DETECTION OF WATER-BED CYBER ATTACKS USING ARTIFICIAL NEURAL NETWORKS

A PROJECT REPORT

Submitted by

RANJITH KUMAR (211420205122)

RONAL REGAN.S (211420205128)

SUNIL KUMAR.V (211420205157)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

PANIMALAR ENGINEERING COLLEGE, POONAMALLEE

ANNA UNIVERSITY : CHENNAI 600 025

MARCH 2024

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **DETECTION OF WATER-BED CYBER ATTACKS USING ARTIFICIAL NEURAL NETWORKS** is the bonafide work of **"RANJITH KUMAR.R (211420205122), RONAL REGAN.S (211420205128) and SUNIL KUMAR.V (211420205157)"** who carried out the project under my supervision.

SIGNATURE

Dr. M. HELDA MERCY M.E., Ph.D.,
HEAD OF THE DEPARTMENT

Department of Information Technology

Panimalar Engineering College

Poonamallee, Chennai - 600 123

SIGNATURE

Mr. A. BALAJI M.E.,
SUPERVISOR
ASSISTANT PROFESSOR

Department of Information Technology

Panimalar Engineering College

Poonamallee, Chennai - 600 123

Submitted for the project and viva-voce examination held on _____

SIGNATURE

INTERNAL EXAMINER

SIGNATURE

EXTERNAL EXAMINER

DECLARATION

I hereby declare that the project report entitled "**DETECTION OF WATER-BED CYBER ATTACKS USING ARTIFICIAL NEURAL NETWORKS** " which is being submitted in partial fulfilment of the requirement of the course leading to the award of the 'Bachelor Of Technology in Information Technology ' in **Panimalar Engineering College, Autonomous institution Affiliated to Anna university- Chennai** is the result of the project carried out by me under the guidance of **Mr. A. Balaji, Assistant Professor in the Department of Information Technology**. I further declared that I or any other person has not previously submitted this project report to any other institution/university for any other degree/ diploma or any other person.

(RANJITH KUMAR.R)

Date**:** (RONAL REGAN.S)

Place**:** Chennai (SUNIL KUMAR.V)

It is certified that this project has been prepared and submitted under my guidance.

Date: **Mr. A. Balaji M.E.,**

Place: Chennai (ASSISTANT PROFESSOR / IT)

ACKNOWLEDGEMENT

A project of this magnitude and nature requires kind co-operation and support from many, for successful completion . We wish to express our sincere thanks to all those who were involved in the completion of this project.

Our sincere thanks to **Our Beloved Secretary and Correspondent, Dr. P. CHINNADURAI, M.A., Ph.D.,** for his sincere endeavor in educating us in his premier institution.

We would like to express our deep gratitude to **Our Dynamic Directors , Mrs. C. VIJAYA RAJESHWARI and Dr. C. SAKTHI KUMAR, M.E.,M.B.A.,Ph.D.,**and **Dr.saranya sree sakthikumar.,B.E.,M.B.A.,Ph.D.,** for providing us with the necessary facilities for completion of this project.

We also express our appreciation and gratefulness to **Our Principal Dr. K. MANI, M.E., Ph.D.,** who helped us in the completion of the project. We wish to convey our thanks and gratitude to our head of the department, **Dr. M. HELDA MERCY, M.E., Ph.D.,** Department of Information Technology, for her support and by providing us ample time to complete our project.

We express our indebtedness and gratitude to our Project co-ordinator **Mr. M. DILLI BABU**, **M.E.,(P.hD.,)** Associate Professor, Department of Information Technology for his guidance throughout the course of our project. We also express sincere thanks to our supervisor **Mr. A Balaji** M.E., Assistant Professor, for providing the support to carry out the project successfully. Last, we thank our parents and friends for providing their extensive moral support and encouragement during the course of the project.

ABSTRACT

Cyber threat intelligence gathered from previous attacks may provide light on the methods and tools employed by cybercriminals, which can aid in both the reconstruction of assaults and the prediction of their future trajectory. Consequently, to mitigate these consequences, cyber security analysts use threat intelligence, alert correlations, machine learning, and enhanced visualisations.The impact and development of machine learning algorithm are booming in the current scenario. These algorithm are working perfectly by identifying the pattern based on instance based or model based learning. Well the model based learning is more efficient then the instance based learning. Because the instance based learning identify the pattern by memorising its pattern where the model based training try to form a boundary between these and try to classify them. In addition to the utilization of machine learning algorithms, the integration of advanced analytics techniques such as anomaly detection and behavior analysis further fortifies cyber defense mechanisms. Anomaly detection algorithms sift through vast datasets to identify deviations from normal behavior, flagging potentially malicious activities that might evade traditional rule-based systems. Meanwhile, behavior analysis algorithms scrutinize user actions and system interactions in real-time, discerning subtle deviations indicative of suspicious or unauthorized behavior.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OFABBREVATIONS

ANN	Artificial Neural Network
NLP	Natural Language Processing
CF	Collaborative Filtering
CBCF	Cluster Based Collaborative Filtering
RSs	Recommender Systems
PCC	Pearson Correlation Coefficient
CBPCC	Cluster Based Pearson Correlation Coefficient
MAE	Mean Absolute Error
RMSE	Root Mean Squared Error
KNN	K nearest Neighbors
VSS	Vector Space Similarity
HDFS	Hadoop Distributed File System
PD	Personality Diagnosis
AM	Aspect Model
JSC	Jac card similarity coefficient
JSON	Java Script Object Notation
HTML	Hyper Text Markup Language
IDE	Integrated Development Environment

CHAPTER 1

1.INRODUCTION

1.1 OVERVIEW AND SCOPE

The primary aim of this project is to design, implement, and evaluate an effective cyber-attack detection system leveraging the power of Artificial Neural Networks (ANNs). The system will analyze network traffic patterns and behavior to accurately identify and classify potential cyber threats, enhancing the security posture of the targeted network.
The primary objectives of implementing an Artificial Neural Network (ANN)-based system for cyber-attack detection are to enhance proactive threat identification, improve the accuracy of anomaly detection in network traffic, and establish a resilient defense mechanism against evolving cyber threats.
This involves training the ANN to recognize patterns indicative of attacks, ensuring real-time monitoring capabilities, and creating an adaptive model that can evolve with the dynamic nature of cyber threats, ultimately fortifying cybersecurity measures for safeguarding sensitive digital assets.
The scope of "Detection of Cyber Attacks Using Artificial Neural Networks" involves developing and implementing an artificial neural network-based system to identify and thwart cyber threats. This encompasses the design, training, and evaluation of the neural network for effective detection, contributing to enhanced cybersecurity measures. The focus is on leveraging advanced machine learning techniques to augment existing security frameworks and mitigate the evolving challenges posed by cyber-attacks.

1.2 DOMAIN OVERVIEW

Data Science

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from structured and unstructured data, and apply knowledge and actionable insights from data across a broad range of application domains.

The term "data science" has been traced back to 1974, when Peter Naur proposed it as an alternative name for computer science. In 1996, the International Federation of Classification Societies became the first conference to specifically feature data science as a topic. However, the definition was still in flux.

The term “data science” was first coined in 2008 by D.J. Patil, and Jeff Hammerbacher, the pioneer leads of data and analytics efforts at LinkedIn and Facebook. In less than a decade, it has become one of the hottest and most trending professions in the market. Data science is the field of study that combines domain expertise, programming skills, and knowledge of mathematics and statistics to extract meaningful insights from data.

Data science can be defined as a blend of mathematics, business acumen, tools, algorithms and machine learning techniques, all of which help us in finding out the hidden insights or patterns from raw data which can be of major use in the formation of big business decisions.

Data Scientist

Data scientists examine which questions need answering and where to find the related data. They have business acumen and analytical skills as well as the ability to mine, clean, and present data. Businesses use data scientists to source, manage, and analyze large amounts of unstructured data.

Artificial Intelligence

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term

may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving.

Artificial intelligence (AI) is intelligence demonstrated by machines, as opposed to the natural intelligence displayed by humans or animals. Leading AI textbooks define the field as the study of "intelligent agents" any system that perceives its environment and takes actions that maximize its chance of achieving its goals. Some popular accounts use the term "artificial intelligence" to describe machines that mimic "cognitive" functions that humans associate with the human mind, such as "learning" and "problem solving", however this definition is rejected by major AI researchers.

Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. Specific applications of AI include expert systems, natural language processing, speech recognition and machine vision.

AI applications include advanced web search engines, recommendation systems (used by Youtube, Amazon and Netflix), Understanding human speech (such as Siri or Alexa), self-driving cars (e.g. Tesla), and competing at the highest level in strategic game systems (such as chess and Go), As machines become increasingly capable, tasks considered to require "intelligence" are often removed from the definition of AI, a phenomenon known as the AI effect. For instance, optical character recognition is frequently excluded from things considered to be AI, having become a routine technology.

