
- Router
- File server
- Firewall
- Customer/services such a VISA card and Master Card.

Based on technical information gathered at the NCR, the components of an ATM system can be grouped into two: thus, the upper part and the lower part. The upper part comprises the central processing unit (CPU) and the card reader. The lower part of the ATM consists of two parts: suction technology and domain safe.

The Suction Technology

The suction technology is a mechanism the ATM uses to dispense money to the upper part of the ATM. It enables the cassettes to present the dispensed cash in belts [8].

The Domain Safe

This part of the ATM contains the presenter, the network card, and other networking components. The NCR ATMs run a middleware that enhances the network communication between the switch and the card reader [8].

2.2 ATM Incident Detection

Incident detection is finding an attacker's activities who deliberately seek to circumvent protocols on network infrastructure. Some malicious activities are mitigated by retracing, containing the threats, and removing their foothold [9]. Learning how attackers compromise systems and move around the network can better detect and stop attacks before valuable data is stolen or the system collapses.

2.2.1 Object Detection

Object detection is one of the classical problems of computer vision and is often described as a difficult task. Object detection is a computer vision task because it involves creating a solution invariant to deformation and changes in lighting and viewpoint. Object detection is a problem because it involves locating and classifying an image's regions [10]. We need to know where the object might be and how the image is segmented to detect the object. That creates a type of chicken-and-egg problem.

To recognize the shape (class) of an object, its location must be known, and to identify the location of an object, its shape must be known [11]. Some visually dissimilar features, such as the clothes and faces of a human being, may be parts of the same object, but it is difficult to know this without recognizing the object first. On the other hand, some objects stand out slightly from the background, requiring separation before recognition[12]. Low-level visual features of an image, such as a saliency map, may be used as a guide for locating candidate objects [13].

The location, shape, and size are typically defined using a bounding box stored in corner coordinates. Using a rectangle is more straightforward than using an arbitrarily shaped polygon,

and many operations, such as convolution, are performed on rectangles in any case. The sub-image in the bounding box is then classified by an algorithm trained using machine learning [14].

The boundaries of the object can be further refined iteratively after making an initial guess [15]. During the 2000s, popular solutions for object detection utilized feature descriptors, such as scale-invariant feature transform (SIFT) [16] developed by David Lowe in 1999, and histogram of oriented gradients (HOG) popularized in 2005 [16]. In the 2010s, there has been a shift towards utilizing convolutional neural networks [14][10][17].

Before the wide-scale adoption of CNNs, there were two competing solutions for generating bounding boxes. A dense set of region proposals is generated in the first solution, and most of these are rejected [18]. This typically involves a sliding window detector. The second solution generates a sparse set of bounding boxes using a region proposal method, such as Selective Search [12]. Combining sparse region proposals with convolutional neural networks has provided good results and is currently popular [10].

2.2.2 Computer Vision

Computer vision is divided into object detection and extracting meaningful information from digital images or video content. This is distinct from mere image processing, which involves manipulating visual information on the pixel level. Computer vision application includes image classification, visual detection, 3D scene reconstruction from 2D images, image retrieval, augmented reality, machine vision, and traffic automation[19].

Today, deep learning and machine learning are necessary for many computer vision algorithms. The algorithms can be described as image processing and machine learning. Effective solutions

require algorithms that can cope with the vast amount of information in visual images and, critically for many applications, can perform the computation in real time[20].

2.3 Machine Learning Classification Algorithms

Classification is a technique to categorize datasets into a desired and distinct number of classes where labels can be assigned to each class. Classification applications are image classification, speech recognition, handwriting recognition, and others [21].

Classification algorithms weigh the input features so that the output separates one class from the other. Classifier training is performed to identify the weights (functions) that provide the data classes' most accurate and best separation. Classification decisions are made based on the results of classifiers. Such classification algorithms are Linear Discriminant Analysis (LDA), Support Vector Machines (SVM), Random Forest, and Artificial Neural Networks [21][22]. LDA is the most basic classifier that identifies the linear weighting of multifactorial data as a means to maximize the distance between the means of the classes. Classification algorithms such as SVM, ANN, and others perform well on large datasets and are the recent computational approaches that generate more complex divisions between classes of the datasets[15][19][23]. Multiple classifiers can also be used, and training and classification decisions can be made based on the results of all the classifiers. The success of the classification algorithms is determined by the performance metrics[22].

The classification algorithm processes the features to produce a class decision. These processes involve a statistical algorithm that maps the different classes into different regions of feature space.

2.3.1 SVM-OVO-OVR Approach

The most accurate methodology for ATM defect incidents datasets is a combination of the OVR approach and SVM using all components provides a powerful technique. A multi-class classification problem can be defined as:

Given n i.i.d. data points:

$$(x_1, y_1),....,(x_n, y_n), \text{such that } x_i \in R^d \text{ for } i=1,...,n \text{ and } y_i \in \{1,....,K\} \quad (2.1)$$

The class label for the data point x_i determines a classifier with the decision function, $f(x)$ such that $y = f(x)$ where y is the class label for x [24]. The performance of the classifier is measured in terms of total classification error or classification accuracy over a set of testing data. The classification error for data point x is defined as:

$$E(y, f(x)) = \begin{cases} 0 & \text{if } y = f(x) \\ 1 & \text{otherwise} \end{cases} \quad (2.2)$$

There are two approaches suggested for multi-class SVM in literature [21]. One is considering all data in one optimization. The other is decomposing multi-class into a series of binary SVMs, such as "One-versus-Rest" (OVR) and "One-versus-One" (OVO) [24][25][22].

OVO-SVM is the most basic scheme used for the implementation of SVM-multiclass classification. With this simplest SVM extension to the $k-class$ problem, k binary SVM models are constructed. In $j^{th}\ binary-class$ SVM problem $class\ C_j$ is separated from the remaining classes. All k binary SVM classifiers are combined to make a final multiclass

classifier. The remaining classifiers mean that all the data points from classes other than c_j are combined to form one $class\, C_j$. Given the ATM defects training dataset, correspondingly to $Xn : R^n \in F$ in the feature space F, the calculated optimal hyperplane that separates data points from the $class\, C_j$, and the $combined\ class\, C_i$, is found using SVM methodology'. The optimal separating hyperplane distinguishing the $class\, c_j$ and the $combined\ class\, c_i$ $combined\ class\, c_i$ is represented as:

$$g^j(x) = w^j.\phi(x) + b^j,\ j = 1,.....k \tag{2.3}$$

Assuming the training dataset is correctly classified, as shown in Figure 4.2.4, the SVC computes the hyperplane to maximize the margin separating the classes [26] (area of failure description, cause code description, and problem code description).

The problem is a quadratic optimization problem (QP) which gives the dual formulation of the Langrangian. The dual Langrangian (L_D) is maximized with non-negativity $g^j(x)$ and can be determined by solving the following dual from:

$$\text{Maximize} : L_D = \sum_i^n \alpha_i - \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^m \alpha_i \alpha_j y_i y_j k(x_i, x_j) \tag{2.4}$$

$$Subject\ to \quad 0 \le \alpha_i \le C_i = 1,...., n$$

And

$$\sum_{i=1}^n \alpha_i y_i = 0 \tag{2.5}$$

The decision rule $f^j(x)$ that assigns the vector x to the $class\, C_j$ or the combined $class\, c_i$ is given by:

$$f^j(x) = sign\left(g^j(x)\right) \tag{2.6}$$

If there are no positive votes or more than one classifier with positive votes, then no decision about the class label is made.

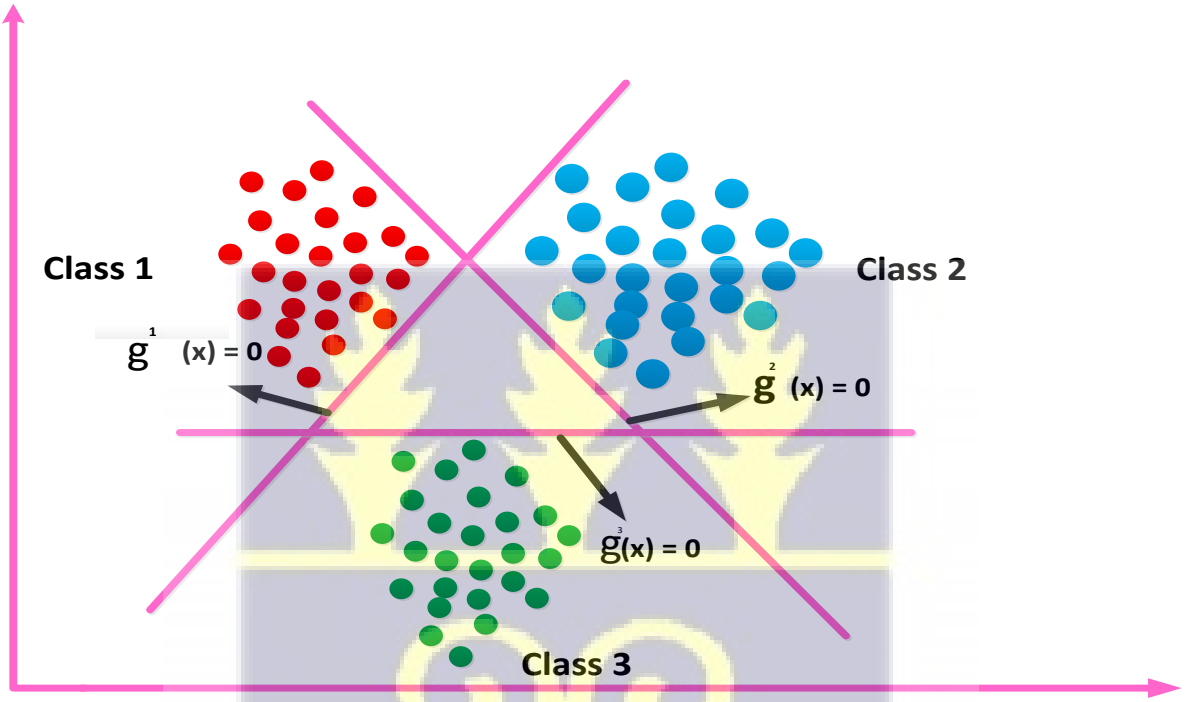


Figure 2. 2 Class boundaries for SVM-OVR formulation of the three-class problem

The main difficulty in this approach is that the outputs of the classifiers $f^j(x)$ are binary values. The usual way to handle this problem is to ignore the sign operator in the equation. After finding all the optimal hyperplanes given by $g^i(x)$ *for* $j = 1,.....k$, we say x is in the class which has the largest value of the decision function and is given by:

$$class\, label\, x = \arg\max\, j = 1,....k, g^j x \tag{2.7}$$

In this approach, the index of the largest component of the discriminant functions:

$$g^j(x), j = 1,....k \tag{2.8}$$

This is assigned to the data point x [24]. This approach is called winner-takes-all. To make a final decision, k binary problem should be solved. A dual problem has to be translated to solve each binary problem containing n data points.

Hence, $(k.n)$ variable quadratic programming problems are to be solved. Class boundaries for the three-class problem are shown above in *Figure: 2.2.*

Mathematical Foundation for Support Vector Machines (SVM)

Support vector machines (SVM) are statistical machine learning algorithms that classify by finding a hyper-plane that maximizes the margins between classes". "SVM is limited to very large datasets due to the dense nature and memory requirement of the quadratic form of the dataset" [26]. Given the ATM defects datasets, correspondingly to $Xn : R^n \in F$ in the feature space F .

A hyperplane is defined by its normal vector ω. "Given a hyperplane ω and a point x define x_0 to be the closest point to x the hyperplane, which is the closest point to x that satisfies $w.x_0 = 0$ as in figure 2.3.

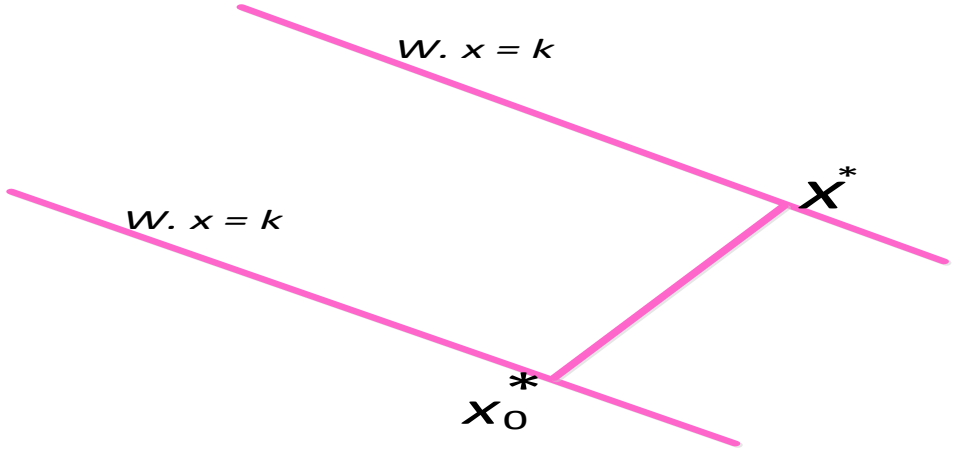


Figure 2. 3 The distance between and of the multiclass classification

The following two equations are obtained:

$$w.x = \text{for some } k,$$

$$w.x_0 = 0 \tag{2.9}$$

Subtracting the above equation, we obtained the following:

$$w.(x - x_0) = k \tag{2.10}$$

Dividing by the norm of w, this is obtained:

$$\frac{w}{\|w\|}.(x - x_0) = \frac{k}{\|w\|} \tag{2.11}$$

$\frac{w}{\|w\|}$ are a unit vector and $x - x_0$ is parallel to w

$$|| x - x_0 || = \frac{k}{|| w ||} \tag{2.12}$$

Nevertheless, SVM is an excellent example of supervised learning that maximizes the generalization by maximizing the margin and supports the nonlinear separation using kernelization. SVM tries to avoid over-fitting and under-fitting. The margin in SVM denotes the distance from the boundary to the closest data point in the feature space [22][21][25]

In general, there may be many separating hyperplanes. In this problem, this separating hyperplane is the boundary separating a given ATM defect incident class from the rest (OVR) or separating two different ATM defect incident classes (AP). The hyperplane computed by the SVM is the maximal margin hyperplane, the hyperplane with maximal distance to the nearest data point. Finding the SVM solution requires training an SVM, which entails solving a convex quadratic program with as many variables as training points[27].

Using the OVR methodology to combine binary SVM classifiers into a multiclass classifier. A separate SVM is trained for each class, and the winning class has the largest margin, which can be considered a signed confidence measure. In the experiments described in this thesis, there were few data points in many dimensions. Therefore, a kernel corresponding to a linear (regularized) classifier was used as the SVM solution. Although we did allow the hyperplane to make misclassifications, in all cases involving the full 16,063 dimensions, each OVR hyperplane fully separated the training data with no errors. Some training errors in some experiments involved explicit feature selection with very few features. This may indicate that we could select a very small number of features and then use a nonlinear kernel function to improve classification; however, preliminary experiments with this approach yielded no improvement over the linear case. An SVM is trained using all features. The features are ranked

according to the magnitude of the elements of the resulting hyper plane, so the importance of feature i is the weight of the i^{th} element of the hyper-plane.

2.3.2 Convolutional Neural Networks (CNN)

The convolutional neural network is a deep learning algorithm that can take an input image, assign weights and biases to various aspects or objects in the image, and distinguish one from the other, as indicated in figure 2.4. A convolutional neural network is mainly for image classification.

A concept inspired the basic idea of CNN in biology called the receptive field [13][20]. Receptive fields are a feature of the animal visual cortex [28]. They act as detectors sensitive to certain stimulus types, for example, edges. They are found across the visual field and overlap each other.

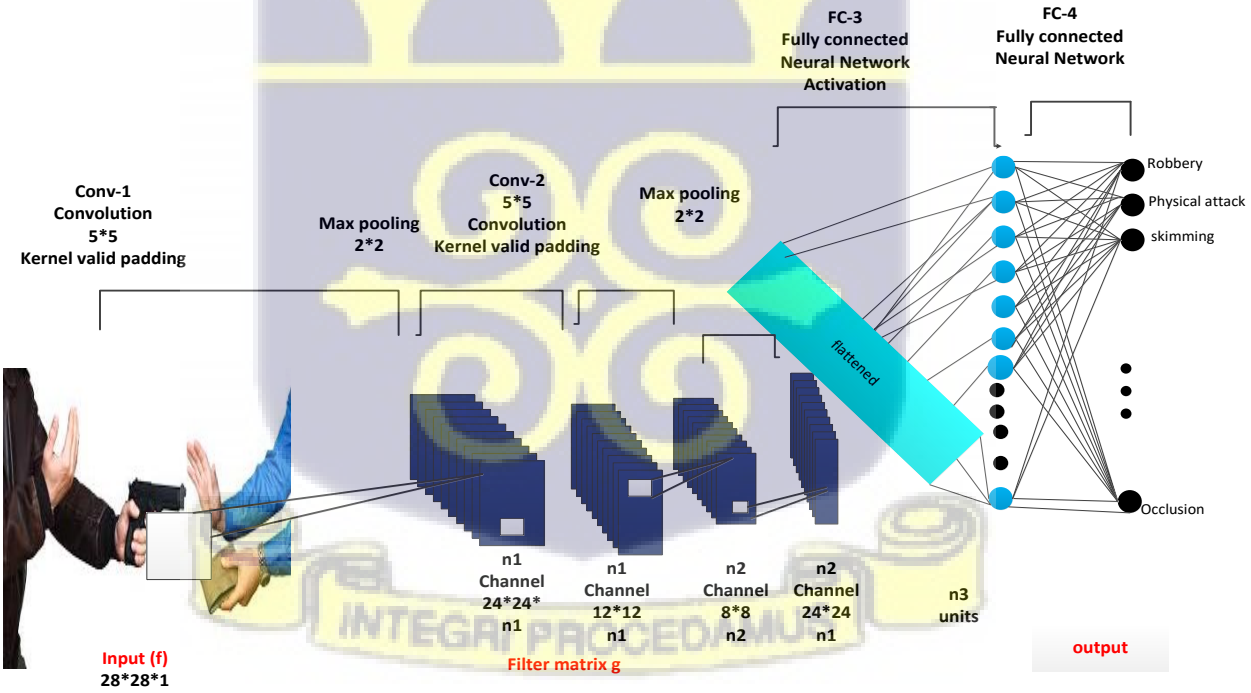


Figure 2. 4 CNN Architecture

This biological function can be approximated in computers using convolution [20]. In image processing, images can be filtered using convolution to produce different visible effects. Figure 2.4 shows how a hand-selected convolutional filter detects horizontal edges from an image, functioning similarly to a receptive field. The discrete convolution operation between an Image f and a filter matrix g is defined as:

$$h[x,y] = f[x,y] * g[x,y] = \sum_{m=1}^{M}\sum_{n=1}^{N} f[n,m]\,g[x-n,y-m] \qquad (2.13)$$

In effect, the dot product of the filter g and a sub-image of f (with same dimensions as g) entered on coordinates $x;y$ produces the pixel value of h at coordinates $x;y$. The size of the filter matrix adjusts the size of the receptive field. Aligning the filter successively with every sub-image of f produces the output pixel matrix h. In neural networks, the output matrix is also called a feature map or an activation map after computing the activation function). Edges need to be treated as a special case [20]. The output size decreases slightly with every convolution if an image is not padded.

2.3.3 Regional Convolutional Neural Networks (RCNN)

Regional Convolutional Neural Network (R-CNN) is a state-of-art visual object detection system that combines bottom-up region proposal with rich features computed by a convolutional neural network. R-CNN forward computation has several stages [10][13][20]. First, the regions of interest (ROIs) are generated. The ROIs are category-independent bounding boxes with a high likelihood of containing an interesting object. A separate method called Selective Search is used to generate these regions [12][17][29].

Given an image, the R-CNN uses selective search to generate around 2000 region proposals to compute features using a Convolutional Neural Network (CNN). Region proposals are regions

that include the potential object. It will be wrapped as 277×277 RGB to fit into CNN. Feature extraction is done in the CNN layers and passed to multiple binary classifiers to determine the class for particular regions. Figure 2.5 illustrates the R-CNN stages.

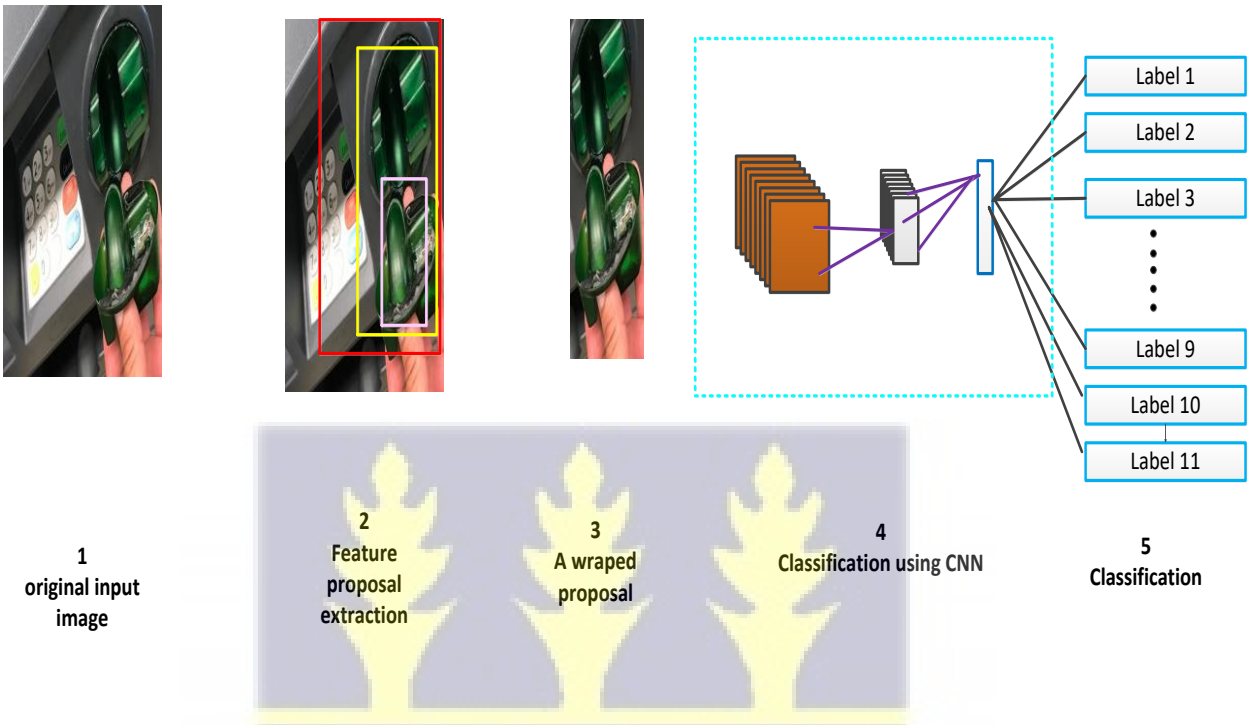


Figure 2. 5 Stages of R-CNN forward Computation

R-CNN forward computation has several stages. First, the regions of interest are generated. The Regions of interest are category-independent bounding boxes with a high likelihood of containing an interesting object. Selective Search [12] is used for generating these.

Next, a convolutional network extract features from each region proposal. The sub-image contained in the bounding box is warped to match the input size of the CNN and then fed to the network. After the network has extracted features from the input, the features are input to the regional convolutional neural network (R-CNN) that provides the final classification. The requirements in the pre-processing stage in a convolutional neural network are much lower than

in other classification algorithms. CNNs have the ability to their filters or detectors through enough training.

A concept inspired the architecture of CNN in biology called the receptive field [10]. Receptive fields are a feature of the animal visual cortex [28]. They act as detectors sensitive to certain stimulus types, for example, edges. They are found across the visual field

and overlap each other [20].

2.4 Transfer Learning

Transfer learning (TL) is a research problem in machine learning (ML) that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. Transfer learning is an approach to transferring a part of the network that has already been trained on a similar task while adding one or more layers at the end and then re-train the model. Transfer learning is the idea of overcoming the isolated learning paradigm and utilizing knowledge acquired for one task to solve related ones[30][31]. Transfer learning is something that data scientists and researchers believe can further our progress toward Artificial general intelligence (AGI)[32]. AGI is the hypothetical intelligence that can understand or learn any intellectual task that a human being can do [31]. It is a primary goal of some artificial intelligence research and a common topic in science fiction and future studies. AGI can also be referred to as strong AI, full AI, or general intelligent action [31]. Some academic sources reserve "strong AI" for machines that can experience consciousness. Today's AI is speculated to be decades away from AGI [31].

In transfer learning, the neural network (NN) such as CNN is trained in two stages, namely:

- Pre-training
- Fine-tuning.

The network is trained on a large-scale benchmark dataset representing a wide range of categories with the pre-training. For fine-tuning stage, the network is further trained on the specific target task of interest, which usually has fewer labeled examples than the pre-training datasets. Such neural network architectures include ALEXNET, ImageNet, ResNet50, Inception, Mobilenet V2, and others.

Researching in transfer learning, some studies suggest that deep learning models trained for a classification task can be employed for classification. Thus, the CNNs models trained on a specific dataset or task can be fine-tuned for a new task, even in a different domain [33]. It has been applied successfully for visual categorization tasks in object recognition, image classification, and human action recognition[33].

2.5 Performance Metrics for Security Incidents Detection

Among the commonly used performance metrics for security incident detection are the mini-batch size and epoch.

Mini-Batch size: Mini-batch size is the subset of the training images at every epoch. It is used to update the weights. A different mini-batch is used in each iteration. The mini-batch accuracy reported during training relates to the accuracy of the particular mini-batch at the given iteration[34].

Epoch: An epoch is a hyperparameter characterized before training a model in deep learning. One epoch is the point at which a whole dataset is passed forward and reversed through the neural network system once[34][33].

Confidence Score: A confidence Score is an ordered set of values that can be easily compared. It is a decimal number between 0 and1, which can be interpreted as a percentage of confidence.

For defects incident detection, in a multiclass classification, there are four (4) main categories a predicted sample of a classifier will belong to [35]:

- ❑ True positives (TP)
- ❑ False positives (FP)
- ❑ True negatives (TN)
- ❑ False negatives (FN)

These four categories are used to form the confusion metrics [35] as indicated in the table below:

Table 2. 1 Confusion Matrix

	Actual condition positive	Actual condition negatives
Predicted positive condition	TP	FP
Predicted negative condition	FN	TN

Among the commonly used performance metrics are accuracy and error rate. The number of correctly classified instances determines the accuracy of the classifier, and the error rate is the number of instances that are incorrectly classified [35][25].

The overall accuracy is given by:

$$A = \frac{TP + TN}{TP + T + FP + FN} \tag{2.14}$$

$$E_r = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.15}$$

However, accuracy can be misleading for highly imbalanced datasets as these metrics favor the majority class [36][35].

For instance, in a dataset where the number of majority instances largely outnumbers the number of minority instances by a ratio $99:1$, the classifier will likely classify all the majority instances correctly and misclassify the only one available minority instance. Hence there will be 99 true negatives, 0 false negatives, 1 false-positive, and 0 true positives, resulting in an accuracy of 99%, which is very misleading since the accurate prediction of the positive class is usually more desirable [35].

$$Acc = \frac{0 + 99}{0 + 99 + 1 + 0} = 0.99 \tag{2.16}$$

To avoid this inconsistency, it is recommended that performance measures based on class metrics are used [35] [37]. These metrics are listed below:

$$TPR = \frac{TP}{TP + FN} \tag{2.17}$$

$$TNR = \frac{TN}{TN + FP} \tag{2.18}$$

$$FPR = \frac{FP}{TN + FP} \quad FNR = \frac{FN}{TP + FN} \tag{2.19}$$

$$AUC = \frac{1 + TPR - FPR}{2} \tag{2.20}$$

$$G - Mean = \sqrt{\frac{TPR \times N}{1}} \tag{2.21}$$

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{2.22}$$

2.6 Summary of Papers

Table 2.2 shows the summary of relevant journal articles that were reviewed.

Table 2. 2 Summary of Papers

SECURITY INCIDENT PAPER TITLE				
	GENERAL OVERVIEW	**METHODOLOGY**	**STRENGTH AND WEAKNESSES**	**ACCURACY**
1. Fast R-CNN for object detection **Author(s):** **Girshick, Ross**	The author(s) looked at ways to improve the efficacy of neural network models in object detection. **Datasets used in the project:** 1. VGG16 2. ImageNet [10]	The paper proposed the Fast Region-based Convolutional Network method (Fast R-CNN) for object detection. Three experiments are used to pre-trained ImageNet models that are available online. The first is the CaffeNet (essentially AlexNet from R-CNN. The second net- work is VGG CNN M 1024 The final network is the very deep VGG16 model. Fast R-CNN was implemented in Python and C++ (using Caffe) and is	Strength: 1. Fast R-CNN trains very deep VGG16 network 9 times faster than R-CNN. 2. Simulations of Fast R-CNN show 213 times faster at test-time and achieve a higher Mean Average Precision (mAP) on PASCAL VOC 2012. 3. Compared to SPPnet, Fast R-CNN trains VGG16 3× faster, tests 10× faster, and is more accurate. Weakness: The author(s) acknowledge the need to further develop this method due to undiscovered techniques that allow dense	R-CNN achieved a Mean Average Precision (mAP) of 66.0%, SPPnet, 63.1% And Fast R-CNN, 66.6%

		available under the open-source MIT License.	boxes to perform and sparse proposals.	
2. ATM- Security using machine learning techniques in IoT **Author(s):** Udhaya Kumar N, Sri Vasu R, Subash S, Sharmila Rani D.	In this paper, the authors designed a security system that gives access to the user of an ATM only after identifying the user's image taken by the CCTV. The image captured will be compared to the image stored in a database. This project will give access to the user only after identifying the user's image taken by the CCTV in the ATM and comparing the specified image with the user's image stored in the database created during the account creation, which comes under the banking session of banks. Sometimes, the authorized user cannot use the ATM for emergency purposes. In such cases, the OTP is sent to the user's registered mobile	This paper proposed a security system for ATMs using deep learning techniques for face detection and recognition. IoT components like a Camera, RFID reader, Tag, Relay, Motor, and a Raspberry pi 3 (2015 version) were used. The authors used the OpenCV platform and Python to implement the Local Binary Patterns (LBP) algorithm. And an alert message is sent to the authorized user as a text message if the user is found to be the third	**Strength:** The proposed solution operates in real-time, thereby providing security in an active state. **Weaknesses:** The system is fully reliant on face detection and recognition system. This makes it imperative for the user to be physically present and granted access to the ATM. As much as they provided an alternative with an OTP sent to the user in such circumstances, it is less effective in the real-world scenario.	The method detected faces without given accuracies

	number, and the person who came instead of the authorized user has to enter the OTP that the authorized user received. This method will reduce the risk of ATM usage by the common people [38]. **Datasets used in the project:** 1. Dataset1 2. Dataset2			
3. Fast Edge Detection Using Structured Forests Author(s): Piotr Dollar and C. Lawrence Zitnick	In this paper, the author(s) analyzed edge detection as a critical component of many vision systems, including object detectors and image segmentation algorithms. Patches of edges exhibit well-known forms of local structure, such as straight lines or T-junctions. This paper takes advantage of the structure in local image patches to learn an accurate and	We predict local edge masks in a structured learning framework applied to random decision forests. Our novel approach to learning decision trees robustly maps the structured labels to a discrete space where standard information gain measures may be evaluated. A result is an approach that obtains real-time performance that is orders of magnitude faster than many competing state-of-the-art approaches while also achieving state-of-the-art edge detection results on the	**Strength:** The proposed solution operates in real-time, thereby providing security in an active state. **Weaknesses:** The system is fully reliant on face detection and recognition system. This makes it imperative for the user to be physically present and granted access to the ATM. As much as they provided an alternative with an OTP sent to the user in such circumstances, it is less effective in the real-world scenario.	Initially, their methodology achieved 71% ODS accuracy and increased to 75% ODS after the parameter sweep.

	computationally efficient edge detector. Finally, we show the potential of our approach as a general-purpose edge detector by showing our learned edge models generalize well across datasets [39]. **Datasets used in the project:** 1. BSDS500 Segmentation dataset 2. NYU Depth dataset			
4. Deep Learn Helmets-Enhancing Security at ATMs Author(s): K. Bavithra Devi, S. Mohamed Mansoor Roomi, M. Meena.	The authors took a comprehensive survey on ATM infrastructure in India, with over 1.2 billion people. Their survey took into perspective the perennial security challenges of ATMs. The author(s) advanced their analysis that real-time intelligent video analytics offers advanced monitoring capabilities that give sophisticated video surveillance to	The author(s) designed an image detection of the helmet using Deep Learning Convolutional Neural Network (CNN) architecture such as VGGNET (Visual Geometry Group) and ALEXNET. The helmet region is detected using (Region Convolutional Neural Network) RCNN with 15 layers. The performance of this technique has been tested on 880 test images out of 1880 images in a database. The parameters are chosen to compare the different mini-	**Strength**: The proposed solution operates in real-time, thereby providing security in an active state. **Weaknesses**: The system is fully reliant on a helmet detection and recognition system. Suppose the person's head is bald or any object resembling a helmet. A person without a helmet is detected with high accuracy.	The accuracy for helmet detection using ALEXNET is 96.03

	recognize abnormal activities. Persons wearing helmets in the ATM center is one of the anomalous activities. In such a scenario, an automatic helmet detection algorithm is required to alert the person wearing a helmet in ATM. [39].	batch sizes and epochs in ALEXNET		
5. Selective Search for Object Recognition Author(s): J. R. R. Uijlings · K. E. A. van de Sande · T. Gevers A. W. M. Smeulders	The authors in this paper addressed the problem of generating possible object locations for use in object recognition. Our selective search results in a small set of data-driven, class-independent, high-quality locations, yielding 99% recall and a Mean Average Best Overlap of 0.879 at 10,097 locations. Compared to an exhaustive search, the reduced number of locations enables more robust machine-learning techniques and appearance	The authors introduced selective search, combining the strength of an exhaustive search and segmentation. Like segmentation, their work used image structure to guide the sampling process. The exhaustive search was aimed at the capture of all possible object locations. Instead of a single technique to generate possible object locations, we diversify our search and use various complementary image partitioning to deal with as many image conditions as possible.	**Strength:** The proposed solution operates in real-time, thereby providing security in an active state. **Weaknesses:** The system is fully reliant on a helmet detection and recognition system. Suppose the person's head is bald or any object resembling a helmet. A person without a helmet is detected with high accuracy.	The selective search for object recognition is 99%

	models for object recognition. In this paper, we show that our selective search allows the powerful Bag-of-Words model for recognition. The selective search software is made publicly available[12].			
6. Anomaly Detection on ATMs via Time Series Motif Discovery. Dirk Walther, Maximilian Riesenhuber, Tomaso Poggio	The authors looked at a 2014 incident where skimming attacks on ATMs resulted in approximately 280 million Euros in losses within the European Union sub-region[40].	The authors used an innovative piezoelectric sensor network to capture the ATM state and analyze the occurring vibrations. The complex quad-tree wavelet packet transform inspected the captured signals, which provided them with a broad frequency analysis of a signal in various scales. Features were extracted from the selected scale based on the information content to detect motifs. The detected motifs provided the detected prototype patterns for anomaly detection or classification tasks.	**Strength:** The practical results showed that the proposed approach could classify normal and abnormal signals via motif discovery **Weaknesses:** The system is fully reliant on signal classification and does not and do not report on time.	Motif achieved 60.0% classification results, 58.8 F-Measure, 71.4 sensitivity, and 50.0 precision.

PAPER TITLE				
7. Automated Teller Machine Analysis under Host-Bank Systems through Telephone Network Author(s): Kuldeep Nagiya, Mangey Ram.	This work demonstrates the performance of an ATM network. The different types of component failure, such as an ATM, telephone network, power supply, etc., were taken for the study. The author(s) considered ATM functionality problems of power supply: 1. Power supply through electricity board and 2. Power supply through a generator. [41].	The various performance and reliability characteristics of the ATM network were accessed by using a supplementary variable technique, Laplace transformation, and the Markov process. The authors used mathematical modeling, Laplace transformation, supplementary variable techniques, and the Markov process	**Strength:** The reliability characteristics of the ATM network were found through the approach. **Weaknesses**: The technique relies on the power supply components of the ATM and not the essential components such as the card reader, dispenser, and others which are the main components that users of ATMs access. The technique is inefficient because detecting only power failure does not make the ATM defective.	The technique achieved the following reliabilities: At time 0, reliability is 1, at time 1; reliability is 0.985042, time 2, reliability is 0.970173; through time 15, with the reliability of 0.787659
8. ATM management prediction using Artificial Intelligence techniques: A survey Author(s): Seyed Mohammad Hossein	In this paper, the author(s) discussed forecasting cash demand, fraud detection, ATM failure, user interface, replenishment strategy, ATM	Artificial Intelligence (AI) techniques were discussed to detect fraud, failure, replenishment, and crash prediction. Several statistical methods used to evaluate these forecasts are also covered in this paper.	**Strength:** The techniques used yielded appreciable results **Weaknesses:**	The ANN model of 20.6 mean average percentage error (MAPE) was achieved, SVR achieved 25.1, Stepwise Autoregressive achieved 46.55,

Hasheminejad and Zahra Reisjafari	location, and customer behavior[42].	Moreover, we review AI techniques such as neural networks, regressions, and support vector machines and their results in graphs in different sections. The literature covered in this paper is related to the past ten years (2006-2016). The approaches studied in this paper are compared regarding data sets and prediction performance, accuracy, etc. We also provide a list of data sets available for the scientific community to research in this field. Finally, open issues and future works are presented in each of these items.	Different data sets were used for each technique; hence, there is no comparison point.	Holt-Winters Additive achieved 53.05, and Exponential Smoothening of 55.87.
9. Agent-Based Faults Monitoring in Automatic Teller Machines. Author(s): Bashir Sulaimon Adebayo, Mohammed Idris Kolo.	In this paper, the authors worked on ATM systems in Nigeria. The main area of this research was the challenges in maximizing the uptime of ATMs due to a wide gap in fault detection, notification, and correction of the ATMs. One way to alleviate this situation is	The authors proposed architecture for rule-based and intelligent agent-based monitoring and management of ATMs. The agents remotely monitor the ATMs and control functions such as software maintenance. A system administrator can securely modify agents' monitoring policies and control functions. The framework presented	**Strength**: Reduction in the mean time to repair (MTTR) by quickly isolating problems in critical business transactions. • Ability to use remote diagnosis information to minimize the number of trips made to the ATM. • Ability to monitor individual processes on the ATM and reset when necessary.	The paper is purely a software approach and does not have a prototype to determine the accuracy of the technique.

	through intelligent monitoring of ATMs by resident software agents that monitor the device and report faulty components in real-time to facilitate quick response [43].	includes a software fault monitor, hardware fault monitor, and transaction monitor. Finally, a set of utility support agents, caller, and log agents alert the network operator, log error, and transaction information in a database.	• Ability to dynamically update diagnosis rule by changing to remote diagnosis information to minimize the number of trips made to the ATM. • Ability to monitor individual processes on the ATM and reset when necessary. • Ability to dynamically update diagnosis rules with changing environments. **Weaknesses**: The agents residing on the ATM devices can declare the state of the ATM after the failures of the components, which does not minimize the downtime experience.	
10. ATM Availability Management System Author(s): Sujata Rao1 and Hrushikesh Mane2	The author(s) worked on the ATM monitoring process as a key ATM Availability Management Solution component. Handling the incidents related to ATMs and monitoring the device level health of the entire ATM fleet is the prime study of this paper. The Master View helps a financial	Various reports and statistics about the transactions across multiple demographics provide helpful information like a) the cities using ATMs heavily, b) the most popular transactions, c) ATMs having heavy transaction volumes, etc. Monitoring ATMs helps significantly identify the various causes behind slow or failed customer transactions to reduce service incidents and maximize the ATM uptime	**Strength**: The Master View Resolve (MVR) can carry out maintenance tasks. **Weaknesses:** The MVR has the limitation of detecting ATM problems at the onset.	Master View Resolve (MVR) can handle ATM network failure of more than 90%.

	institute improve the availability of their self- service terminals and minimize downtime. [44].	through Master View Resolve (MVR).		

2.7 Conclusion and Summary of Literature Review

The body of academic literature in the aforementioned papers presented in this chapter proves a point for the need to adopt the machine learning approach for this work. The machine learning approach provides an avenue for a dynamic, efficient, lightweight methodology. From the literature, some significant setbacks in using a machine learning approach are the lack of a dataset for specific problems and a biased dataset in some instances. These challenges have been solved using some machine learning techniques and also the use of synthetic data. In effect, the primary technique for this work uses machine learning models for incident detection.

CHAPTER THREE

METHODOLOGY

3.0 Introduction

This chapter discusses the research strategy, the research method, the research approach, the methods of data collection, and the sample selection. It also includes the research process, the type of data analysis, the ethical considerations, and the research limitations of the project. This study involves a field survey and software development:

3.1 Proposed System Design

The flowchart for the SRSAID system design is shown in figure *3.1. The figure shows the various stages in blocks as well as the processes involved for each block.*

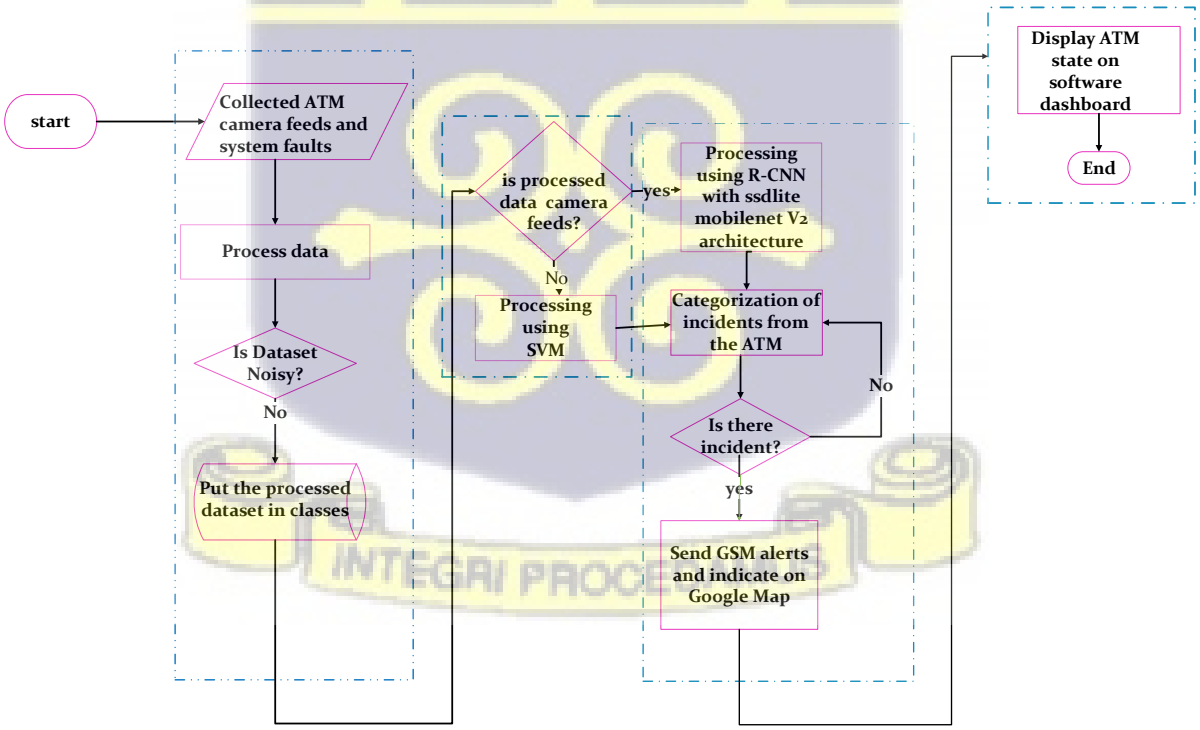


Figure 3. 1 Flowcharts for Proposed Design

3.2 Field Survey

A field survey was conducted to solicit information on the user experience of persons who have used ATMs over the past 12 months. The population group for this survey was a cross-section of bank customers on the University of Ghana campus, some of the banks' staff, and NCR staff. A structured questionnaire was presented to the selected people, and the answers provided were used in the analysis.