Artificial intelligence was founded as an academic discipline in 1956, and in the years since has experienced several waves of optimism, followed by disappointment and the loss of funding (known as an "AI winter"), followed by new approaches, success and renewed funding. AI research has tried and discarded many different approaches during its lifetime, including simulating the brain, modeling human problem solving, formal logic, large databases of knowledge and imitating animal behavior. In the first decades of the 21st century, highly mathematical statistical machine learning has dominated the field, and this technique has proved highly

successful, helping to solve many challenging problems throughout industry and academia.

The various sub-fields of AI research are centered around particular goals and the use of particular tools. The traditional goals of AI research include reasoning, knowledge representation, planning, learning, natural language processing, perception and the ability to move and manipulate objects. General intelligence (the ability to solve an arbitrary problem) is among the field's long-term goals. To solve these problems, AI researchers use versions of search and mathematical optimization, formal logic, artificial neural networks, and methods based on statistics, probability and economics. AI also draws upon computer science, psychology, linguistics, philosophy, and many other fields.

The field was founded on the assumption that human intelligence "can be so precisely described that a machine can be made to simulate it". This raises philosophical arguments about the mind and the ethics of creating artificial beings endowed with human-like intelligence. These issues have been explored by myth, fiction and philosophy since antiquity. Science fiction and futurology have also suggested that, with its enormous potential and power, AI may become an existential risk to humanity.

As the hype around AI has accelerated, vendors have been scrambling to promote how their products and services use AI. Often what they refer to as AI is simply one component of AI, such as machine learning. AI requires a foundation of specialized hardware and software for writing and training machine learning algorithms. No one programming language is synonymous with AI, but a few, including Python, R and Java, are popular.

In general, AI systems work by ingesting large amounts of labeled training data, analyzing the data for correlations and patterns, and using these patterns to make predictions about future states. In this way, a chatbot that is fed examples of text chats can learn to produce life like exchanges with people, or an image recognition tool can learn to identify and describe objects in images by reviewing millions of examples.

AI programming focuses on three cognitive skills: learning, reasoning and self-correction.

Learning processes. This aspect of AI programming focuses on acquiring data and creating rules for how to turn the data into actionable information. The rules, which are called algorithms, provide computing devices with step-by-step instructions for how to complete a specific task.

Reasoning processes. This aspect of AI programming focuses on choosing the right algorithm to reach a desired outcome.

Self-correction processes. This aspect of AI programming is designed to continually fine-tune algorithms and ensure they provide the most accurate results possible.

AI is important because it can give enterprises insights into their operations that they may not have been aware of previously and because, in some cases, AI can perform tasks better than humans. Particularly when it comes to repetitive, detail-oriented tasks like analyzing large numbers of legal documents to ensure relevant fields are filled in properly, AI tools often complete jobs quickly and with relatively few errors.

Artificial neural networks and deep learning artificial intelligence technologies are quickly evolving, primarily because AI processes large amounts of data much faster and makes predictions more accurately than humanly possible.

NLP

Natural language processing (NLP) allows machines to read and understand human language. A sufficiently powerful natural language processing system would enable natural-language user interfaces and the acquisition of knowledge directly from human-written sources, such as newswire texts. Some straightforward applications of natural language processing include information retrieval, text mining, question answering and machine translation. Many current approaches use word co-occurrence frequencies to construct syntactic representations of text. "Keyword spotting" strategies for search are popular and scalable but dumb; a search query for "dog" might only match documents with the literal word "dog" and miss a document

with the word "poodle". "Lexical affinity" strategies use the occurrence of words such as "accident" to assess the sentiment of a document. Modern statistical NLP approaches can combine all these strategies as well as others, and often achieve acceptable accuracy at the page or paragraph level. Beyond semantic NLP, the ultimate goal of "narrative" NLP is to embody a full understanding of commonsense reasoning. By 2019, transformer-based deep learning architectures could generate coherent text.

Machine Learning

Machine Learning combines computer science, mathematics, and statistics. Statistics is essential for drawing inferences from the data. Mathematics is useful for developing machine learning models and finally, computer science is used for implementing algorithms.

However, simply building models is not enough. You must also optimize and tune the model appropriately so that it provides you with accurate results. Optimization techniques involve tuning the hyper parameters to reach an optimum result.

The world today is evolving and so are the needs and requirements of people. Furthermore, we are witnessing a fourth industrial revolution of data. In order to derive meaningful insights from this data and learn from the way in which people and the system interface with the data, we need computational algorithms that can churn the data and provide us with results that would benefit us in various ways. Machine Learning has revolutionized industries like medicine, healthcare, manufacturing, banking, and several other industries. Therefore, Machine Learning has become an essential part of modern industry.

Data is expanding exponentially and in order to harness the power of this data, added by the massive increase in computation power, Machine Learning has added another dimension to the way we perceive information. Machine Learning is being utilized

everywhere. The electronic devices you use, the applications that are part of your everyday life are powered by powerful machine learning algorithms.

With an exponential increase in data, there is a need for having a system that can handle this massive load of data. Machine Learning models like Deep Learning allow the vast majority of data to be handled with an accurate generation of predictions. Machine Learning has revolutionized the way we perceive information and the various insights we can gain out of it.

These machine learning algorithms use the patterns contained in the training data to perform classification and future predictions. Whenever any new input is introduced to the ML model, it applies its learned patterns over the new data to make future predictions. Based on the final accuracy, one can optimize their models using various standardized approaches. In this way, Machine Learning model learns to adapt to new examples and produce better results.

1.3 Types of Machine Learning

Machine Learning Algorithms can be classified into 3 types as follows –

1. Supervised Learning
2. Unsupervised Learning
3. Reinforcement Learning

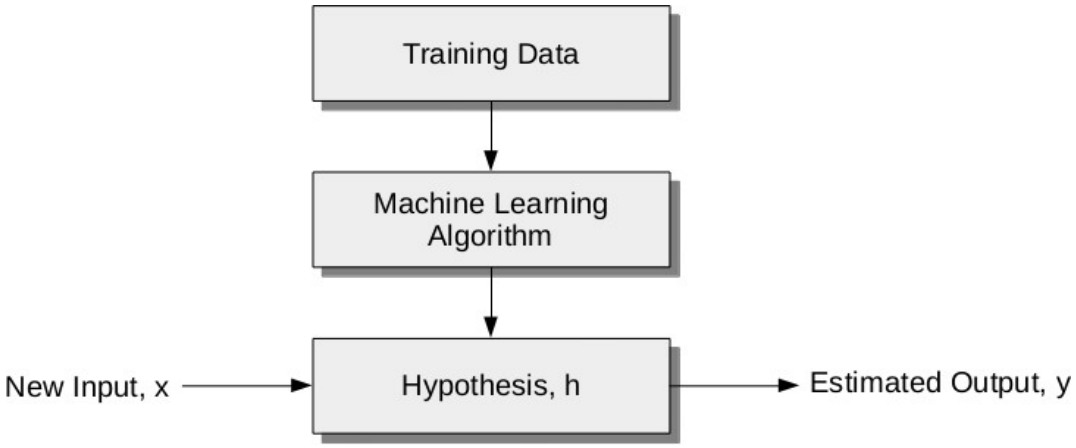

FIG 1.1 TRAINING DATA

Supervised Learning

In the majority of supervised learning applications, the ultimate goal is to develop a finely tuned predictor function h(x) (sometimes called the "hypothesis"). "Learning" consists of using sophisticated mathematical algorithms to optimize this function so that, given input data x about a certain domain (say, square footage of a house), it will accurately predict some interesting value h(x) (say, market price for said house).

$$h(x_1, x_2, x_3, x_4) = \theta_0 + \theta_1 x_1 + \theta_2 x_3^2 + \theta_3 x_3 x_4 + \theta_4 x_1^3 x_2^2 + \theta_5 x_2 x_3^4 x_4^2$$

This function takes input in four dimensions and has a variety of polynomial terms. Deriving a normal equation for this function is a significant challenge. Many modern machine learning problems take thousands or even millions of dimensions of data to build predictions using hundreds of coefficients. Predicting how an organism's genome will be expressed, or what the climate will be like in fifty years, are examples of such complex problems.

Under supervised ML, two major subcategories are:

- Regression machine learning systems: Systems where the value being predicted falls somewhere on a continuous spectrum. These systems help us with questionsof “How much?” or “How many?”.
- Classification machine learning systems: Systems where we seek a yes-or-no prediction, such as “Is this tumer cancerous?”, “Does this cookie meet our quality standards?”, and so on.

In practice, x almost always represents multiple data points. So, for example, a housing price predictor might take not only square-footage (x1) but also number of bedrooms (x2), number of bathrooms (x3), number of floors (x4), year built (x5), zip code (x6), and so forth. Determining which inputs to use is an important part of ML design. However, for the sake of explanation, it is easiest to assume a single input value is used.

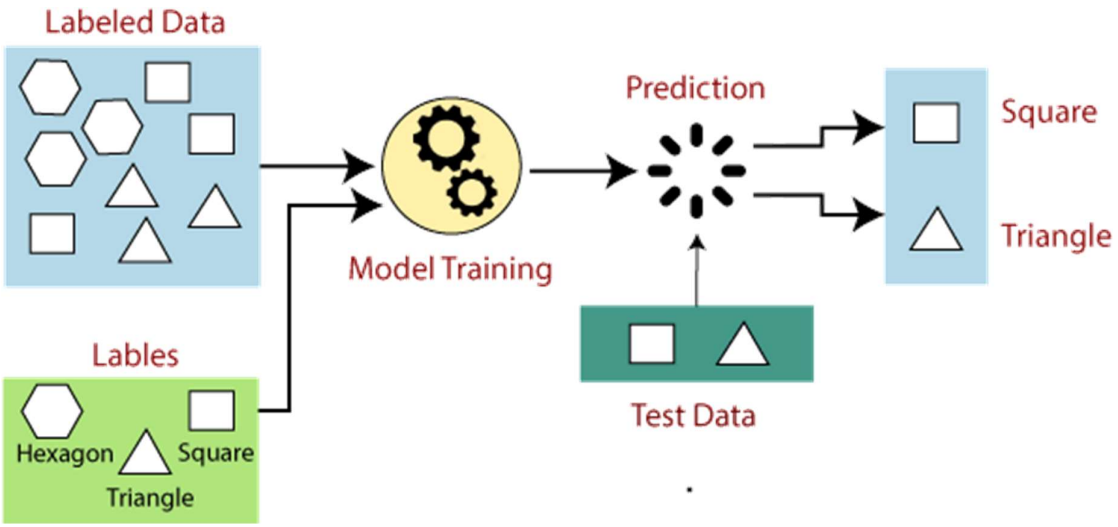

FIG 1.2. MACHINE LEARNING TESTING

Steps Involved in Supervised Learning:

- First Determine the type of training dataset

- Collect/Gather the labelled training data.

- Split the training dataset into training dataset, test dataset, and validation dataset.

- Determine the input features of the training dataset, which should have enough knowledge so that the model can accurately predict the output.
- Determine the suitable algorithm for the model, such as support vector machine, decision tree, etc.
- Execute the algorithm on the training dataset. Sometimes we need validation setsas the control parameters, which are the subset of training datasets.
- Evaluate the accuracy of the model by providing the test set. If the model predictsthe correct output, which means our model is accurate.

Deep Learning

Deep learning is a branch of machine learning which is completely based on artificial neural networks, as neural network is going to mimic the human brain so deep learning is also a kind of mimic of human brain. It's on hype nowadays because earlier we did not have that much processing power and a lot of data. A formal definition of deep learning is- neurons Deep learning is a particular kind of machine learning that achieves great power and flexibility by learning to represent the world as a nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones. In brain approximately 100 billion neurons all together this is a picture of an individual neuron and each neuron is connected through thousands of their neighbors. The question here is how it recreates these neurons in a computer. So, it creates an artificial structure called an artificial neural net where we have nodes or neurons. It has some neurons for input value and some for output value and in between, there may be lots of neurons interconnected in the hidden layer.

It need to identify the actual problem in order to get the right solution and it should

be understood, the feasibility of the Deep Learning should also be checked (whether it should fit Deep Learning or not). It needs to identify the relevant data which should correspond to the actual problem and should be prepared accordingly. Choose the Deep Learning Algorithm appropriately. Algorithm should be used while training the dataset. Final testing should be done on the dataset

Deep learning (also known as deep structured learning) is part of a broader family of machine learning methods based on artificial neural networks with representation learning. Learning can be supervised, semi-supervised or unsupervised.

Deep-learning architectures such as deep neural networks, deep belief networks, deep reinforcement learning, recurrent neural networks and convolutional neural networks have been applied to fields including computer vision, speech recognition, natural language processing, machine translation, bioinformatics, drug design, medical image analysis, material inspection and board game programs, where they have produced results comparable to and in some cases surpassing human expert performance.

Artificial neural networks (ANNs) were inspired by information processing and distributed communication nodes in biological systems. ANNs have various differences from biological brains. Specifically, neural networks tend to be static and symbolic, while the biological brain of most living organisms is dynamic (plastic) and analogue.

The adjective "deep" in deep learning refers to the use of multiple layers in the network. Early work showed that a linear perceptron cannot be a universal classifier, but that a network with a non-polynomial activation function with one hidden layer of unbounded width can. Deep learning is a modern variation which is concerned with an unbounded number of layers of bounded size, which permits practical application and optimized implementation, while retaining theoretical universality under mild conditions. In deep learning the layers are also permitted to be heterogeneous and to deviate widely from biologically informed connectionist models, for the sake of efficiency, trainability and understandability, whence the "structured" part.

Deep learning is a class of machine learning algorithms that uses multiple layers to progressively extract higher-level features from the raw input. For example, in image processing, lower layers may identify edges, while higher layers may identify the concepts relevant to a human such as digits or letters or faces.

1.4 INTERPRETATIONS

Deep neural networks are generally interpreted in terms of the universal approximation theorem or probabilistic inference.

The classic universal approximation theorem concerns the capacity of feed-forward neural networks with a single hidden layer of finite size to approximate continuous functions. In 1989, the first proof was published by George Cybenko for sigmoid activation functions and was generalised to feed-forward multi-layer architectures in 1991 by Kurt Hornik. Recent work also showed that universal approximation also holds for non-bounded activation functions such as the rectified linear unit.

The universal approximation theorem for deep neural networks concerns the capacity of networks with bounded width but the depth is allowed to grow proved that if the width of a deep neural network with ReLU activation is strictly larger than the input dimension, then the network can approximate any Lebesgue integrable function; If the width is smaller or equal to the input dimension, then deep neural network is not a universal approximator.

The probabilistic interpretation derives from the field of machine learning. It features inference, as well as the optimization concepts of training and testing, related to fitting and generalization, respectively. More specifically, the probabilistic interpretation considers the activation nonlinearity as a cumulative distribution function. The probabilistic interpretation led to the introduction of dropout as regularizer in neural networks. The probabilistic interpretation was

introduced by researchers including Hopfield, Widrow and Narendra and popularized in surveys such as the one by Bishop.

1.5 DEEP LEARNING REVOLUTION

In 2012, a team led by George E. Dahl won the "Merck Molecular Activity Challenge" using multi-task deep neural networks to predict the biomolecular target of one drug. In 2014, Hochreiter's group used deep learning to detect off-target and toxic effects of environmental chemicals in nutrients, household products and drugs and won the "Tox21 Data Challenge" of NIH, FDA and NCATS.

Significant additional impacts in image or object recognition were felt from 2011 to 2012. Although ANNs trained by back-propagation had been around for decades, and GPU implementations of NNs for years, including ANNs, fast implementations of ANNs on GPUs were needed to progress on computer vision. In 2011, this approach achieved for the first time superhuman performance in a visual pattern recognition contest. Also in 2011, it won the ICDAR Chinese handwriting contest, and in May 2012, it won the ISBI image segmentation contest. Until 2011, ANNs did not play a major role at computer vision conferences, but in June 2012, a paper by Ciresan et al. at the leading conference CVPR showed how max-pooling ANNs on GPU can dramatically improve many vision benchmark records. In October 2012, a similar system by Krizhevsky et al. won the large-scale ImageNet competition by a significant margin over shallow machine learning methods. In November 2012, Ciresan et al.'s system also won the ICPR contest on analysis of large medical images for cancer detection, and in the following year also the MICCAI Grand Challenge on the same topic. In 2013 and 2014, the error rate on the ImageNet task using deep learning was further reduced, following a similar trend in large-scale speech recognition.

Image classification was then extended to the more challenging task of generating descriptions (captions) for images, often as a combination of ANNs and LSTMs.

Some researchers state that the October 2012 ImageNet victory anchored the start of a "deep learning revolution" that has transformed the AI industry.

CHAPTER 2

2. LITERATURE SURVEY

REFERENCE 1:

TITLE: Mitigating Adversarial Attacks on Data-Driven Invariant Checkers for Cyber-Physical Systems

AUTHOR: Rajib Ranjan Maiti , Cheah Huei Yoong

YEAR: 2023

DESCRIPTION:

The use of invariants in developing security mechanisms has become an attractive research area because of their potential to both prevent attacks and detect attacks in Cyber-Physical Systems (CPS). In general, an invariant is a property that is expressed using design parameters along with Boolean operators and which always holds in normal operation of a system, in particular, a CPS. Invariants can be derived by analysing operational data of various design parameters in a running CPS, or by analysing the system's requirements/design documents, with both of the approaches demonstrating significant potential to detect and prevent cyber-attacks on a CPS. While data-driven invariant generation can be fully automated, design-driven invariant generation has a substantial manual intervention. In this paper, we aim to highlight the shortcomings in data-driven invariants by demonstrating a set of adversarial attacks on such invariants. We propose a solution strategy to detect such attacks by complementing them with design-driven invariants. We perform all our experiments on a real water treatment testbed. We shall demonstrate that our approach can significantly reduce false positives and achieve high accuracy in attack detection on CPSs.

REFERENCE 2:

TITLE: Cyber Restoration of Power Systems: Concept and Methodology for Resilient Observability.