The main reason for using these instruments was to collect enough firsthand information from respondents. Drawing from [45], it is argued that with a semi-structured interview, the interviewer has more freedom to pursue his idea and can improvise the questions [46] confirmed the use of the interview by stating that it is a face-to-face questioning of respondents to obtain information. The study was based on primary and secondary sources of data.

The primary source of data was obtained from interviews and questionnaires, which were administered to ten (10 officials of NCR and ten (10) clients and staff from different banks who are using the electronic-banking service (ATMs) of the banks. Secondary data was also collected from research reports, Agricultural Development Bank (ADB), Ghana, Ghana Commercial Bank (GCB), Barclays Bank, Ghana, Republic Bank, Ghana, and other published materials.

The purposive sampling method that allows the researcher to select particular participants needed for specific information was adopted to select the NCR officials directly related to ATM banking services of all the banks in Ghana. Ten (10) officials of NCR, Ghana, which included the manager of the ATMs in Ghana, the Head of engineering, and eight (8) staff from NCR as

primary respondents. Random sampling was adopted to select 10 customers of different banks that use NCR ATMs.

3.2.1 Data Collection

Computer vision base approaches depend on collected data or image datasets. It is essential to analyze object features and to review the performance of detection algorithms. Some databases are available for object, character, and scene recognition[34]. There is no database for ATM incidents (different images showing security and defect incidents). Hence, thousand (1000) plus images (images showing the various ATM security incidents) were collected from the internet, and the faults dataset in excel format was collected from the ATM service provider (NCR).

3.3 Data Pre-processing

3.3.1 Security Incident Data Pre-processing

The data preprocessing stage is the first significant stage in developing the incident detection system. This stage involves using data mining techniques to transform the data from its raw form into the required format used by the convolutional neural network classifier (CNN-C) to detect and identify ATM security incidents. The data preprocessing stage involves removing unwanted images and creating bounding boxes to select the parts of the images that fit for security incidents. This ensures that only valid and relevant information is extracted for the next process.

Before the data preprocessing, images were downloaded from the internet and others with TECHNO POP 2 Plus camera into a folder on the raspberry pi named image2. The image data preprocessing involves these steps:

- Image data filtering and selection
- Feature selection and extraction
- Feature adjustment.

3.3.2 Image Dataset and Description

The security incident experimental setup used one thousand, one hundred and sixty-five (1,165) image datasets. One thousand, one hundred and fifty-five were obtained from the internet, and the ten were the images captured with the raspberry pi for testing purposes. The images from the internet depict all the activities of security incidents of the ATM. For this study, the multiclass datasets sourced from the internet were improved and grouped into a multiclass classification problem. Table 3.1 shows a summary of the image datasets and their class distributions:

Table 3. 1 Summary of Image Dataset

Dataset	Number of Attributes
ATM out of service	62
card trapping	50
cash trapping	64
Jackpotting	46
logical attack	44
Malware	78

Occlusion	73
physical attack	204
Robbery	126
Skimming	55
transaction reversal fraud	50

Samples of Skimming Attack Images

Samples of Physical Attack Images

Samples of Robbery Images

Figure 3. 2 Sample Images for ATM Incident Attacks

Defects datasets

Eleven thousand, four hundred and fifty-two (11452) 2018 ATM defects incidents were used in the experimental setup obtained from NCR, Ghana. The NCR defects datasets consist of various ATM systems technical problems encountered in 2018, obtained through the company's database or logs. The multi-class datasets from the NCR Ghana were modified into binary classification problems for this study. Figure 3.3 shows part of the multiclass datasets used by the company and their class distributions:

```
In [19]: data.head(10)
```

Out[19]:

	Area_of_Failure_Description	Count1	Cause_Code_Description_Global	Count2	Repair_Code_Description_Global	Count3	Problem_Code_Description	Count4
0	WCS SPECIAL USAGE	3700.0	NCR NON-REMEDIAL WO	3700	NON-REMEDIAL CALL	3701.0	NON-REMEDIAL CALL	3742.0
1	DISPENSER-PICK MOD	1639.0	PICK MECHAN TIMING	188	ADJUST/CALIBRATE	2316.0	DISPENSE PROBLEM	4060.0
2	SOFTWARE	719.0	S/W OS/WINDOWS	245	REIMAGE	31.0	SCREEN/DISPLAY	281.0
3	CARDREAD-MOTORIZED	977.0	CR-M READ/WRITE HEAD	413	CLEANED	1739.0	CARD READER JAM	957.0
4	RECYCLER/GBNA/GBRU	210.0	PICK DRIVE MECHANISM	537	RESEAT PART/MODULE	100.0	DEPOSITORY PROBLEM	447.0
5	DISPLAY/TOUCH	234.0	GREC NOTE VALIDATOR	50	INSTALL NEW SW/FW/CW	229.0	OPSYS PROBLEM	692.0
6	PRINTER-RECEIPT	369.0	GREC SEPARATOR	39	REPLACE DEF PART	1865.0	RECEIPT PRINTER PROB	291.0
7	DISPENSR-PRESENTER	2294.0	MNTR CABLE/CONNECT	49	ALIGNED MODULES	369.0	PIN PAD FAILURE	191.0
8	KYB/ ENCRYPT DEV	197.0	RPT EXIT TRAN/PRES	52	CLEAR MEDIA/DEBRIS	305.0	HYGENIC CLEAN REQD	10.0
9	CORE ELECTRONICS	230.0	CR-M SHUTTER	197	RELOAD SW/FW/CW	232.0	UNDEFINED PROBLEM	193.0

Figure 3. 3 Summaries of defect incident datasets

Figure 3. 4 Plot of summaries of defect incident datasets

In addition to the images, data obtained from NCR Ghana for ATM Defects was used for the system defect aspect of this project.

3.4 Machine Learning Classification Algorithms

To assess the effectiveness and efficiency of the proposed technique, Machine Learning techniques are used. These algorithms were Regional Convolutional Neural Network (R-CNN) and Support Vector Machines (SVM) for defects classifications. These Machine Learning algorithms are considered: Deep learning Regional Convolutional Neural Network (R-CNN), which regression model for image classifications fine-tuned, and Support Vector Machines (SVM) for defects classifications.

3.4.1 Regional Convolutional Neural Network

R-CNN forward computation has several stages, shown in figure 10. First, the regions of interest (RoIs) are generated. The RoIs are category-independent bounding boxes that are highly likely to contain an exciting object [10]. This study uses a different method called the integrated method, which works similarly to selective search [20] [12], to generate these through Labelimg. These methods are discussed in further detail in figure 3.5. Next, a convolutional network extract features from each region proposal. The sub-image contained in the bounding box is warped to match the input size of the CNN and then fed to the network. After the network has extracted features from the input, the features are input to a faster regional convolutional network (F-RCNN) that provides the final classification.

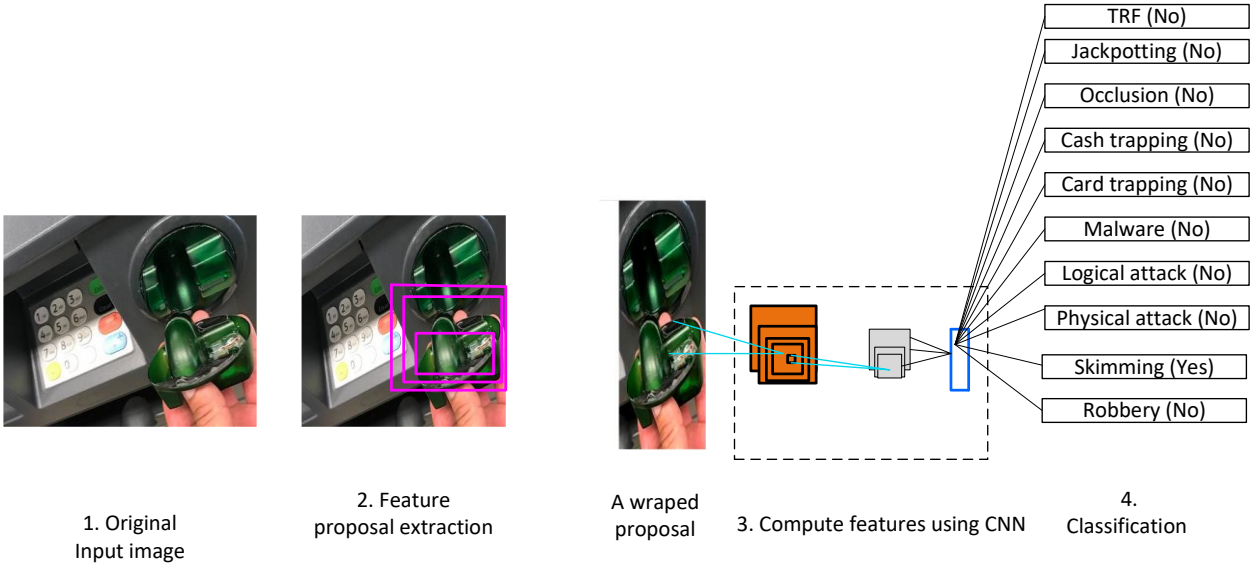


Figure 3. 5 Stages of R-CNN forward computation

The method is trained in multiple stages, beginning with the convolutional network [10][14][17]. After the CNN has been trained, the Faster-RCNN is fitted to the CNN features. Finally, the region proposal-generating method is trained.

Fast R-CNN

The method receives an input image plus regions of interest computed from the image. As in R-CNN, the RoIs are generated using an external method [10][14][17]. The image is processed using a CNN containing several convolutional and max-pooling layers. The convolutional feature map generated after these layers is input to an RoI pooling layer, as shown in the figure. This extracts a fixed-length feature vector for each RoI from the feature map [20]. The feature vectors are then input to fully connected layers that are connected to two output layers: a SoftMax layer that produces probability estimates for the object classes and a real-valued layer that outputs bounding box co-ordinates computed using regression (meaning refinements to the initial candidate boxes)[20] [47][12][10].

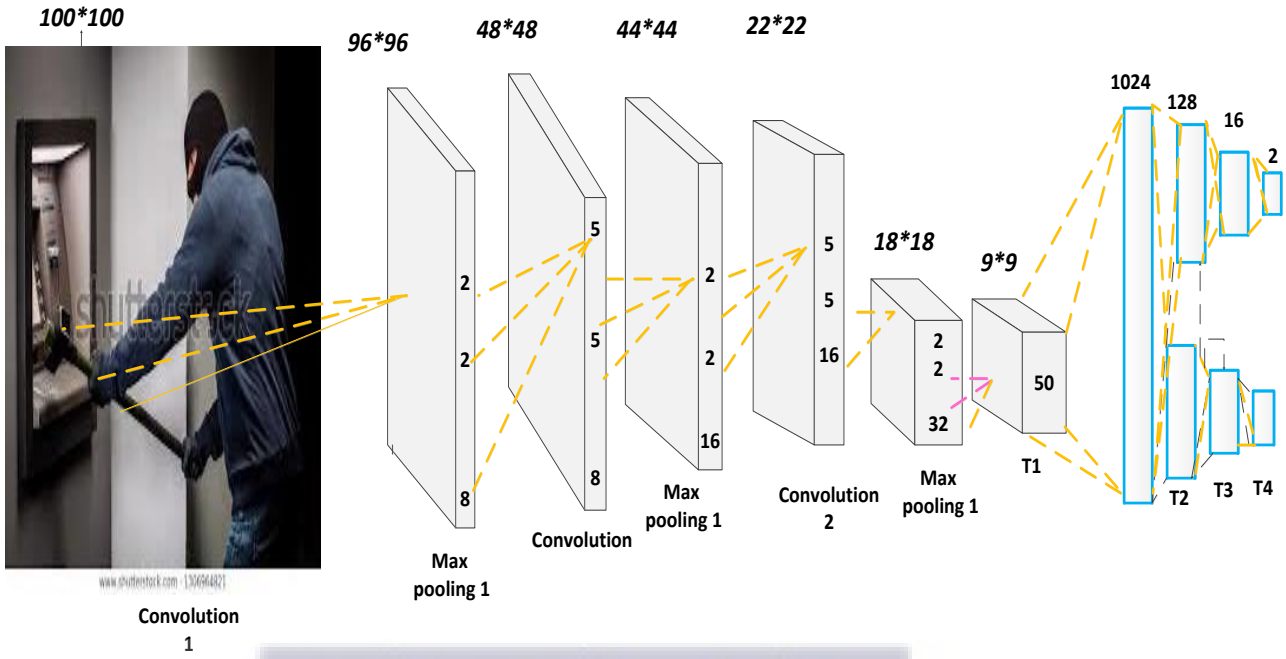


Figure 3. 6 RoI Pooling Layers

3.4.2 Support Vector Machines

Support Vector Machines (SVM) is a machine learning tool for classification and regression. Support Vector Machine is based on supervised learning, which classifies points to one of two disjoint half-spaces [24][22][21]". It uses nonlinear mapping to convert the original data into a higher dimension. Its objective is to construct a function that correctly predicts the class to which the new and old points belong. With an appropriate nonlinear mapping, two data sets can always be divided by hyperplane.

Hyperplane separates the tuples of one class from another and defines decision boundaries. Many hyper planes separate the data, but only one will achieve maximum separation. The main reason behind maximum margin or separation is that if we use a decision boundary to classify, it may end up nearer to one set of datasets than others[21][24]. This was the case when data is

linear, but we mostly find that data is non-linear and the data set is inseparable, so we use kernels.

The core purpose of SVM is to separate the data with decision boundaries and extend it to non-linear boundaries using kernel trick [24]. The significant benefit of SVM is its versatility meaning that different kernel functions can be specified for the decision function. Available kernels are provided, but it is also possible to specify custom kernels. SVM becomes prominent when we use pixel maps as input; it gives accuracy equivalent to neural networks with elaborated features in a handwriting recognition task. Support vector machine is used for many applications, such as text categorization, pattern recognition, face recognition, and handwriting analysis, especially for classification and regression applications.

Neural Networks are more accessible to apply than support vector machine, but sometimes it provides unsatisfactory results. For example, even in perceptron learning algorithms, gradient descent is slower than SVM learning. SVM is unbeaten when used for pattern classification problems. One of the significant challenges is choosing a suitable kernel for a given application [24]. But there are many standard or default choices, such as Gaussian or polynomial kernel, but if these prove worthless, more elaborate kernels are needed. Traditional Classification approaches perform poorly when working directly because of high data dimensionality, but support vector machines can avoid very high dimensionality representations. Support vector machine is the most promising technique and approach compared to others. Support vector machine scales fairly well to high dimensional data, and the trade-off between classifier complexity and error can be controlled explicitly. Another benefit of SVMs and kernel methods is that one can design and use a kernel for a particular problem that could be applied directly to the data without needing a feature extraction

process. It is imperative in situations where a lot of data structure is lost by the feature extraction process. An example is text processing. Limitations of SVM are speed and size in training and testing [24][21]. Discrete data presents another problem. The most severe difficulty with SVMs is the high algorithmic complexity and extensive memory requirements. The development of SVM is utterly different from standard algorithms used for learning, and SVM provides fresh insight into this learning.

SVM is an excellent example of supervised learning that maximizes generalization by maximizing the margin and supports kernelization's nonlinear separation. SVM tries to avoid overfitting and underfitting. The margin in SVM denotes the distance from the boundary to the closest data points in the feature space. Given the 2018 incident dataset correspondingly to $Xn : R^n \in F$ in the feature space F .

The calculated linear hyperplane dividing them into two labeled classes (problem code description and all other classes) can be mathematically obtained as:

$$w^T x_i + b = 0, w \in R^n, b \in R \qquad (3.1)$$

Assuming the training dataset is correctly classified, as shown in *figure 3.7:*

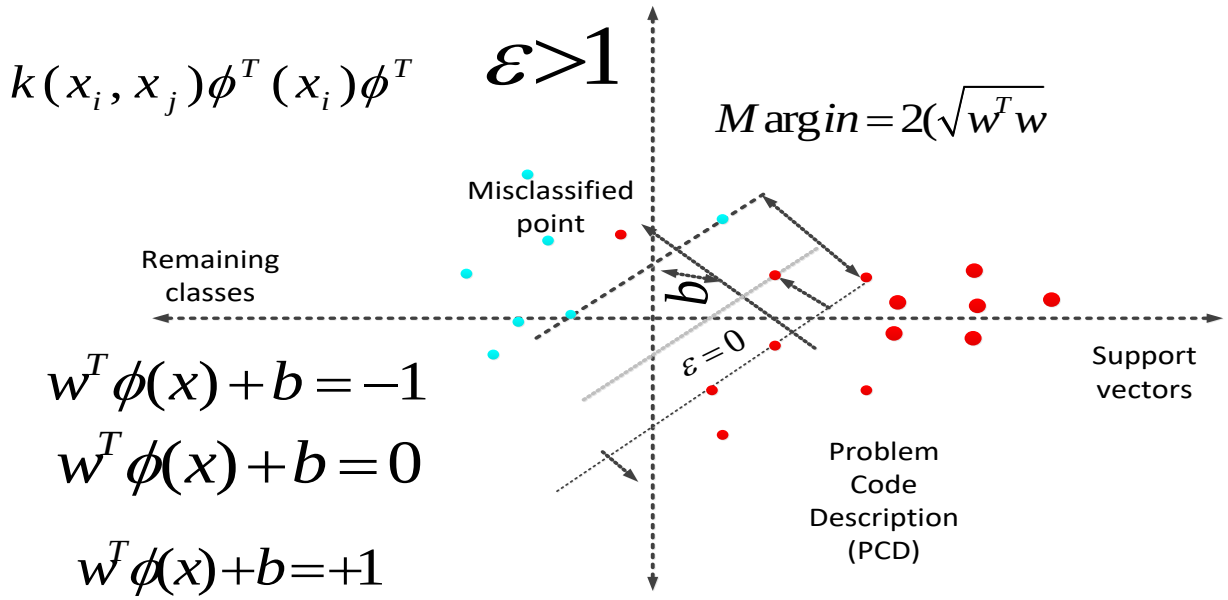


Figure 3. 7 Standard formulation of SVM

This means the SVC computes the hyperplane to maximize the margin separating the classes (problem and all other classes). The SVC is a hyperplane that separates the problem state from all other classes with a maximum margin in the simplest linear form. Finding this hyperplane involves obtaining two parallel hyperplanes, as shown in Figure 3.7 above, equal distance to the maximum margin. If all the training dataset satisfies the constraints as follows:

$$\begin{cases} w^T x_i + b \le 1, \text{for } y_i = +1 \\ w^T x_i + b \ge -1, \text{for } y_i = -1 \end{cases} \quad (3.2)$$

Where ω is the normal to the hyperplane, is $|b|/w$ the perpendicular distance from the hyperplane to the origin, and ||w|| is the Euclidean norm of w. The separating hyperplane is defined by the plane $w^T x_i + b = 0$, and the above constraints in *(2)* are combined to form:

$$y_i\left(w^T x_i + b\right) + 1 \quad (3.3)$$

The pair of hyperplanes that gives the maximum margin (c) can be found by minimizing $||w||2$ subject to constraint in (9)". This leads to a quadratic optimization problem formulated as:

Minimize

$$f(w,b)=\frac{\|w\|^2}{2} \tag{3.5}$$

Subject to

$$y_i\left(w^T x_i + b\right) \geq 1, \forall i = 1,\ldots\ldots\ldots n \tag{3.6}$$

This problem is reformulated by introducing Lagrange multipliers, $\alpha_i\left(i=1,...,n\}\right)$ for each constraint and subtracting them from the function". This $f(x)\left(w^T x_i+b\right)$ results in establishing the primal Lagrangian function:

$$L_P(w,b,\alpha)=\frac{w^2}{2}+\sum_i^n\left(\alpha_i\left(1-y_i\{(w^T x+b)\}\right)\right), \tag{3.7}$$

$$\forall i=1,\ldots\ldots\ldots n$$

Taking the partial derivatives of $L_P(w,b,\alpha))$ with respect to $w,b\,\&\,\alpha,$ respectively, and applying the duality theory yields:

$$\frac{\partial L_P}{\partial w}=0\rightarrow w=\sum_{i=1}^n \alpha_i\, y_i\, x_i \tag{3.8}$$

The problem defined in (5) is a quadratic optimization (QP) problem. "Maximizing the primal problem L_P with respect to αi, subject to the constraints that the gradient of L_P with respect to w and b vanish, and that $\alpha_i \geq 0,$ gives the following two conditions":

$$w=\sum_{i-1}^n \alpha_i\, y_i\, x_i$$

(3.9)

$$w = \sum_{i-1}^{n} \alpha_i \, y_i = 0, \tag{3.10}$$

Substituting these constraints gives the dual formulation of the Lagrangian:

$$maximize_{\alpha} \quad L_P(w,b,\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \, \alpha_j \, y_i \, y_j \left(x_i \, x_j \right), \tag{3.11}$$

$$\text{subject to } \sum_{i-1}^{n} \alpha_i \, y_i = 0, \alpha \geq 0; i = 1, ...n \tag{3.12}$$

But the values of $\alpha_i, w,$ and b are obtained from these respective equations, namely:

$$w = \sum_{i-1}^{n} \alpha_i y_i x_i \tag{3.13}$$

$$b = \frac{1}{2} \left(Min_i y_i = 1 w^T x_i + Max_i \, y_i = 1 w^T x_i \right) \tag{3.14}$$

Also, the Lagrange multiplier is computed using the following:

$$\alpha \left(1 - y_i \left(w^T x_i + b \right) \right) = 0 \tag{3.15}$$

Hence, this dual Lagrangian is *LD* maximized with respect to its nonnegative α_i to give a standard quadratic optimization problem". "The respective training vectors are called support vectors. With the input dataset x_i as a nonzero Lagrangian multiplier α_i,

$$y_i \left(w^T x_i + b \right) = 1 \tag{3.16}$$

The equation above gives the support vectors (SVs). Although the SVM classifier can only have a linear hyperplane as its decision surface, its formulation can be extended to build a nonlinear SVM. SVMs with nonlinear decision surfaces can classify nonlinearly separable

data by introducing a soft margin hyperplane, as shown in Figure 3.8: Introducing the slack variable into the constraints yields:

$$w^T x_i + b \geq 1 - \xi_i, \text{for } y_i = 1,$$

$$w^T x_i + b \geq -1 + \xi_i, \text{for } y_i = -1,$$

$$\xi_i \geq 0 \forall i. \tag{3.17}$$

These slack variables help to find the hyperplane that provides the minimum number of training errors. "Modifying equation (4) to include the slack variable yields:

$$\underset{\alpha, b, \xi}{Mininmize} \quad \frac{w}{2} + C\sum_{i-1}^{n} \xi_i$$

$$\text{subject to} \quad \alpha_i \left(1 - y_i (w^T x_i + b)\right) + \xi_i - 1 \geq 0, \xi_i \geq 0. \tag{3.18}$$

The parameter C is a regularization parameter that trades off the wide margin with few margin failures. The parameter C is finite. "The larger the C value, the more significant the error". "The Karush–Kuhn–Tucker (KKT) conditions are necessary to ensure the optimality of the solution to a nonlinear programming problem;

$$\left(y_i \left(w^T x_i + b\right)\right) - 1 \geq 0, \quad 1 = 1, 2, \ldots . l, \ \forall i,$$

$$\alpha_i \left(y_i \left(w^T x_i + b\right)\right) - 1 = 0, \alpha \geq 0, \forall_i \tag{3.19}$$

The KKT conditions for the primal problem are used in the non-separable case, after which the primal Lagrangian becomes:

$$L_P = \frac{w^2}{2} + C\sum_{i=1}^{n} \xi_i + \left(\alpha_i \left(1 - y_i \{w^T x_i + b\}\right)\right) + \sum_{i=1}^{n} \beta_i \xi_i \tag{3.20}$$

With βi as the Lagrange multipliers to enforce positivity of the slack variables $\left(\xi_i\right)$ and applying the KKT conditions to the primal problem yields":

$$\partial L_P = w_u = -\sum_{i-1}^{n} \alpha_i y_i x_{iu} = 0,$$

$$\frac{\partial L_P}{\partial w_u} = C - \alpha_i y_i = 0 \tag{3.21}$$

$$\frac{\partial L_P}{\partial wu} = C - \alpha - \beta_i y_i = 0,$$

$$\alpha_i \left(y_i (\{ w^T x_i + b \}) \right) - 1 + \xi = 0$$

$$y_i \left(w^T x_i + b \right) - 1 + \xi > 0,$$

$$\alpha_i \beta_i \xi_i \geq 0, C, \xi_i \geq 0, \tag{3.22}$$

$$i = 1, 2, \ldots\ldots n \ \ and \ \ u = 1, 2, \ldots\ldots, d,$$

$$\frac{\partial L_P}{\partial w_u} = w_u - y_i x_{iu} = 0,$$

$$\frac{\partial L_P}{\partial w_u} = C - \alpha_i y_i = 0, \tag{3.23}$$

Where the parameter d represents the dimension of the dataset". "Observing the expressions obtained above after applying KKT conditions yields $\xi = 0$ for $\alpha_i < C, \text{since } \beta_i = C - \alpha_i \neq 0$". This implies that any training point for which $0 < \alpha_i < C$ will be taken to compute for b as a data point that does not cross the boundary:

$$y_i \left(w^T x_i + b \right)^{\alpha_i = 0} - 1 + \xi > 0. \tag{3.34}$$

This does not participate in the derivation of the separating function with $\alpha_i = C \, and \, \xi_i > 0.$

$$y_i \left(w^T x_i + b \right)^{\alpha_i = 0} - 1 + \xi > 0. \tag{3.25}$$

Nonlinear SVM maps the training samples from the input space into a higher-dimensional feature space via a kernel mapping function F". In the dual Lagrangian function, the kernel function replaces the inner products:

$$\left(\phi(x_i).\phi(x_j)\right) = k\left(x_i x_j\right). \tag{3.26}$$

Effective kernels are used in finding the separating hyperplane without high computational resources. "The non- linear SVM dual Lagrangian:

$$L_P(\alpha) = \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j k\left(x_i.x_j\right),$$

subject to

$$\sum_{i-1}^{n} \alpha_i y_i = 0, 0 \le \alpha_i \; i = 1,...n. \tag{3.27}$$

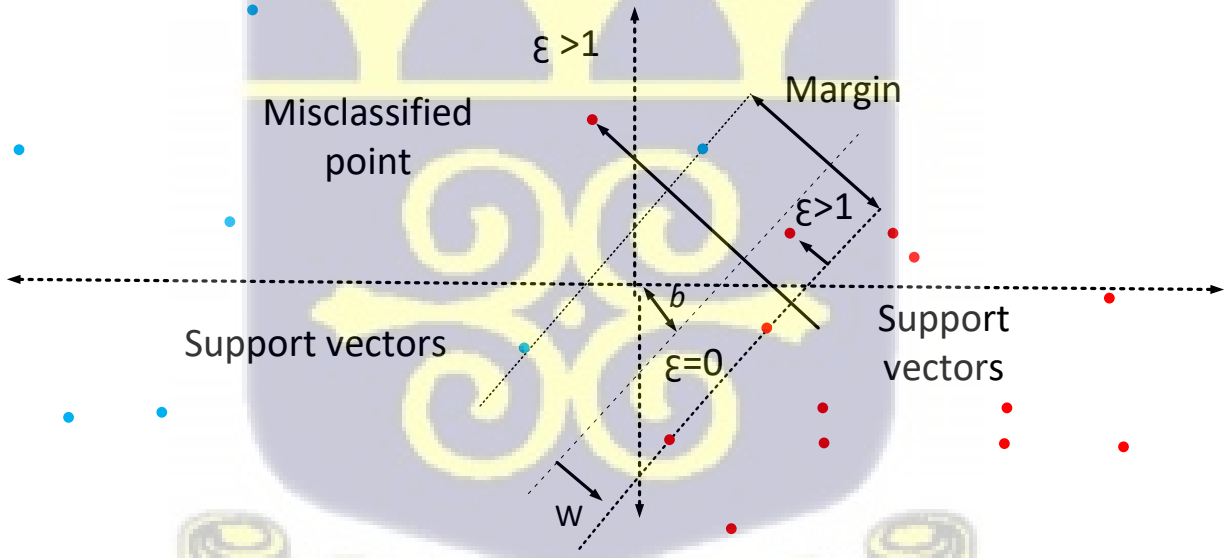


Figure 3. 8 "Linear separating hyperplanes for the nonseparable case of SVC by introducing the slack variable (ξ)".

In this study, the dataset being used is multiclass. With a multiclass dataset, the machine should classify an instance as only one of three classes or more. In this case, the classification

is done here as in figures 3.7, 3.8, and 3.9, respectively classifying the defect incident dataset as problem code description (PCD), area of failure description (AFD), and cause code description (CCD.

This is like that of the generalized linear case". "The nonlinear SVM separating hyperplane is illustrated in Figure 3.9 with the support vectors, class labels, and margin". "This model can be solved by optimization in the separable case". "Therefore, the optimal hyperplane has the following form:

$$f(x) = \sum_{i-1}^{n} \alpha_i y_i k(x_i, x) + b, \tag{3.28}$$

Where b is the decision boundary from the origin. "Hence, separating newly arrived dataset x implies that

$$g(x) = sign(f(x)). \tag{3.29}$$

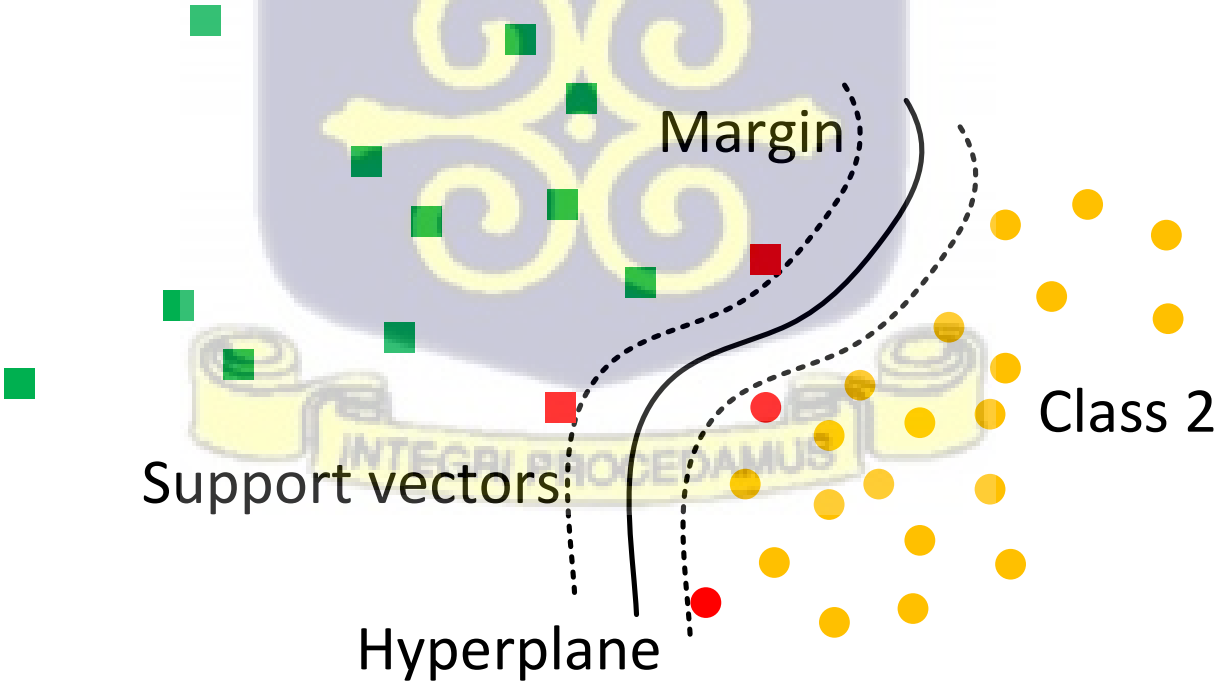


Figure 3. 9 Nonlinear separating hyperplane for the nonseparable case of SVM

However, feasible kernels must be symmetrical, which implies; the matrix *K* with the component $k\left(x_i, x_j\right)$ is positive semi-definite," as indicated in the table below:

Table 3. 2 Kernel Functions

Kernel	Parameter	Kernel Function
Linear	C	$k\left(x^T x_i\right)$ or $k\left(x_i, x_j\right) = x_i.x_j$
Radial Basis Function	$y \in R$	$\exp\left(\frac{-\| x_i - x_j \|^2}{2\sigma^2}\right)$ or $k\left(x_i, x_j\right) = e^{-\gamma\|x_i, x_j\|^2}$
Polynomial	$C \in R, d \in N$	$\left(x_i^T x_j + r\right)^d Or$ $k\left(x_i, x_j\right) = \left(x_i.x_j + C\right)^d$

The training of SVMs used the solution of the quadratic programming optimization problem. The above mathematical formulations form the foundation for developing and deploying support vector machines as the decision support tool for detecting and classifying ATM system defect incidents. In recent times, the decision-making activities of knowledge-intensive enterprises depend holistically on the successful classification of data patterns, despite the time and computational resources required to achieve the results due to the complexity associated with the dataset and its size.

SVM does not support multiclass classification natively. It supports binary classification and separating data points into two classes. The same principle discussed above is utilized for

multiclass classification after breaking down the multi-classification problem into multiple binary classification problems[48].

The idea is to map data points to high dimensional space to gain mutual linear separation between every two classes. This is called a *One-versus-One (OVO)* approach, which breaks down the multiclass problem into multiple binary classification problems, here, a binary classifier per each pair of classes. Another approach utilizes *One-versus-Rest* (OVR) or *One-versus-Rest* (OVR). This approach sets the breakdown to a binary classifier per class [48]. In this study, OVO was the option taken, and a hyperplane is needed to separate between a class and all others at once, as indicated in *figure 3.10*.

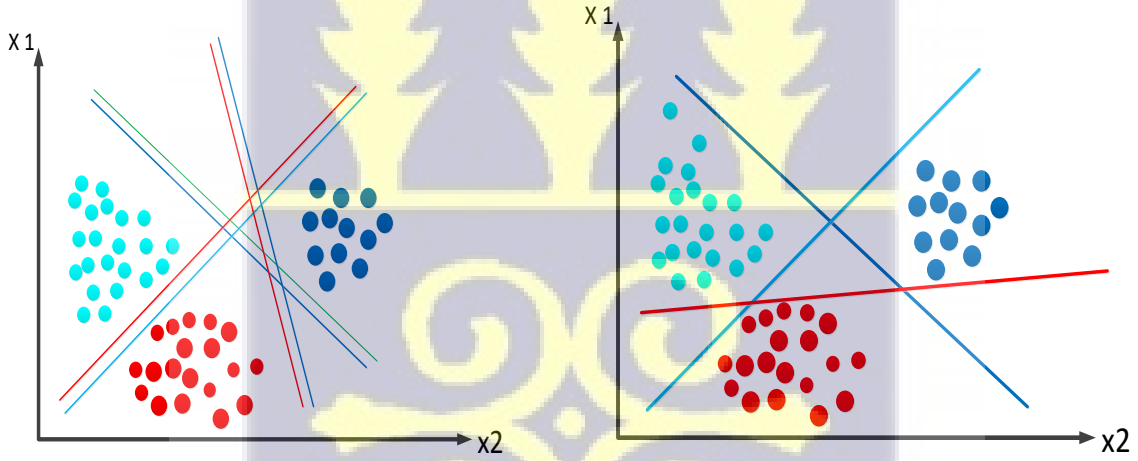


Figure 3. 10 OVO approach on multiclass

Figure 3. 11 OVR approach on multiclass

Figure 3.10 means the separation takes all points into account, dividing them into two groups; a group for the class points and a group for all other points. The green line tries to simultaneously maximize the separation between green points and all other points. For the OVO technique on the ATM system defect multiclass dataset:

$$\frac{m(m-1)}{2}$$

This would lead to binary SVM classification problems, and each one is trained on data from two classes. Figures 3.10 and 3.12 depict the OVO multiclass SVM.

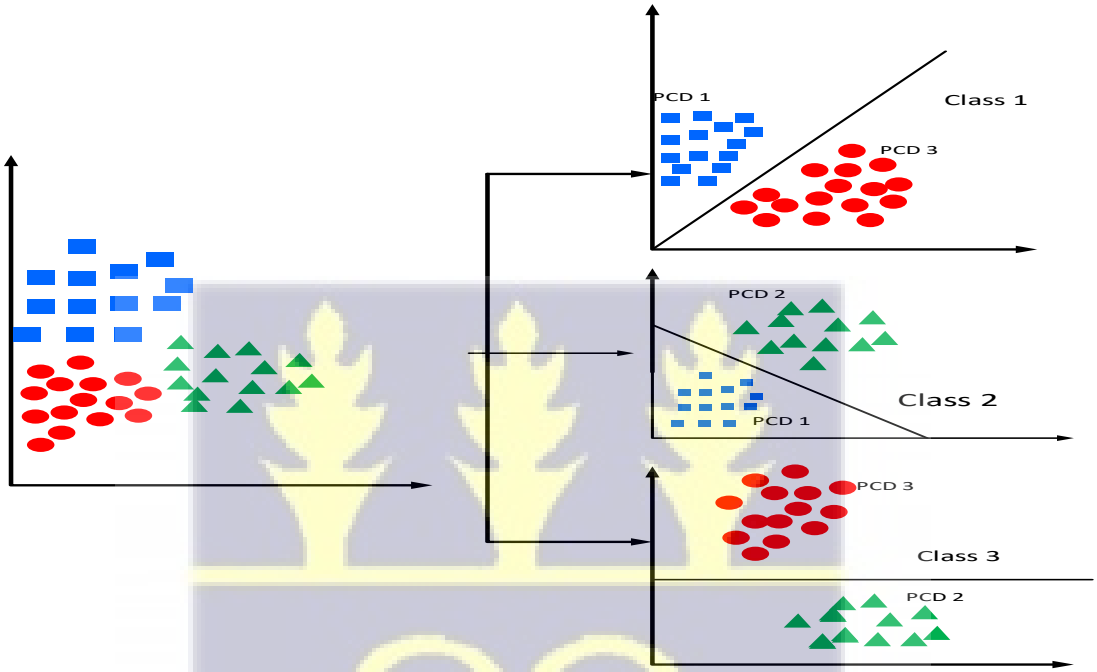


Figure 3. 12 OVO approach on multiclass taking all points into account.

For training the ATM system incident dataset from the i[th] and j[th] classes, the problem is formulated as follows:

$$\min_{w^{i,j},b^{i,j},g^{i,j}} \quad \frac{1}{2}\left(w^{i,j}\right)+C\sum_t \xi_t^{i,j}$$

$$\text{subject to } \left(w^{i,j}\right)^T x+b^{i,j}) \geq 1-\xi^{i,j} \text{ for } y=1,$$

$$\left(\left(w^{i,j}\right)\right)^T x+b^{i,j}) \leq -1+\xi^{i,j} \text{ for y =-1}$$

$$\xi_{i,j} \geq 0 \tag{3.30}$$

Where $t \in x_i, j$ and x_i, j is the set of data points with labels i *and* j.

All the classifiers used in this study are implementations from the *scikit-learn* Python library. Unless otherwise stated, the default parameters for all classifiers were used.

3.5 Tools Used

3.5.1 Software Component

In this study, the tools used include a Python IDE running on a Raspberry pi for security incident detection and Google Colaboratory running on 64-bit Windows 10 for defect detection and scikit-learn, a machine learning module written in Python with fully integrated a wide range of machine learning algorithms [35][49]. Python is a relatively simple interpretive language that allows writing more readable and typically much shorter code than equivalent C or C++ programs[35]. The classification algorithms used from the scikit-learn library include Support Vector Machines, Random Forest, and Decision Tree for the defect aspect of this study. The default parameters for these classifiers were used in this study. Labellmg platform was used as well for the region of interest generation.

Django, a high-level Python Web Framework, was used to write the codes for the dashboard and API to show the results.

3.5.2 Hardware Component

Raspberry pi

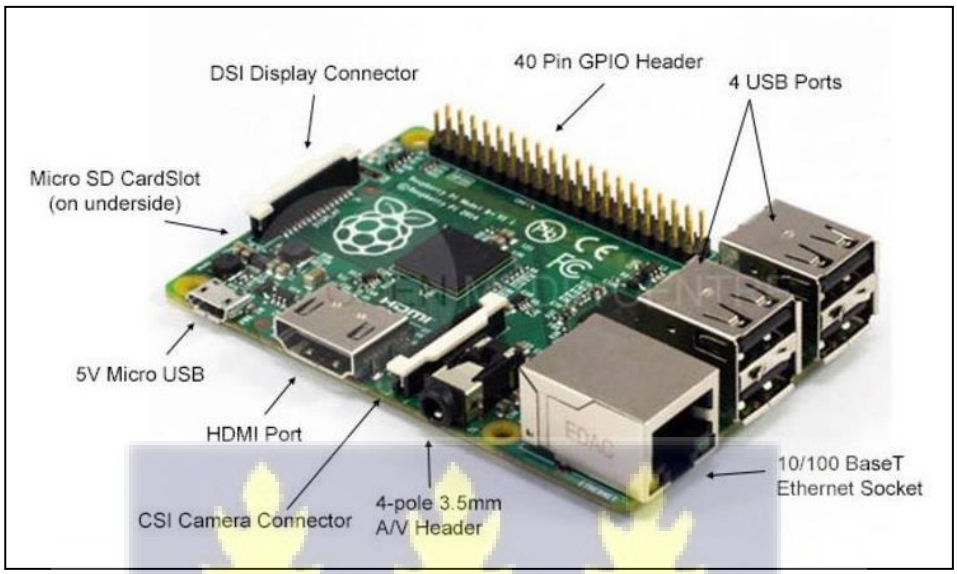


Figure 3. 13 Raspberry pi 3 model B

Raspberry pi is an efficient and cost-effective credit-sized computer developed by the United Kingdom – Raspberry pi Foundation to enlighten and empower engineering projects. Its core architectures are based on Broadcom BCM 2837B0 System on Chip (SOC)". "The processor is Broadcom BCM 2387 chipset with 1.2GHz Quad-core ARM cortex-A53 (64-bit). The PI contains a USB Micro SD card which includes the operating system.

The Raspberry Pi boots from the SD card running Linux or Raspbian operating system, which holds programs for the compilation and running of projects and the documents for the project.

The raspberry pi serves as the storage location for codes for security incident detection, defect incident detection, GPS, pi camera, and the GSM modules, which form the IoT components.

The Pi Camera

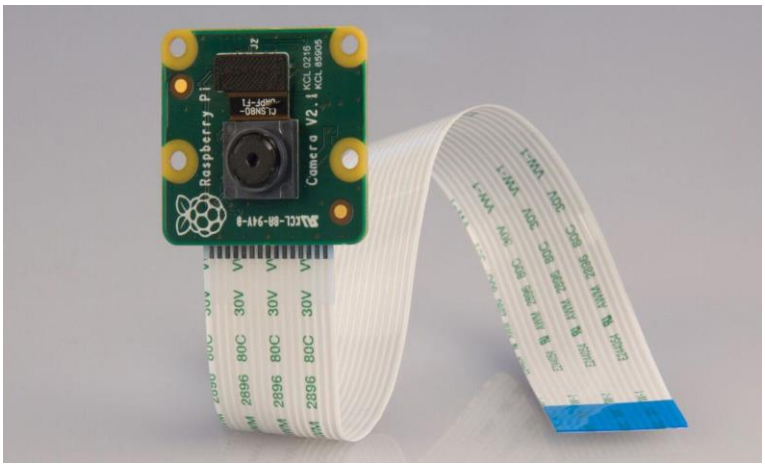


Figure 3. 14 Pi Camera

The PI camera communicates with Pi using the MIPI camera serial interface protocol. It has a 5MP colour camera module, MIPI Camera serial interface, OmniVision 5647 Camera Module, Resolution: 2592 x 1944, and supports 1080p, 720p, and 480p and Less payload.

It provides a pure Python interface to the raspberry pi. For this project, the Pi camera is responsible for capturing image processing and the deep learning algorithms used.

Global Positioning System (GPS) and Global System for Mobile Communication (GSM) Module

Figure 3. 15 GPS and GSM module

Figure: 3.15 shows the GPS and GSM modules used in this project. It is GPS/GSM module SIM808 from SIMCOM. It has high GPS receive sensitivity which is ideal for GSM applications. It supports GPS/GSM Quad – Band Network and combines GPS technologies for satellite navigation. It is used in this project to establish communication between a mobile device (user's mobile phone) and the raspberry pi, the GSM, and the GPS.

The GPS is a navigation system using satellites, receivers, and algorithms to synchronize location, velocity, and time data for sea, air, and land travel. The GPS is used in this project to locate or determine the position of ATMs based on their latitude and longitude and track or monitor objects. Users' movements with the ATMs about the incidents reported. The latitude and the longitude are coordinate systems through which the position or the location of any place on the earth's surface can be determined and described. The latitude for the positions of the ATMs with serial numbers given in the listing codes was calculated using this expression:

DD / MM / SS

Where

DD represents degrees, MM represents minutes, and SS represents seconds.

AND

The longitudes used the expression: DD / MM.MMM.

Where the DD refers to degrees and MM.MMM, decimal minutes.

GPS or GSM module SIM 808 uses serial communication, sending information one at a time or in series. Based on command, it determines where the signal should go, either GSM or GPS. An application that uses serial communication, such as Python, is used too. Mini-com was used to test. Mini-com is a serial communication program.

The GSM

The GSM is an open digital technology for transmitting mobile voice and data services. It operates at 850MHz, 900MGHz, 1800MGHz, and 1900MGHz frequency bands. It is used for sending information from the ATM to the ATM security or the owner of the ATM.

3.5.3 System Specifications

The system specification of the platform on which the proposed technique was developed is shown in the table below.

Table 3. 3 System Specifications

Hardware	**Think pad Laptop**	**Raspberry pi model B**
Operating System	Windows 10	Raspbian
System Architecture	64-bit	1.2GHz CPU
RAM	I6 Gigabytes	512MB
CPU	Intel (I) Core (TM) i5 – 8250U	Broadcom BCM2835 Single core (ARM11) SoC

3.6 Experimental Method

3.6.1 Security Incident Detection Training

To use R-CNN and Fast R-CNN for the training, methods for generating the class agnostic regions of interest are needed, namely Multi-Box and integrated methods. Convolutional Neural Network was used to generate the regions of interest. CNN was used for calculating the bounding boxes [50]. CNN comprises the convolution layer, region proposal Network (RPN), and classes and bounding box predictions.

In the convolution layers, filters are trained, and appropriate features are extracted from the images. The filters learn through training shapes and colors that exist in the images.

Region proposal network (RPN) is the second layer in the faster R-CNN architecture which represents a small neural network sliding on the last feature map of the convolution layers and predicting whether there is an object or not and predicting the bounding boxes of the objects as well. The third layer is the classes and bounding box predictions. This is where the fully connected layer is used from the regions proposed by the RPN and predicts the object class or the classification and the bounding boxes or the regression. The object detection methods, such as Faster R-CNN [40] described above, use parts of the same convolutional network for generating the region proposals and detection. We call these kinds of methods integrated methods.

Region of Interest (RoI)

Data labeling is an essential step in a supervised machine learning task. **Garbage-In-Garbage-Out** (GIGO) is a phrase commonly used in the machine learning space, which means that the training data quality determines the model's quality. The same is true for annotations used for

data labeling. As a machine learning model learns by looking at examples, the result of the model depends on the labels we feed in during its training phase[51][52].

This study used the Labelimg platform to select a region to annotate the rectangular boxes. The annotations were saved as XML files in Pascal Visual Object Classes (PASCAL VOC) format to a specified folder on the Raspberry pi. The PASCAL VOC format is used by ImageNet and supports the YOLO format [51][52]. Labellmg is a graphical image annotation tool written in Python and uses Qt for its graphical user interface.

Region Proposal Extraction

The first step of training the network is wrapping the region proposal from the original image with reference to the bounding box's points, width, and height and stored in the *train.cvs* file. The region of interest is defined in this stage. For this, the ROI label is first assigned to the region proposal. After the label definition, each image in the training dataset is annotated with the rectangle bounding box and the label. For each image, the points, the width, and the height of the bounding box are separated and saved in the table. After annotating the data table with the image location, bounding box coordinates and position are exported to the workspace and saved to a particular path in the *.mat file*. A regression model is trained to correct the predicted detection windows on bounding box correction offset using CNN features to reduce the localization errors. The confidence of robbery, physical attack, skimming, ATM out of service, transaction reversal fraud, jackpotting, logical attack, malware, occlusion, and other images are shown in figures 3.17 and 3.21.

Faster R-CNN

Faster R-CNN is used to faster the speed of object detection. Faster R-CNN uses N bounding boxes at each location. Bounding boxes are translation invariant; it uses the same ones at every location. Regression gives offsets from bounding boxes, and classification gives the probability that each regressed bounding box shows an object. This study imported a pre-trained model called ssdlite_mobilenet_V2, configured for the imageNet dataset. It is retrained with the downloaded ATM tampering image datasets. All the weights are fine-tuned by continuing the backpropagation with new datasets.

Transfer learning is efficient as many low-level features are common in many classification tasks. Transfer learning saves a lot of computation power and time for training. Observation suggests that the pre-trained model performs better than the model trained from scratch. There are a few things to take care of while doing transfer learning. The pre-trained model, trained on the different datasets, cannot be used because there will not be any common features in the two other datasets. Moreover, the learning rate while retraining the model should be low [29][47][9][10].

Faster R-CNN with ssdlite_mobilenet_V2, configured for imageNet dataset, was used which the fine-tune checkpoint field was configured in the train configuration and the *label_map_path* and *input_path* fields in the *train_input_reader* and *eval_input_reader* modules. The path *PATH_TO_BE_CONFIGURED*" was searched to find the fields that should be configured.

Faster R-CNN is an integrated method [17]. The main idea is to use shared convolutional layers for region proposal generation and detection. Feature maps generated by object detection networks can also generate region proposals[17][34]. The fully convolutional part of the Faster

R-CNN network that generates the feature proposals is called a region proposal network (RPN)[20].

Fast R-CNN architecture was used for the detection network. A Faster R-CNN network is trained by alternating between RoI generation and detection training. First, two separate networks are trained. Then, these networks are combined and fine-tuned. During fine-tuning, certain layers are kept fixed, and certain layers are trained in turn. The trained network receives a single image as input. The shared fully convolutional layers generate feature maps from the image. These feature maps are fed to the RPN. The RPN outputs region proposals, input with the said feature maps, to the final detection layers[20]. These layers include an RoI pooling layer and output the final classifications as indicated in figure: 3.22.

Ssdlite_mobilenet_V2

Ssdlite_mobilenet_V2 is a very effective feature extractor for object detection and segmentation. "It comprises two main features, namely, linear bottlenecks between the layers and shortcut connections between the bottlenecks". It uses depthwise separable convolution, and its main block is shown in the figure below:

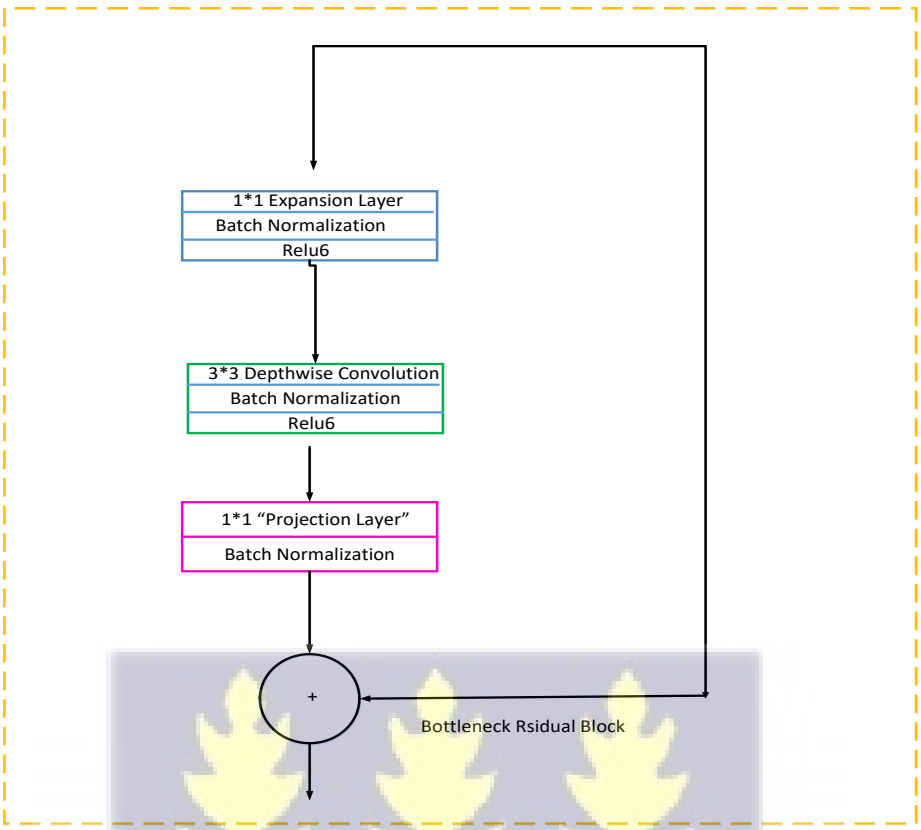


Figure 3. 16 Main Building Block of ssdlite_MobileNet_V2

Mobilenet V2 is made up of three convolution layers in the block. The last two layers: a depthwise convolution that filters the input followed by a 1*1 pointwise convolution which makes the number of channels smaller. For example, the depthwise convolution layer may work on tensor with 144 channels; the projection layer will then shrink down to 24 channels. The projection layer is also known as the bottleneck layer because it reduces the amount of data that flows through the network. Hence, the output of this layer forms the bottleneck residual block. From figure 3.16, the expansion layer is the first layer that expands the number of channels in the data before it goes into the depthwise convolutional layer. Hence it has more output channels than input channels. How much the data gets expanded is given by the expansion factor. The default expansion factor is 6. For instance, if a tensor with 24 channels goes into a block, the

expansion layer first converts this into a new tensor with 24 * 6 = 144 channels. Next, the depthwise convolution applies its filters to that 144 channel tensor, and then finally, the projection layer projects the 144 filtered channels back to a smaller number, say 24 again, as indicated in figure 3.17 below:

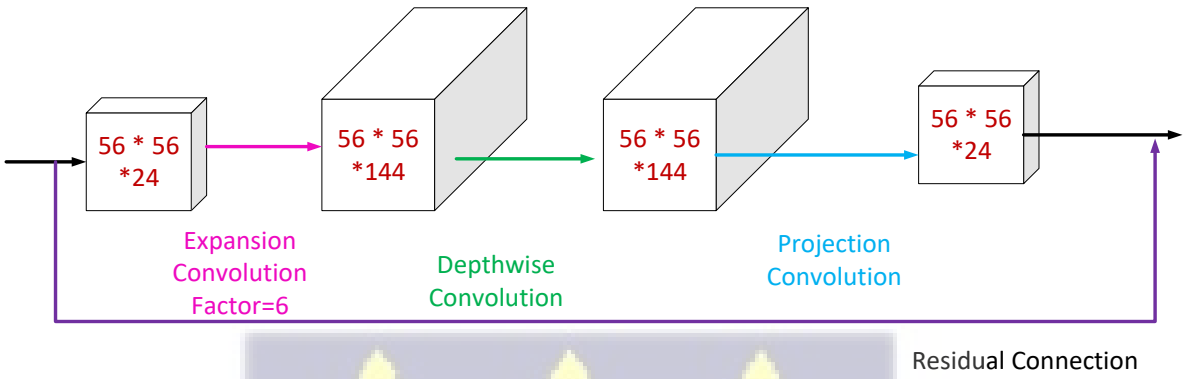


Figure 3. 17 Operations of MobileNet V2

Ssdlite_mobilenet_V2 is usually referred to as the deep convolutional neural network for object detection and achieves excellent performance on the imageNet dataset. Each layer has batch normalization, and the activation function is ReLu6. The next layer is the Maxpooling layer, which downsamples the output from the layer. The next step is Dropout which randomly drops units along with their connection during training. It helps to learn more robust features by reducing complex co-adaption of units. Ssdlite_mobilenet_V2 has ReLU6 (Rectified Linear Unit) for the non-linear part. ReLU is a component-wise operation (applied per pixel) and replaces all negative pixel values in the element map with zero.

$$f(x)=\begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{3.31}$$

The next layer is a SoftMax layer, turning the raw values for 1000 classes. This layer will predict the probability of the given images. The last layer takes these probabilities and returns the most likely class as the network output. The last layer is fully connected, each node fully connected to all input nodes and computes the weighted sum of all input nodes. It has a two-dimensional structure. It helps to classify input patterns with high-level features extracted by the previous layer. The 23rd layer is a fully-connected layer with 1000 neurons. This will take the extracted feature from the previous layers and maps them to the 1000 output classes.

For this study, the pre-trained model uses batch normalization after the depthwise convolution. The 1 * 1 convolution use bias instead, and for inference, the batch norm operation gets folded into the convolution layer. The full architecture of this model consists of 17 of these building blocks in a row. This is followed by 1*1 convolution, a global average pooling layer, and a classification layer and uses a 3 * 3 convolution with 32 channels instead of the expansion layer.

The Region-based CNN (RCNN) algorithm used in the model is one of the CNN-based deep learning object detection approaches. Given this, the deep learning method has different types of RCNN. They are Fast, Faster, and Mask RCNN. Faster RCNN is used to faster the speed of object detection. The general purpose of Mask RCNN is object instance segmentation. Other object detection approaches are You Only Look Once (YOLO) and Single Shot Multibox Detector (SSD). This proposed work consists of RCNN with 15 layers. The fundamental idea is made up of two steps. Initially, utilizing selective search will search for the object of our interest, and it will show the Region of Interest using a bounding box. It independently separates CNN features from each region for classification, as indicated in figure 3.16.

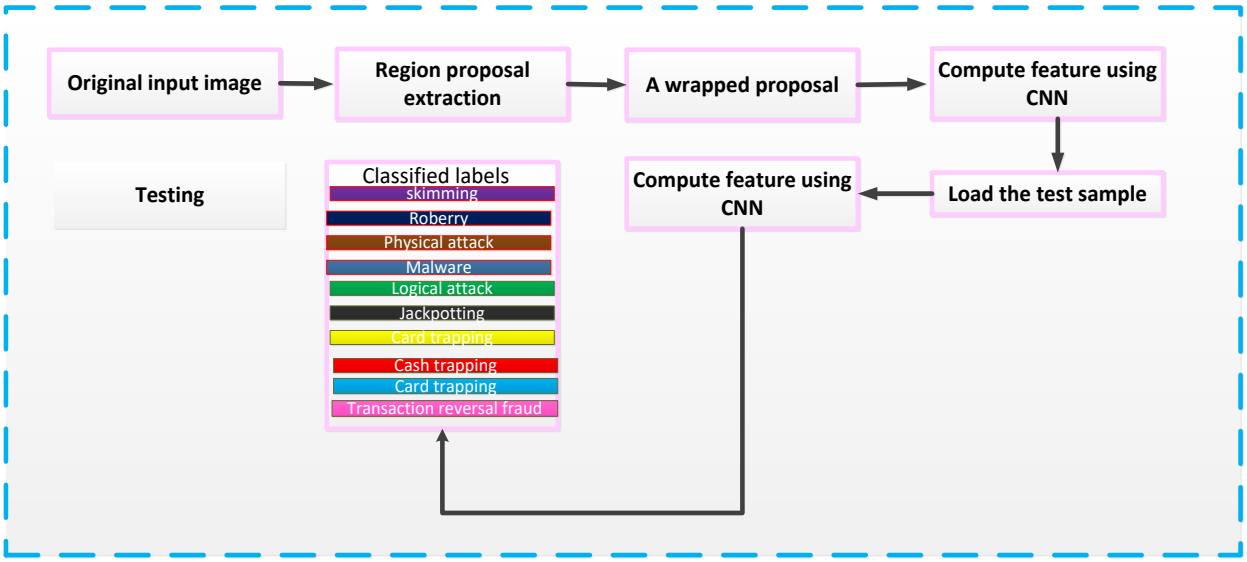


Figure 3. 18 Architecture for R-CNN

Using shared convolutional layers, region proposals are computationally almost cost-free. Computing the region proposals on a CNN has the benefit of being realizable on a GPU.

The Label-Image platform was used to create the bounding boxes feature selection and extraction, while the data processing codes were written in Python 3 in the TensorFlow environment. To identify the security incident class in an image, the advantage was taken of transfer learning with Faster Regional Convolutional Neural Network (R-CNN), which is the state-of-the-art object detector that classifies candidate object hypothesis generated by appropriate region proposal algorithms. The (R-CNN) with ssdlite mobilenet architecture leverages several advantages of computer vision development and CNN, including the feature expression capability from a pre-trained CNN, fine-tuning flexibility for a specific object to be detected, and the ever-increasing efficiency of object proposal generation schemes.

The bounding boxes technique was chosen among the off-the-shelf object proposal generation algorithms, attracting much interest recently. Bounding boxes are built on a structural Edge

Map to locate object boundaries and find object proposals. The number of enclosed edges inside bounding boxes is used to rank the likelihood of the box containing the security incident image.

R-CNN forward computation has several stages. First, the regions of interest are generated. The Regions of interest are category-independent bounding boxes with a high likelihood of containing an interesting object.

(1) ATM security incident camera feeds are downloaded from the internet, and some are taken with mobile phones and normalized. Their feature vectors are extracted using a regional-convolutional network (RCNN) and normalized. The final feature vector is classified as skimming, card trapping, cash trapping, Physical attacks, logical attacks, and robbery.

(2) ATM defect incident dataset in excel sheet noise was removed, feature vectors were extracted using Python, imported into TensorFlow environment in Python, and the final feature vector is classified using SVM-OVO, feature vector is classified as problem code description.

Selective Search [12] is used for generating these. A selective Search is an approach used to localize objects in object detection. It uses both Exhaustive search and segmentation, a method to separate objects of different shapes in an image by assigning them different colors.

Selective Search [12] utilizes a hierarchical partitioning of an image to create a sparse set of object locations. The main design philosophy is not to use a single strategy but to combine the best features of bottom-up segmentation and exhaustive search. The authors had three primary design considerations: the search should capture all scales, be diverse, i.e., not use any single strategy for grouping regions and be fast to compute.

The algorithm begins by creating a set of small initial regions using Graph-Based Image Segmentation [53][20], designed by Felzenszwalb and Huttenlocher. The method creates a set of regions called superpixels. The superpixels are internally nearly uniform. Combined, they span the entire image, but individually they should not span different objects.

Selective Search then continues by iteratively grouping the regions using a greedy algorithm, beginning with the two most similar regions. Many complementary measures are used to compute the similarity. These measures consider color similarity (by computing a color histogram), texture similarity (by computing a SIFT-like measure), size of the regions (small regions should be merged earlier), and how well the regions fit together (gaps should be avoided). The grouping phase ends when every region has been combined. The hypothetical object locations thus generated are then ordered by the likelihood of the location containing an object.

In practice, the locations are ordered based on the order in which the different measures grouped them. A certain element of randomness is added to prevent large objects from being favored too much. Lower-ranking duplicates are removed. The region-generating method and the similarity measures were selected to be fast to compute, making the method fast in general. In addition to using diverse similarity measures, the search can be further diversified by using complementary color spaces (to ensure lighting invariance) and complementary starting regions.

Secondly, a convolutional neural network extract features from each region proposal. The sub-image in the bounding box is wrapped to match the input size of the CNN and then fed to the network. After the network has extracted features from the input, the features are input to the

Faster regional convolutional neural network (R-CNN) that provides the final classification [20]. Figure 3.17 shows the stages of R-CNN forward computation:

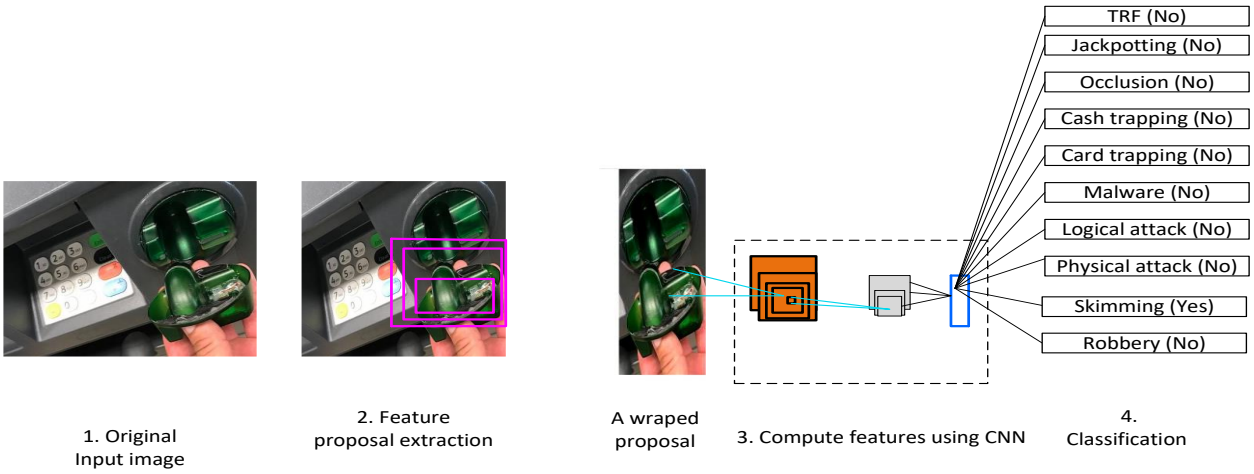


Figure 3. 19 Stages of R-CNN forward computation

The method is trained in multiple stages, beginning with the convolutional network [10][20]. After the CNN has been trained, R-CNN features are fitted. Finally, the region proposal-generating method is trained.

The systematic approach adopted for the design and development of the CNN and R-CNN for ATM security incident detection is presented in the conceptual framework in figure 3.18

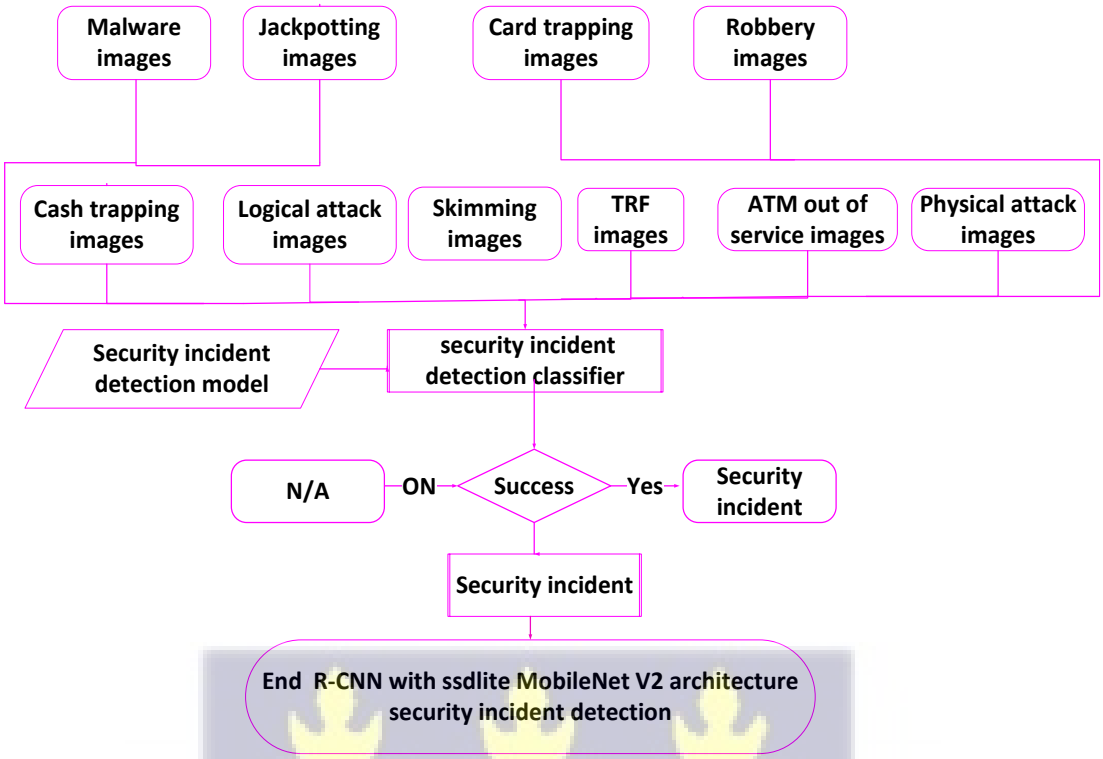


Figure 3. 20 Conceptual model design and development of CNN with ssdlite MobileNet V2 Architecture.

The conceptual framework incorporates the design and development of key algorithms that enable the annotated images to be analyzed and a model to be developed for testing validation.

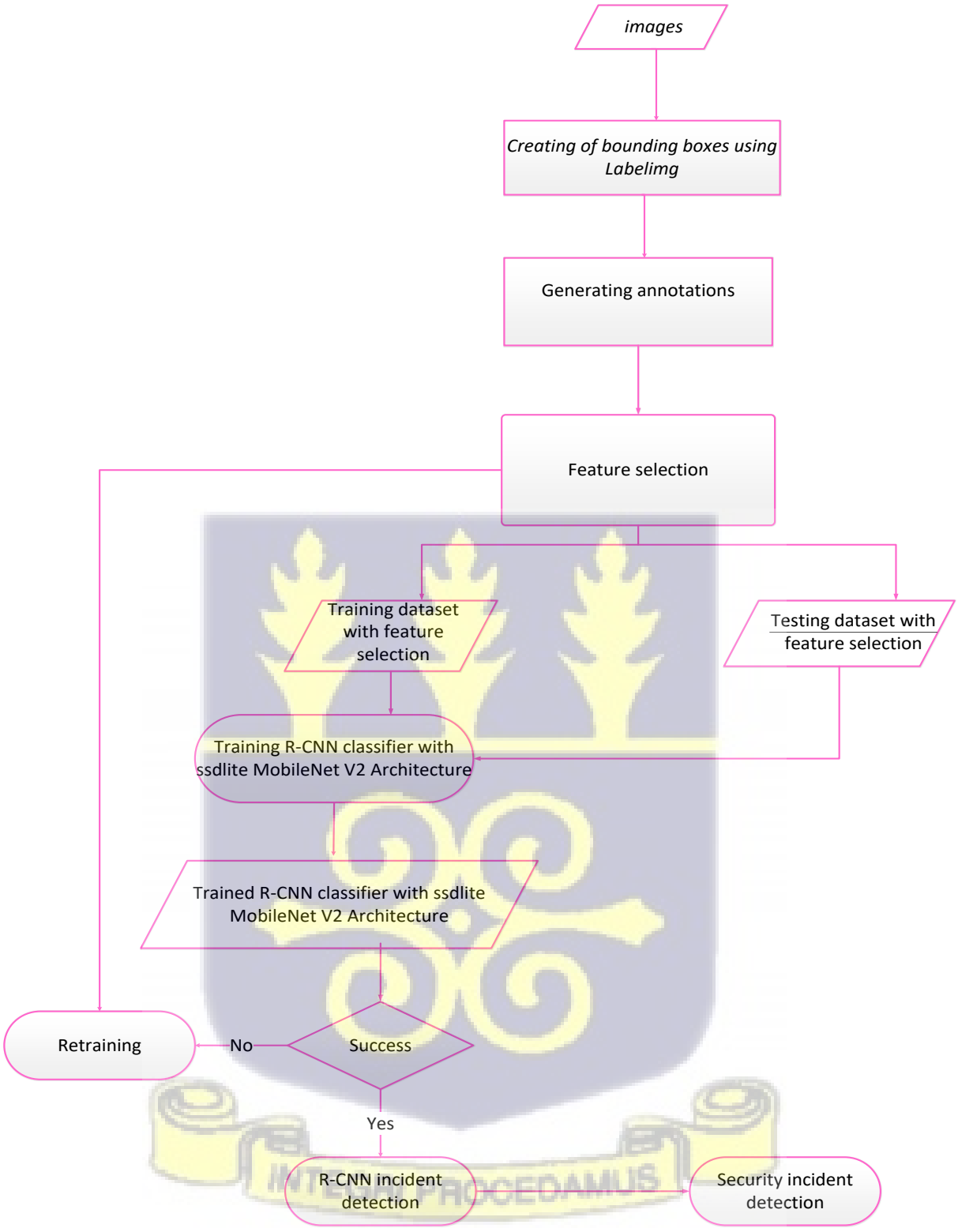


Figure 3. 21 Flow chart model design and development of CNN / R-CNN

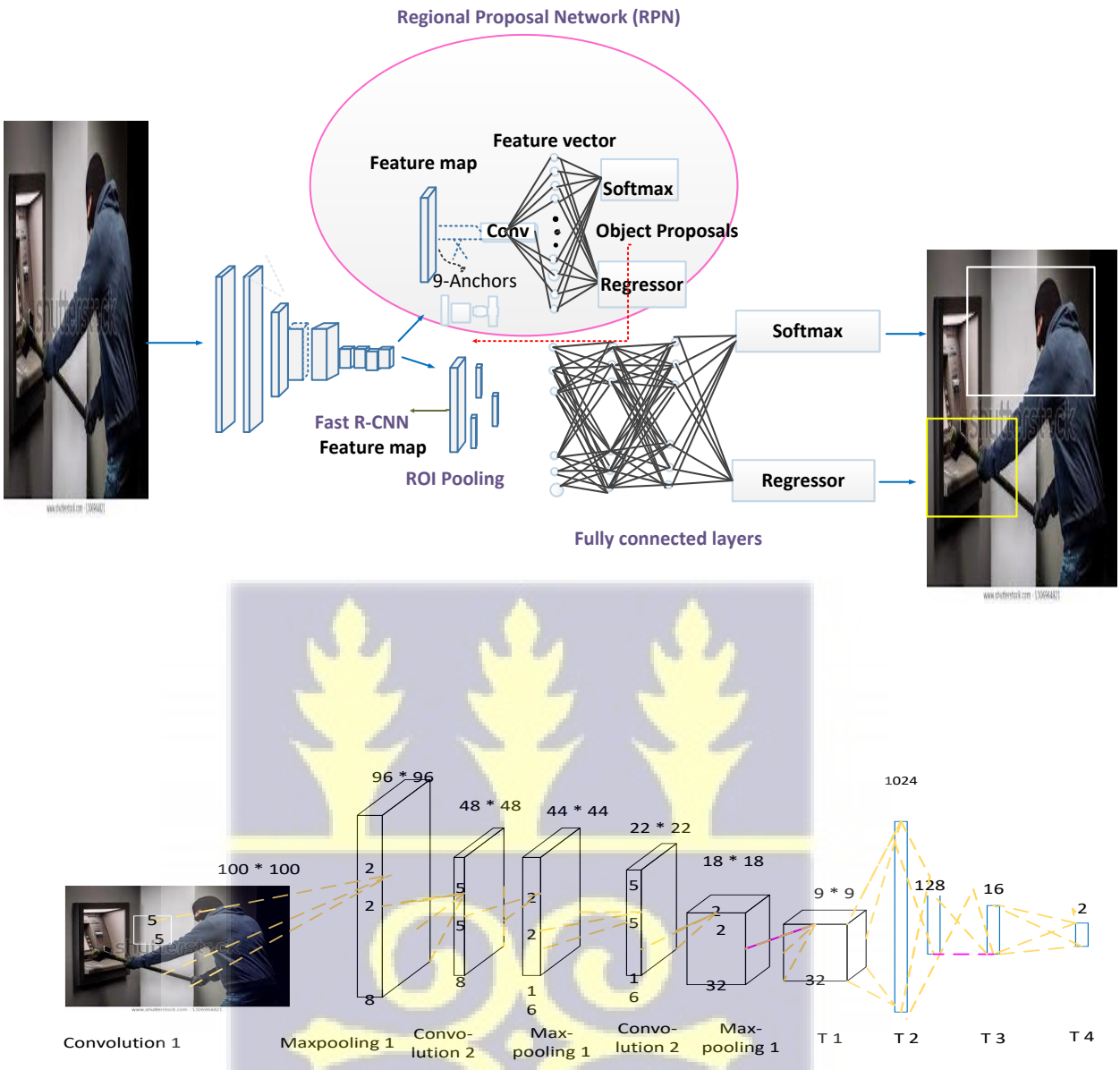


Figure 3. 22 CNN Security Incident Detection Architecture

The network consists of three (3) convolution stages followed by three fully connected convolutional layers; one processing step is usually the layer which could be the convolutional, pooling layer.

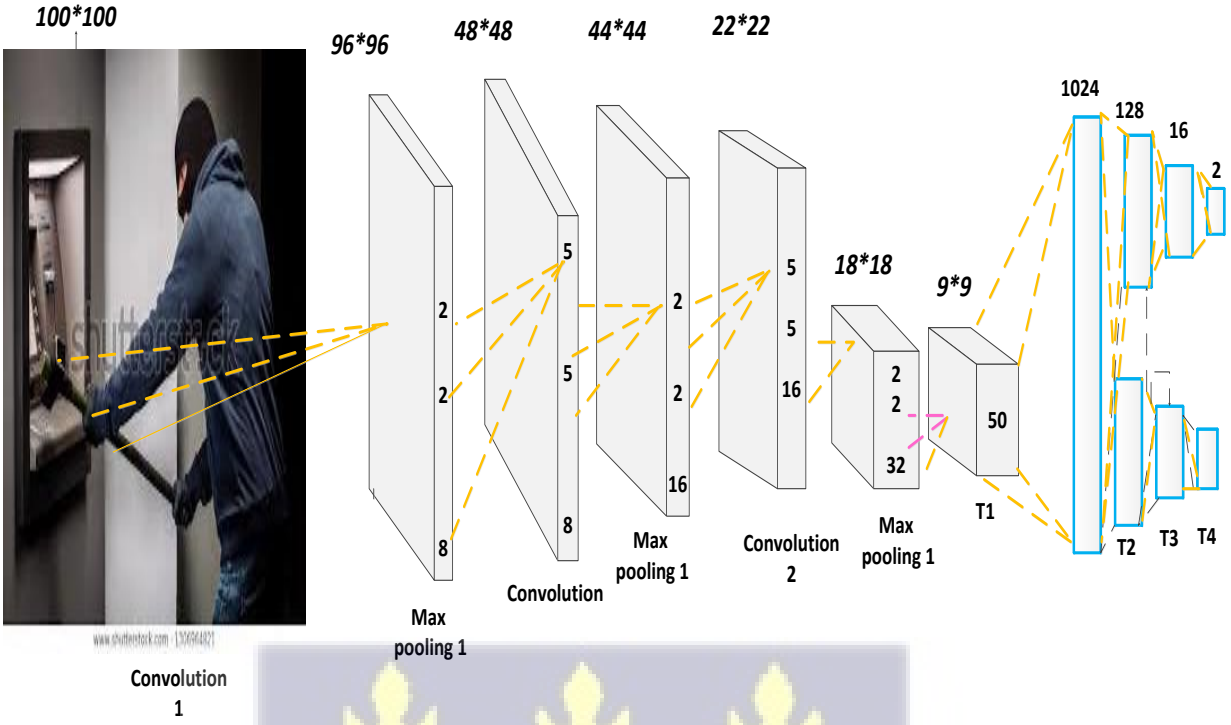


Figure 3. 23 CNN Classification Model Architecture

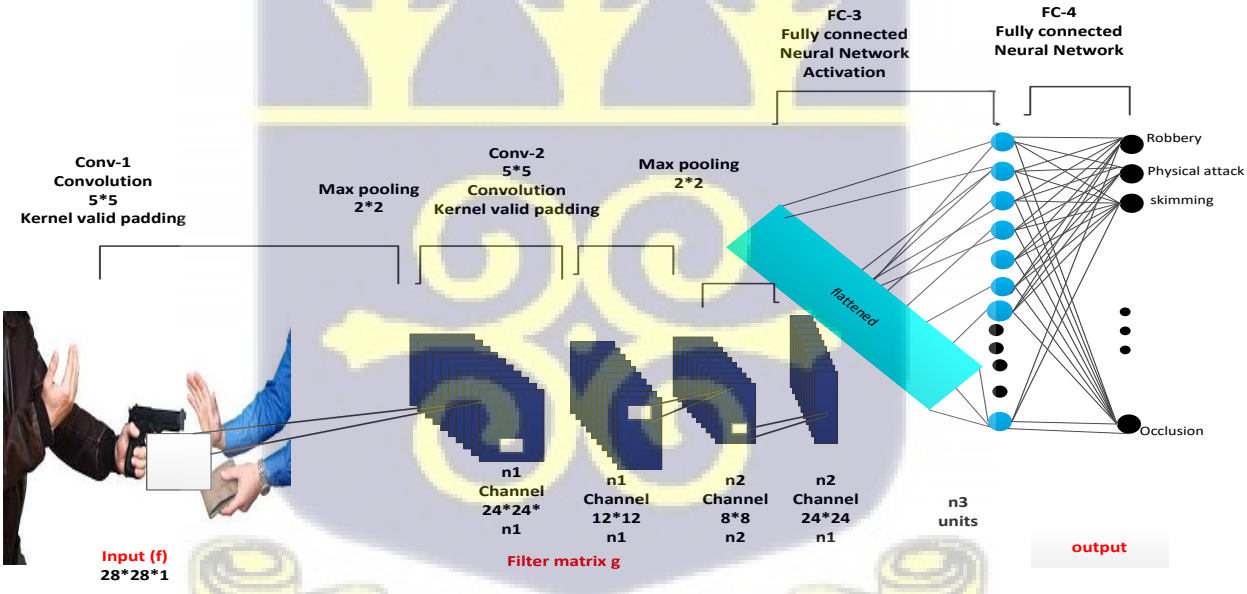


Figure 3. 24 CNN Classification Model Architecture

Mathematical Foundation for Convolutional Neural Network (CNN) And Regional-Convolutional Neural Network (R-CNN) Operations

R-CNN creates anchors or bounding boxes around the images, and it is used to generate and create a training classifier

$$(f.g) \overset{def}{=} \int_{\infty}^{\infty} f(r)\, g(t-r)\, dr" \tag{3.32}$$

The feature detector filters the original part of the input image that is integral to it and excludes the rest, and it is used to adjust the images.

$$h[x,y] = f[x,y].g[x,y] = \sum_{n}\sum_{m}\left(f[n,m]\right)g[x-n,y-m])) \tag{3.33}$$

The dot product of the filter g and a sub-image of f (with the same dimensions as g) centered on coordinates $x; y$. "This produces the pixel value of that coordinates $x; y$". "The filter matrix size adjusts the receptive field's size, aligning the filter successively with every sub-image of f to produce the output pixel matrix h".

The flow chart presents the algorithm implemented based on theoretical foundations incorporating convolutional neural networks (CNN) and regional neural network (R-CNN) for security incident detection and the two useful machine learning algorithms necessary for ATM security incident detection. The combined use in the detection process generates accurate results.