AUTHOR: Shamsun Nahar Edib , Graduate Student Member IEEE, Yuzhang Lin

YEAR: 2023

DESCRIPTION:

In order to have a properly functioning cyber– physical power system, the operational data need to be properly measured, transmitted, and processed. In case of a malicious event on the cyber layer of the power system, such as the wide-area monitoring system, cyber components, such as phasor measurement units (PMUs), communication routers, and phasor data concentrators (PDCs) may be compromised, leading to an unobservable power system. This article proposes the concept of cyber restoration of power systems, and an optimal restoration scheme to recover the system observability swiftly after massive interruptions. The cyber restoration problem is formulated as a mixed integer linear programming (MILP) problem considering PMU measurability, communication network connectivity, and PDC processability conditions, as well as cyber restoration resources as constraints. Results in the IEEE 57-bus system validate that the proposed optimization method can provide solutions that recover system observability much faster than heuristic methods, demonstrating the need for systematic cyber restoration planning research and implementation.

REFERENCE 3:

TITLE: Attack Hypotheses Generation Based on Threat Intelligence Knowledge Graph.

AUTHOR: Florian Klaus Kaiser, Uriel Dardik, Aviad Elitzur, Polina Zilberman, Nir Daniel, Marcus Wiens, Frank Schultmann, Yuval Elovici, and Rami Puzis

YEAR: 2022

DESCRIPTION:

Cyber threat intelligence on past attacks may help with attack reconstruction and the prediction of the course of an ongoing attack by providing deeper understanding of the tools and attack patterns used by attackers. Therefore, cyber security analysts employ threat intelligence, alert correlations, machine learning, and advanced visualizations in order to produce sound attack hypotheses. In this paper, we present AttackDB, a multi-level threat knowledge base that combines data from multiple threat intelligence sources to associate high-level ATT&CK techniques with low-level telemetry found in

behavioral malware reports. We also present the Attack Hypothesis Generator which relies on knowledge graph traversal algorithms and a variety of link prediction methods to automatically infer ATT&CK techniques from a set of observable artifacts. Results of experiments performed with 53K VirusTotal reports indicate that the proposed algorithms employed by the Attack Hypothesis Generator are able to produce accurate adversarial technique hypotheses with a mean average precision greater than 0.5 and area under the receiver operating characteristic curve of over 0.8 when it is implemented on the basis of AttackDB. The presented toolkit will help analysts to improve the accuracy of attack hypotheses and to automate the attack hypothesis generation process. Index Terms—cyber threat intelligence, data fusion, attack hypotheses, link prediction of the future which leads to a very vast change.

REFERENCE 4:

TITLE: A STUDY OF CYBER SECURITY CHALLENGES AND ITS EMERGNING TRENDS ON LATEST TECHNOLOGIES.

AUTHOR: G.NIKHITA REDDY1 , G.J.UGANDER REDDY2

YEAR: 2021

DESCRIPTION:

Cyber Security plays an important role in the field of information technology .Securing the information have become one of the biggest challenges in the present day. When ever we think about the cyber security the first thing that comes to our mind is ‘cyber crimes’ which are increasing immensely day by day. Various Governments and companies are taking many measures in order to prevent these cyber crimes. Besides various measures cyber security is still a very big concern to many. This paper mainly focuses on challenges faced by cyber security on the latest technologies .It also focuses on latest about the cyber security techniques, ethics and the trends changing the face of cyber security. Keywords: cyber security, cyber crime, cyber ethics, social media, cloud computing, android apps.

REFERENCE 5:

TITLE: A comprehensive review study of cyber-attacks and cyber security; Emerging trends and recent developments.

AUTHOR: Yuchong Li a,b , Qinghui Liu c

YEAR: 2021

DESCRIPTION:

At present, most of the economic, commercial, cultural, social and governmental activities and interactions of countries, at all levels, including individuals, non-governmental organizations and government and governmental institutions, are carried out in cyberspace. Recently, many private companies and government organizations around the world are facing the problem of cyber-attacks and the danger of wireless communication technologies. Today's world is highly dependent on electronic technology, and protecting this data from cyber-attacks is a challenging issue. The purpose of cyber-attacks is to harm companies financially. In some other cases, cyber-attacks can have military or political purposes. Some of these damages are: PC viruses, knowledge breaks, data distribution service (DDS) and other assault vectors. To this end, various organizations use various solutions to prevent damage caused by cyberattacks. Cyber security follows real-time information on the latest IT data. So far, various methods had been proposed by researchers around the world to prevent cyber-attacks or reduce the damage caused by them. Some of the methods are in the operational phase and others are in the study phase. The aim of this study is to survey and comprehensively review the standard advances presented in the field of cyber security and to investigate the challenges, weaknesses and strengths of the proposed methods. Different types of new descendant attacks are considered in details. Standard security frameworks are discussed with the history and early-generation cyber-security methods. In addition, emerging trends and recent developments of cyber security and security threats and challenges are presented. It is expected that the comprehensive review study presented for IT and cyber security researchers will be useful.

CHAPTER 3

3. SYSTEM ANALYSIS

➢ 3.1 EXISTING SYSTEM:

The utilization of invariants in the development of security mechanisms has garnered significant interest in the context of Cyber-Physical Systems (CPS), owing to their potential to prevent and identify attacks. Invariants are properties expressed using design parameters and Boolean operators, which remain true during normal CPS operation. These invariants can be derived from operational data or system requirements/design documents. While data-driven invariant generation is automated, design-driven methods require manual input. This paper exposes vulnerabilities in data-driven invariants by showcasing adversarial attacks. To address this, the paper proposes a solution involving both data-driven and design-driven invariants, aiming to enhance attack detection accuracy. Experiments conducted on an actual water treatment testbed demonstrate the effectiveness of this approach in reducing false positives and achieving reliable attack detection for CPSs.

➢ 3.2 Drawbacks:

- They have used ROC algorithm which is one of the machine learning algorithm.
- They did not implement the deployment process.
- Implementing a hierarchical network with both communication and physical layers in a distributed and decentralized manner can introduce increased complexity to the system design and operation.
- This complexity might lead to challenges in system maintenance, troubleshooting, and scalability.

➢ 3.3Proposed system:

The proposed system focuses on utilizing Artificial Neural Networks (ANNs) for the detection of cyber-attacks. ANNs are a type of machine learning technique inspired by the human brain's neural structure. The objective here is to leverage ANNs to identify and counteract cyber-attacks effectively. This research

acknowledges the rising threat of cyber-attacks and aims to enhance cybersecurity by using ANNs as a robust and adaptable tool. The study likely explores how ANNs can learn patterns from historical attack data and subsequently identify deviations from normal behaviour, enabling timely detection and response to potential cyber threats. Overall, the abstract highlights the potential of ANNs to contribute significantly to cyber-attack detection and prevention strategies.

Advantages

- We implemented ANN architecture algorithm which can handle more hierarchical data.
- We implemented the one of best architecture of neural network to reduce the attack based on the cyber security in water management.
- We develop a framework based application for deployment purpose.

CHAPTER 4

4. REQUIREMENT ANALYSIS

➢ 4.1 HARDWARE REQUIREMENT

- **Processor : Pentium IV/III**
- **Hard disk : minimum 80 GB**
- **RAM : minimum 4 GB**

➢ 4.2 SOFTWARE REQUIREMENT

- **Operating System : Windows**
- **Tool : Anaconda with Jupyter Notebook**

refactorings and rich navigation capabilities.

4.2.1 PYTHON

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically-typed and garbage-collected. It supports multiple programmingparadigms,including structured (particularly, procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a DETECTION OF CYBER ATTACKS collection system using reference counting. Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible. Python 2 was discontinued with version 2.7.18 in 2020.

Python consistently ranks as one of the most popular programming languages.

CHAPTER 5

5. SYSTEM DESIGN

5.1 BLOCK DIAGRAM

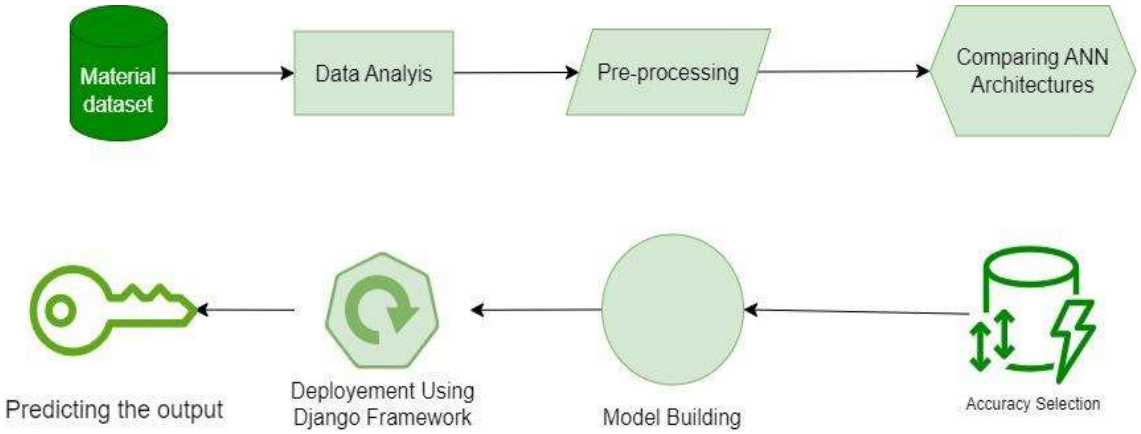

FIG 5.1 BLOCK DIAGRAM

5.2 FEASIBILITY STUDY

5.2.1 SPLITTING THE DATASET

The data use is usually split into training data and test data. The training set contains a known output and the model learns on this data in order to be generalized to other data later on. It has the test dataset (or subset) in order to test our models and it will do this using Tensorflow library in Python using the Keras method.