Bounding Box Regression

Given a predicted bounding box coordinates

$$p = \left(p_x, p_y, p_w, p_h \right) \tag{3.34}$$

("Center coordinate, width, height) and its corresponding ground-truth box coordinates":

$$g = \left(g_x, g_y\, g_w\, g_h \right), \tag{3.35}$$

"The regressor is configured to learn scale-invariant transformation between two centers and log scale transformation between width and height". "All $gh = p_h \exp(d_h(p))$ transformation functions take P as input"

$$\begin{aligned} \hat{g}_x &= p_w d_w(p) + p_x \\ \hat{g}_y &= p_h d_y(p) + p_y \\ \hat{g}_w &= p_w \exp\left(d_x(p)\right) \\ \hat{g}_h &= p_h \exp\left(d_h(p)\right) \end{aligned} \tag{3.36}$$

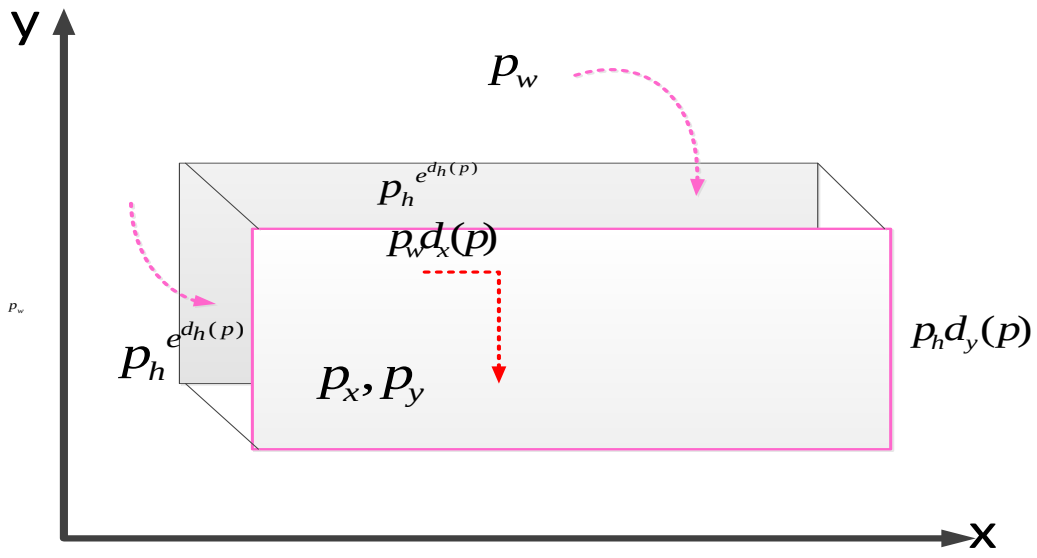


Figure 3. 25 Illustration of transformation between predicted and ground-truth bounding boxes

An obvious benefit of applying such transformation is that all the bounding box correction functions $d_i(p)$ where $i \in \{x, y, w, h\}$ can take any value between" $[-\infty, +\infty]$. The targets for them to learn are:

$$t_x = (g_h - p_x) / p_w$$

$$t_y = (g_y - p_y) / p_h$$

$$t_w = \log\left(\frac{g_w}{p_w}\right)$$

$$t_w = \log\left(\frac{g_h}{p_h}\right) \quad (3.37)$$

Astandard regression model can solve the problem by minimizing the loss with regularization

$$lreg = \sum_{i\in\{x,y,w,h\}} \left(t_i - d_i(p)\right)^2 + \lambda \|w\|^2 \text{”} \quad (3.38)$$

The regularization term is critical here, and RCNN is picked as the best λ cross-validation”. “It is additionally imperative that not all the predicted bounding boxes have to compare ground truth boxes”. “For instance, running a bounding box regression does not make sense if there is no overlap”. “Here, a predicted box with a nearby ground truth box with at least 0.6 IoU is kept for training the bounding box regression model”.

The inbuilt Python code for R-CNN classifiers was integrated as one function for CNN and Faster R-CNN”. “The datasets were partitioned for the classifier training and testing, and 70% of the dataset was used for training and 30% for testing for the classification process”.

3.8.4.1 Defects Dataset Training

The systematic approach adopted and used for the design and development of the SVM model for ATM defect incident detection is presented in the two flow diagrams; conceptual and implementation, below:

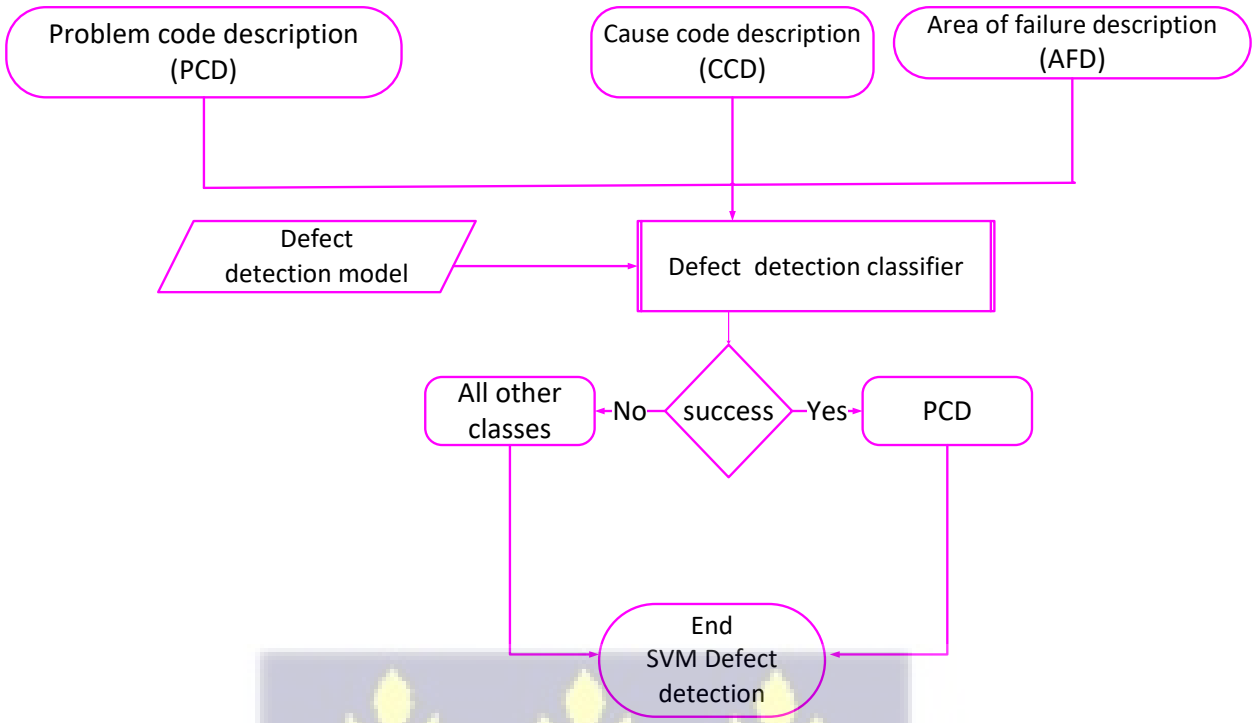


Figure 3. 26 *Conceptual model design and development of the support vector machines.*

The conceptual framework incorporates the design and development of key algorithms that enable defects data to be analyzed and a model to be developed for testing. The flow chart presents the algorithm implemented based on theoretical foundations incorporating support vector machines, two useful machine learning algorithms for defect detection. The use in the detection process generates accurate results. The methodology for the design and development of support vector machines, as presented above, consists of three (3) significant steps, namely, (i) data preprocessing, (ii) classification engine development, and (iii) data post-processing [54].

Data Preprocessing

Data preprocessing is the first significant stage in developing the defect detection system. This stage involves using data mining techniques to transform the data from its raw form into the required format to be used by the SVC for the detection and identification of ATM defective detection. The data preprocessing stage involves the removal of unwanted fields and data smoothening. This ensures that only useful and relevant information is extracted for the next process.

Before the preprocessing, the data were imported from Microsoft Excel in xlsx format into the Google Colab Python environment. The data preprocessing involves the following steps: (1) defects data filtering and selection, (2) feature selection and extraction, and (3) feature adjustment.

The Google Colab machine learning and knowledge analysis environment were used for feature selection and extraction. At the same time, the data processing codes are written in the Python IDE running on Linux, and scikit-learn, a machine learning module written in Python with fully integrated a wide range of machine learning algorithms, was used to load the scikit-learn libraries such as pandas and the ATM defects dataset needed to build the model. The features and the target group were defined. The dataset was split into train and test using *sklearn* before building the SVM algorithm model. The support vector classifier function or SVC function from the *sklearn* SVM module was Imported, and the Support Vector Machine model was built with the help of the SVC function[54] in Figure 3.25 below:

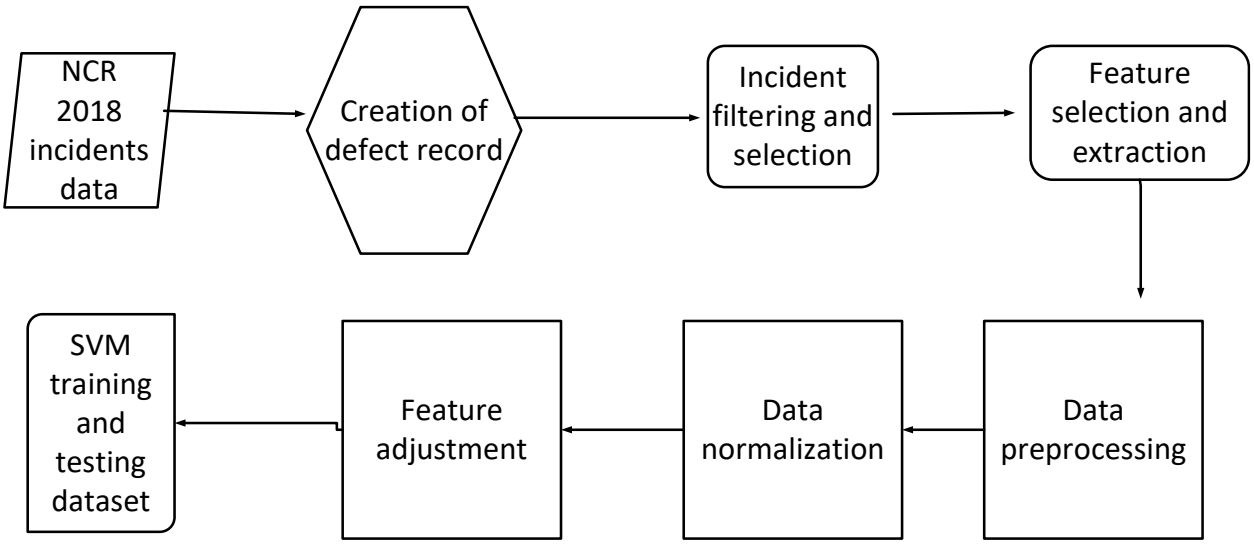


Figure 3. 27 Data preprocessing for SVM training and testing.

Preprocessing the raw data involves checking the cause, problem, and area of failure of the defective components. The data mining techniques require extracting a smaller and optimized set of features obtained by removing largely redundant, irrelevant, and unnecessary features for the class prediction. The filter method, a feature selection method, was utilized to extract a minimal subset of attributes. The resulting probability distribution of data classes is close to the original distribution obtained using all attributes. The filter methods select the features based on the scores in various statistical correlations. The selected features based on SVM algorithms are "Area of failure (AFD)," "Cause code description (CCD)," and "Problem Code Description (PCD)." These are the features selected, extracted, and used as the basis for the optimization problem formulated below:

$$minimize\ incidents(PCD) = f\left(CCD, AFD\right),$$

$$subject\ to \quad \sum_{i=1}^{n}\left(CCD_i\right) \le C_i\ PCD, \forall i = 1,2,....n,$$

$$\sum_{j-1}^{n}\left(AFD_j\right) \leq A_j\, PCD, \forall j,\ j = 1, 2, \ldots . n, \tag{3.39}$$

The NCR 2018incident dataset is subjected to SVM training, using 70% of the dataset and 30% for testing, as depicted in Figure 3.25. The NCR 2018 incident dataset, which passes the preprocessing stage, that is, the incidents, was used for SVM training and testing. The best data that meet the SVM criteria are classified first. Each record of this dataset is classified as either "PCD" or "CCD." Or "AFD." The same SVM training and the testing dataset are applied to the SVM algorithm for its performance analysis. The hyperparameter(C) from problem code description (PCD) was performed for each classifier.

The performance measured on PCD optimization tested several parameters for the optimal SVM. The SVC training aims for the best SVC hyperparameter *(C)* in building the defect detection classifier model. The developed classifier is evaluated using testing and precision. The inbuilt Python code for SVM classifiers was integrated as one function for linear, polynomial, and RBF kernels. The incident datasets were partitioned for the classifier training and testing, 70% of the dataset was used for training, and 30% was used for testing. The linear, polynomial, and radial basis function SVM classification kernels were used for each kernel, averaging the results. "For the polynomial classification kernel, a cubic polynomial was used. The RBF classification kernel used the Sequential Minimal Optimization (SMO) method" [40]. "This method ensures the 2018incident dataset is subjected to SVM training, using 70% of the dataset and 30% for testing". "The dataset was used for the SVM training and testing". The same SVM training and the testing dataset are applied to the SVM algorithm for its performance analysis, as shown in figure 3.26 below:

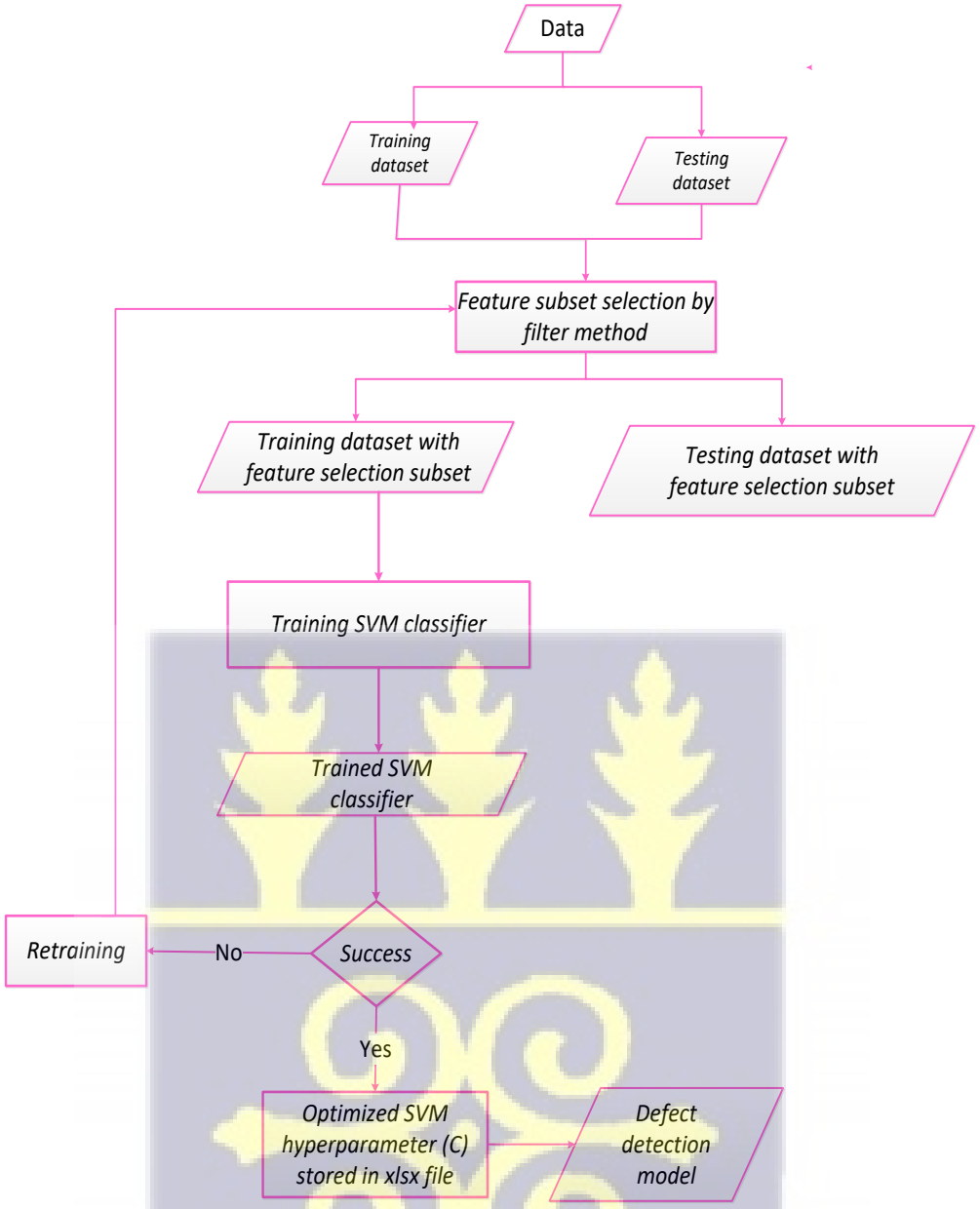


Figure 3. 28 Flow chart for design and development of the support vector machines

The inbuilt Python code for SVM classifiers was integrated as one function for linear, polynomial, and RBF kernels. The datasets were partitioned for classifier training and testing, and 70% of the dataset was used for training and 30% for testing. Since the dataset is multiclass,

a decomposition approach, OVO with a hyperplane, was adopted to separate the class and all others simultaneously, as indicated in the flow diagram below.

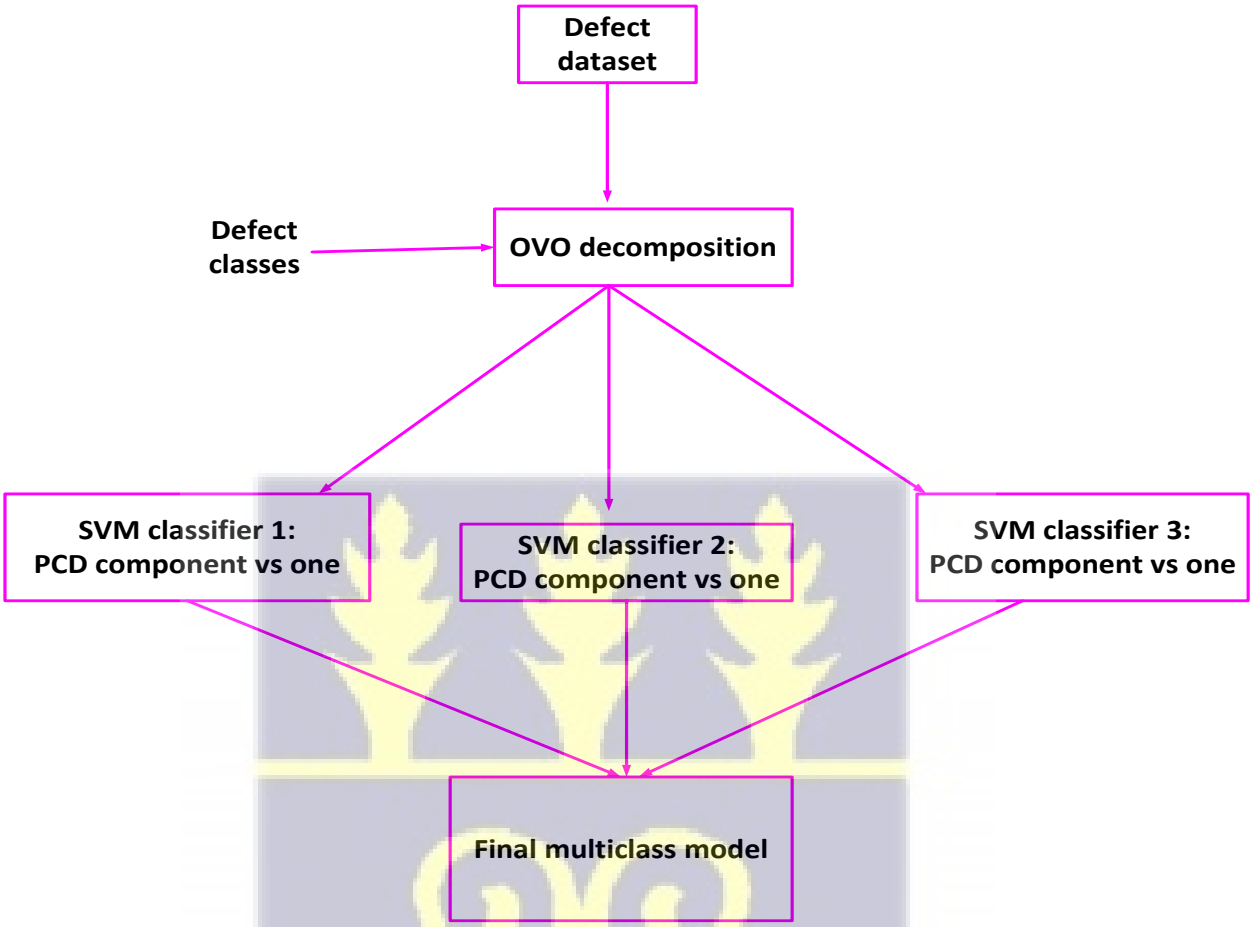


Figure 3. 29 OVO classification for the multiclass problem.

The linear, polynomial, and radial basis function SVM classification kernels were used for each kernel, averaging the results. For the polynomial classification kernel, a cubic polynomial was used. The RBF classification kernel used the Sequential Minimal Optimization (SMO) method. This method ensures handling large data sizes as it does data transformation through kernelization. After running many instances and varying parameters for RBF, a variance of 0.9 gave better results as it corresponded well with the datasets for the classification. Using the

classes for the dataset for this study, the correct rate is calculated after each classification, and the confusion matrix is extracted. The confusion matrix gives a count, for example, the true PCDs, and false PCDs.

(i) True PCD: this consists of the number of "true PCDs" correctly classified as "True PCD" by the classifier.

(ii) False True PCD: this consists of the number of "true one class" correctly classified as "true PCD" by the classifier.

(iii) This consists of the classified as "PCD" even though they are not. These are wrongly classified as "other PCD class" by the kernel used.

The correct rate: this is calculated as the total number of correctly classified PCDs, namely, the true PCD, and divided by the total number of the dataset used for the classification:

$$Correct\ rate = \frac{number\ of\ TPCD + number\ of\ PCDs}{total\ number\ of\ data\ ued} \tag{3.40}$$

$$Accuracy = (1 - Error) = \frac{TP + TN}{TP + TN + FP + FN} \mathrm{P}_r(C) \tag{3.41}$$

Sensitivity: This is the statistical measure of the proportion of actual PCD which are correctly detected:

$$specificity = \frac{TN}{TN + FP} \tag{3.42}$$

Specificity: This is the statistical measure of the proportion of negative PCD which are correctly classified":

$$sensitivity = \frac{TP}{TP + TN} \tag{3.43}$$

Precision: this is the percentage of correct detection of a positive class among all detection of that class.

$$precision = \frac{TP}{(TP + FP)} \tag{3.44}$$

CHAPTER FOUR

SYSTEM DESIGN AND DEVELOPMENT

4.0 Introduction

This chapter gives an in-depth analysis of the design of a self-reporting system for ATM incident detection.

4.1 Proposed System Design

The Self-reporting System for ATM Incidents Detection (SRSAID) aims to improve ATM security and downtime experience due to system component defects. Its core functionality focuses on incident detection, which reduces the delays in using the machine. Incidents at the ATM terminal are categorized into security incidents and defective incidents. ATMs under any of these conditions cause dissatisfaction to the user, banks, and ATM Service Provider Company. The SRSAID is a hybrid system that solves ATM security incident problems with Faster-Regional Convolutional Neural Network (Faster-RCNN) and ATM defect problems with Support Vector Machines (SVM) algorithms.

The ATM security incident side of the SRSAID uses Deep Learning built on the TensorFlow platform. Its workflow is listed as follows:

- Preprocessing data
- Building the model
- Training and estimating.

4.2 Incident Detection

4.2.1 CNN and SVM Incident Detection System Implementation and Testing.

The system's security and defective incidents aspects comprise four main modules integrated. They are algorithm implementation using Python technical computing platform, development of graphical user interface (GUI) for the SRSAID system, which consists of uploading and processing of ATM state management, system administrator management, and post-processing of detection and classification results. Figure 4.1 shows the system implementation architecture for the SRSAID.

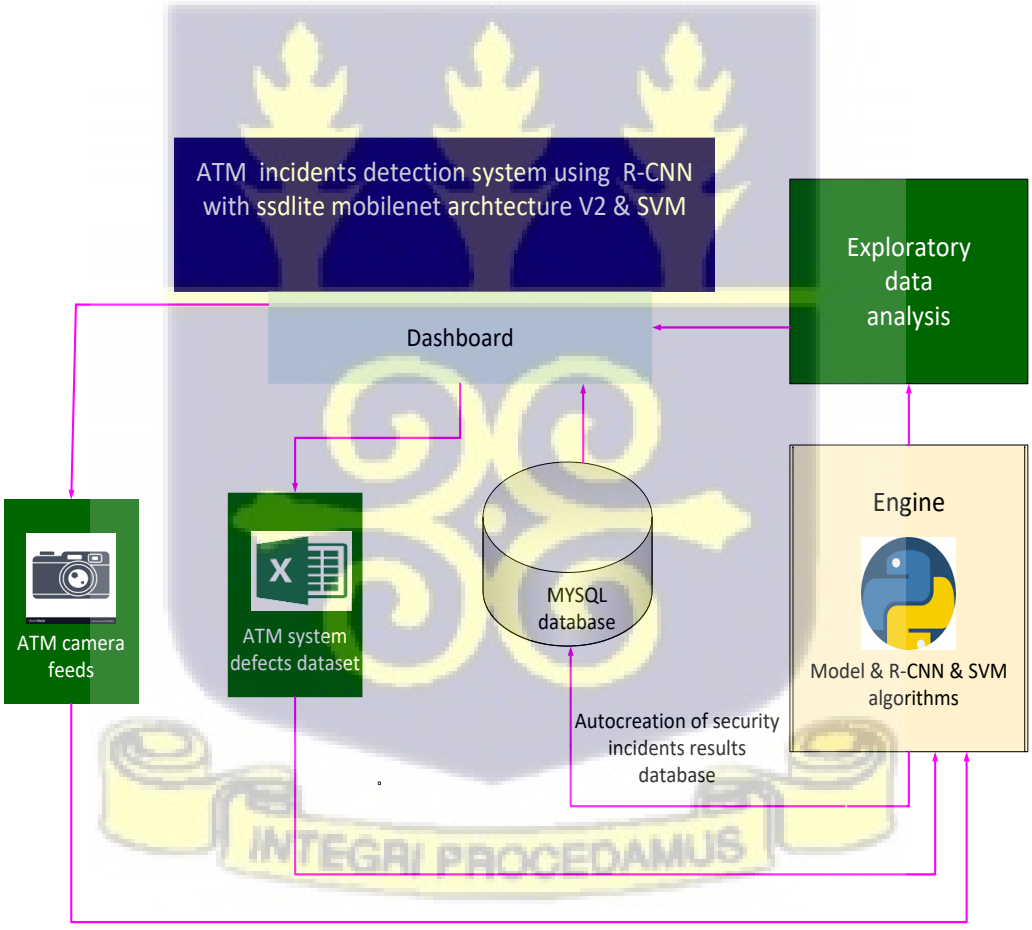


Figure 4. 1 System Implementation Architecture for SRSAID

4.2.2 Detection Results

The SRSAID system consists of a web application developed in Django for which detected incidents are reported. The web application uses HTML, CSS, and Bootstrap4 in its dashboard interface and API written in Python at the backend.

The SRSAID consists of several functional components: a function for computing the descriptive statistics of raw and processed data, preprocessing wrapper function for data handling and processing, and Python functions and SVM Classification processes. The SRSAID components are shown in *Figure 4.2*. The results generated by the SRSAID are stored in the MYSQL database. The results comprise five parts: security incidents detection, location of incidents on GPS displayed on a map, text messages through GSM and mobile phone, and statistics of all the results. These results are shown in Figure 4.2. The GUI portal for analyzing results obtained from the submitted ATM incidents classification is displayed in Figures 5.2 and 5.3. A pop-up menu generating the labeled is obtained for security datasets by clicking on the incidents button in the GUI. It shows the grouping of detected incident types in the dataset.

In the graphical user interface of the system, the user is allowed to carry out the actions mentioned above, and the associated use case for users is represented in figure 4.3.

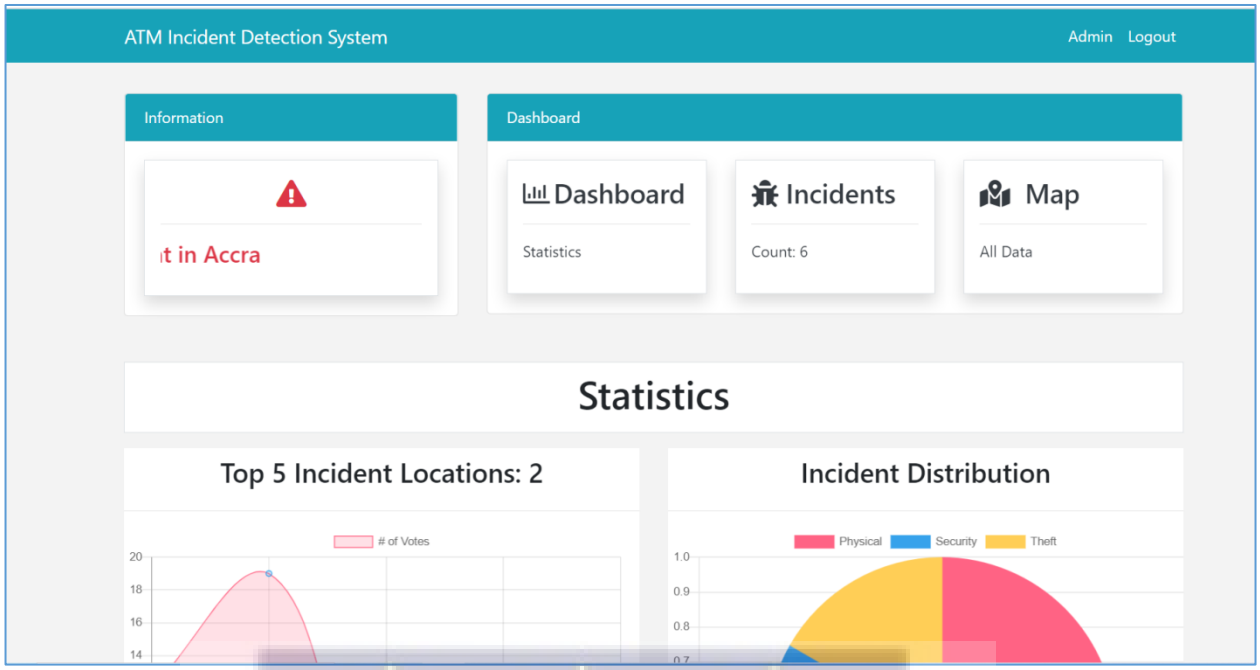


Figure 4. 2 Detection results control portal interface.

Figure 4.3 shows the SRSAID block diagram. The entire architecture of SRSAID incorporates Machine Learning techniques, a software application, and a hardware setup built with Raspberry Pi 3 Model B, Pi camera Global System for Mobile Communication (GSM), and Global Positioning System (GPS) hardware modules.

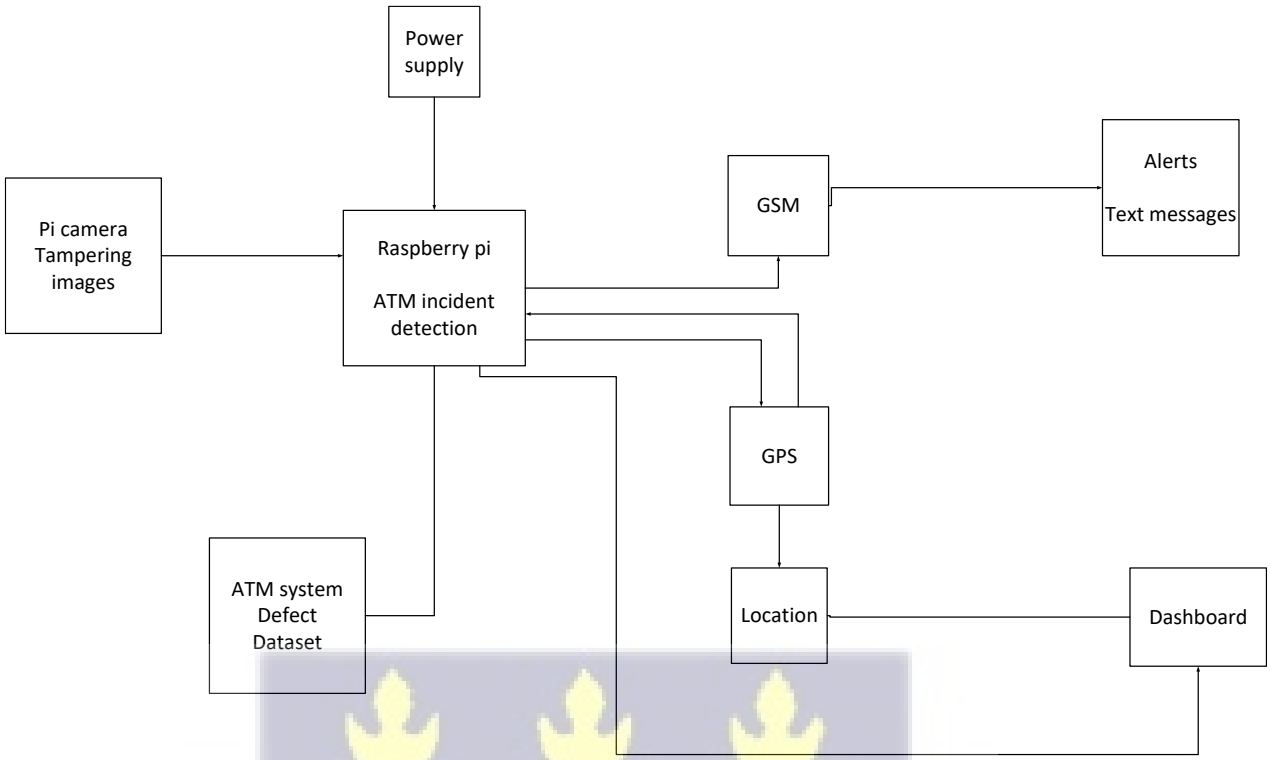


Figure 4. 3 Proposed System Block Diagram

The raspberry pi 3 is a mini-computer and user-friendly. It is very compact. The raspberry pi 3 holds the operating system, programs for the compilation and running of the project, and the documents. The Raspberry Pi 3 stores the code for security incident detection, the type of security incident, and the location and alert messages module with other IoT components, GPS and GSM, respectively. The defect incident detection model is built on a Windows laptop due to a lack of storage space.

4.3 Implementation of SRSAID

The complete source code for implementing SRSAID and the required import modules are provided in Appendix A, B, and C. SRSAID was implemented in Python 3 using TensorFlow

2019 running in a 64bit Linux and Windows 10 environment on Raspberry pi and laptop, respectively.

Initially, a Selective search algorithm is used to scan the input image of possible objects for generating 1000 region proposals.

Secondly, apply the CNN algorithm for the generated region proposals to create the bounding boxes for feature selection which generated the XML file. The XML was then converted to a CSV file, the tfrecord.py was used, and the security incident detection classification was generated. The sigma value is calculated at the end of the iteration, as described in chapter 4. The output of each CNN is fed into RCNN. The input image is fed into the pre-trained CNN with ssdlite_mobilenet_V2 architecture using the transfer learning technique. The different parameters are chosen for comparing various mini-batch sizes and epochs in ssdlite_mobilenet_V2. The ssdlite_mobilenet_V2 detects whether a security incident is present in the input image or not. If a security incident is present in the image, the class or type of the incident is detected using RCNN. Based on the region of interest (ROI) location, the system will decide whether the security incident type is skimming, physical attack, out of service, logical attack, robbery, jackpotting, cash trapping, card trapping, and others. The alert will be generated if there are security incidents and reports on the dashboard. The code snippet that performs this operation is shown below.

Code Listing

```
 #  prepare to log coordinates, ATM serial number, etc., in the
database
        "import requests"
        url = "http://localhost/config/api2.php"
        my_post_data = {
```

```
                        "serial_number": atmSerialNum,
                        "city": cityName,
                        "lat": gpsFx.getLatitudeDeg(),
                        "lng": gpsFx.getLongitudeDeg(),
                        "type": detection_name
                    }

        " r = requests.post(url, my_post_data)"

         "if "Records added successfully" in r.text:"
             print("\nSuccessfully added record in database! Will send S
MS next\n")
             sleep(1.000)
         else:
            " print("Failed to add record to database.")
             sleep(1.000)

         #Add coordinates to message prefix and send SMS to destination
phone
         newMessage = "{}{},{}".format(msgPrefix, gpsFx.getLatitudeDeg()
, gpsFx.getLongitudeDeg())
         print("\nPreparing and sending message to {}...\n".format(destP
hone))
        " print("Message to send is: {}\n".format(newMessage))"
         gsmFx.composeSMS(destPhone, newMessage)
         gsmFx.sendSMS()

         #turn off GPS to save power
         #gpsFx.gpsOff()

                 #exit(0)
         sleep(2.000)
         captureImage()
```

During the defect incident detection process, the defect datasets were imported into the Google Colab workspace. Python libraries were imported. The Python data mining technique was used to extract the features. SVM algorithms were used to determine the defect incident detection classification. The code snippet that performs this operation is shown below.

Code Listing for Defects Incident Detection

```
clf =” sklearn.svm.SVC(kernel='rbf',C=study.best_params['C'])
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print(f"score: {clf.score(X_test, y_test)}")
print(f"accuracy: {accuracy_score(y_test, y_pred)}")
```

4.4 The Hardware Architecture

This study's security incident detection models are processed with IoT components such as Raspberry Pi, Pi Camera, GSM module, and the GPS module performing the functions indicated in figures 4.4 and 4.5. From figure 4.4, the GSM/GPS modules are interfaced with the raspberry pi to form the IoT components of the hardware architecture. The Raspberry pi uses a Universal asynchronous receiver transmitter (UART). This enables interfacing with the GPS and the GSM module. This study used a minicom to connect the GPS/GSM and testing. Minicom is a text-based serial port communications program. It is used to talk to external RS-232 devices such as mobile phones, routers, and serial console ports. It also uses the pynmea2 library to parse the received data.

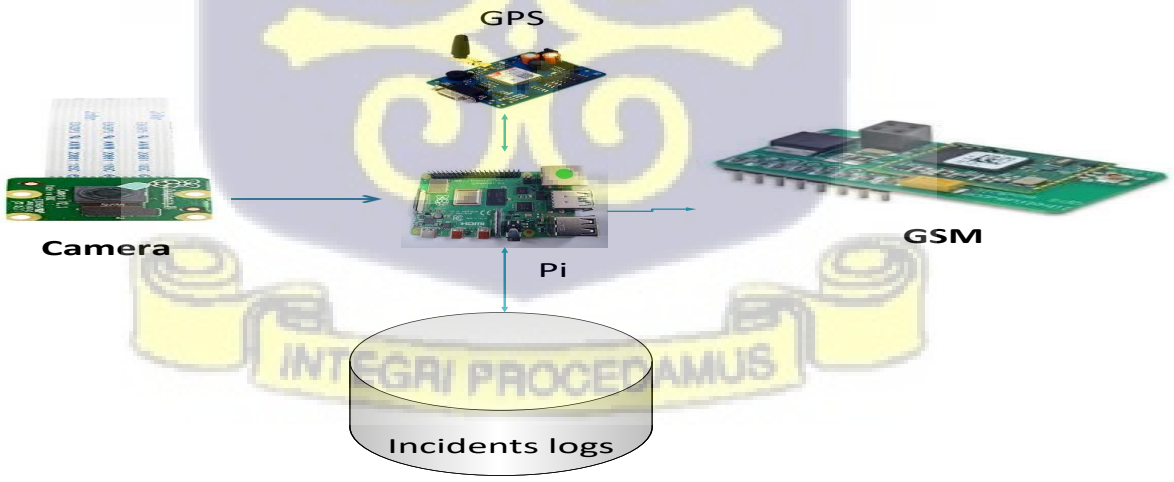


Figure 4. 4 Hardware Implementation Architecture for SRSAID

4.5 System Integration Architecture

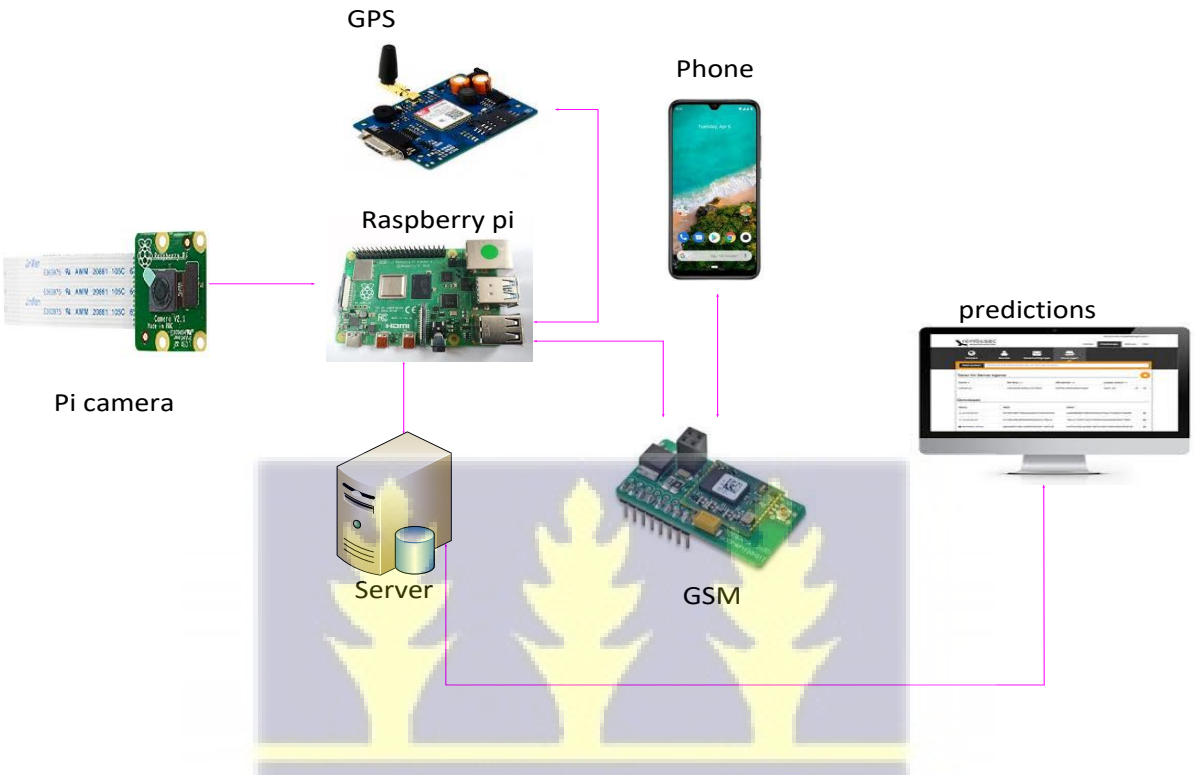


Figure 4. 5 The SRSAID Integration Architecture

Figure 4.5 shows the overall system integration of the SRSAID. The Pi camera captures images as input and feeds them to the Raspberry Pi for analysis. If a security incident is determined, the location of the incident is projected on Map through the GPS module. An alert is sent to a mobile phone through the GSM, and eventually, logs in the MYSQL database, and analysis is reported on a software dashboard for decision making.

CHAPTER FIVE

SYSTEM IMPLEMENTATION AND TESTING

5.0 Introduction

This section gives an in-depth implementation of the Self-Reporting System for ATM Incident Detection (SRSAID) and discusses the results from the experiments performed.

5.1 Testing of SRSAID

To examine the efficiency of the proposed technique, several test experiments were performed across a wide range of input performance parameters. As indicated in chapter 3, the Pi camera would capture ATM security incident images and analyze them on the Raspberry Pi. The algorithms used include R-CNN, as indicated in the code listing below:

```
# Current working directory
CWD_PATH = os.getcwd()

IMAGE_NAME = "cam_image.jpg"
IMAGE_PATH = os.path.join(CWD_PATH, IMAGE_NAME)

def captureImage():
    with picamera.PiCamera() as camera:
        print("Capturing image...")
        sleep(1.000)
        camera.vflip = True
        camera.capture(IMAGE_NAME)
        print("Image captured")
        IMAGE_CV2  = cv2.imread(IMAGE_PATH)
        cv2.imshow("Captured Image", cv2.resize(IMAGE_CV2, (800,600)) )
        cv2.waitKey(3000)               # Show the captured image for 3s
        cv2.destroyAllWindows()
        return
```

With defect incident detection, the model would predict numerous components to determine what is possible to be defective. The algorithm used is the Support Vector Machines (SVM):

Figure 5.1 shows the Use case diagram of the SRSAID. For its intended purposes, the system will be used by a bank official who will have all the activities shown in the diagram.

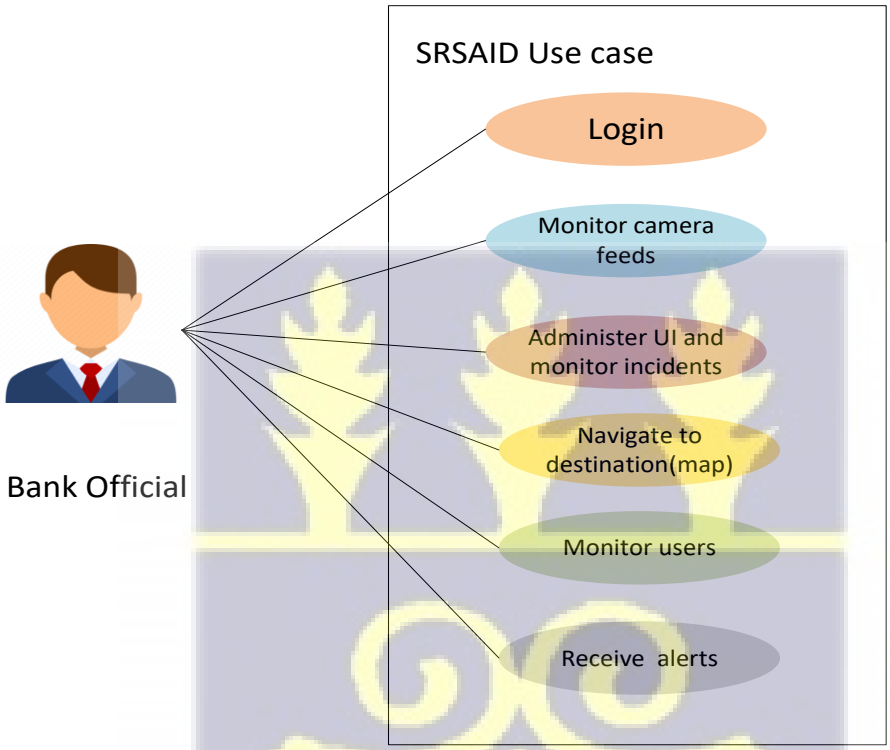


Figure 5. 1 Use case Diagram for SRSAID

5.2 Results and Discussions for Security Incident Detection

The deep learning Faster R-CNN with ssdlite_mobilenet_V2, configured for the ATM security incident dataset used for this study, will detect a security incident and its class in the input images. Suppose a security incident is detected in the image captured by the pi camera, in that case, the image is fed into the RCNN model designed and developed on the Raspberry pi to

find the class of the security incident in the image and then into the GPS to determine the location of the ATM where the security incident occurs, and then the alert will be generated via GSM to the connected mobile phone.

To assess the extent to which SRSAID improves ATM incidents detection, captured images are analyzed, and detection results are achieved through alert messages, dashboard, and location on the map, as shown in Figures 5.2, 5.3, 5.4, and Figure 5.5.

Figure 5. 2 GSM alerts received from the Raspberry PI on a mobile phone

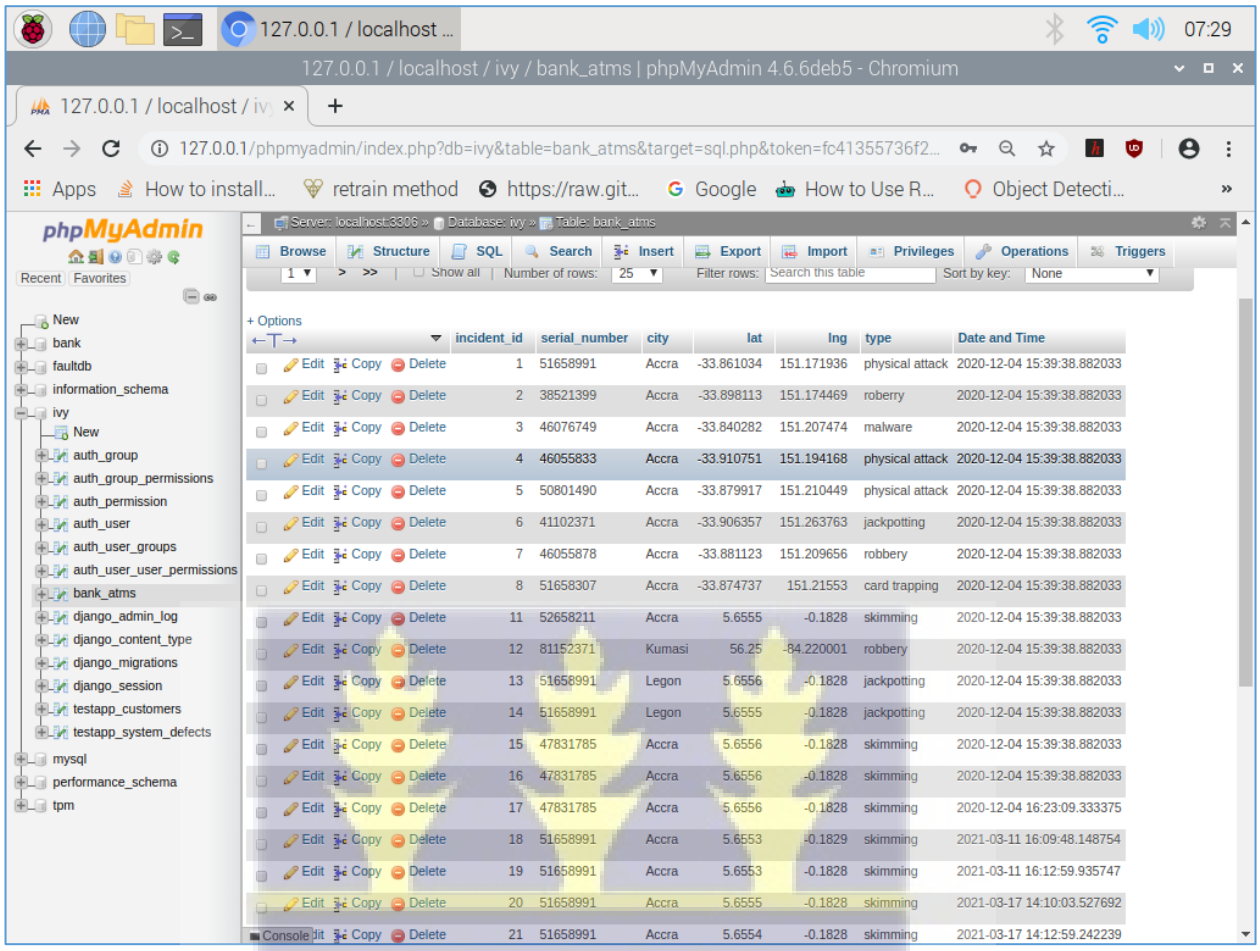


Figure 5. 3 Autocreation of Results Database

Detail	City	Category	Date
View	Accra	occlusion	April 17, 2021
View	Accra	skimming	April 17, 2021
View	Accra	skimming	March 22, 2021
View	Accra	skimming	March 22, 2021
View	Accra	skimming	March 22, 2021
View	Accra	skimming	March 17, 2021
View	Accra	robbery	March 17, 2021
View	Accra	robbery	March 17, 2021
View	Accra	skimming	March 17, 2021
View	Accra	occlusion	March 17, 2021
View	Accra	skimming	March 17, 2021
View	Accra	skimming	March 17, 2021

Figure 5. 4 Dashboard Results

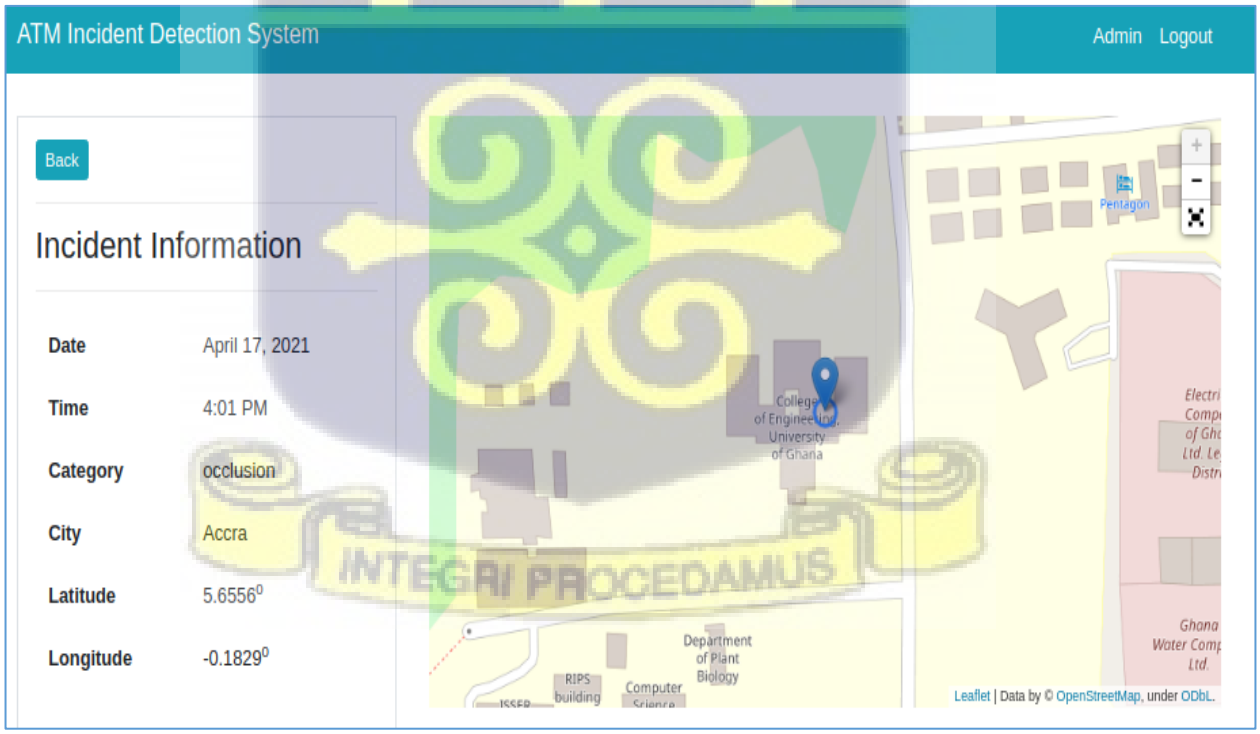


Figure 5. 5 Location on a map through GPS.

5.3 Security Incident Detection Using ssdlite_mobilenet_V2

There are different issues in security incident detection. Suppose there are so many activities in a captured image, tampering activity or security incident is detected with high accuracy, as shown in *figure 5.6*

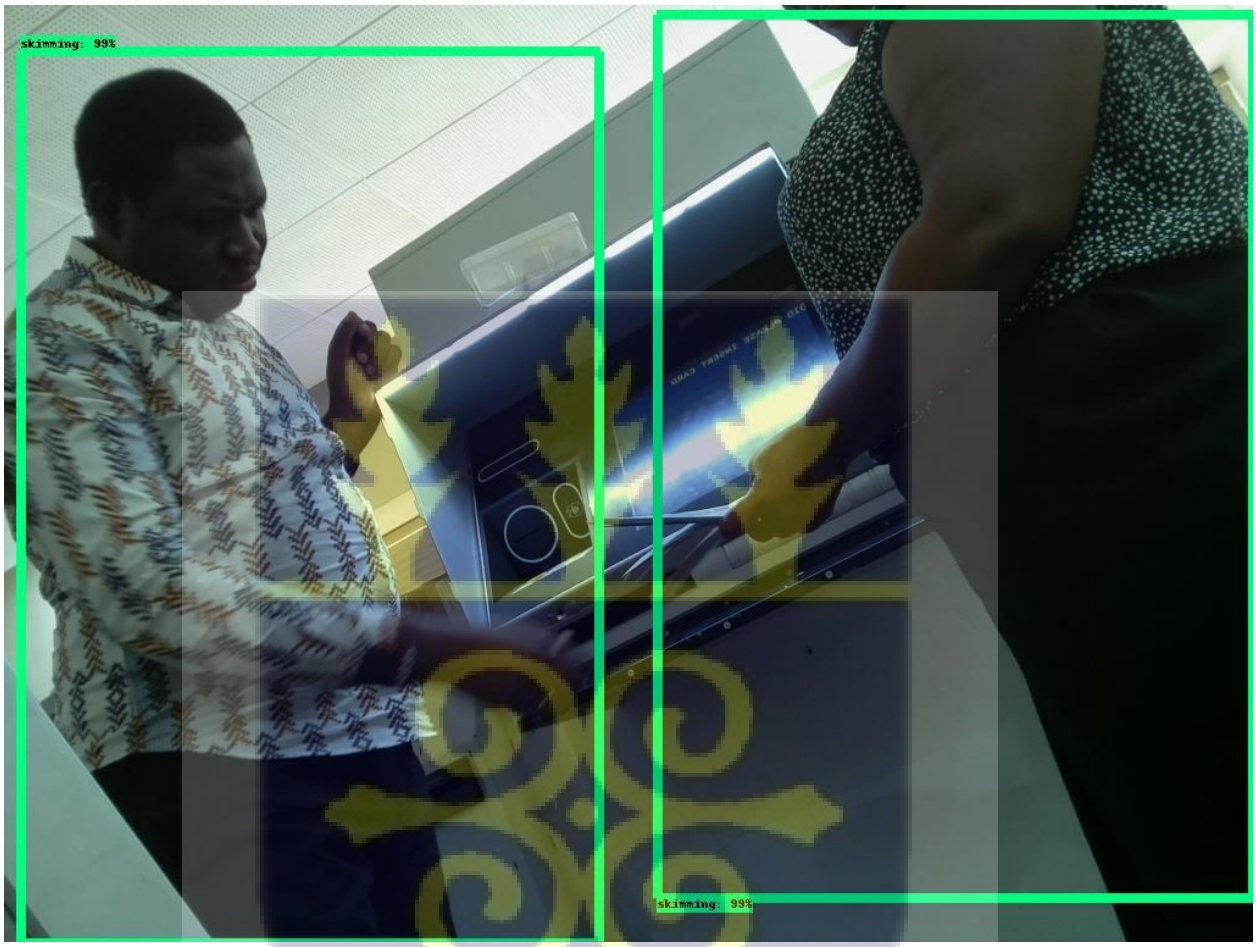


Figure 5. 6 Security Incident Detection Using Faster R-CNN

5.3.1 Regional Proposal Extraction

The first step of training the network is wrapping the region proposal from the original image with reference to the bounding box's points, width, and height and stored in a *.csv* file. The region of interest is defined in this stage. For this, the ROI label is first assigned to the region proposal. After the label definition, each image in the training dataset is annotated with the rectangle bounding box and the label. For each image, the points, the width, and the height of the bounding box are separated and saved in the table. After annotating the data table with the image location, bounding box coordinates and position are exported to the workspace and saved to a particular path in a *.csv* file. A regression model is trained to correct the predicted detection windows on bounding box correction offset using CNN features to reduce the localization errors. The confidence of security incident images is shown in figure 5.7.

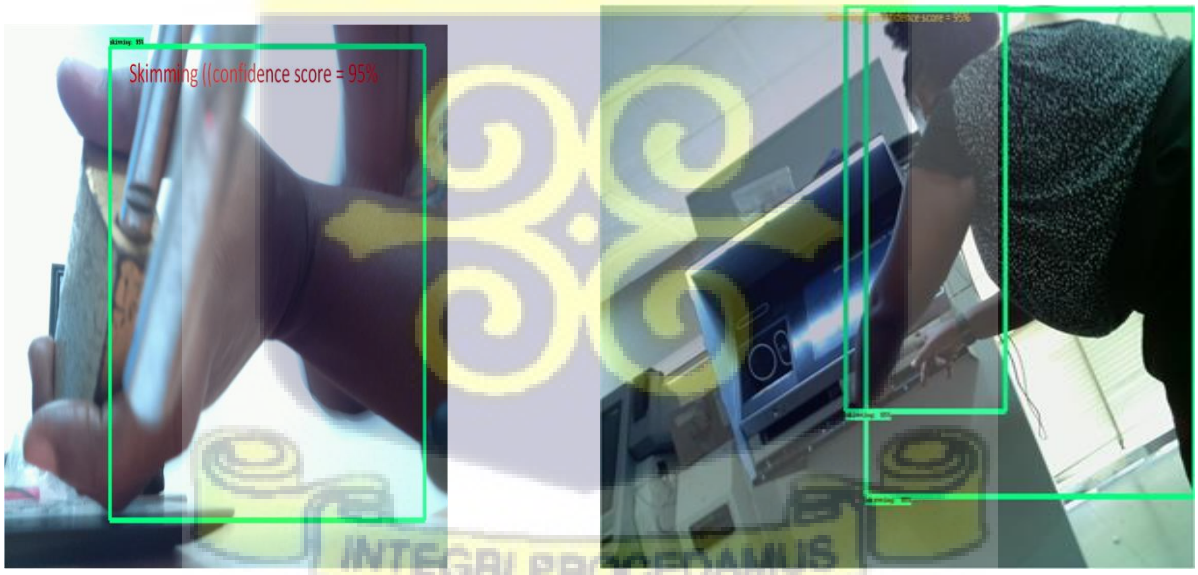


Figure 5. 7 Region Proposal Extraction using types of security incidents.

5.3.2 Performance Analysis for Different Mini Batch Sizes and Epoch

Estimating different parameters using mini-batch size and epoch was done on the Raspberry Pi.

Table 5. 1 Security Incident Detection using ssdlite_mobilenet_V2

Batch Size	***Epoch***	***Training Time(min)***	***Testing Time(min/sec)***	***Accuracy (%)***
500	50	6	1.05	96

Table 5. 2 Security Incident Detection using ALEXNET

Batch Size	***Epoch***	***Training Time(min)***	***Testing Time(min/sec)***	***Accuracy (%)***
500	50	8.30	*1.5*	*80*

Mini- Batch size

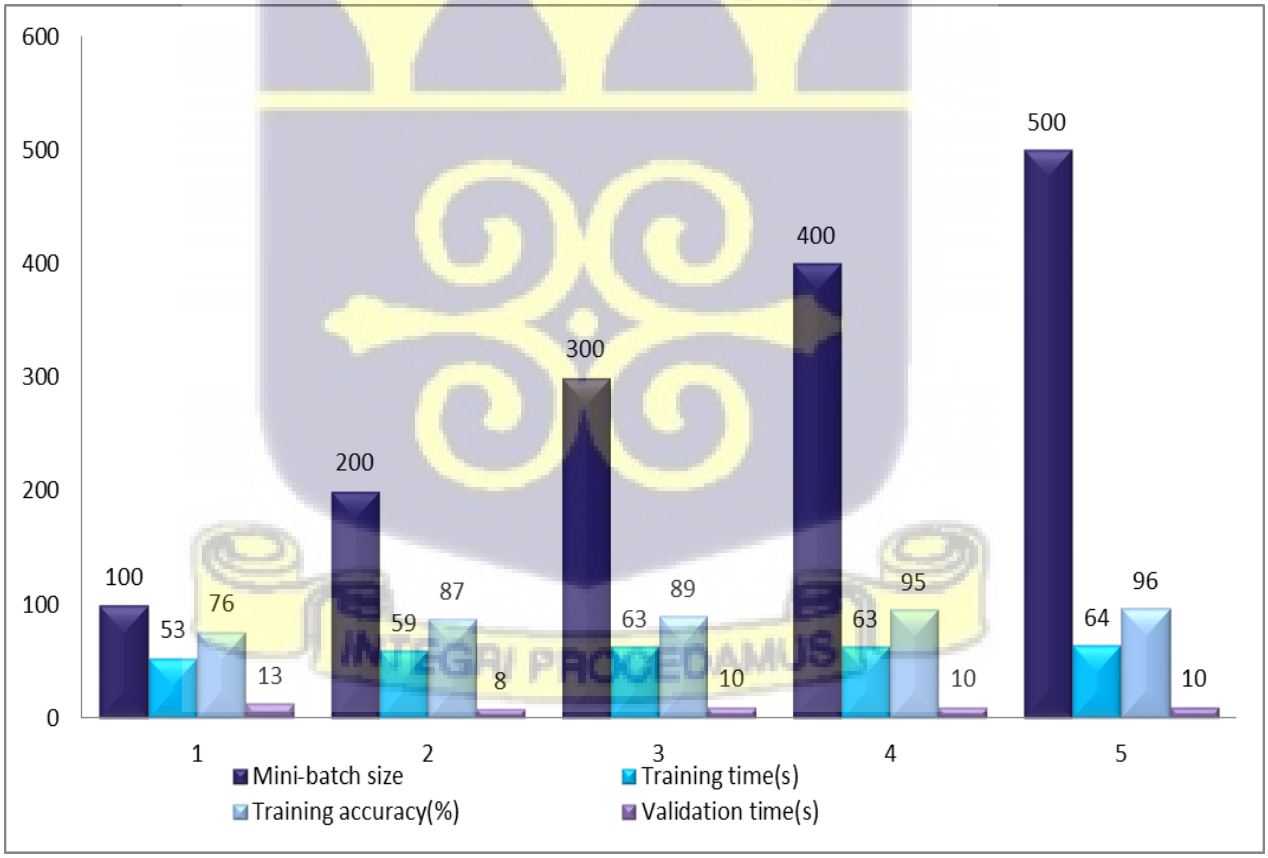
Figure 5.8 illustrates the mini-batch size of both CNN architectures and the other parameters. At every epoch, the subset of the training images is known as mini-batch size. It is used to update the weights. Each iteration uses a different mini-batch. The mini-batch accuracy reported during training relates to the accuracy of the particular mini-batch at the given iteration.

Epoch

Figures 5.8 through 5.12 illustrate the comparison analysis of epochs of the R-CNN training for both CNN architectures and the other parameters used in this study. Epoch is a hyperparameter that defines the number of times the learning algorithms will work through the entire training dataset. An epoch is composed of one or more batches.

Cross-Entropy

Figure 5.13 shows the cross-entropy during the training process. In deep learning, cross-entropy measures how far the predicted value is from the true label. The weights are changed to reduce cross-entropy, thus reducing the difference between the prediction and true labels and, therefore, better accuracy.



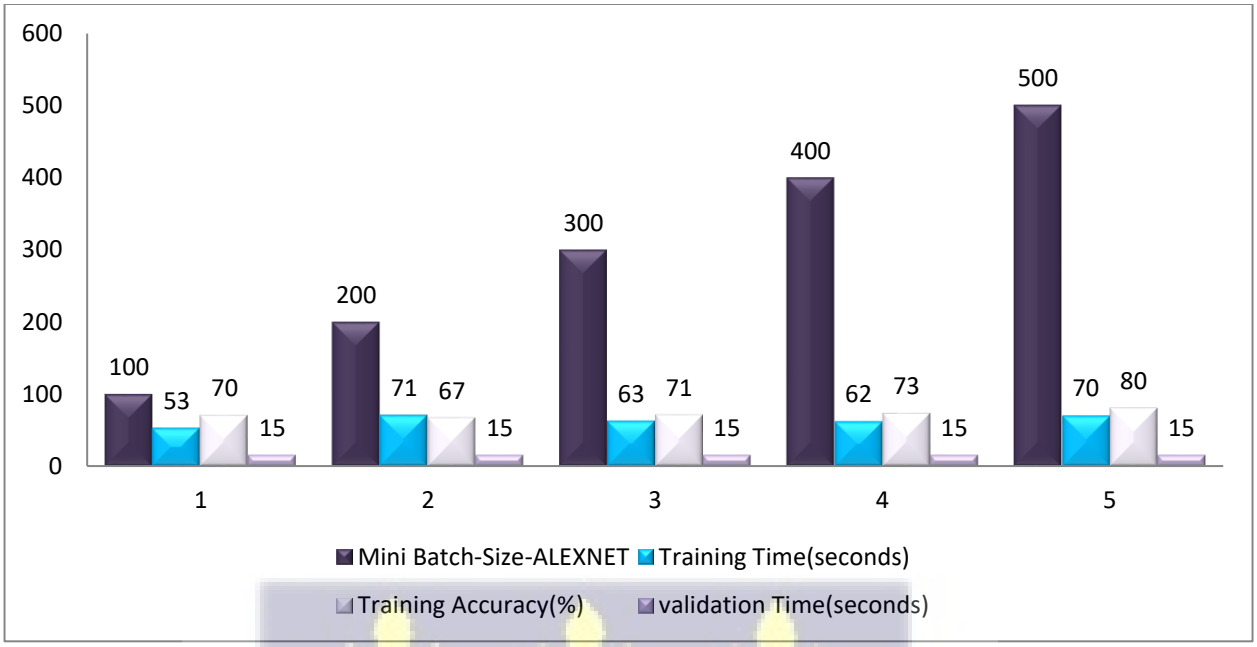
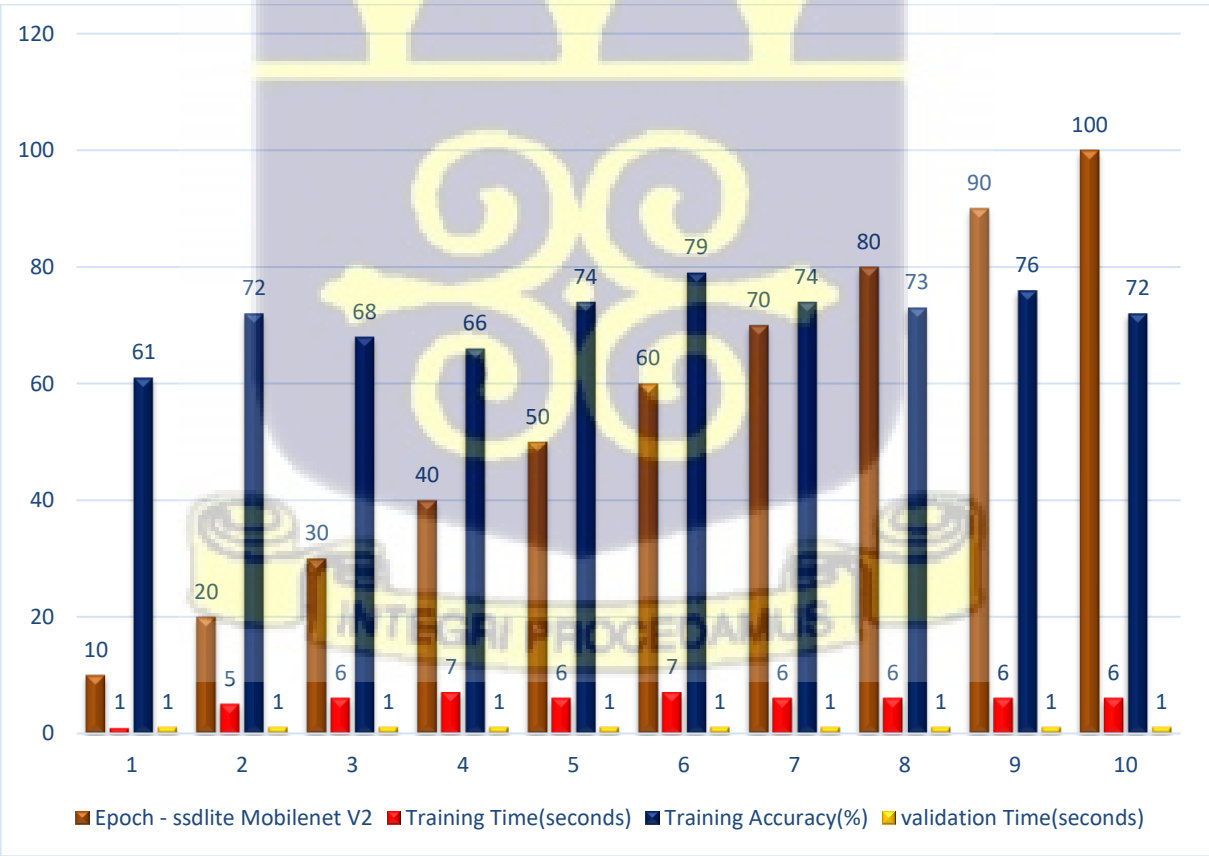


Figure 5. 8 Performance analysis mini-batch size of ssdlite_mobilenet_V2 and ALEXNET.



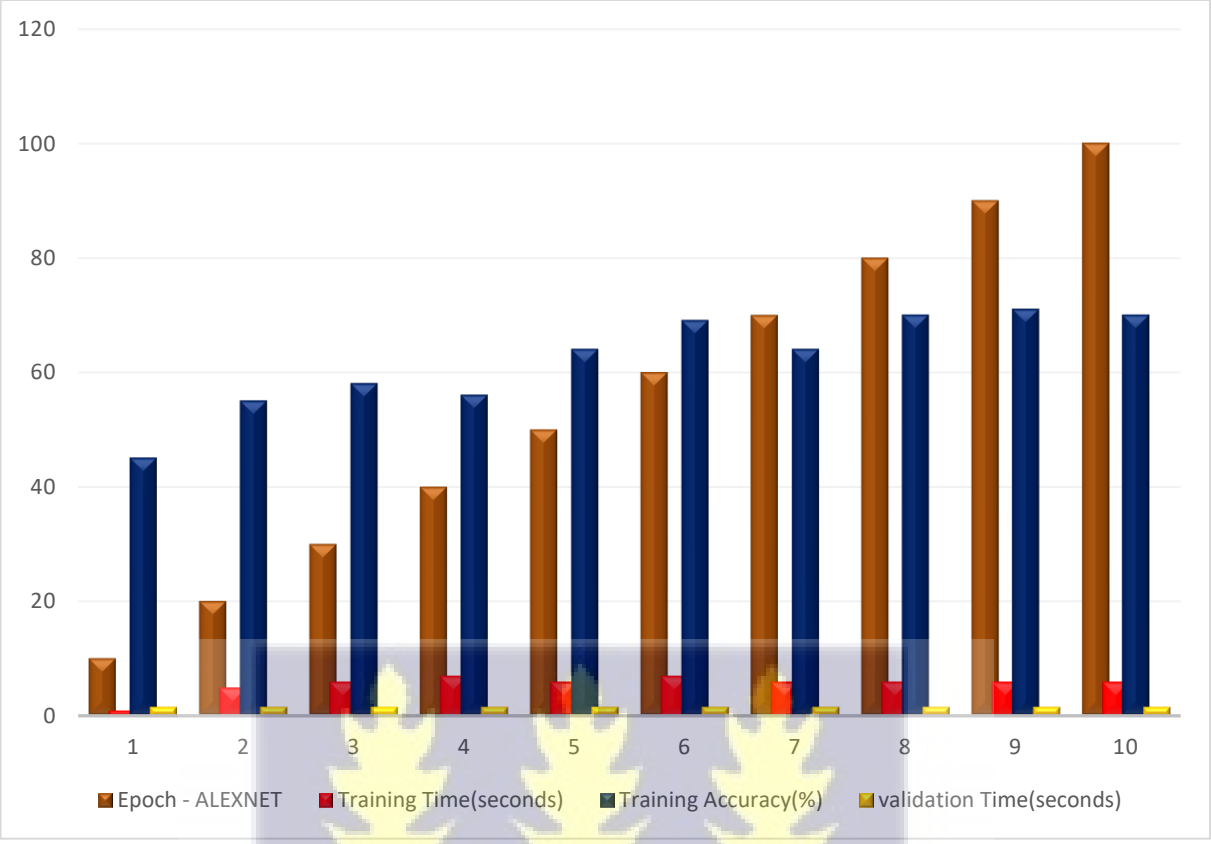
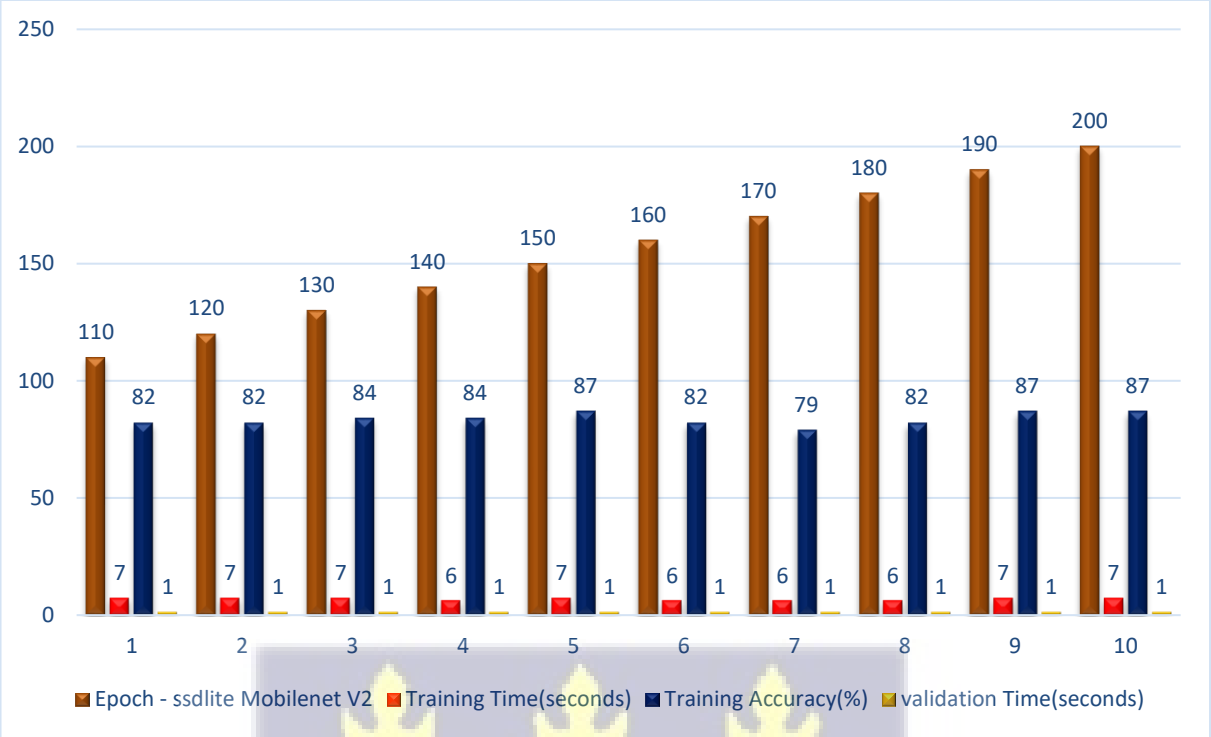


Figure 5. 9 Performance analysis of Epoch from steps 10-100.