5.2.2 CONSTRUCTION OF A DETECTING MODEL

Deep learning needs data gathering have lot of past image data's. Training and testing this model working and predicting correctly.

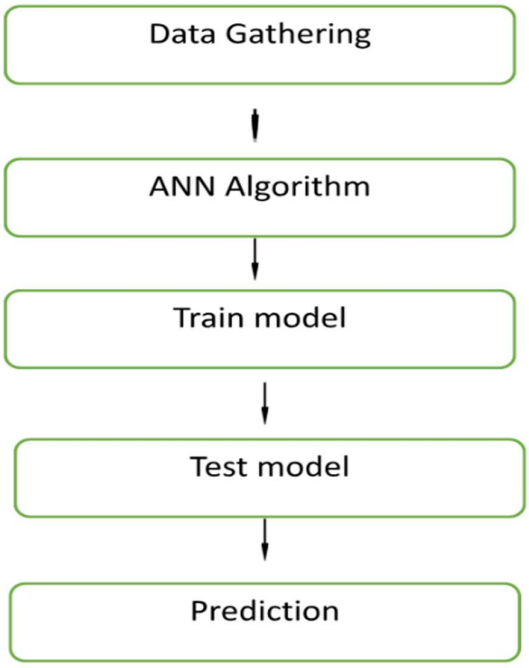

FIG 5.2 DETECTING MODEL

5.3 UML DIAGRAMS

The Unified Modeling Language (UML) is used to specify, visualize, modify, constructand document the artefacts of an object-oriented software intensive system under development. UML offers a standard way to visualize a system's architectural blueprints, including elements such as:

- Actors
- Business processes
- (logical) components
- Activities
- Programming language statements
- Database schemas, and
- Reusable software components.

UML combines best techniques from data modeling (entity relationship diagrams),

business modeling (work flows), object modeling, and component modeling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies. UML has synthesized the notations of the Booch method, the Object-modeling technique (OMT) and Object-oriented software engineering (OOSE) by fusing them into a single, common and widely usable modeling language. UML aims to be a standard modeling language which can model concurrent and distributed systems.

5.3.1 DATA FLOW DIAGRAM

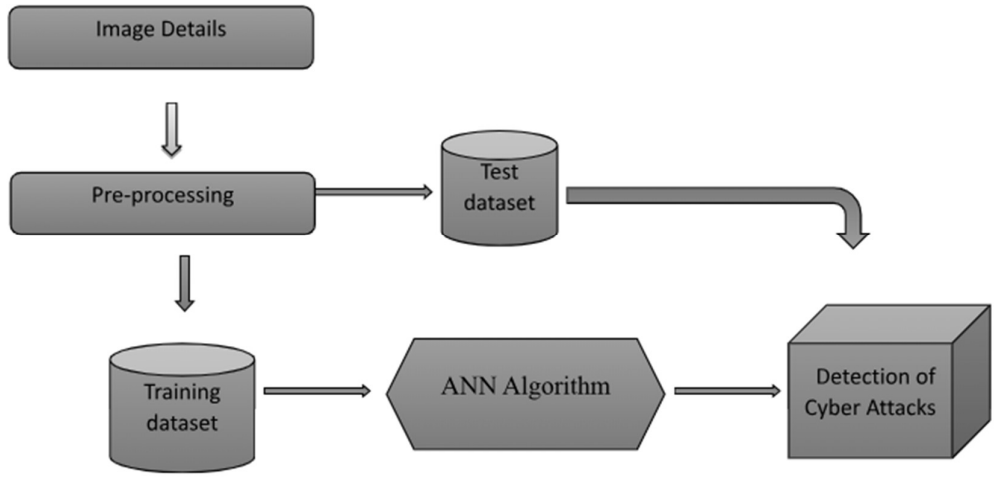

FIG 5.3 UML Data Flow Diagram

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design). A DFD shows what kind of information will be input to and output from the system, how the data will advance through the system, and where the data will be stored. It does not show information about process timing or whether processes will operate in sequence or in parallel, unlike a traditional structured flowchart which focuses on control flow, or a UML activity workflow diagram, which presents both control and data flows as a unified model. Data flow diagrams are also known as

bubble charts. DFD is a designing tool used in the top down approach to Systems Design. Symbols and Notations Used in DFDs Using any convention's DFD rules or guidelines, the symbols depict the four components of data flow diagrams.

External entity: an outside system that sends or receives data, communicating with the system being diagrammed. They are the sources and destinations of information entering or leaving the system. They might be an outside organization or person, a computer system or a business system. They are also known as terminators, sources and sinks or actors. They are typically drawn on the edges of the diagram.

Process: any process that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules.

Data store: files or repositories that hold information for later use, such as a database table or a membership form.

Data flow: the route that data takes between the external entities, processes and data stores. It portrays the interface between the other components and is shown with arrows, typically labeled with a short data name, like "Billing details."

DFD levels and layers A data flow diagram can dive into progressively more detail by using levels and layers, zeroing in on a particular piece. DFD levels are numbered 0, 1 or 2, and occasionally go to even Level 3 or beyond. The necessary level of detail depends on the scope of what you are trying to accomplish. DFD Level 0 is also called a Context Diagram. It's a basic overview of the whole system or process being analyzed or modeled. It's designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities. It should be easily understood by a wide audience, including stakeholders, business analysts, data analysts and developers.

5.3.2 WORK FLOW DIAGRAM

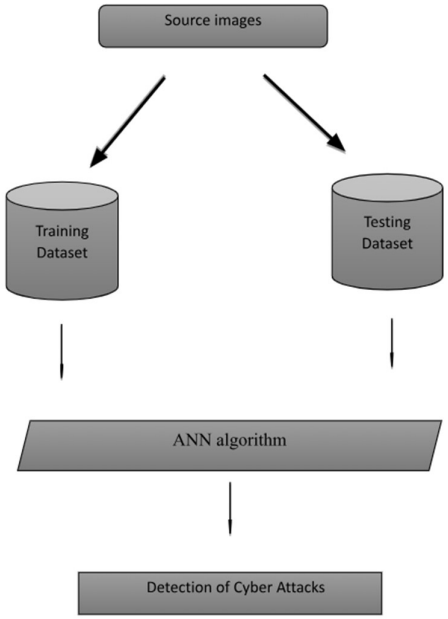

FIG 5.4 UML Work Flow Diagram

5.3.3 USECASE DIAGRAM:

A use case is a set of scenarios that describing an interaction between a user and a system. A use case diagram displays the relationships among actors and use cases. A use case is an external view of the system that represents some action the user might perform in order to complete the task.

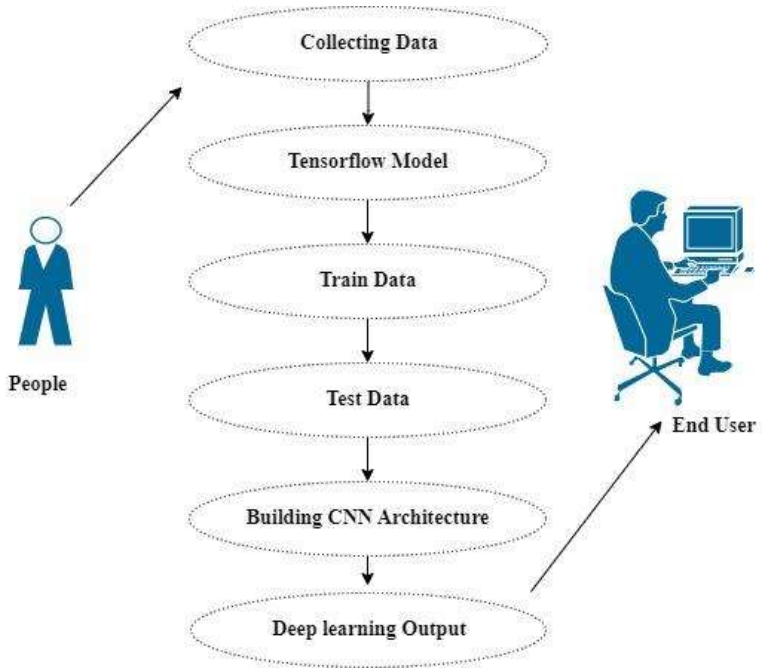

FIG 5.5 UML Use Case Diagram

5.3.4 SEQUENCE DIAGRAM:

Sequence Diagram:

Sequence Diagrams Represent the objects participating the interaction horizontally and time vertically. A Use Case is a kind of behavioural classifier that represents a declaration of an offered behaviour. Each use case specifies some behaviour, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behaviour of the subject without reference to its internal structure. These behaviours, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behaviour, including exceptional behaviour and error handling.

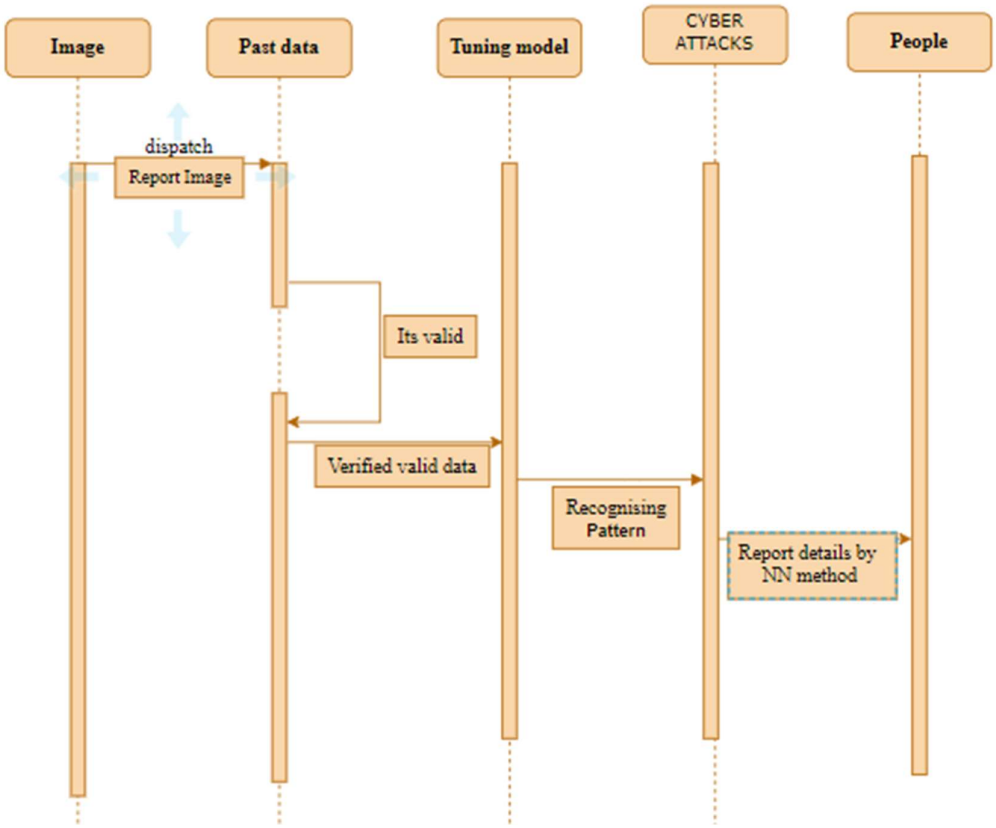

FIG 5.6 UML Sequence Diagram

5.3.5 ACTIVITY DIAGRAM:

Activity Diagrams-:

Activity diagrams are graphical representations of Workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