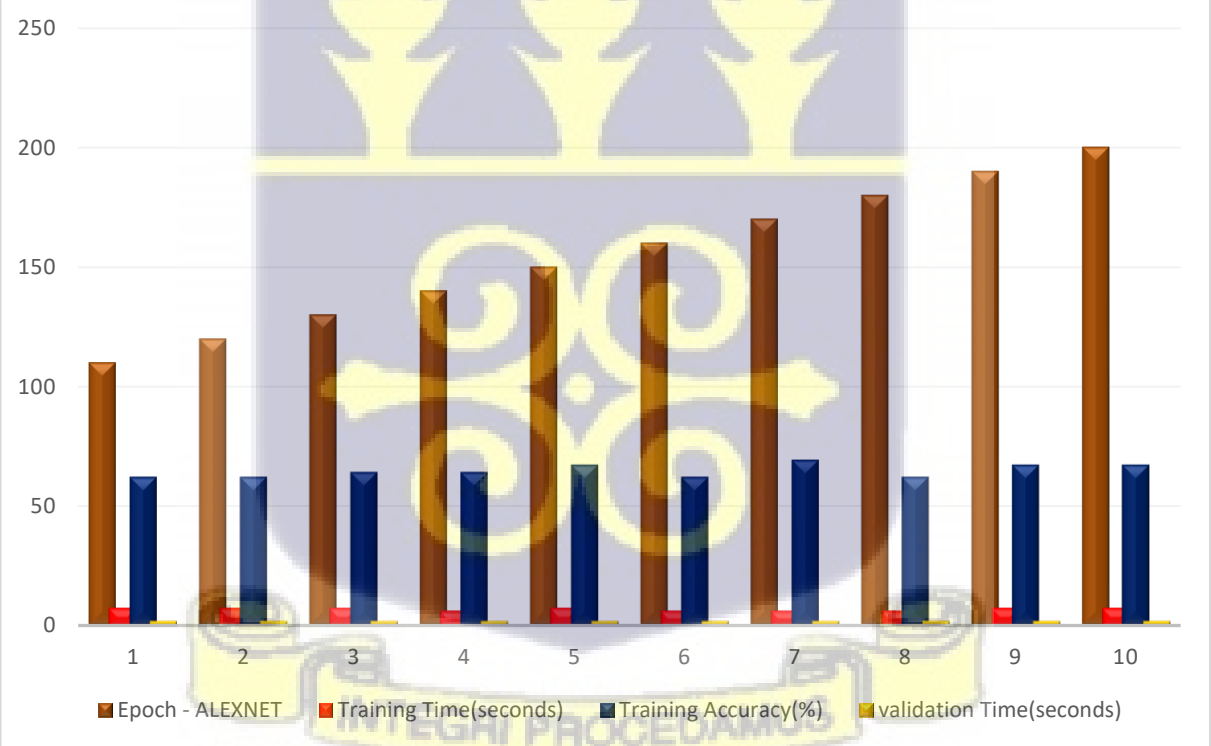


Figure 5. 10 Performance analysis of Epoch from step 110-200

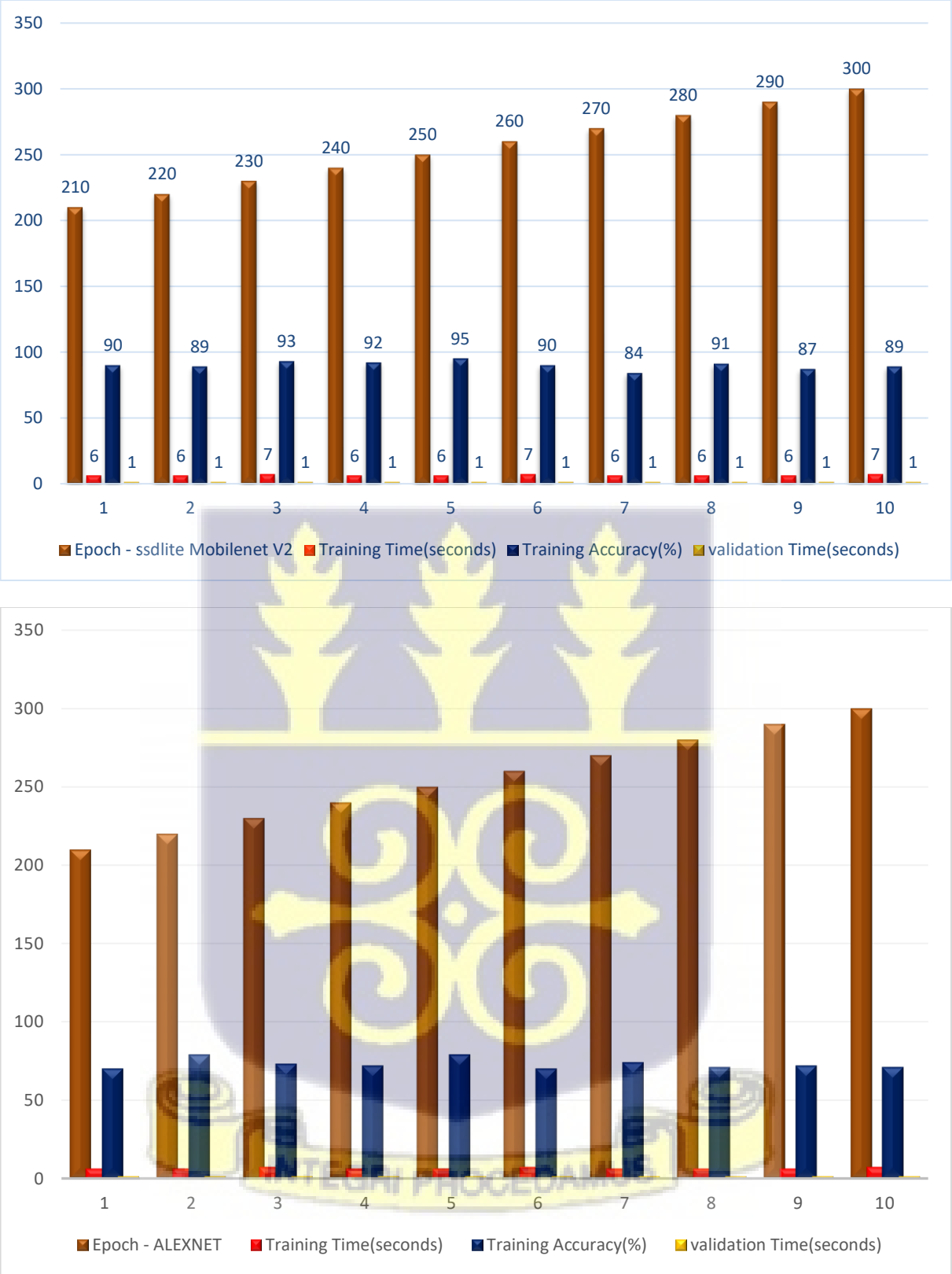


Figure 5. 11 Performance analysis of Epoch from step 210-300

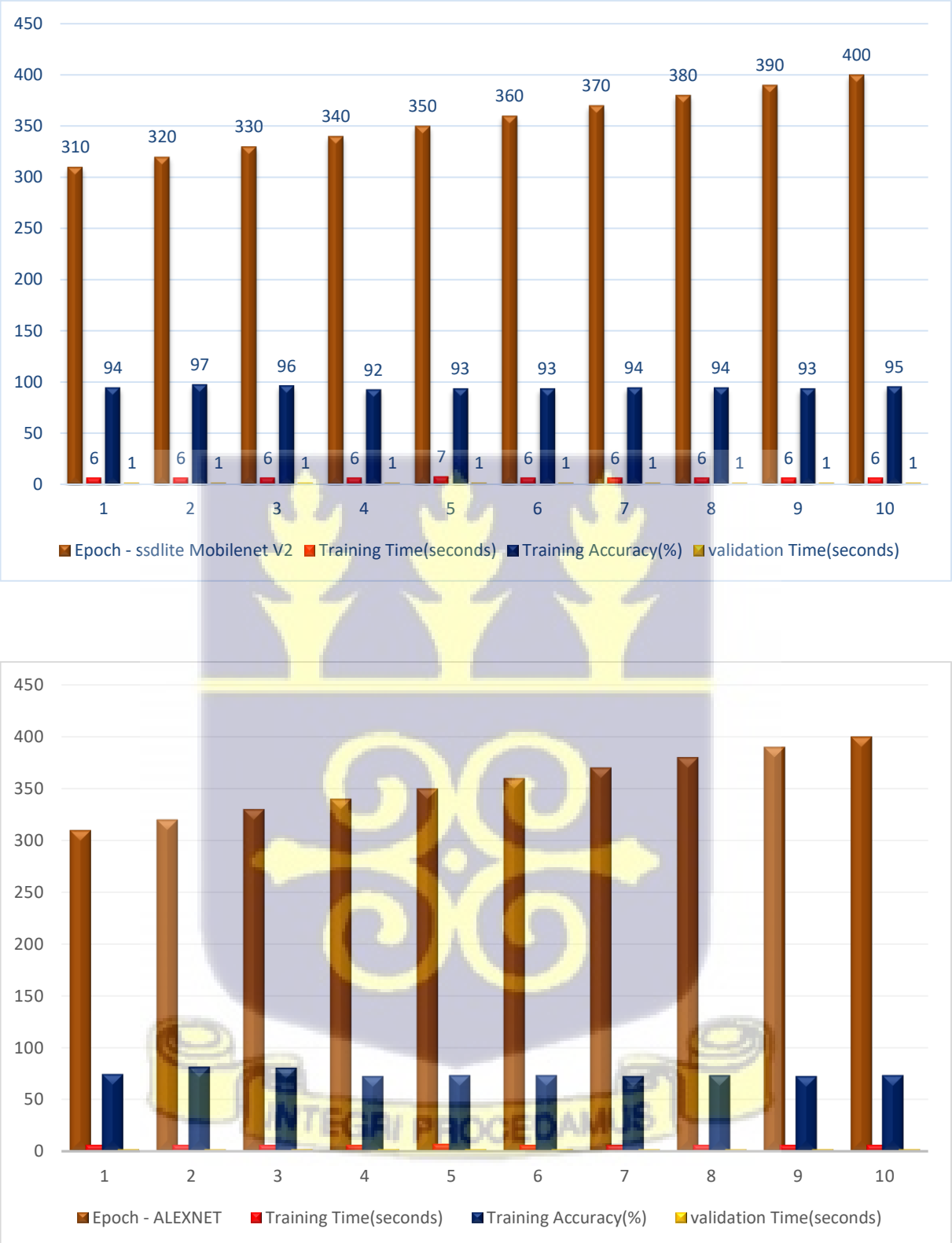
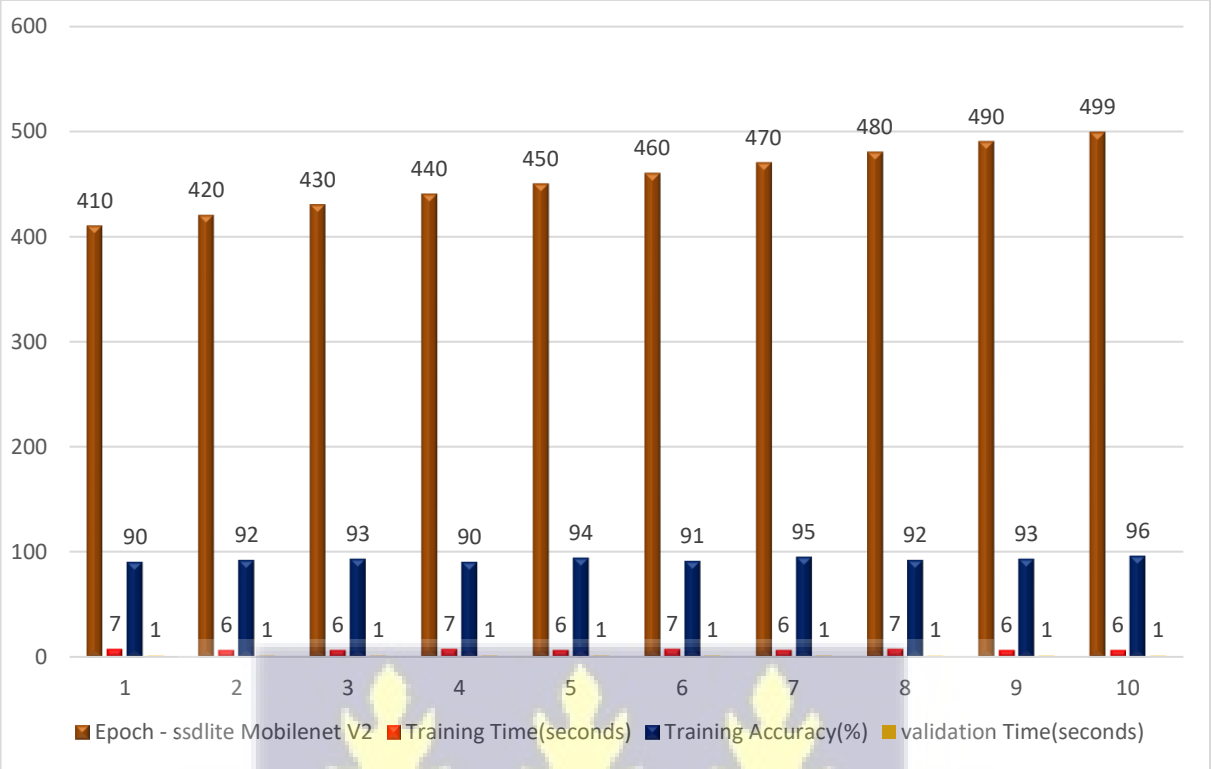


Figure 5. 12 Performance analysis of Epoch from step 310-400



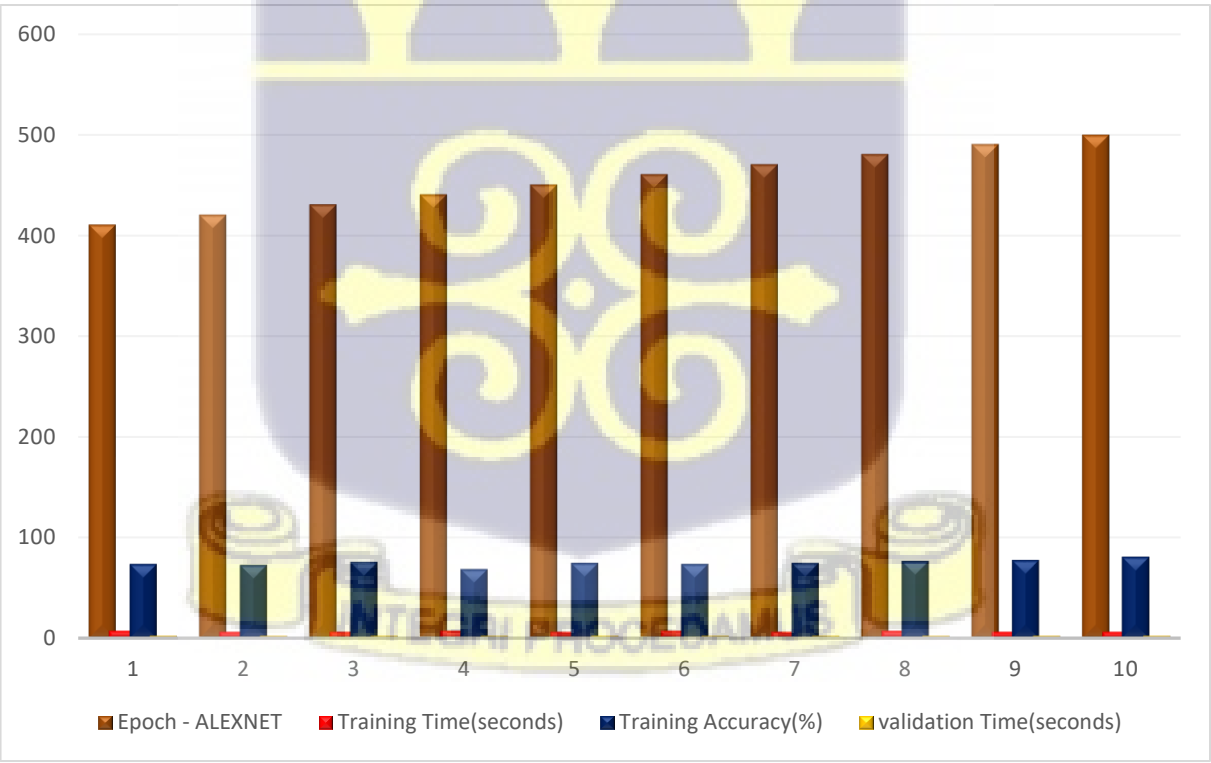


Figure 5. 13 Performance analysis of Epoch from step 410-499

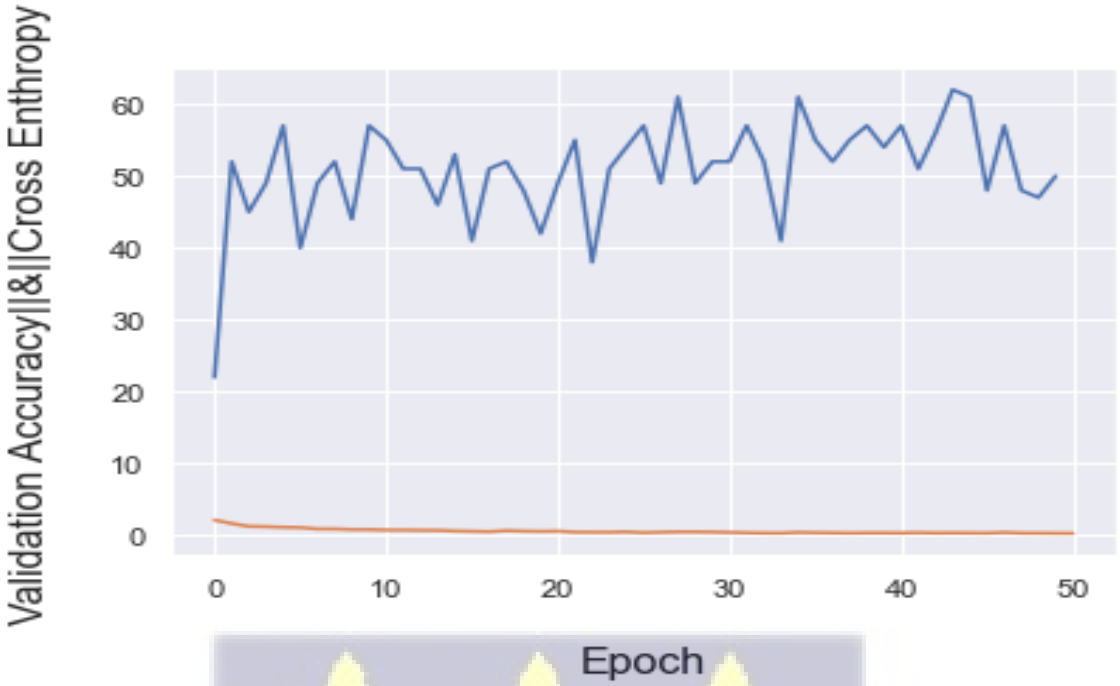


Figure 5. 14 Performance analysis of cross-entropy and validation accuracy at every epoch.

5.4 Results and Discussion for Defects Incident Detection

Figure 5. 15 The SVM Optimization History Plot

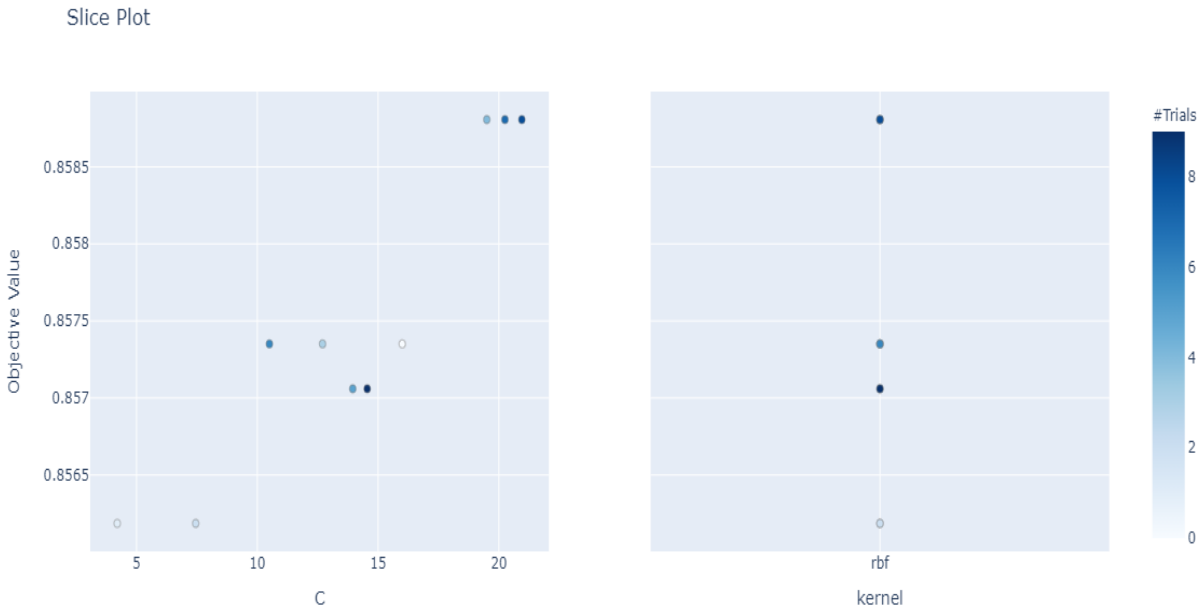


Figure 5. 16 The slice plot of the SVM model optimization

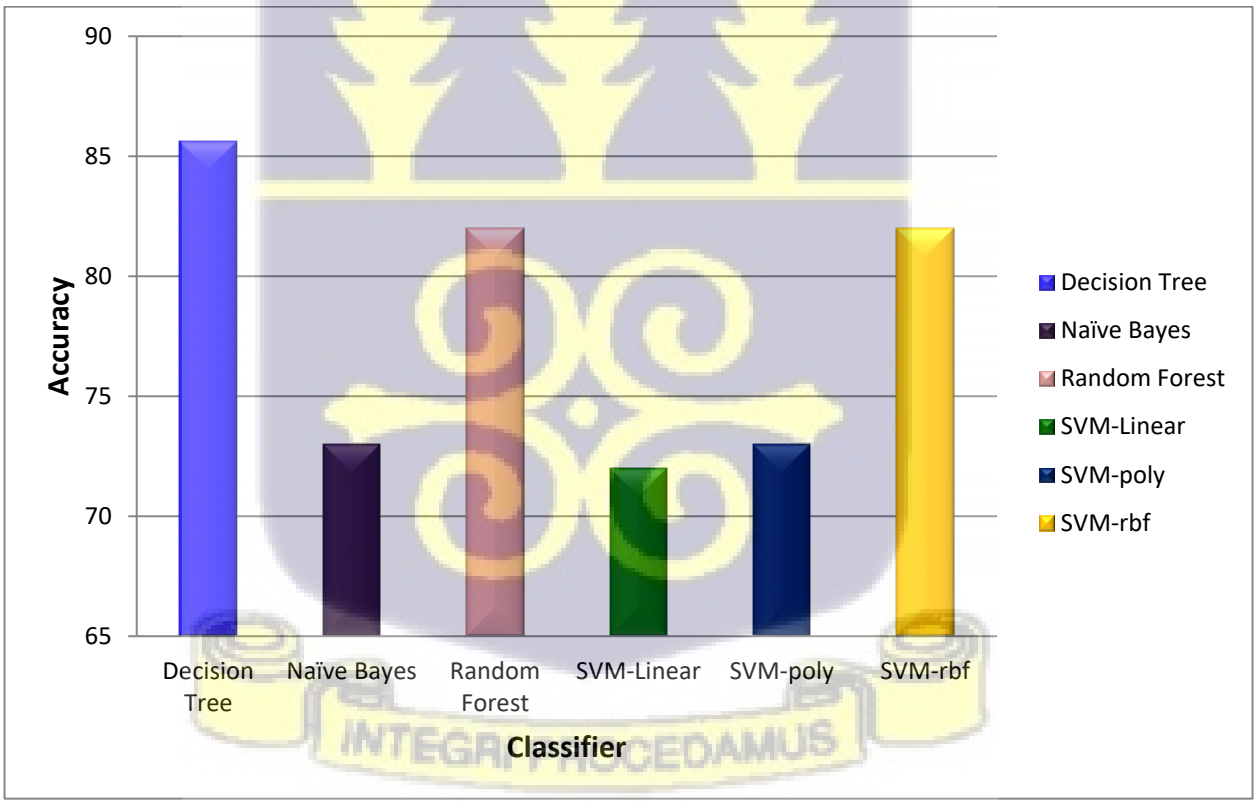


Figure 5. 17 Graph showing the comparison of accuracy for classifiers in the raw state.

The figure below shows SVC plots on the various classifiers (linear, polynomial, and radial basis function) on the ATM system 2018incidents dataset. From the performance metrics and overall statistics presented, it is observed that the support vector machine performs better classification with an accuracy of 86% using the RBF kernel function, followed by the polynomial kernel with 78%, and linear SVM emerged as the least SVM classifier with an accuracy of 77%.

A summary of results and comparison with SVM and other machine learning algorithms such as Naïve Bayes, Decision Tree, and Random Forest is presented in Figures 5.16 and 17.

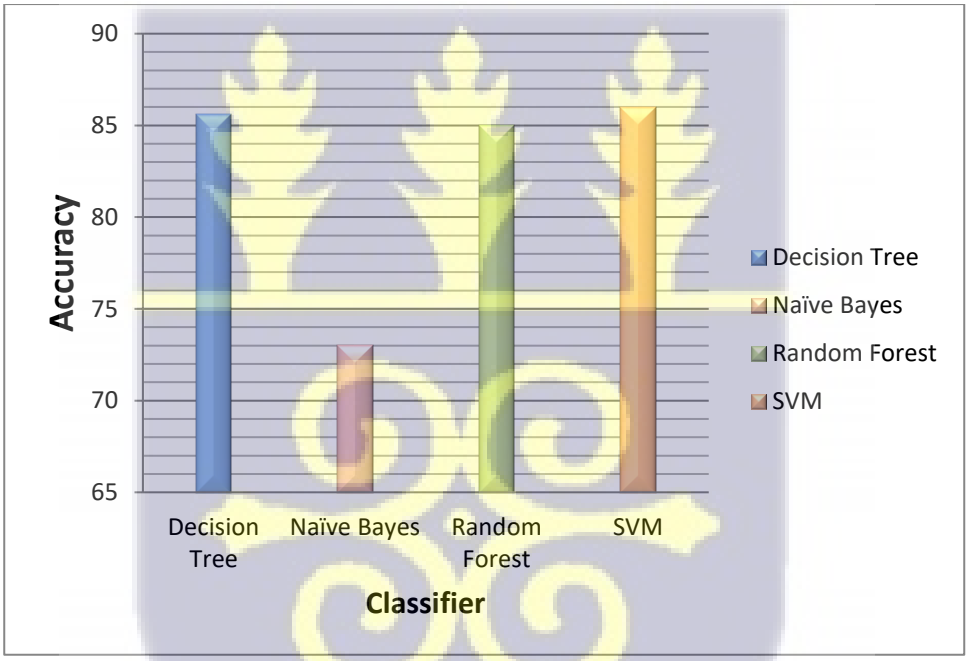


Figure 5. 18 Graph showing the comparison of accuracy for classifiers after hyperparameter tuning.

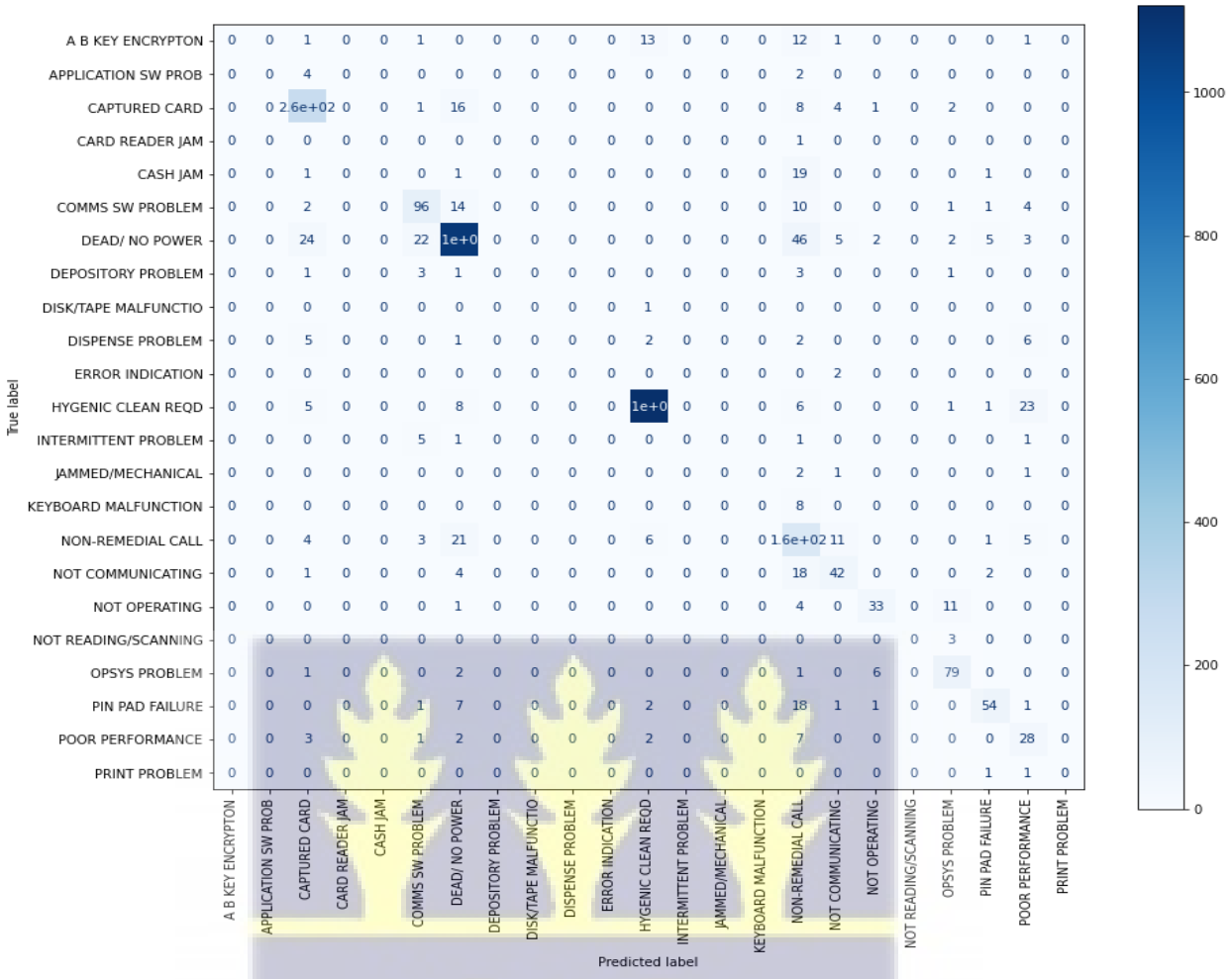


Figure 5. 19 Plot of Confusion Matrix with class labels

Figure 5.18 shows the confusion matrix of the SVM classifiers utilized to compute the performance metric of the SVM classifiers. The confusion matrix allows visualization of the performance of the supervised machine learning algorithm.

Table 5. 3 Averages performance analysis of SVM classifiers.

Kernels used	Data size Description	Average accuracy rate (%)
Linear	11452	0.77
Polynomial	11452	0.78
Radial basis function	11452	0.86

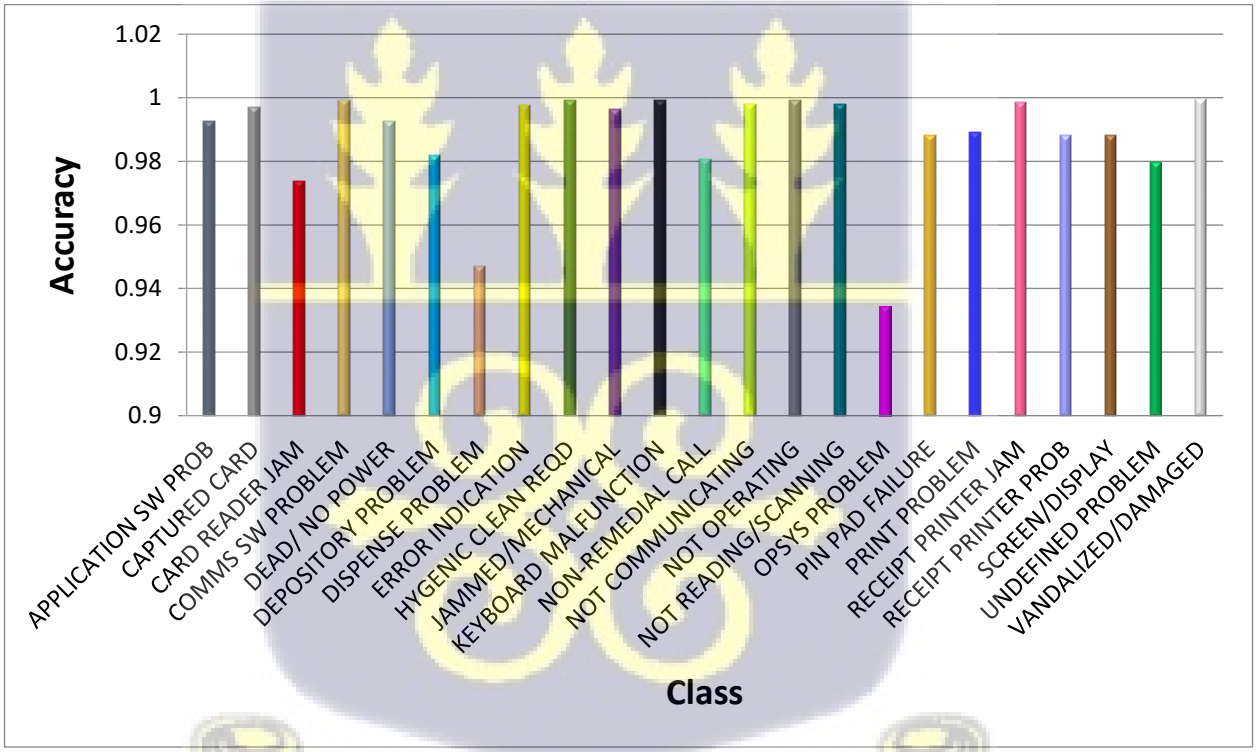
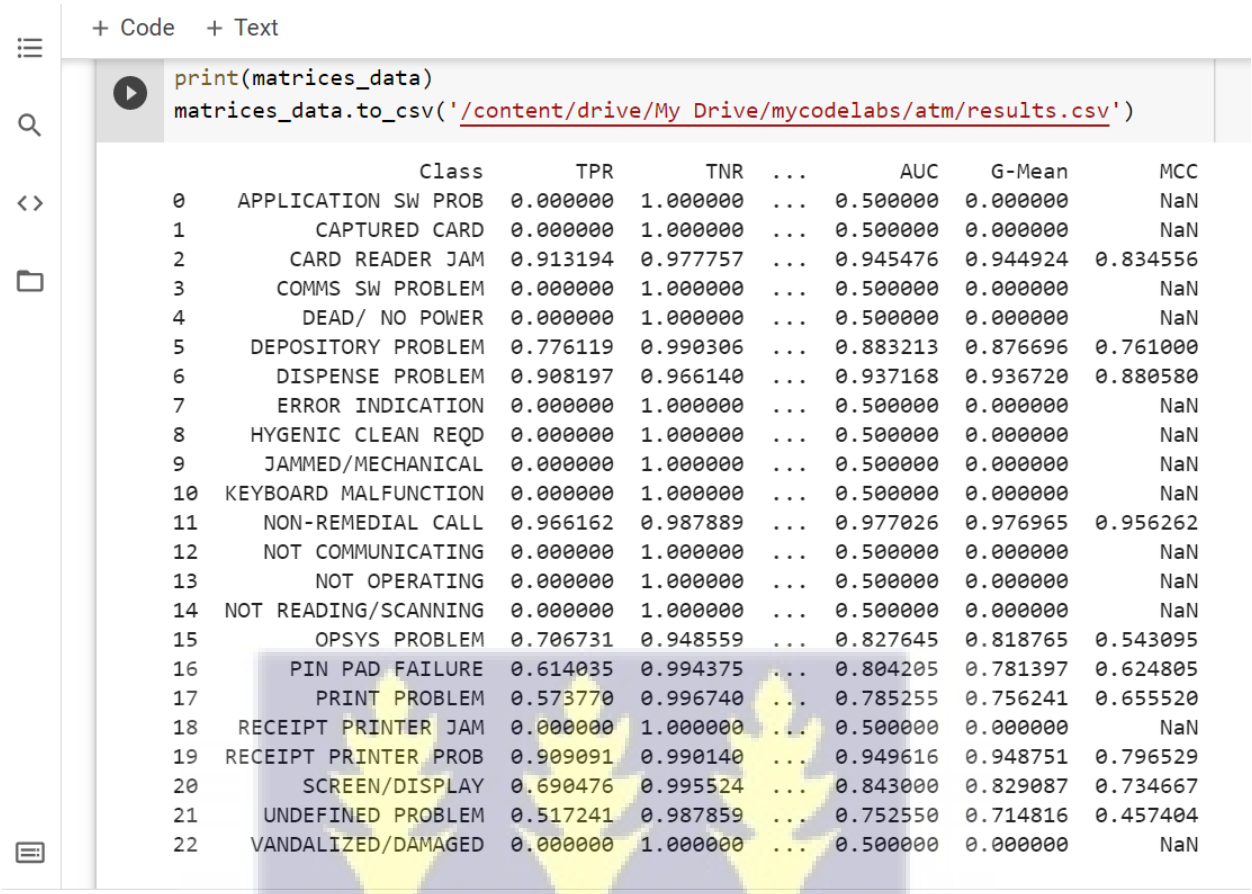


Figure 5. 20 Plot of ACC of the SVM classifier of the predicted labels.

Other performance metrics and comparative analyses of the machine learning models are provided in Appendix B.



+ Code + Text

```
print(matrices_data)
matrices_data.to_csv('/content/drive/My Drive/mycodelabs/atm/results.csv')
```

	Class	TPR	TNR	...	AUC	G-Mean	MCC
0	APPLICATION SW PROB	0.000000	1.000000	...	0.500000	0.000000	NaN
1	CAPTURED CARD	0.000000	1.000000	...	0.500000	0.000000	NaN
2	CARD READER JAM	0.913194	0.977757	...	0.945476	0.944924	0.834556
3	COMMS SW PROBLEM	0.000000	1.000000	...	0.500000	0.000000	NaN
4	DEAD/ NO POWER	0.000000	1.000000	...	0.500000	0.000000	NaN
5	DEPOSITORY PROBLEM	0.776119	0.990306	...	0.883213	0.876696	0.761000
6	DISPENSE PROBLEM	0.908197	0.966140	...	0.937168	0.936720	0.880580
7	ERROR INDICATION	0.000000	1.000000	...	0.500000	0.000000	NaN
8	HYGENIC CLEAN REQD	0.000000	1.000000	...	0.500000	0.000000	NaN
9	JAMMED/MECHANICAL	0.000000	1.000000	...	0.500000	0.000000	NaN
10	KEYBOARD MALFUNCTION	0.000000	1.000000	...	0.500000	0.000000	NaN
11	NON-REMEDIAL CALL	0.966162	0.987889	...	0.977026	0.976965	0.956262
12	NOT COMMUNICATING	0.000000	1.000000	...	0.500000	0.000000	NaN
13	NOT OPERATING	0.000000	1.000000	...	0.500000	0.000000	NaN
14	NOT READING/SCANNING	0.000000	1.000000	...	0.500000	0.000000	NaN
15	OPSYS PROBLEM	0.706731	0.948559	...	0.827645	0.818765	0.543095
16	PIN PAD FAILURE	0.614035	0.994375	...	0.804205	0.781397	0.624805
17	PRINT PROBLEM	0.573770	0.996740	...	0.785255	0.756241	0.655520
18	RECEIPT PRINTER JAM	0.000000	1.000000	...	0.500000	0.000000	NaN
19	RECEIPT PRINTER PROB	0.909091	0.990140	...	0.949616	0.948751	0.796529
20	SCREEN/DISPLAY	0.690476	0.995524	...	0.843000	0.829087	0.734667
21	UNDEFINED PROBLEM	0.517241	0.987859	...	0.752550	0.714816	0.457404
22	VANDALIZED/DAMAGED	0.000000	1.000000	...	0.500000	0.000000	NaN

Figure 5. 21 Classification Report

Figure 5.20 summarizes the average results and comparisons based on precision with other machine learning algorithms such as Naïve Bayes, Decision Tree, and Random forest. For this study, some of the classes are much larger (more instances) than others; hence the metric is weighted toward the largest ones; the micro average precision is considered.

However, the plots of accuracy, AUC, G-mean, and MCC of the classifier are shown in APPENDIX B.

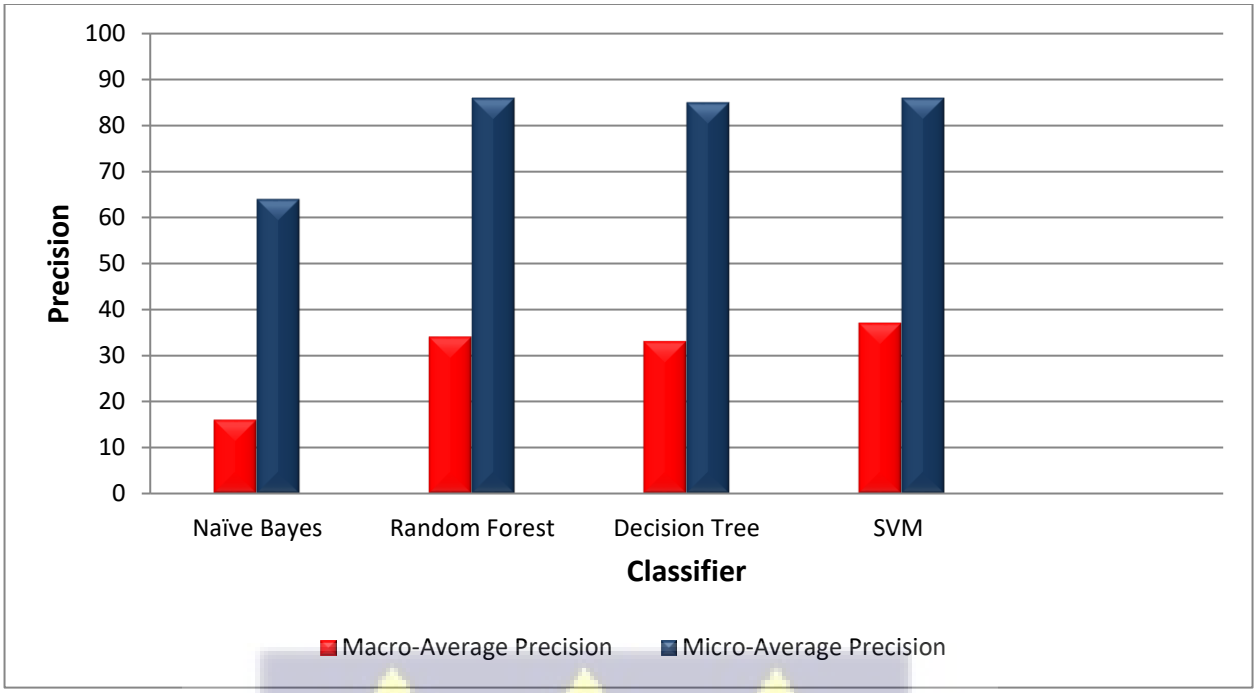


Figure 5. 22 Comparative Analysis on precision of the classifiers.

Table 5. 4 Comparative Analysis on *roc_auc_score* of the classifiers

Classifier	Roc_auc_score
Decision Tree	0.8474
Random Forest	0.8070
Gaussian Naïve Bayes	0.8403
SVM	0.9060

In evaluating the classifiers obtained with the analyzed methods, the most widely employed performance measures are precision, as shown in figure 5.21and receiver operating characteristics and area under the curve score(roc_auc_score). The figure below shows the plot of the classifiers of the NCR2018incident dataset. From performance metrics and overall statistics presented in the table above, it is observed that SVM performs better classification with the roc_auc_score as indicated in the table above. The figure below shows a plot of the classifiers.

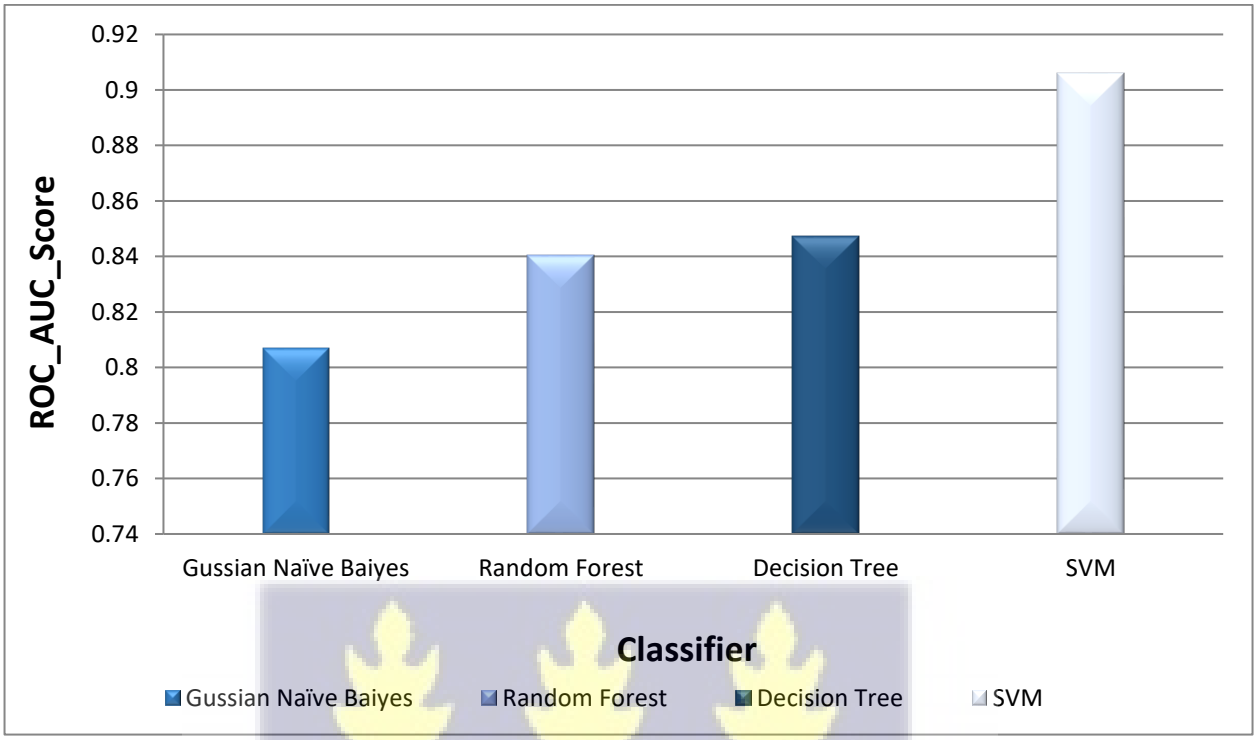


Figure 5. 23 Comparative Analysis on roc_auc_score of the classifiers.

5.5 Analysis of Results

The results and performance metrics obtained for Regional Convolutional Neural Network (R-CNN) with ssdlite_mobilenet_V2, and Support Vector Machines (SVM) are further analyzed to determine their statistical significance. The one-way analysis of variance (ANOVA) is a case of the linear model. It tests the hypothesis that all group means are equal versus the alternative hypothesis that at least one group is different from the others. Table 5.6 shows the ANOVA analysis of the models in terms of AUC and F1-Score. The columns of tables represent the classifier and the corresponding sum of squares (SS), degree of freedom (DF), mean squares (MS), F-values (F), and P values of the models.

Table 5. 5 ANOVA Analysis for Machine Learning Classifier

Groups	*Count*	*Sum*	*Average*	*Variance*
R CNN	2	1.411	0.643	0.006962
ssdlite_mobilenet_V2	2	1.231	0.8935	0.004901
SVM (Linear Kernel)	2	1.648	0.9745	0.000761
SVM (Polynomial Kernel)	2	1.372	0.966	0.00045
SVM (RBF Kernel)	2	1.722	0.9785	0.000421

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.028671	11	0.002606	1.456044	0.263912	2.717331
Within Groups	0.021481	12	0.00179			
Total	0.050152	23				

The results from Table 5.5 show that the source of variation between groups of the classifiers has a p-value of 0.263912. This means that, with a 5% alpha level, if the p-value is greater than or equal to 0.05, then the performance of the classifier will not be significantly affected when using the AUC and F1 score performance metrics.