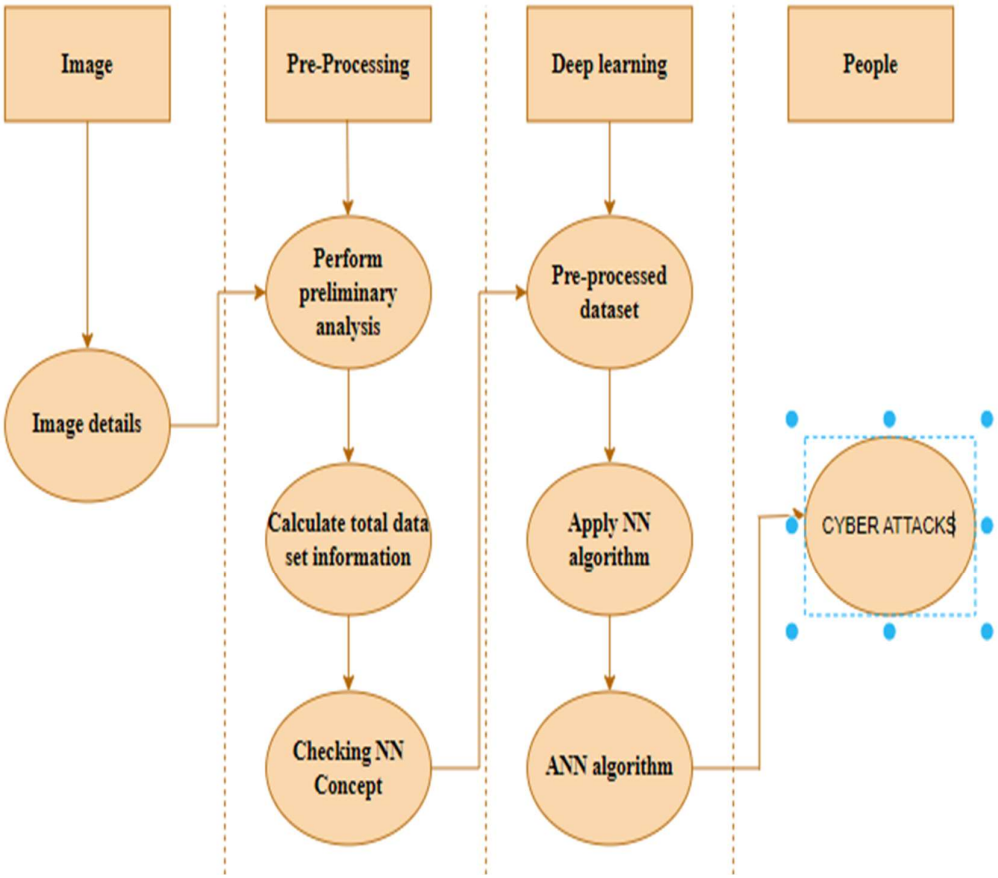

FIG 5.7 UML Activity Diagram

5.3.6 CLASS DIAGRAM:

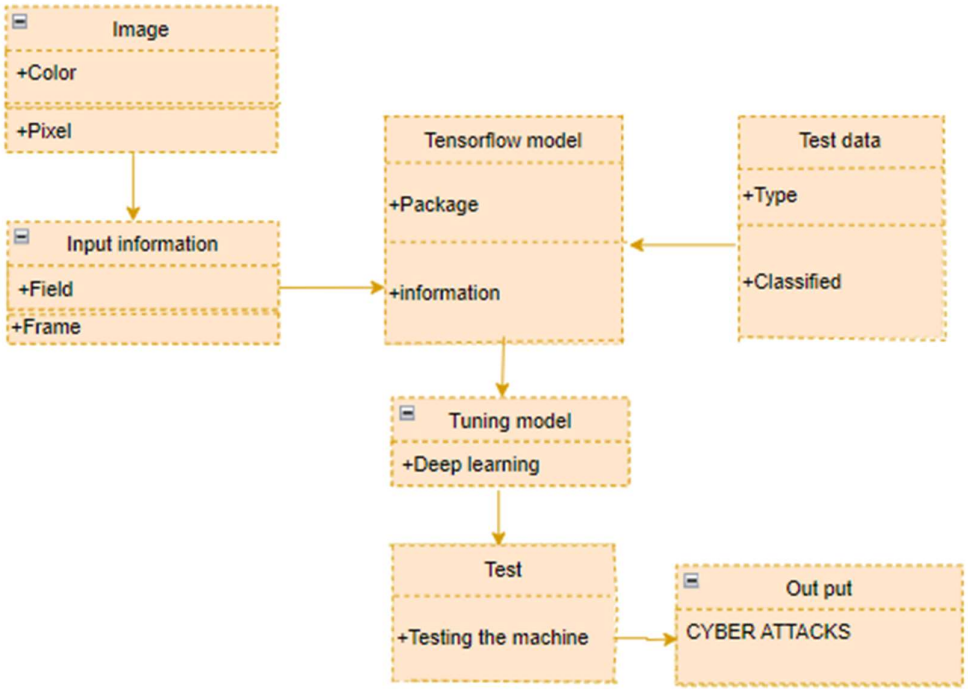

FIG 5.8 UML Class Diagram

5.3.7 ER DIAGRAM:

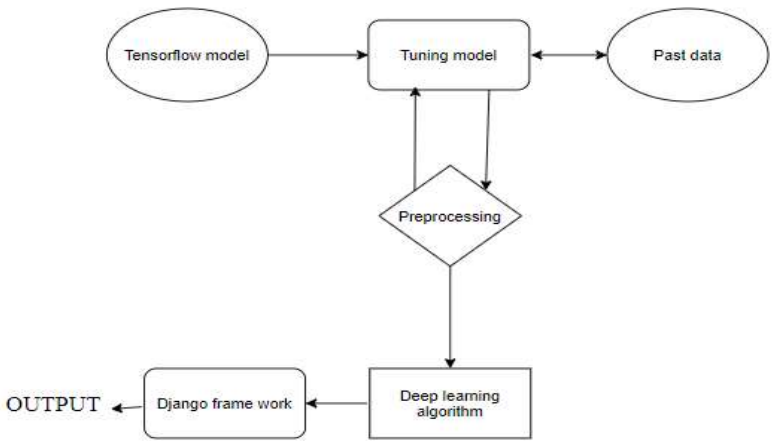

FIG 5.9 UML ER Diagram

5.3.8 COLLABORATION DIAGRAM:

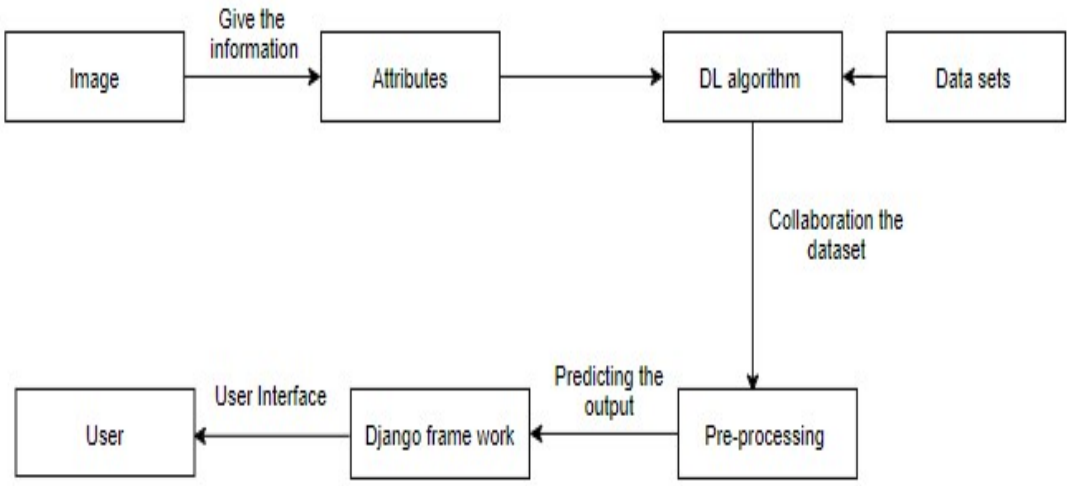

FIG 5.10 UML Collaboration Diagram

5.4 MODULE DESCRIPTION

5.4.1 PRE-PROCESSING

We have to import our data set using keras preprocessing image data generator function also we create size, rescale, range, zoom range, horizontal flip. Then we import our image dataset from folder through the data generator function. Here we set train, test, and validation also we set target size, batch size and class-mode from this function we have to train using our own created network by adding layers of ANN.

DATA PREPROCESSING AND DATA CLEANING

```
[1]: import pandas as pd
     import numpy as np

     import warnings
     warnings.filterwarnings('ignore')

[2]: df = pd.read_csv('A.csv')
     del df['DATETIME']
     del df['F_PU9']
     del df['S_PU9']
     del df['F_PU5']
     del df['S_PU5']
     del df['F_PU3']
     del df['S_PU3']
     del df['S_PU1']
     df.head()
```

[2]:

	L_T1	L_T2	L_T3	L_T4	L_T5	L_T6	L_T7	F_PU1	F_PU2	S_PU2	...	P_J256	P_J289	P_J415	P_J302	P_J306	P_J307	P_J317	P_J14	P_J422	ATT_FLAG
0	2.44	5.24	3.19	4.10	2.86	5.50	4.39	93.63	93.65	1	...	70.00	28.22	85.87	21.69	82.72	21.58	71.99	39.33	29.64	0
1	2.66	4.53	3.20	4.18	3.29	5.44	4.53	89.41	89.43	1	...	87.73	24.45	84.87	29.81	86.62	29.81	59.76	42.17	26.15	0

FIG 5.11 Pre Processing

5.4.2 TRAIN THE IMAGE DATASET

To train our dataset using classifier and fit generator function also we make training steps per epoch's then total number of epochs, validation data and validation steps using this data we can train our dataset.

DATA VISUALIZATION AND DATA ANALYSIS

```
[1]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns

[2]: df = pd.read_csv('A.csv')
     del df['DATETIME']
     del df['F_PU9']
     del df['S_PU9']
     del df['F_PU5']
     del df['S_PU5']
     del df['F_PU3']
     del df['S_PU3']
     del df['S_PU1']
     df.head()
```

[2]:

	L_T1	L_T2	L_T3	L_T4	L_T5	L_T6	L_T7	F_PU1	F_PU2	S_PU2	...	P_J256	P_J289	P_J415	P_J302	P_J306	P_J307	P_J317	P_J14	P_J422	ATT_FLAG
0	2.44	5.24	3.19	4.10	2.86	5.50	4.39	93.63	93.65	1	...	70.00	28.22	85.87	21.69	82.72	21.58	71.99	39.33	29.64	0
1	2.66	4.53	3.20	4.18	3.29	5.44	4.53	89.41	89.43	1	...	87.73	24.45	84.87	29.81	86.62	29.81	59.76	42.17	26.15	0
2	3.11	3.66	3.66	4.21	3.87	5.15	3.22	89.88	89.89	1	...	89.29	23.90	87.11	29.85	87.64	29.85	58.50	42.00	25.56	0

FIG 5.12 Train the Image Dataset

5.4.3 WORKING OF LAYERS

A Convolutional Neural Network (ConvNet/ANN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics. The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in

the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. Their network consists of four layers with 1,024 input units, 256 units in the first hidden layer, eight units in the second hidden layer, and two output units.

- INPUT LAYER: Input layer in ANN contain image data. Image data is represented by three dimensional matrixes. It needs to reshape it into a single column. Suppose you have image of dimension 28 x 28 =784, it need to convert it into 784 x 1 before feeding into input.
- CONVO LAYER: Convo layer is sometimes called feature extractor layer because features of the image are get extracted within this layer. First of all, a part of image is connected to Convo layer to perform convolution operation as we saw earlier and calculating the dot product between receptive field (it is a local region of the input image that has the same size as that of filter) and the filter. Result of the operation is single integer of the output volume. Then the filter over the next receptive field of the same input image by a Stride and do the same operation again. It will repeat the same process again and again until it goes through the whole image. The output will be the input for the next layer.
- POOLING LAYER: Pooling layer is used to reduce the spatial volume of input image after convolution. It is used between two convolution layers. If it applies FC after Convo layer without applying pooling or max pooling, then it will be computationally expensive. So, the max pooling is only way to reduce the spatial volume of input image. It has applied max pooling in single depth slice with Stride of 2. It can observe the 4 x 4 dimension input is reducing to 2 x 2 dimensions.
- FULLY CONNECTED LAYER (FC): Fully connected layer involves weights, biases, and neurons. It connects neurons in one layer to neurons in another layer. It is used to classify images between different categories by training.

- SOFTMAX/LOGISTIC LAYER: Softmax or Logistic layer is the last layer of ANN. It resides at the end of FC layer. Logistic is used for binary classification and softmax is for multi-classification.
- OUTPUT LAYER: Output layer contains the label which is in the form of one-hot encoded. Now you have a good understanding of ANN.

Convolutional layers:

Convolutional layers are the layers where filters are applied to the original image, or to other feature maps in a deep ANN. This is where most of the user-specified parameters are in the network. The most important parameters are the number of kernels and the size of the kernels.

Pooling layers:

Pooling layers are similar to convolutional layers, but they perform a specific function such as max pooling, which takes the maximum value in a certain filter region, or average pooling, which takes the average value in a filter region. These are typically used to reduce the dimensionality of the network.

Dense or Fully connected layers:

Fully connected layers are placed before the classification output of a ANN and are used to flatten the results before classification.

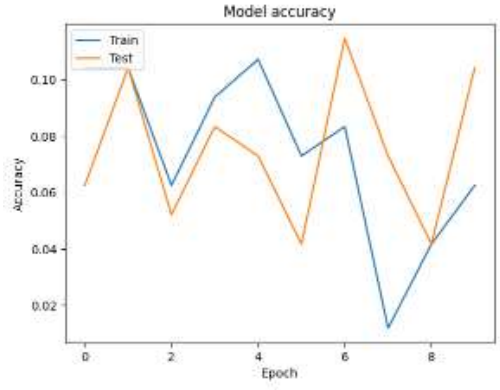

FIG 5.13 MODEL ACCURACY

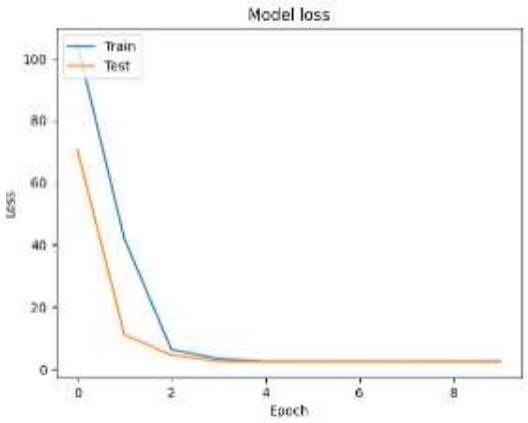

FIG 5.14 MODEL LOSS

5.4.4 TENSORFLOW

The detection of cyber attacks using artificial neural networks in TensorFlow involves leveraging the power of deep learning to analyze and identify patterns indicative of malicious activities. TensorFlow, an open-source machine learning library, facilitates the implementation of complex neural network architectures for robust cyber threat detection. By training neural networks on diverse datasets, the system can learn and adapt to evolving cyber threats, enhancing its accuracy and effectiveness. TensorFlow's flexibility and scalability make it a preferred platform for building sophisticated models capable of discerning subtle indicators of cyber attacks in real-time. The integration of artificial neural networks in TensorFlow thus contributes to a proactive and dynamic defense against cybersecurity threats.