In summary, the results also imply that the performances of the classifiers based on AUC and F1 scores are justifiable based on the ANOVA results.

5.6 Research Contribution

The research, therefore, achieved its objectives by providing detailed knowledge regarding Machine Learning techniques and algorithms (Regional Convolutional Neural Network (R-CNN) with ssdlite_mobilenet_V2 architecture for security incident detection and Support Vector Machines (SVM)) for detecting ATM system defect incidents. The use of Internet of

Things (IoT) components such as Raspberry Pi, Global Positioning System (GPS), and Global System for Mobile Communications (GSM) have been demonstrated as well to self-report ATM incidents. The proposed technique, with the security incident detection aspect, the R-CNN with ssdlite_mobilenet_V2 detected security incidents such as skimming, robbery, and other categories of security incidents with a confidence score of 95% and above. This indicates that the model will detect security incidents efficiently when higher-resolution cameras capture activities around the ATM terminal.

Again, the proposed model designed and developed using the SVM classifier for ATM system defect incidents detection yielded appreciable accuracies for the predicted labels. This indicates that the saved model, when to run on a real ATM network, will efficiently predict system defects scenarios.

The hardware components that form the IoT components interfaced well with the methodology used. The GPS / GSM module communicated by transporting the analysis generated by the prototype (Raspberry pi) to a mobile device and reported on a dashboard for decision-making.

These findings could inform Bank officials, Security Services and ATM Service Providers, and computer engineers about the importance of Machine Learning and Deep Learning techniques while evaluating the Machine Learning and Deep Learning algorithms proposed for industrial applications at research and development levels.

CHAPTER SIX

CONCLUSION AND RECOMMENDATION

6.1 Conclusion

In conclusion, the proposed design has met all the requirements and specifications. The work has demonstrated a Machine Learning technique using the R-CNN algorithm with ssdlite_mobilenet_V2 architecture for security incident detection and SVM for ATM defect incidents detection. The models designed and developed are used to analyze incident detection. Results from the models designed and developed for the collected datasets verify the results of the model. The IoT components used for communication between the Raspberry Pi and the GPS/GSM module provided information on the map and the messages received from the mobile phone.

This work aimed at developing a novel incident detection model for ATMs incidents based on R-CNN with ssdlite_mobilenet_V2 architecture and support vector machines, which hybridizes and draws on the strengths of both R-CNN with ssdlite_mobilenet_V2 architecture and support vector machines. Both algorithms have been investigated and applied in the development of incident detection. This thesis used R-CNN with ssdlite_mobilenet_V2 architecture and support vector machines to detect and classify ATM security incidents and defects incidents into classes under each aspect.

With this study's security incident detection aspect, Security incident detection for ATMs using Deep Learning Algorithm is proposed. The detection of security incidents is obtained using R-CNN with ssdlite_mobilenet_V2 architecture. Security detection using ALEXNET has 80%

accuracy. The disadvantage of ALEXNET, it contains around 160M parameters. Most of these are where every input is connected to every output by a learnable weight in fully connected layers. The total number of ALEXNET layers is more than the ssdlite_mobilenet_V2, so training takes a substantial amount of time. Because of this, the ssdlite_mobilenet_V2 architecture is used. The Accuracy of security incident detection using ssdlite_mobilenet_V2 is 96%. The performance analysis among different Mini batch sizes and epoch estimation of different parameters such as training time, testing time, and Accuracy are evaluated. The RCNN algorithm detects the region where the security incident occurs.

Again, this paper proposed an approach for ATM incident detection to self-report security activities and defective state of system components for ATMs. The technique consists of a security incident detector and a defect incident detector classifier. The security incident detector is implemented with bounding boxes, a regional proposal, and Regional Convolutional Neural Network (R-CNN). The technique used in this research achieved detection accuracy for security incidents such as Skimming, Physical Attacks, and others of 96%, 99%, and others on tampering activities captured by the Pi camera.

In this study of defect incident detection aspect, SVMs have been considered preferable to other classification techniques due to several advantages. They enable the separation (classification) of defects incidents into problem code description and the other classes using the soft margin. Thus, it has a nonlinear dividing hyperplane, which prevails over the discrimination within the dataset. The generalization ability of any newly arrived data for classification was considered over other classification techniques. Thus, the defect detection system combines two computational intelligence schemes and achieves higher defect detection accuracy. The average classification accuracies achieved by the SVCs are 77%, 78%, and 86%, which show the

performance capability of the SVCs model. These classification accuracies are obtained due to the careful selection of the features for training and developing the model and fine-tuning the SVCs' parameters using the hyperparameter tuning approach. These results are much better than those obtained using Decision Tree, Random Forest, and Naive–Bayes. The behavior of the dataset has a significant impact on classification results. Based on this work, the developed SVM model was tested using NCR2018 incident data. The study sought the best-performing classifier for analyzing the ATM defect incidents datasets for incidents. The RBF kernel was adjudged the best, with an average accuracy rate of 86%. The RBF kernel is therefore recommended.

Moreover, the technique used achieved accuracies of 99%, 99%, 97%, 99%, 99%, 98%, 94%, 99%, 99%, 99%, 99%, 98%, 99%, 99%, 99%, 93%, 98%, 98%, 99%, 98%, 98%, 97% and 99% for the problems; APPLICATION SW PROB, CAPTURED CARD, CARD READER JAM, COMMS SW PROBLEM, DEAD/ NO POWER, DEPOSITORY PROBLEM, DISPENSE PROBLEM, ERROR INDICATION, HYGENIC CLEAN REQD, JAMMED/MECHANICAL, KEYBOARD MALFUNCTION, NON-REMEDIAL CALL, NOT COMMUNICATING, NOT OPERATING, NOT READING/SCANNING, OPSYS PROBLEM, PIN PAD FAILURE, PRINT PROBLEM, RECEIPT PRINTER JAM, RECEIPT PRINTER PROB, SCREEN/DISPLAY, UNDEFINED PROBLEM and VANDALIZED/DAMAGED respectively.

6.2 Recommendations

The performance of the proposed SRSAID solution improved the overall security of ATM systems. It would therefore be of interest to carry out further studies using different datasets with varying sizes and degrees of imbalance and classification algorithms to confirm and further establish the practicality and other possible limitations of SRSAID. As this study focused much on addressing ATM incidents, assessing the applicability and efficiency of SRSAID in different application areas of system security is highly recommended. Considering the computational complexity of the machine learning algorithms on image and video processing, it is also of interest to research how the time requirements of SRSAID can be minimized by tackling it at the algorithmic level or how other outlier detection techniques with lesser computational time can be used.

REFERENCES

[1] Bernard Marr, "The 7 Biggest Technology Trends To Disrupt Banking & Financial Services In 2020," *Forbes*, 2019. [Online]. Available: https://www.forbes.com/sites/bernardmarr/2019/11/08/the-7-biggest-technology-trends-to-disrupt-banking--financial-services-in-2020/#7453943c2c42. [Accessed: 20-Jul-2020].

[2] G. Odachi, "ATM Technology and Banking System in West African Sub-Region: Prospects and Challenges," *African Res. Rev.*, vol. 5, no. 2, pp. 104–114, 2011.

[3] N. S. Bhati, "Innovations in Service Accessibility: A Role of Information Technology with Special Reference to Banking Industry in India," *Proc. 2019 Int. Conf. Comput. Intell. Knowl. Econ. ICCIKE 2019*, pp. 160–163, 2019.

[4] N. T. Device, "Remote Automated Teller Machine (ATM) Monitoring Solution MapmyIndia Tamper Detected !!"

[5] NCR Corporation, "ABOUT NCR We turn everyday transactions into meaningful relationships," 2020. [Online]. Available: https://www.ncr.com/about. [Accessed: 20-Jul-2020].

[6] E. Attah-Botchwey, "Electronic Banking and the Challenges of the Ghanaian Business Environment," *Int. J. Humanit. Soc. Sci.*, vol. 4, no. 9, 2014.

[7] B. Editorial Committee, "Bank of Ghana Annual Report 2018," Accra, 2018.

[8] C. Logan, "ATM Technology," 2018. [Online]. Available: https://www.kal.com/en/atm-technology. [Accessed: 01-Apr-2020].

[9] Y. Xia, B. Zhang, and F. Coenen, "Face Occlusion Detection Using Deep Convolutional Neural Networks," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 30, no. 9, pp. 1–24, 2016.

[10] R. Girshick, "Fast R-CNN," *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2015 Inter, pp. 1440–1448, 2015.

[11] D. Walther, L. Itti, M. Riesenhuber, T. Poggio, and C. Koch, "Attentional selection for object recognition A gentle way," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 2525, pp. 472–479, 2002.

[12] J. R. R. Uijlings, K. E. A. Van De Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *Int. J. Comput. Vis.*, vol. 104, no. 2, pp. 154–171, 2013.

[13] K. Fukushima, "Neocognitron: A hierarchical neural network capable of visual pattern recognition," *Neural Networks*, vol. 1, no. 2, pp. 119–130, 1988.

[14] R. Girshick, J. Donahue, T. Darrell, J. Malik, U. C. Berkeley, and J. Malik, "1043.0690," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 1, p. 5000, 2014.

[15] D. Huttenlocher, "Computer vision," *Comput. Sci. Handbook, Second Ed.*, pp. 43-1-43–23, 2004.

[16] R. A. Hughes, "Geoscience data and derived spatial information: Societal impacts and benefits, and relevance to geological surveys and agencies," *Spec. Pap. Geol. Soc. Am.*, vol. 482, pp. 35–40, 2011.

[17] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017.

[18] W. Liu *et al.*, "SSD: Single shot multibox detector," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9905 LNCS, pp. 21–37, 2016.

[19] Y. Xia, B. Zhang, S. Engineering, X. Jiaotong-Liverpool, F. Coenen, and C. N. Networks, "International Journal of Pattern Recognition and Artificial Intelligence c World Scientific Publishing Company 3," 2016.

[20] C. Münker and O. Stenroos, "Object detection from images using convolutional neural networks," *Thesis*, pp. 1–87, 2017.

[21] Y. Ahuja and S. Kumar Yadav, "Multiclass Classification and Support Vector Machine," *Glob. J. Comput. Sci. Technol. Interdiscip.*, vol. 12, no. 11, pp. 14–19, 2012.

[22] A. Rosales-Pérez, H. Terashima-Marin, S. García, F. Herrera, and C. A. C. Coello, "MC2ESVM: Multiclass Classification Based on Cooperative Evolution of Support Vector Machines," *IEEE Comput. Intell. Mag.*, vol. 13, no. 2, pp. 18–29, 2018.

[23] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2, pp. 2169–2178, 2006.

[24] N. Neighbor, P. W. Estimation, C. Trees, K. Neighbor, C. Trees, and C. Trees, "Multi-class Support Vector Machines," pp. 22–28, 2002.

[25] A. Rocha and S. K. Goldenstein, "Multiclass from binary: Expanding One-versus-all, one-versus-one and ECOC-based approaches," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 25, no. 2, pp. 289–302, 2014.

[26] R. A. Sowah *et al.*, "Decision Support System (DSS) for Fraud Detection in Health Insurance Claims Using Genetic Support Vector Machines (GSVMs)," vol. 2019, no. January 2007, 2019.

[27] R. Rifkin *et al.*, "An analytical method for multiclass molecular cancer classification," *SIAM Rev.*, vol. 45, no. 4, pp. 706–723, 2003.

[28] "HubelWiesel1968." pp. 1–35, 2005.

[29] R. Kumaresan, "ATM- Robbery prevention using machine learning technique," vol. 5, no. 1, pp. 781–783, 2019.

[30] D. (DJ) Sarkar, "No TitleBuilding Robust Production-Ready Deep Learning Vision

Models in Minutes," 2019. [Online]. Available: https://medium.com/google-developer-experts/building-robust-production-ready-deep-learning-vision-models-in-minutes-acd716f6450a. [Accessed: 08-Oct-2020].

[31] D. (DJ) Sarkar, "A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning," 2019. [Online]. Available: A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning%0A. [Accessed: 08-Oct-2020].

[32] B. Philip, "How artificial intelligence works First wave : Symbolic artificial intelligence," no. March, 2019.

[33] R. Saha, "Transfer Learning – A Comparative Analysis," no. December, 2018.

[34] K. B. Devi, S. M. M. Roomi, M. Meena, and S. Meghana, "Deep Learn Helmets-Enhancing Security at ATMs," *2019 5th Int. Conf. Adv. Comput. Commun. Syst. ICACCS 2019*, pp. 1111–1116, 2019.

[35] B. Kuditchar, "Hybrid Cluster-Based Sampling Technique for Class Imbalance Problems By This Thesis Is Submitted To the University of Ghana, Legon in Partial Fulfilment of the Requirement for the Award of Mphil Computer Engineering Degree," no. July, 2019.

[36] B. Miguel and S. D. C. Machado, "Extremely Imbalanced Smell-based Defect Prediction," no. September, 2019.

[37] P. Phoungphol, "ScholarWorks @ Georgia State University," 2013.

[38] U. K. N and S. R. D, "ATM- Security using machine learning technique in IoT," vol. 5, no. 2, pp. 150–153, 2019.

[39] P. Dollár and C. L. Zitnick, "Fast edge detection using structured forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 8, pp. 1558–1570, 2015.

[40] S. Torkamani, A. Dicks, and V. Lohweg, "Anomaly detection on ATMs via time series motif discovery," *IEEE Int. Conf. Emerg. Technol. Fact. Autom. ETFA*, vol. 2016-Novem, no. September, 2016.

[41] W. Journal, "Automated Teller Machine Analysis under Host-Bank Systems through Telephone Network," no. February, 2018.

[42] S. M. H. Hasheminejad and Z. Reisjafari, "ATM management prediction using Artificial Intelligence techniques: A survey," *Intell. Decis. Technol.*, vol. 11, no. 3, pp. 375–398, 2017.

[43] B. S. Adebayo and M. I. Kolo, "Agent-Based Faults Monitoring in Automatic Teller Machines," *Comput. Inf. Syst. Dev. Informatics*, vol. 3, no. 4, pp. 1–6, 2013.

[44] S. Rao and H. Mane, "ATM Availability Management System," *IOSR J. Comput. Eng.*, vol. 2, no. 2278–8727, pp. 21–31, 2017.

[45] J. Levashina, C. J. Hartwell, F. P. Morgeson, and M. A. Campion, "The structured employment interview: Narrative and quantitative review of the research literature,"

Pers. Psychol., vol. 67, no. 1, pp. 241–293, 2014.

[46] G. Chitiyo, "Research in education," *Educ. Zimbabwe 21st Century What Every Educ. Should Know*, pp. 133–144, 2014.

[47] Z. Zhao and P. Zheng, "Object Detection with Deep Learning : A Review," pp. 1–21, 2012.

[48] V. K. Chauhan, A. Sharma, and K. Dahiya, "Problem formulations and solvers in linear SVM : a review," *Artif. Intell. Rev.*, 2018.

[49] F. Pedregosa *et al.*, "Scikit-learn : Machine Learning in Python To cite this version : Scikit-learn : Machine Learning in Python," *J. Mach. Learn. Res.*, vol. 12, 2011.

[50] K. J. Zhang, C. H. Guo, Z. H. Niu, L. F. Liu, and Y. Bin Yang, "SCOD: Dynamical Spatial Constraints for Object Detection," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11295 LNCS, no. November, pp. 17–28, 2019.

[51] T. Lin, "LabelImg is a graphical image annotation tool and label object bounding boxes in images," 2019. [Online]. Available: https://pypi.org/project/labelImg/. [Accessed: 24-Sep-2020].

[52] S. Pokhrel, "Image Data Labelling and Annotation — Everything you need to know | by Sabina Pokhrel | Towards Data Science," *Towards Data Science*, 2020. [Online]. Available: https://towardsdatascience.com/image-data-labelling-and-annotation-everything-you-need-to-know-86ede6c684b1. [Accessed: 24-Sep-2020].

[53] P. F. Felzenszwalb and D. P. Huttenlocher, "Seg-Ijcv," pp. 1–26, 2015.

[54] B. Akella, "What is Support Vector Machine? SVM Algorithm in Machine Learning," 2020. [Online]. Available: https://intellipaat.com/blog/tutorial/machine-learning-tutorial/svm-algorithm-in-python/#Building-a-Support-Vector-Machine-Classification-Model-in-Machine-Learning-Using-Python.

APPENDICES

APPENDIX A

I. EXPERIMENTAL PROCESS IMPLEMENTATION (SECURITY INCIDENT DETECTION)

```
import numpy as np
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, BatchNormalization, AveragePooling2D,
Dense, Dropout, Flatten
from keras.optimizers import Adam
from keras import regularizers
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
import pandas as pd
import matplotlib.pyplot as plt
import os
import pickle
from keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau
import cv2
import picamera
from time import sleep

import numpy as np
import os
import sys

import cv2


# Current working directory
CWD_PATH = os.getcwd()

IMAGE_NAME = "cam_image.jpg"
IMAGE_PATH = os.path.join(CWD_PATH, IMAGE_NAME)

def captureImage():
    with picamera.PiCamera() as camera:
        print("Capturing image...")
        sleep(1.000)
        camera.vflip = True
        camera.capture(IMAGE_NAME)
        print("Image captured")
```

```
        IMAGE_CV2  = cv2.imread(IMAGE_PATH)
        cv2.imshow("Captured Image", cv2.resize(IMAGE_CV2, (800,600)) )
        cv2.waitKey(3000)           # Show the captured image for 3s
        cv2.destroyAllWindows()
        return

captureImage()

print()
print("-----------------------------------------------")
print("Loading TensorFlow engine")
import tensorflow as tf


from utils import label_map_util
from utils import visualization_utils as vis_util
def get_conv_model(dim = (150,150, 3)):
    '''This function will create and compile a CNN given the input dimension'''
    inp_shape = dim
    act = 'relu'
    drop = .25
    kernal_reg = regularizers.l1(.001)
    optimizer = Adam(lr = .0001)
    model = Sequential()
    model.add(Conv2D(64, kernel_size=(3,3),activation=act, input_shape = inp_shape,
                “kernel_regularizer = kernal_reg,
                kernel_initializer = 'he_uniform',  padding = 'same', name = 'Input_Layer'))
    model.add(MaxPooling2D(pool_size=(2, 2),  strides = (3,3)))”
    model.add(Conv2D(64, (3, 3), activation=act, kernel_regularizer = kernal_reg,
                kernel_initializer = 'he_uniform',padding = 'same'))
    model.add(MaxPooling2D(pool_size=(2, 2), strides = (3,3)))
    model.add(Conv2D(128, (3, 3), activation=act, kernel_regularizer = kernal_reg,
                kernel_initializer = 'he_uniform',padding = 'same'))
    model.add(Conv2D(128, (3, 3), activation=act, kernel_regularizer = kernal_reg,
                kernel_initializer = 'he_uniform',padding = 'same'))
    model.add(MaxPooling2D(pool_size=(2, 2), strides = (3,3)))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(32, activation='relu'))
    model.add(Dropout(drop))
    model.add(Dense(3, activation='softmax', name = 'Output_Layer'))
    model.compile(loss = 'categorical_crossentropy', optimizer = optimizer, metrics = ['accurac
y'])
    return model
```

```
# Name of model folder
“MODEL_NAME = 'ssdlite_mobilenet_v2_ imageNet_2018_05_09'“
# Object detection folder path
“OBJ_DETECTION_FOLDER_PATH” = '/home/pi/project/tensorflow/models-master/research/object_detection'


# Below paths are for Ivy's trained model
“PATH_TO_CKPT = “os.path.join(OBJ_DETECTION_FOLDER_PATH,”'ivy_ATMMs','frozen_inference_graph.pb')”
“PATH_TO_LABELS = “os.path.join(OBJ_DETECTION_FOLDER_PATH,”'ivy_ATMMs', 'atm_labelmap.pbtxt')”



#------------------------------------------------------------
NUM_CLASSES = 13


“detection_graph = tf.Graph()”
with detection_graph.as_default():
  od_graph_def = tf.GraphDef()
  with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
    serialized_graph = fid.read()
    od_graph_def.ParseFromString(serialized_graph)
    tf.import_graph_def(od_graph_def, name='')


“label_map = label_map_util.load_labelmap(PATH_TO_LABELS)”
categories = label_map_util.convert_label_map_to_categories(label_map, max_num_classes=
”NUM_CLASSES, use_display_name=True)”
category_index = label_map_util.create_category_index(categories)

print("TensorFlow engine loaded")
print("-----------------------------------------------")
print()

while True:
    with detection_graph.as_default():
      with tf.Session(graph=detection_graph) as sess:
        image_np = cv2.imread(IMAGE_PATH)
        #image_np = cv2.flip(image_np, 0)
        # Expand dimensions since the model expects images to have shape: [1, None, None, 3]
```

```
image_np_expanded = np.expand_dims(image_np, axis=0)
image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')
# Each box represents a part of the image where a particular object was detected.
“boxes = detection_graph.get_tensor_by_name('detection_boxes:0')”
# Each score represent how level of confidence for each of the objects.
# Score is shown on the result image, together with the class label.
scores = detection_graph.get_tensor_by_name('detection_scores:0')
classes = detection_graph.get_tensor_by_name('detection_classes:0')
num_detections = detection_graph.get_tensor_by_name('num_detections:0')

# Actual detection.
(boxes, scores, classes, num_detections) = sess.run(
  [boxes, scores, classes, num_detections],
  feed_dict={image_tensor: image_np_expanded})

print()
print()

if num_detections < 1:
    print("No detections made on this image")
    captureImage()
    continue
    #exit(1)

detection_name     = “category_index[classes[0][0]]["name"]
detection_score    = scores[0][0]
print("The most prominent detection was: '", end="")
print(detection_name, end="")
print("'")
print("With score of: ", end="")
print(int(round(detection_score*100)), end="")
print("%")

print()

print("Visualizing detection")
# Visualization of the results of a detection.
vis_util.visualize_boxes_and_labels_on_image_array(
    image_np,
    np.squeeze(boxes),
    np.squeeze(classes).astype(np.int32),
    np.squeeze(scores),
    category_index,
    use_normalized_coordinates=True,
    line_thickness=8)
```

```
    cv2.imshow('object detection', cv2.resize(image_np, (800,600)))

    if cv2.waitKey(5000) & 0xFF == ord('q'):
        print("Wait key executed")
    cv2.destroyAllWindows()
    print("Visualization done.")
    print()


    print("Attempting to send GSM alert with GPS information")

    import gsmFunctions as gsmFx
    import gpsFunctions as gpsFx

    """
        Phone numbers:
            Dr. Sowah     - 0507714085
            Ivy           - 0244810016


    """
    destPhone      = "0244810016"       #the phone to receive the SMS
    atmSerialNum   = "51658991"
    msgPrefix      = """{}' ({}%) incident detected on ATM with Serial Number: {}.\r\n
“  Location coordinates are: """.format(detection_name, int(round(detection_score*100)),
                                atmSerialNum)
    cityName       = "Accra"

    #initilize GSM module
    gsmFx.initGSM()
    sleep(1.000)

    #initilize GPS system
   “ gpsFx.initGPS”()
    sleep(1.000)

    #attempt to update GPS data
    #report if unsuccessful
   “ if gpsFx.updateGPSData(100) “== False:
        print("GPS data update was unsuccessful. Please try again")
    else:
        print( "Coordinates are: {},{}".format(gpsFx.getLatitudeDeg(),
                                        gpsFx.getLongitudeDeg()))
```

```
#  prepare to log coordinates, ATM serial number etc. in database
import requests
url = "http://localhost/config/api2.php"
my_post_data = {
                "serial_number": atmSerialNum,
                "city": cityName,
                "lat": gpsFx.getLatitudeDeg(),
                "lng": gpsFx.getLongitudeDeg(),
                "type": detection_name
            }

r = requests.post(url, my_post_data)

if "Records added successfully" in r.text:
   print("\nSuccessfully added record in database! Will send SMS next\n")
   sleep(1.000)
else:
   print("Failed to add record to database.")
   sleep(1.000)

#Add coordinates to message prefix and send SMS to destination phone
" newMessage = "{}{},{}".format(msgPrefix, gpsFx.getLatitudeDeg(), gpsFx.getLongitu
deDeg())
print("\nPreparing and sending message to {}...\n".format(destPhone))
print("Message to send is: {}\n".format(newMessage))
" gsmFx.composeSMS(destPhone, newMessage)"
gsmFx.sendSMS()

#turn off GPS to save power
#gpsFx.gpsOff()


#exit(0)
sleep(2.000)
captureImage()
```

APPENDIX B

I. FULL IMPLEMENTATION OF SRSAID (SECURITY INCIDENT DETECTION)

```
from django.db import models

# Create your models here.

Category = (
    ('Security', 'Security'),
    ('Physical', 'Physical'),
    ('Theft', 'Theft'),
)


“class Incident(models.Model):
“serial_number = models.PositiveIntegerField()”
  “  city = models.CharField(max_length=100)”
   “ latitude = models.FloatField(null=True)”
   “ longitude = models.FloatField(null=True)”
   “ category = models.CharField(max_length=50, choices=Category)”
    “date = models.DateTimeField(auto_now_add=True)”

    class Meta:
        ordering = ['-date']

    def __str__(self):
        “return f'{self.category} Incident in {self.city} ’”
```

II. CODE IMPLEMENTATION FOR SOFTWARE: DJANGO VIEW

```
from django.shortcuts import render
from django.contrib.auth.decorators import login_required
from .models import Incident
from . import dates
from django_pandas.io import read_frame
import folium
from folium import plugins


# Create your views here.
“@login_required(login_url='user-login')”
def index(request):

   “incident = Incident.objects.all()””
  “ physical_incident = Incident.objects.filter(category='Physical')”
 “  security_incident = Incident.objects.filter(category='Security')”
  “ theft_incident = Incident.objects.filter(category='Theft')”

  “ count_physical_incident = physical_incident.count()”
  “ count_security_incident = security_incident.count()”
 “  count_theft_incident = theft_incident.count()”

   incident_count = incident.count()

   last_incident = Incident.objects.first()

   bar_labels =” ['January', 'February', 'March', 'April', 'May', 'June',”
            “ 'July', 'August', 'September', 'October', 'November', 'December']”
   bar_data =” [dates.Jan, dates.Feb, dates.Mar, dates.Apr, dates.May, dates.Jun,”
           dates.Jul, dates.Aug, dates.Sep, dates.Oct, dates.Nov, dates.Dec]

   “accra = dates.accra”
   pie_labels = ['Physical', 'Security', 'Theft']
   pie_data = [count_physical_incident,
           count_security_incident, count_theft_incident]
   context = {
      'incident_count': incident_count,
      'bar_labels': bar_labels,
```

```
        'bar_data': bar_data,
        'count_physical_incident': count_physical_incident,
        'count_security_incident': count_security_incident,
        'count_theft_incident': count_theft_incident,
        'pie_data': pie_data,
        'pie_labels': pie_labels,
        'last_incident': last_incident,
        'accra': accra,
    }
    return render(request, 'dashboard/index.html', context)


@login_required(login_url='user-login')
“def incident(request):”
    “incident = Incident.objects.all()”
    “incident_count = incident.count()”

   “ last_incident = Incident.objects.first()”

    context = {
        'incident': incident,
        'incident_count': incident_count,
        'last_incident': last_incident,
    }
   “ return render(request, 'dashboard/incident.html', context)”


@login_required(login_url='user-login')
“def map(request, pk):”
   “ incident = Incident.objects.get(id=pk)
    lat = incident.latitude
    lon = incident.longitude
    city = incident.city”
    tooltip = 'Click for Info'
    m = “folium.Map(location=[lat, lon], zoom_start=17)”
   “ folium.Marker(location=[lat, lon], popup=city, tooltip=tooltip).add_to(m)”
    folium.CircleMarker(location=[lat, lon]).add_to(m)
    plugins.Fullscreen('topright').add_to(m)

    m = m._repr_html_()
    context = {
```

```
        'm': m,
        'incident': incident,
    }
    return render(request, 'dashboard/map.html', context)


@login_required(login_url='user-login')
def allmap(request):
    "incident = Incident.objects.all()"
    df = read_frame(incident, fieldnames=['latitude', 'longitude', 'city'])

    m = "folium.Map(location=[7.689217127736191, -1.25244140625], zoom_start=7)"
    tooltip = 'Click for Info'
    for (index, rows) in df.iterrows():
      "  folium.Marker(location=[rows.loc['latitude'],"
                        "rows.loc['longitude']], popup=rows.loc['city'], tooltip=tooltip).add_to(m)
""""
    plugins.Fullscreen(position='topright').add_to(m)"

    m = m._repr_html_()
    context = {
        'm': m,
    }
    return render(request, 'dashboard/allmap.html', context)
```

APPENDIX C

EXPERIMENTAL PROCESS IMPLEMENTATION (DEFECTS INCIDENT DETECTION)

```
import matplotlib.pyplot as plt
from sklearn.model_selection import cross_val_score
import pandas as pd
import time
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
import pickle
import numpy as np
import lightgbm as lgb
import numpy as np
import sklearn.datasets
import sklearn.metrics
from sklearn.model_selection import train_test_split

import optuna
from optuna.visualization import plot_optimization_history
from optuna.visualization import plot_slice

SEED = 42

np.random.seed(SEED)
""
from google.colab import drive

drive.mount('/content/drive')
import sys
sys.path.insert(0,'/content/drive/My Drive/mycodelabs/atm')

data = pd.read_excel('/content/drive/My Drive/mycodelabs/atm/ncr_wb.xls
x', index_col='Index')
data.PCD.astype('category').cat.codes
""

data['City'] = data.City.astype('category').cat.codes
data['AFD'] = data.AFD.astype('category').cat.codes
data['CCD'] = data.CCD.astype('category').cat.codes
data['RCD'] = data.RCD.astype('category').cat.codes
```

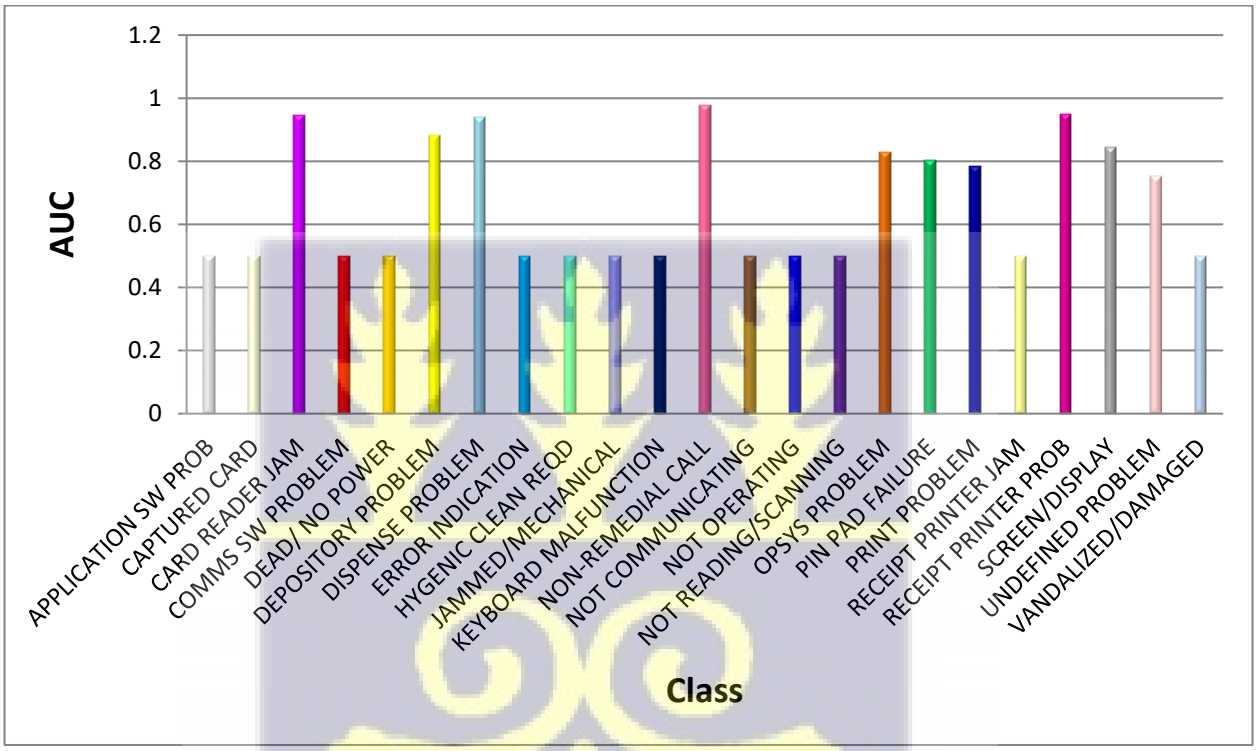
```
data['PCD_Names'] = data.PCD
class_names=data.sort_values(by=['PCD']).PCD_Names.unique()
data['PCD'] = data.PCD.astype('category').cat.codes
print(data.head())
#X = data[['CCD','RCD','AFD']].values
X = data[['AFD','CCD']].values
y = data['PCD'].values
y_names = data['PCD']


name = y_names.unique()
name.sort()
print(name)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3
, random_state=167)
“

def objective(trial):
    kernel = trial.suggest_categorical("kernel", ['rbf'])
    if kernel == 'poly':
        degree = trial.suggest_int("degree", 1, 3)
        clf = svm.SVC(verbose=True, gamma='scale',decision_function_sha
pe='ovo', kernel=kernel, degree=degree)”
    else:
        C = trial.suggest_float("C", 1.0, 25,step=0.05)
        #tol = trial.suggest_float("tol", 1e-5, 1e-3, log=True)
        clf = svm.SVC(verbose=True, gamma='scale',decision_function_sha
pe='ovo', kernel=kernel,C=C)
    clf.fit(X_train, y_train)
    #y_pred = clf.predict(X_test)
    return clf.score(X_test, y_test)

def objective_random_forest(trial):
    criterion = trial.suggest_categorical("criterion", ['gini','entropy
'])
   “ max_samples = trial.suggest_int("max_samples", 10,X_train.shape[0]
”)
    #kernel = trial.suggest_int("max_samples", 10,X_train.shape[0])
    clf = RandomForestClassifier(max_samples=max_samples, criterion=cri
terion, random_state=167)

    clf.fit(X_train, y_train)
    #y_pred = clf.predict(X_test)
    return clf.score(X_test, y_test)
```

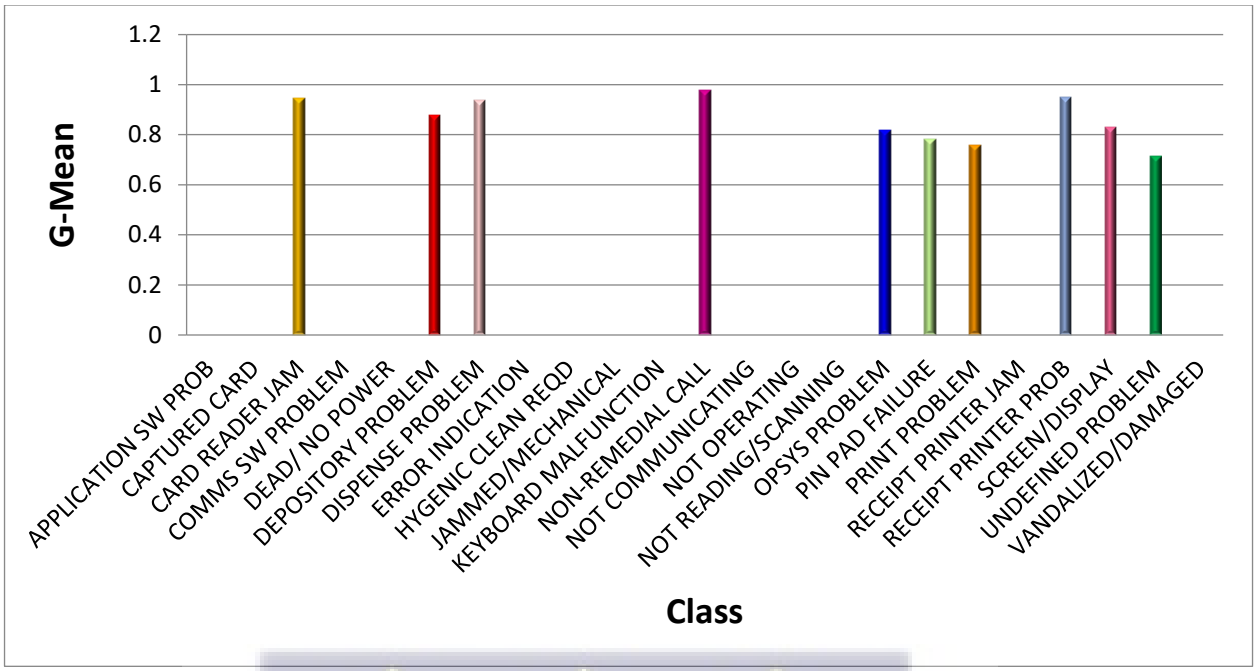
```
study = optuna.create_study(direction="maximize")
study.optimize(objective, n_trials=20)
print(study.best_value)
study.best_params


“X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
3, random_state=167)
“
start_time = time.time()
“clf=RandomForestClassifier(n_estimators=100)”
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
clf.score(X_test, y_test)
end_time = time.time()

computational_time = end_time - start_time

print('Computational Time: ', computational_time)

with open('saved_svm_model', 'wb' ) as f:
    pickle.dump(clf, f)
```

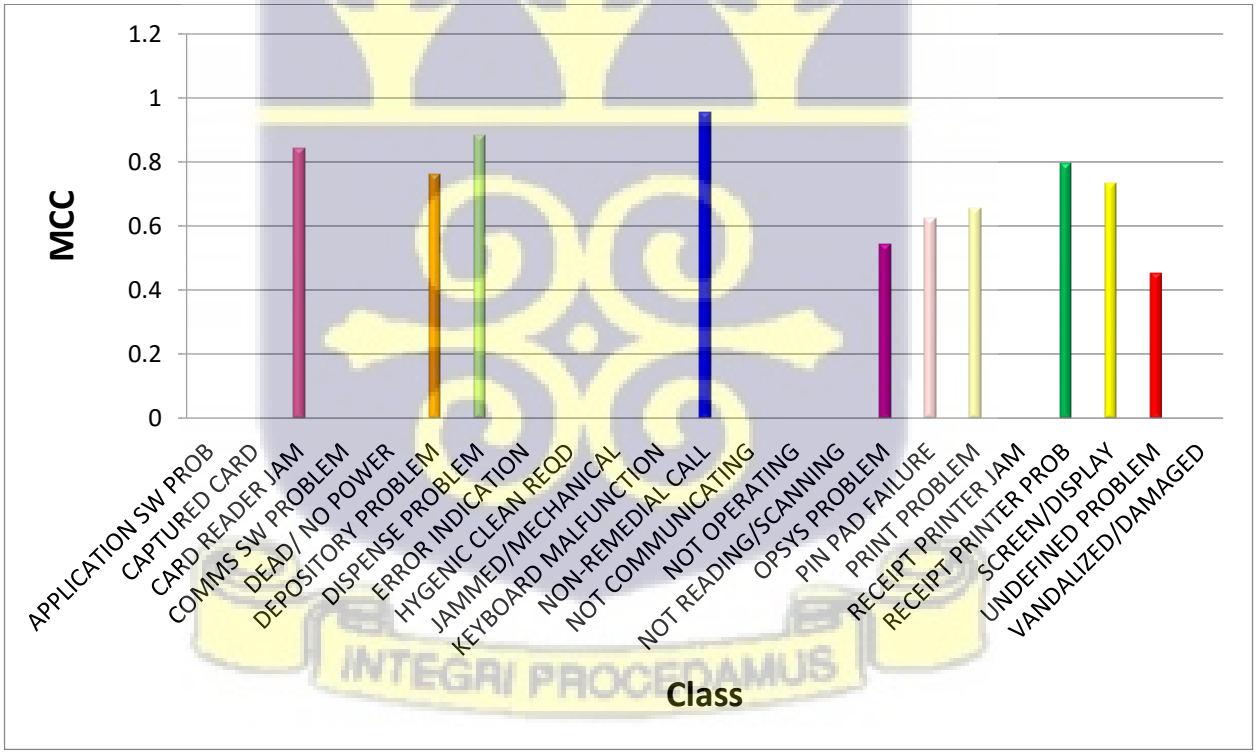
APPENDICES

APPENDIX B

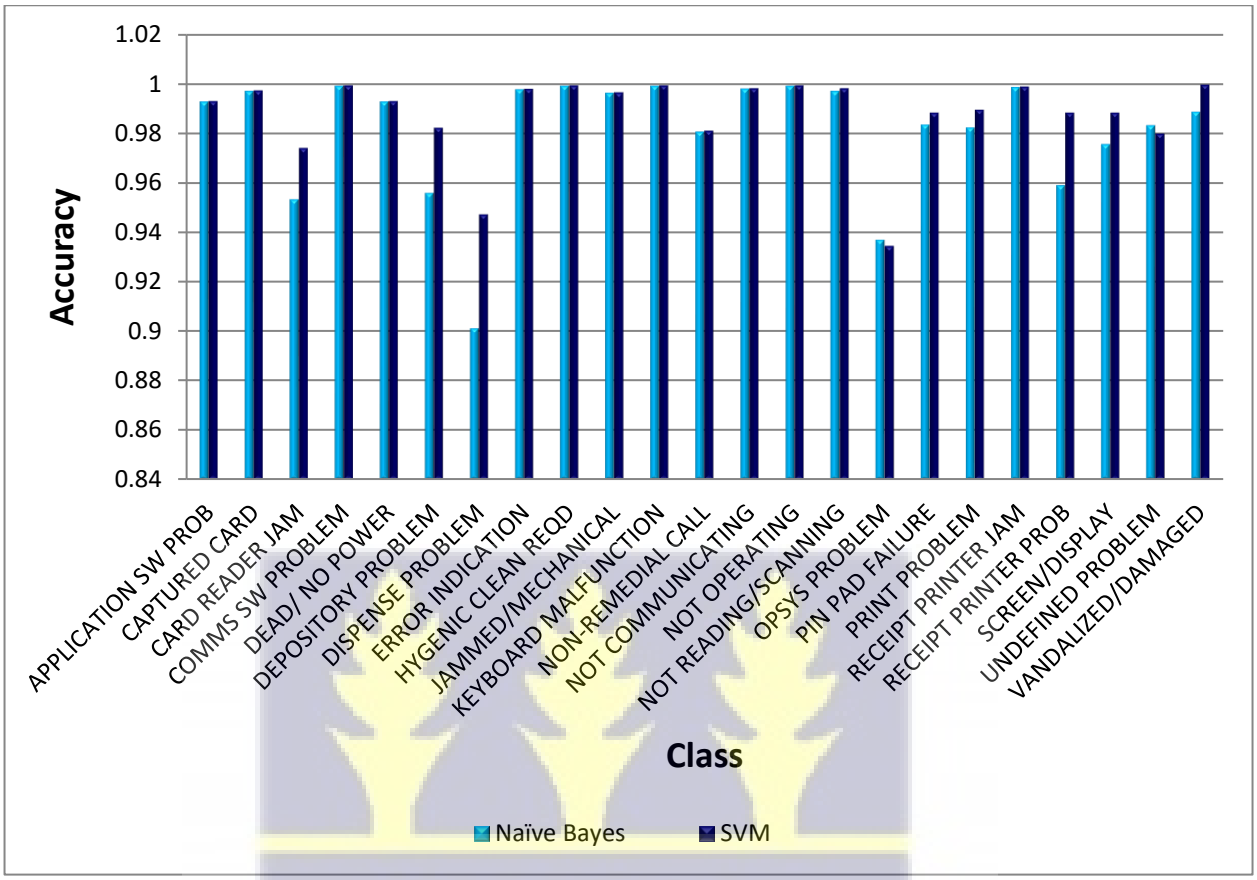
I. PLOT OF PERFORMANCE METRICS FOR DEFECT INCIDENT DETECTION USING SVM CLASSIFIER

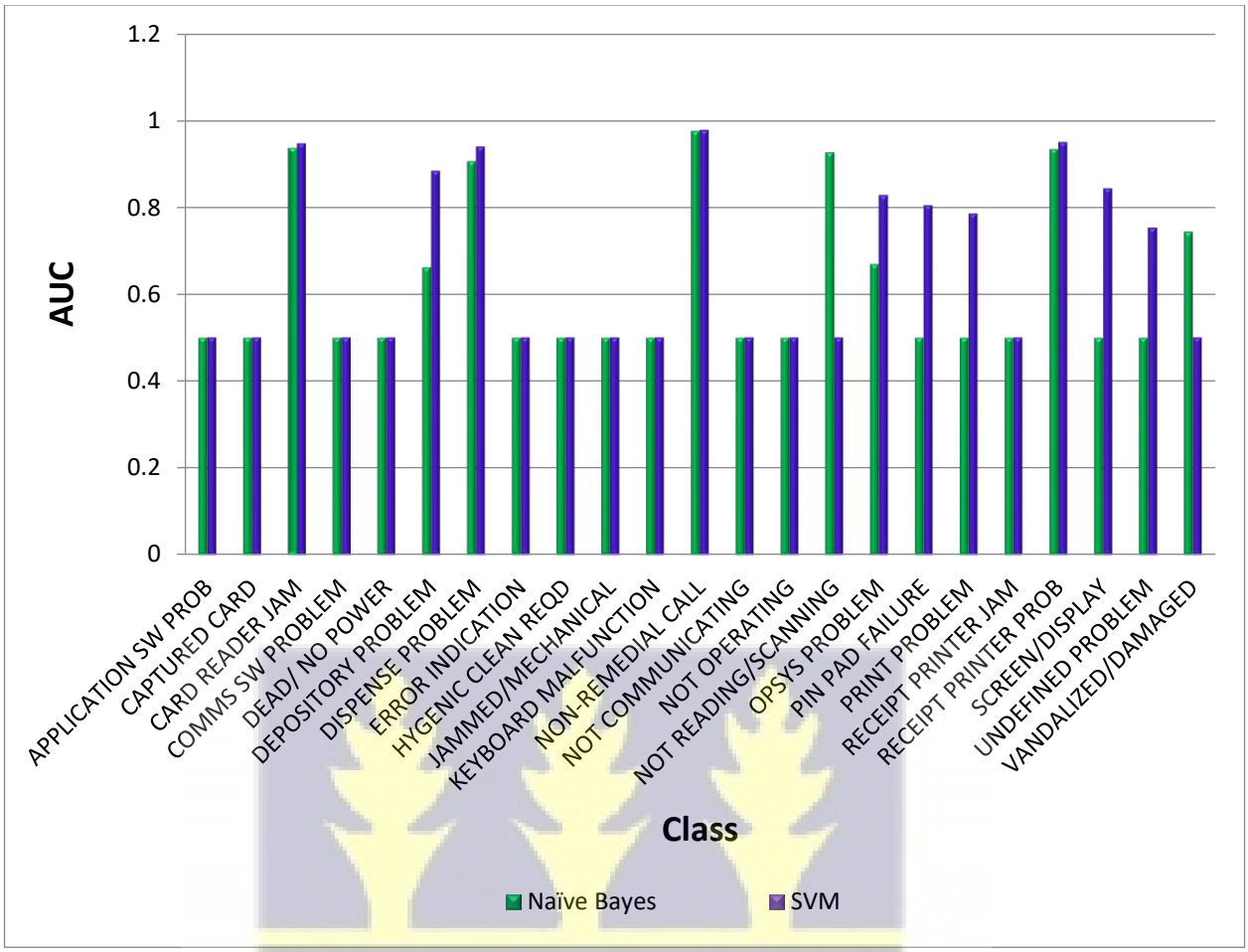


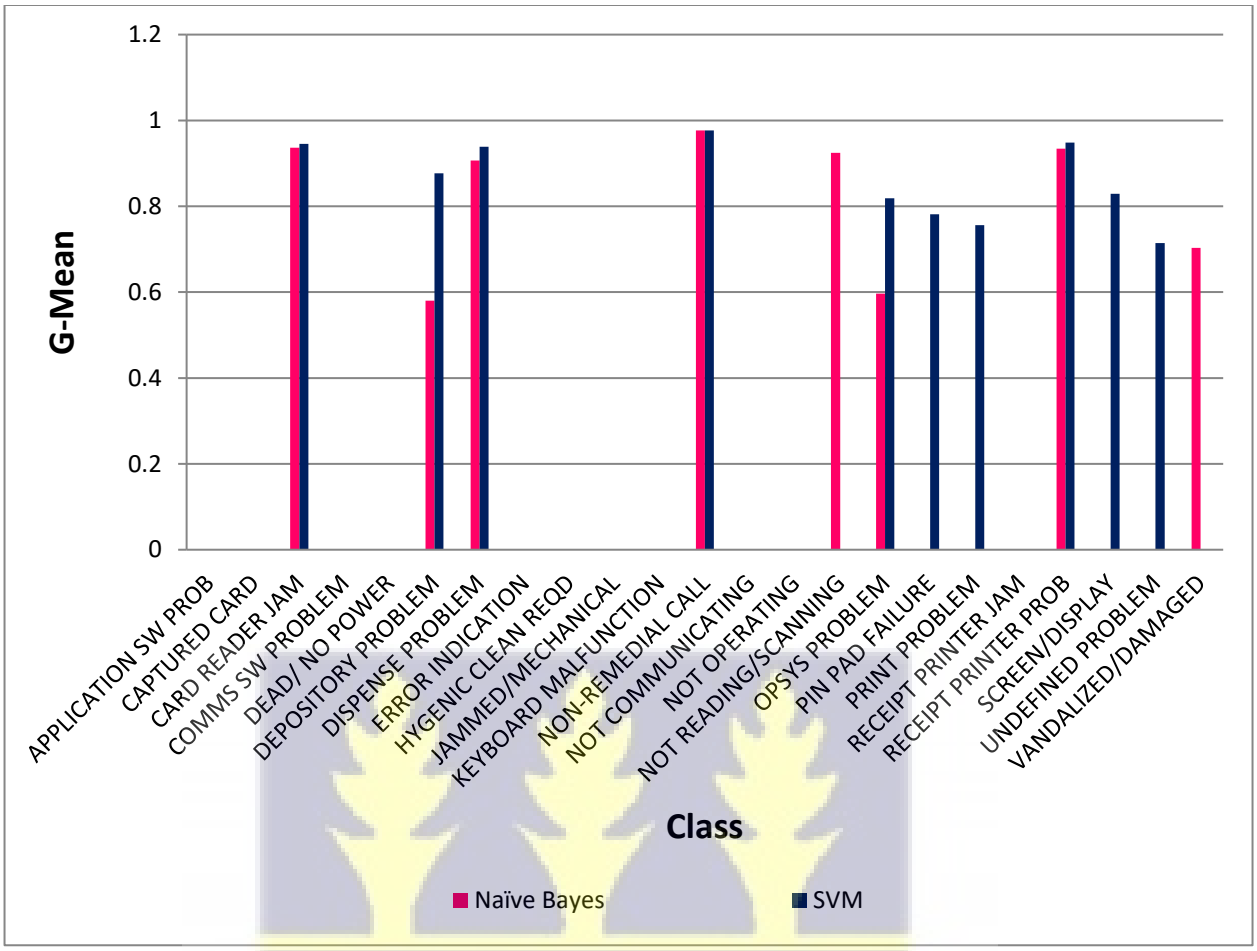


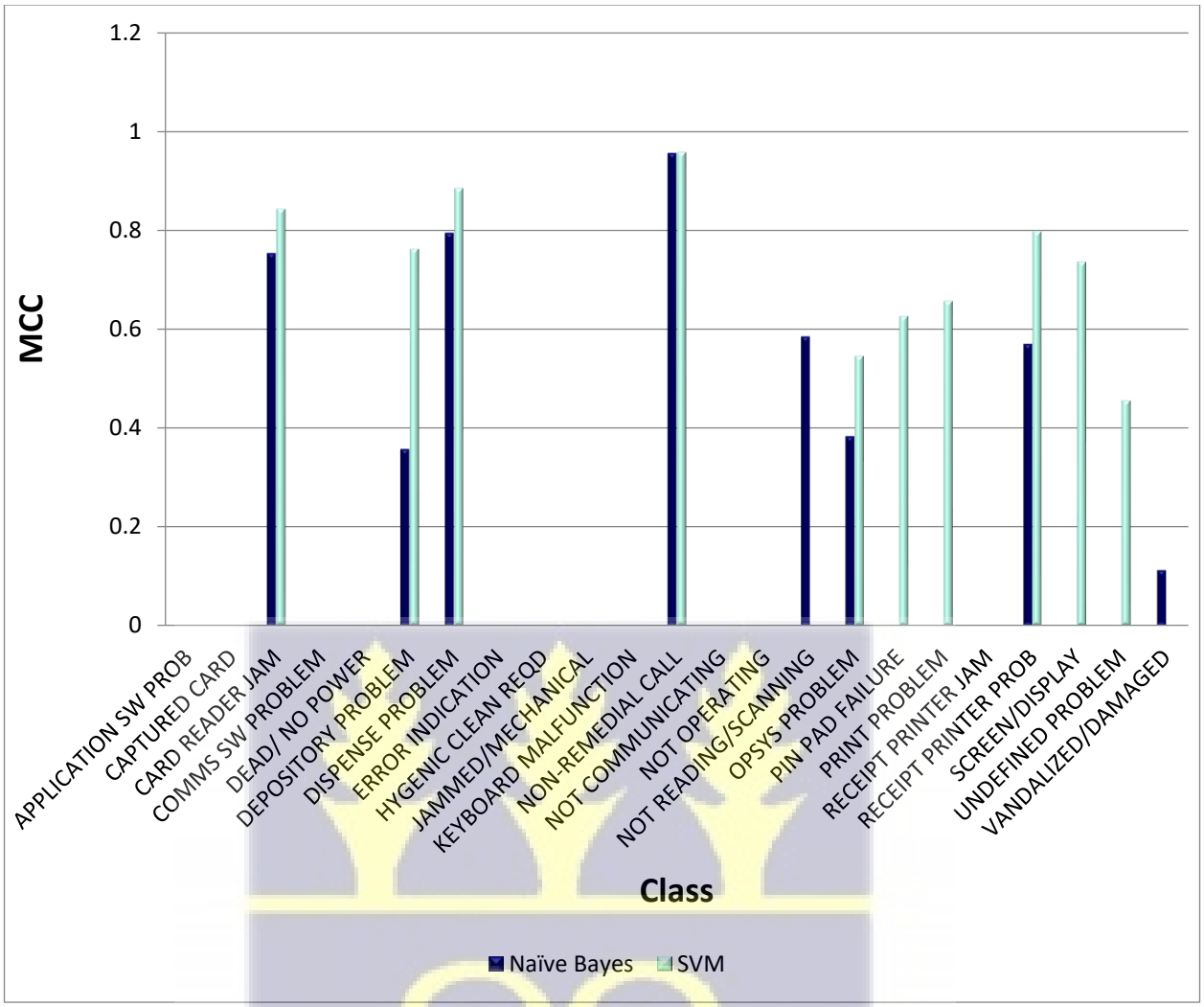


II. Results Comparative Analysis of SVM and Gaussian Naïve Bayes

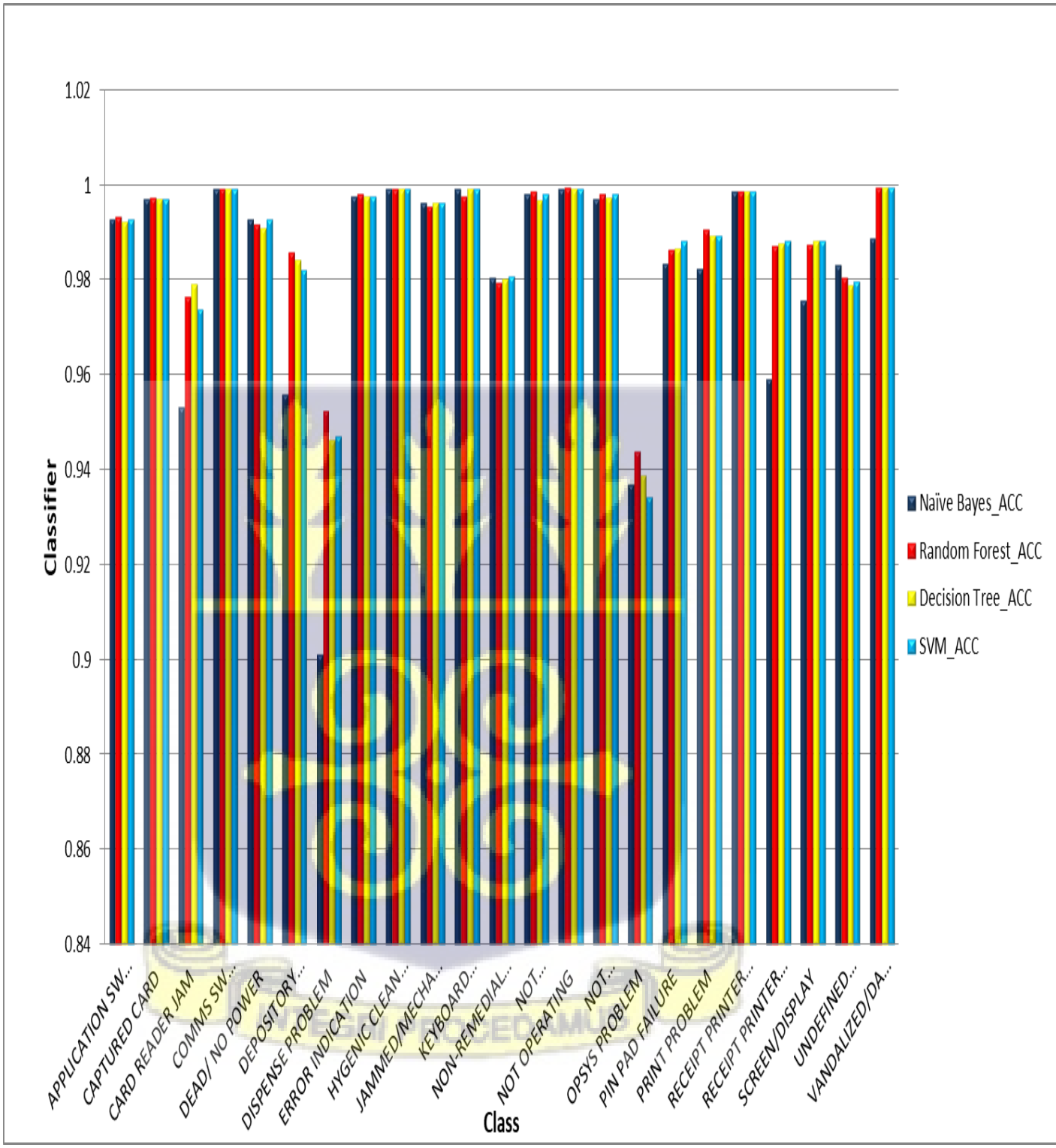


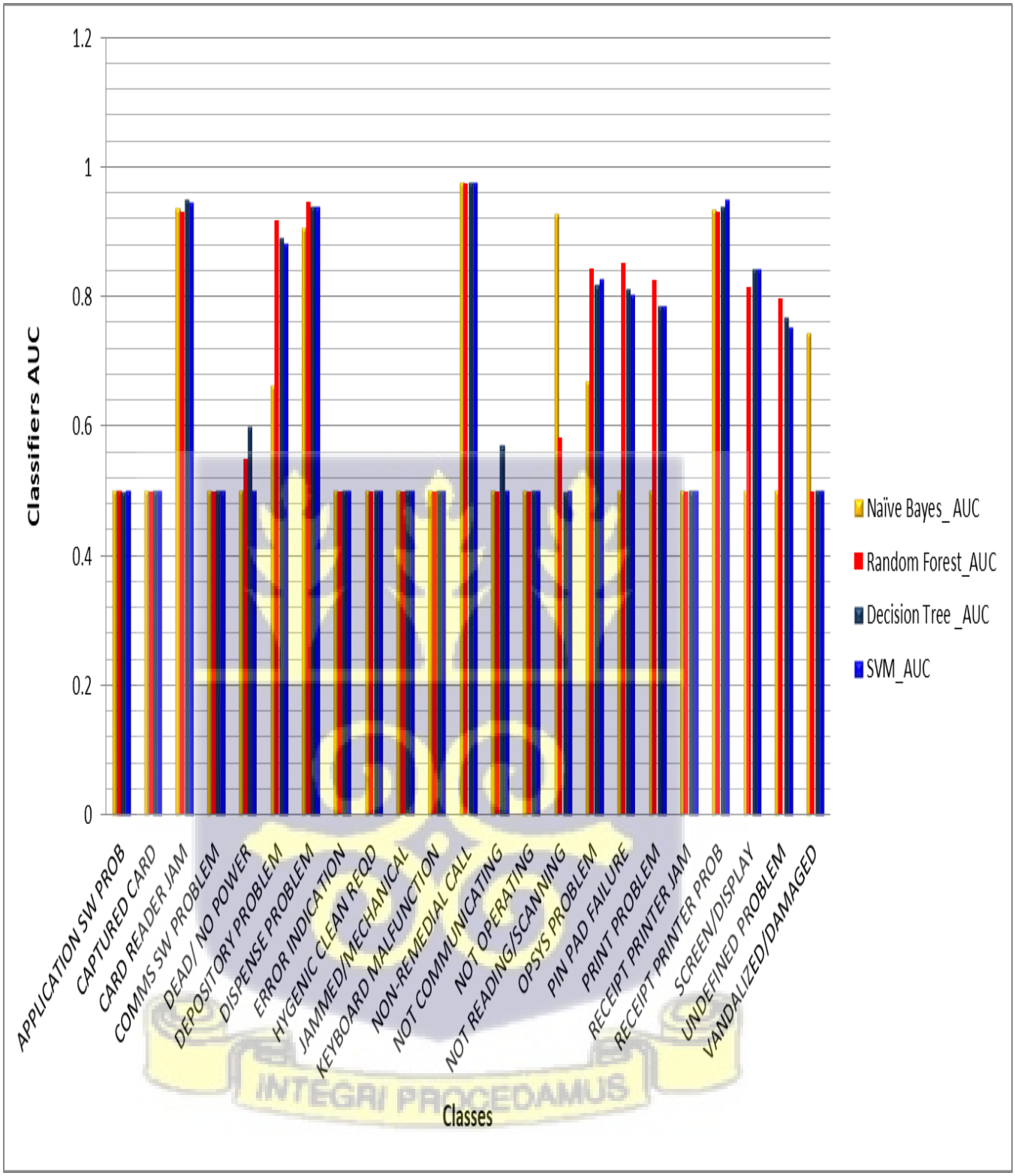




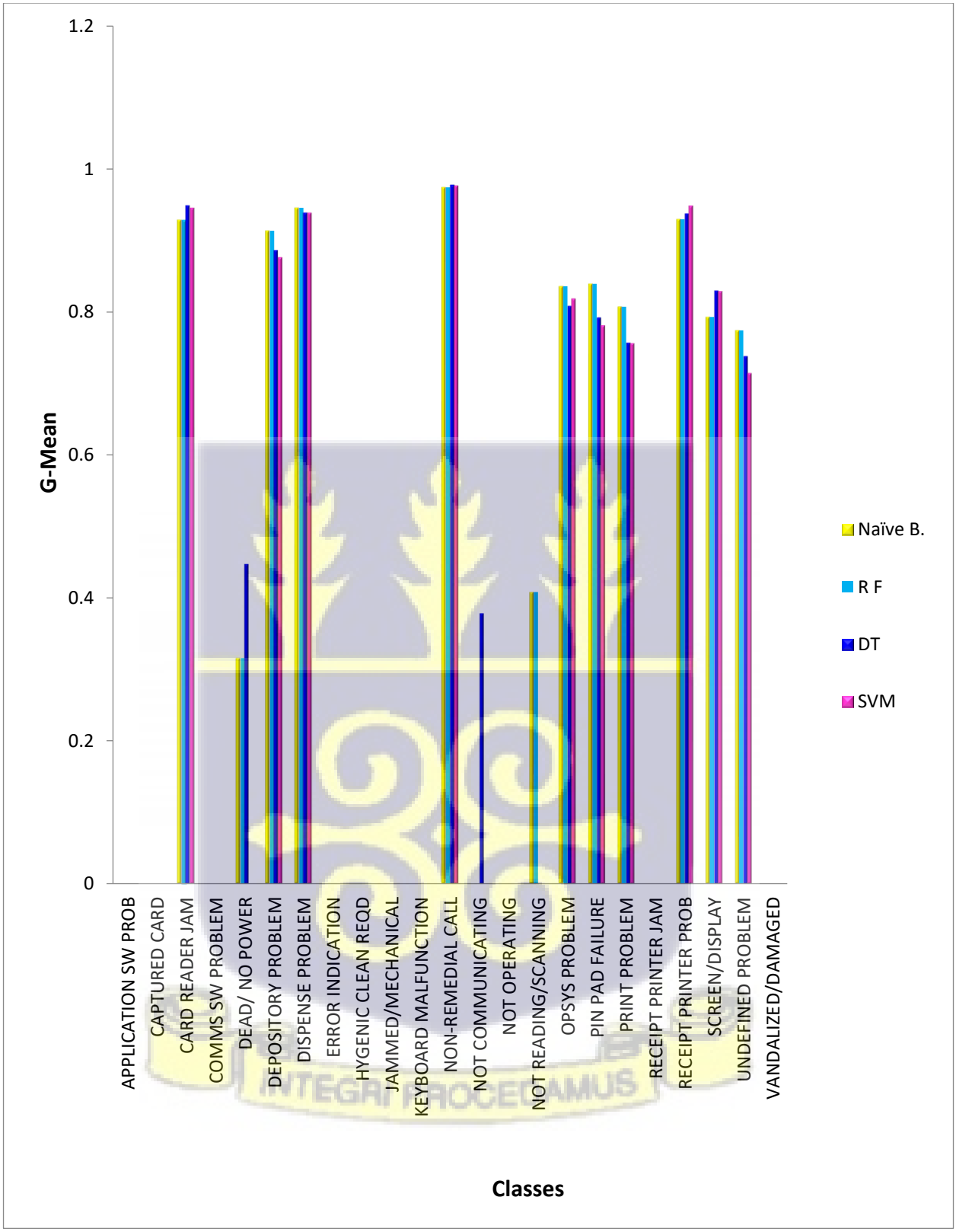


III. Comparative Analysis of Accuracies of the Classifiers

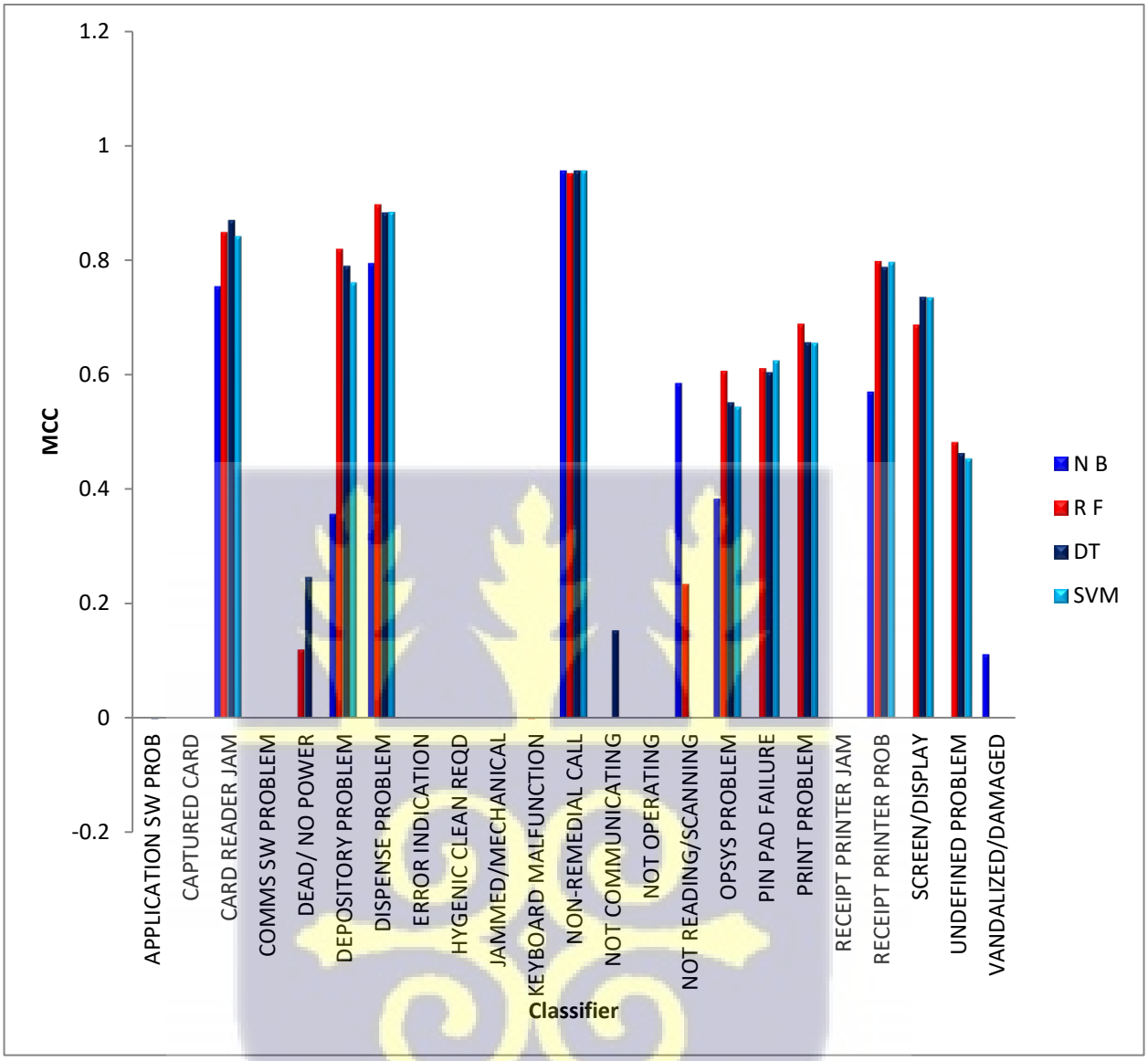




Comparative Analysis of AUCs of the classifiers for the classes



G-Mean of the classifiers



APPENDIX C

I. PERFORMANCE METRIC FOR ATM SECURITY INCIDENTS IMAGE DATASET USING R-CNN (SSDLITE_MOBILENET_V2)

Epoch	Training Time(seconds)	Training Accuracy(%)	Cross Entropy	Validation Accuracy	Validation Time(seconds)
0	3	58	2.16314	22	4
10	1	61	1.688085	52	1
20	5	72	1.289984	45	1
30	6	68	1.247007	49	1
40	7	66	1.163026	57	1
50	6	74	1.123522	40	1
60	7	79	0.928927	49	1
70	6	74	0.929696	52	1
80	6	73	0.846241	44	1
90	6	76	0.846241	57	1
100	6	72	0.777785	55	1
110	7	82	0.777785	51	1
120	7	82	0.743248	51	1
130	7	84	0.732659	46	1
140	6	84	0.646645	53	1
150	7	87	0.598614	41	1
160	6	82	0.555392	51	1
170	6	79	0.692751	52	1
180	6	82	0.627964	48	1
190	7	87	0.597346	42	1
200	7	87	0.621004	49	1
210	6	90	0.48781	55	1
220	6	89	0.47804	38	1
230	7	93	0.472113	51	1
240	6	92	0.523488	54	1
250	6	95	0.415194	57	1
260	7	90	0.475468	49	1
270	6	84	0.523661	61	1
280	6	91	0.526445	49	1
290	6	87	0.493927	52	1
300	7	89	0.450311	52	1
310	6	94	0.392039	57	1
320	6	97	0.359506	52	1
330	6	96	0.345114	41	1
340	6	92	0.447625	61	1

350	7	93	0.412192	55	1
360	6	93	0.379221	52	1
370	6	94	0.353409	55	1
380	6	94	0.374487	57	1
390	6	93	0.370026	54	1
400	6	95	0.364758	57	1
410	7	90	0.406155	51	1
420	6	92	0.363539	56	1
430	6	93	0.371614	62	1
440	7	90	0.36471	61	1
450	6	94	0.347737	48	1
460	7	91	0.436913	57	1
470	6	95	0.35002	48	1
480	7	92	0.341001	47	1
490	6	93	0.32336	50	1
499	6	96	0.319667	50	1

II. OTHER PERFORMANCE METRIC FOR ATM SECURITY INCIDENTS IMAGE DATASET USING R-CNN (SSDLITE_MOBILENET_V2)

Mini Batch-Size-ALEXNET	Training Time(seconds)	Training Accuracy(%)	validation Time(seconds)
100	53	76	13
200	59	87	3
300	63	89	10
400	63	95	10
500	64	96	10

III. PERFORMANCE METRIC FOR ATM TAMPERING IMAGE DATASET USING R-CNN (ALEXNET)

Epoch	Training Time(seconds)	Training Accuracy(%)	Cross Entropy	Validation Accuracy	validation Time(seconds)
0	3	40	2.16314	15	5
10	1	45	1.688085	40	1.5
20	5	55	1.289984	30	1.5
30	6	58	1.247007	39	1.5
40	7	56	1.163026	47	1.5
50	6	64	1.123522	30	1.5
60	7	69	0.928927	39	1.5
70	6	64	0.929696	42	1.5
80	6	70	0.846241	43	1.5
90	6	71	0.846241	47	1.5
100	6	70	0.777785	45	1.5
110	7	62	0.777785	41	1.5
120	7	62	0.743248	41	1.5
130	7	64	0.732659	36	1.5
140	6	64	0.646645	43	1.5
150	7	67	0.598614	31	1.5
160	6	62	0.555392	41	1.5
170	6	69	0.692751	42	1.5
180	6	62	0.627964	38	1.5
190	7	67	0.597346	32	1.5
200	7	67	0.621004	39	1.5
210	6	70	0.48781	45	1.5
220	6	79	0.47804	28	1.5
230	7	73	0.472113	41	1.5
240	6	72	0.523488	44	1.5
250	6	79	0.415194	47	1.5
260	7	70	0.475468	39	1.5
270	6	74	0.523661	51	1.5
280	6	71	0.526445	39	1.5
290	6	72	0.493927	42	1.5
300	7	71	0.450311	42	1.5
310	6	74	0.392039	47	1.5
320	6	81	0.359506	42	1.5
330	6	80	0.345114	31	1.5
340	6	72	0.447625	51	1.5
350	7	73	0.412192	45	1.5
360	6	73	0.379221	42	1.5
370	6	72	0.353409	45	1.5
380	6	73	0.374487	47	1.5

390	6	72	0.370026	44	1.5
400	6	73	0.364758	47	1.5
410	7	73	0.406155	41	1.5
420	6	72	0.363539	36	1.5
430	6	75	0.371614	42	1.5
440	7	68	0.36471	41	1.5
450	6	74	0.347737	38	1.5
460	7	73	0.436913	47	1.5
470	6	74	0.35002	38	1.5
480	7	76	0.341001	37	1.5
490	6	77	0.32336	40	1.5
499	6	80	0.319667	35	1.5

IV. OTHER PERFORMANCE METRIC FOR ATM TAMPERING IMAGE DATASET USING R-CNN (ALEXNET)

Mini Batch-Size-ALEXNET	Training Time(seconds)	Training Accuracy(%)	Validation Time(seconds)
100	53	70	15
200	71	67	15
300	63	71	15
400	62	73	15
500	70	80	15

APPENDIX E

V. PERFORMANCE METRIC FOR 2018 ATM SYSTEM DEFECT DATASETS USING SVM

Class	*Data size*	*Average accuracy rate (%)*	*Sensitivity* **(%)**	*Specificity* **(%)**
APPLICATION SW PROB	*84*	*0.99272198*	*0*	*1*
CAPTURED CARD	*33*	*0.997088792*	*0*	*1*
CARD READER JAM	*957*	*0.973799127*	*0.913194444*	*0.979345408*
COMMS SW PROBLEM	*11*	*0.999126638*	*0*	*1*
DEAD/ NO POWER	*82*	*0.99272198*	*0*	*1*
DEPOSITORY PROBLEM	*447*	*0.981950509*	*0.776119403*	*0.990305968*
DISPENSE PROBLEM	*4060*	*0.947016012*	*0.912295082*	*0.966139955*
ERROR INDICATION	*26*	*0.997671033*	*0*	*1*
HYGENIC CLEAN REQD	*10*	*0.999126638*	*0*	*1*
JAMMED/MECHANICAL	*45*	*0.996215429*	*0*	*1*
KEYBOARD MALFUNCTION	*10*	*0.999126638*	*0*	*1*
NON-REMEDIAL CALL	*3742*	*0.980786026*	*0.966162066*	*0.987889273*
NOT COMMUNICATING	*24*	*0.997962154*	*0*	*1*
NOT OPERATING	*11*	*0.999126638*	*0*	*1*
NOT READING/SCANNING	*21*	*0.997962154*	*0*	*1*
OPSYS PROBLEM	*692*	*0.934206696*	*0.706730769*	*0.948868919*
PIN PAD FAILURE	*191*	*0.988064047*	*0.614035088*	*0.99437537*
PRINT PROBLEM	*203*	*0.98922853*	*0.573770492*	*0.996739775*
RECEIPT PRINTER JAM	*16*	*0.998544396*	*0*	*1*
RECEIPT PRINTER PROB	*291*	*0.988064047*	*0.909090909*	*0.990140424*
SCREEN/DISPLAY	*281*	*0.988064047*	*0.69047619*	*0.995523724*

UNDEFINED PROBLEM	*16*	*0.979621543*	*0.517241379*	*0.987562926*
VANDALIZED/DAMAGED	*6*	*0.999417758*	*0*	*1*

II. OTHER PERFORMANCE METRICS FOR 2018 ATM SYSTEM DEFECT DATASETS USING SVM

Class	ACC	AUC	G-Mean	MCC
APPLICATION SW PROB	0.992722	0.5	0	0
CAPTURED CARD	0.997089	0.5	0	0
CARD READER JAM	0.973799	0.94627	0.945692	0.841714
COMMS SW PROBLEM	0.999127	0.5	0	0
DEAD/ NO POWER	0.992722	0.5	0	0
DEPOSITORY PROBLEM	0.981951	0.883213	0.876696	0.761
DISPENSE PROBLEM	0.947016	0.939218	0.938832	0.883826
ERROR INDICATION	0.997671	0.5	0	0
HYGENIC CLEAN REQD	0.999127	0.5	0	0
JAMMED/MECHANICAL	0.996215	0.5	0	0
KEYBOARD MALFUNCTION	0.999127	0.5	0	0
NON-REMEDIAL CALL	0.980786	0.977026	0.976965	0.956262
NOT COMMUNICATING	0.997962	0.5	0	0
NOT OPERATING	0.999127	0.5	0	0
NOT READING/SCANNING	0.997962	0.5	0	0
OPSYS PROBLEM	0.934207	0.8278	0.818899	0.544135
PIN PAD FAILURE	0.988064	0.804205	0.781397	0.624805
PRINT PROBLEM	0.989229	0.785255	0.756241	0.65552
RECEIPT PRINTER JAM	0.998544	0.5	0	0
RECEIPT PRINTER PROB	0.988064	0.949616	0.948751	0.796529
SCREEN/DISPLAY	0.988064	0.843	0.829087	0.734667

UNDEFINED PROBLEM	0.979622	0.752402	0.714709	0.454018
VANDALIZED/DAMAGED	0.999418	0.5	0	0

OTHER PERFORMANCE METRICS FOR 2018NCR ATM SYSTEM DEFECT DATASETS USING RANDOM FOREST

Class	TPR	TNR	FPR	FNR	ACC	AUC	G-Mean	MCC
APPLICATION SW PROB	0	1	0	1	0.993304221	0.5	0	0
CAPTURED CARD	0	1	0	1	0.997379913	0.5	0	0
CARD READER JAM	0.875	0.985701	0.014299	0.125	0.976419214	0.93035	0.928702	0.848774
COMMS SW PROBLEM	0	1	0	1	0.999126638	0.5	0	0
DEAD/ NO POWER	0.1	0.996779	0.003221	0.9	0.991557496	0.548389	0.315718	0.119917
DEPOSITORY PROBLEM	0.841727	0.991808	0.008192	0.158273	0.98573508	0.916767	0.913691	0.819558
DISPENSE PROBLEM	0.921155	0.970752	0.029248	0.078845	0.952256186	0.945954	0.945629	0.897556
ERROR INDICATION	0	1	0	1	0.997962154	0.5	0	0
HYGENIC CLEAN REQD	0	1	0	1	0.999126638	0.5	0	0
JAMMED/MECHANICAL	0	1	0	1	0.995342067	0.5	0	0
KEYBOARD MALFUNCTION	0	0.998252	0.001748	1	0.997671033	0.499126	0	-0.00101
NON-REMEDIAL CALL	0.962371	0.986931	0.013069	0.037629	0.979330422	0.974651	0.974573	0.951542

NOT COMMUNICATING	0	1	0	1	0.998544396	0.5	0	0
NOT OPERATING	0	1	0	1	0.999417758	0.5	0	0
NOT READING/SCANNING	0.166667	0.999417	0.000583	0.833333	0.997962154	0.583042	0.408129	0.234775
OPSYS PROBLEM	0.728889	0.958879	0.041121	0.271111	0.943813683	0.843884	0.836012	0.606379
PIN PAD FAILURE	0.711538	0.990541	0.009459	0.288462	0.986317322	0.85104	0.839528	0.611023
PRINT PROBLEM	0.654545	0.996154	0.003846	0.345455	0.990684134	0.82535	0.807482	0.688777
RECEIPT PRINTER JAM	0	1	0	1	0.998544396	0.5	0	0
RECEIPT PRINTER PROB	0.872549	0.990699	0.009301	0.127451	0.987190684	0.931624	0.929749	0.798033
SCREEN/DISPLAY	0.631579	0.995534	0.004466	0.368421	0.987481805	0.813557	0.792943	0.687454
UNDEFINED PROBLEM	0.607843	0.986111	0.013889	0.392157	0.980494905	0.796977	0.77421	0.482203

OTHER PERFORMANCE METRICS FOR 2018NCR ATM SYSTEM DEFECT DATASETS USING

DECISION TREE

Class	TPR	TNR	FPR	FNR	ACC	AUC	G-Mean	MCC
APPLICATION SW PROB	0	0.99941349	0.00058651	1	0.992139738	0.499706745	0	-0.00207
CAPTURED CARD	0	1	0	1	0.997088792	0.5	0	0
CARD READER JAM	0.913194444	0.985065141	0.014934859	0.086806	0.979039301	0.949129793	0.948449	0.868842
COMMS SW PROBLEM	0	1	0	1	0.999126638	0.5	0	0
DEAD/ NO POWER	0.2	0.996774194	0.003225806	0.8	0.990975255	0.598387097	0.446492	0.245644
DEPOSITORY PROBLEM	0.791044776	0.992123599	0.007876401	0.208955	0.984279476	0.891584188	0.885897	0.788 84
DISPENSE PROBLEM	0.913114754	0.964334086	0.035665914	0.086885	0.946142649	0.938724442	0.938375	0.881963
ERROR INDICATION	0	1	0	1	0.997671033	0.5	0	0
HYGENIC CLEAN REQD	0	1	0	1	0.999126638	0.5	0	0
JAMMED/MECHANICAL	0	1	0	1	0.996215429	0.5	0	0
KEYBOARD MALFUNCTION	0	1	0	1	0.999126638	0.5	0	0
NON-REMEDIAL CALL	0.967943301	0.986159917	0.013840083	0.032057	0.980203785	0.977059109	0.977009	0.954 98
NOT COMMUNICATING	0.142857143	0.998541424	0.001458576	0.857143	0.996797671	0.570699283	0.377689	0.152706
NOT OPERATING	0	1	0	1	0.999126638	0.5	0	0
NOT READING/SCANNING	0	0.999416569	0.000583431	1	0.997379913	0.499708285	0	-0.00109
OPSYS PROBLEM	0.682692308	0.955376511	0.044623489	0.317308	0.938864629	0.819034409	0.807606	0.550845
PIN PAD FAILURE	0.631578947	0.992599171	0.007400829	0.368421	0.986608443	0.812089059	0.791773	0.603724
PRINT PROBLEM	0.573770492	0.996739775	0.003260225	0.42623	0.989228853	0.785255133	0.756241	0.655 52

RECEIPT PRINTER JAM	0	1	0	1	0.99854 4396	0.5	0	0
RECEIPT PRINTER PROB	0.88636 3636	0.99043 9199	0.00956 0801	0.113 636	0.98777 2926	0.93840 1418	0.936 957	0.786 826
SCREEN/DISPLAY	0.69047 619	0.99552 3724	0.00447 6276	0.309 524	0.98806 4047	0.84299 9957	0.829 087	0.734 667
UNDEFINED PROBLEM	0.55172 4138	0.98608 2322	0.01391 7678	0.448 276	0.97874 818	0.76890 323	0.737 594	0.462 254