5.4.5 PYTORCH

In the realm of cybersecurity, PyTorch, a powerful deep learning framework, is harnessed for the detection of cyber attacks. Leveraging its flexibility and efficiency, artificial neural networks (ANNs) are implemented to analyze intricate patterns in network data, enabling the identification of anomalous activities indicative of potential threats. Through PyTorch's seamless integration, the system facilitates the training and deployment of ANNs, enhancing the accuracy and responsiveness of cyber attack

detection mechanisms. This approach not only showcases the efficacy of PyTorch in neural network development but also underscores its role in fortifying digital security against evolving cyber threats.

Convolutional layers:

Convolutional layers are the layers where filters are applied to the original image, or to other feature maps in a deep ANN. This is where most of the user-specified parameters are in the network. The most important parameters are the number of kernels and the size of the kernels.

Pooling layers:

Pooling layers are similar to convolutional layers, but they perform a specific function such as max pooling, which takes the maximum value in a certain filter region, or average pooling, which takes the average value in a filter region. These are typically used to reduce the dimensionality of the network.

Dense or Fully connected layers:

Fully connected layers are placed before the classification output of a ANN and are used to flatten the results before classification. This is similar to the output layer of an MLP.

```
[1]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns

     import torch
     import torch.nn as nn
     import torch.optim as optim
     from torch.utils.data import Dataset, DataLoader

     import warnings
     warnings.filterwarnings('ignore')

[2]: df = pd.read_csv('A.csv')
     del df['DATETIME']
     del df['F_PU9']
     del df['S_PU9']
     del df['F_PU5']
     del df['S_PU5']
     del df['F_PU3']
     del df['S_PU3']
     del df['S_PU1']
     df.head()

[2]:   L_T1  L_T2  L_T3  L_T4  L_T5  L_T6  L_T7  F_PU1  F_PU2  S_PU2  ...  P_J256  P_J289  P_J415  P_J302  P_J306  P_J307  P_J317  P_J14  P_J422  ATT_FLAG
```

FIG 5.15 Tensor flow

5.4.6 DJANGO DEPLOYMENT

Deploying the model in Django Framework and predicting output

In this module the trained deep learning model is converted into hierarchical data format file (.h5 file) which is then deployed in our django framework for providing better user interface and predicting the output whether the given material image is Fabric / Glass / Plastic / Stone / Wooden.

Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It is free and open source, has a thriving and active community, great documentation, and many options for free and paid-for support.

Django helps you write software that is:

Complete

Django follows the "Batteries included" philosophy and provides almost everything developers might want to do "out of the box". Because everything you need is part of the one "product", it all works seamlessly together, follows consistent design principles, and has extensive and up-to-date documentation.

Versatile

Django can be (and has been) used to build almost any type of website — from content management systems and wikis, through to social networks and news sites. It can work with any client-side framework, and can deliver content in almost any format (including HTML, RSS feeds, JSON, XML, etc). The site you are currently reading is built with Internally, while it provides choices for almost any functionality you might want (e.g. several popular databases, templating engines, etc.), it can also be extended to use other components if needed.

Secure

Django helps developers avoid many common security mistakes by providing a framework that has been engineered to "do the right things" to protect the website automatically. For example, Django provides a secure way to manage user accounts and passwords, avoiding common mistakes like putting session information in cookies where it is vulnerable (instead cookies just contain a key, and the actual data is stored in the database) or directly storing passwords rather than a password hash.

A password hash is a fixed-length value created by sending the password through a cryptographic hash function. Django can check if an entered password is correct by running it through the hash function and comparing the output to the stored hash value. However due to the "one-way" nature of the function, even if a stored hash value is compromised it is hard for an attacker to work out the original password.

Django enables protection against many vulnerabilities by default, including SQL injection, cross-site scripting, cross-site request forgery and clickjacking (see Website security for more details of such attacks).

CHAPTER 6

6. METHODOLOGY

The methodology involves acquiring diverse datasets representing normal and anomalous network behavior. Preprocessing techniques, such as feature extraction and normalization, are applied to enhance data quality. An artificial neural network (ANN) is then trained using the prepared dataset, with a focus on optimizing network architecture and hyperparameters. Rigorous testing and validation are performed to assess the ANN's ability to accurately detect cyber attacks. The methodology also includes fine-tuning the model based on performance metrics to ensure robust and reliable cyber threat detection.

6.1 ANN MODEL

The detection of cyber attacks using Artificial Neural Networks (ANNs) involves leveraging advanced machine learning algorithms to analyze patterns in network data. ANNs, inspired by the human brain, can learn and adapt to the complex and dynamic nature of cyber threats. In this model, the neural network is trained on diverse datasets, enabling it to recognize anomalous activities indicative of potential cyber attacks. The ANN model's ability to generalize and identify patterns in real-time contributes to its effectiveness in enhancing cybersecurity measures. Through continuous learning, the ANN model evolves to stay resilient against evolving cyber threats.

Artificial Neural Network(ANN) uses the processing of the brain as a basis to develop algorithms that can be used to model complex patterns and prediction problems.

6.1.1 ARCHITECTURE

The network architecture has an input layer, hidden layer (there can be more than 1) and the output layer. It is0 also called MLP (Multi Layer Perceptron) because of the multiple layers.The hidden layer can be seen as a “distillation layer” that distills some of the important patterns from the inputs and passes it onto the next layer to see. It makes

the network faster and efficient by identifying only the important information from the inputs leaving out the redundant information.

The activation function serves two notable purposes:

- It captures non-linear relationship between the inputs
- It helps convert the input into a more useful output.

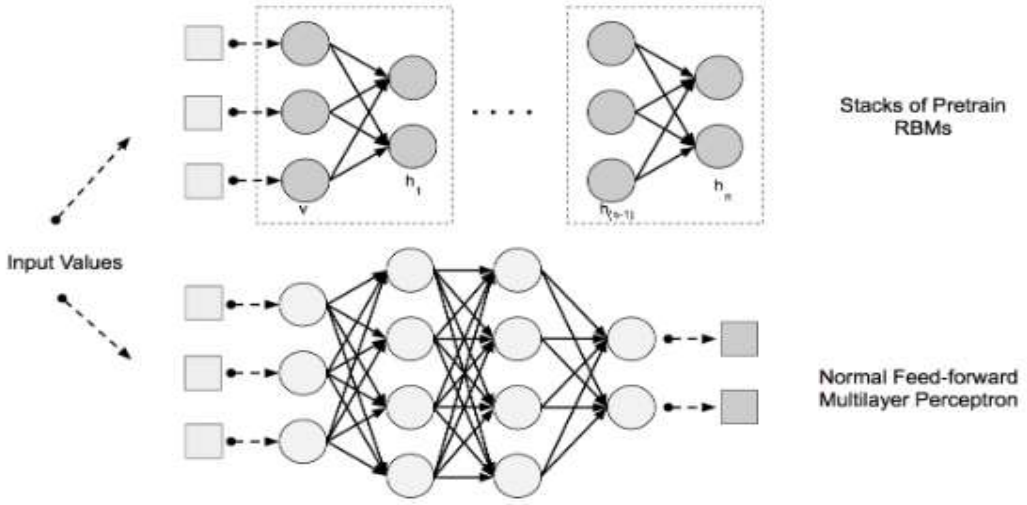

FIG 6.1 ANN ARCHITECTURE

6.1.2 NEURAL NETWORKS

ANNs have some key advantages that make them most suitable for certain problems and situations:

1. ANNs have the abil1ity to learn and model non-linear and complex relationships, which is really important because in real-life, many of the relationships between inputs and outputs are non-linear as well as complex.

2. ANNs can generalize — After learning from the initial inputs and their relationships, it can infer unseen relationships on unseen data as well, thus making the model generalize and predict on unseen data.

3. Unlike many other prediction techniques, ANN does not impose any restrictions on the input variables (like how they should be distributed). Additionally, many studies have shown that ANNs can better model heteroskedasticity i.e. data with high volatility

and non-constant variance, given its ability to learn hidden relationships in the data without imposing any fixed relationships in the data. This is something very useful in financial time series forecasting (e.g. stock prices) where data volatility is very high.

6.2 DATA GENERATOR

It is that rescales the image, applies shear in some range, zooms the image and does horizontal flipping with the image. This Image Data Generator includes all possible orientation of the image.

6.3 TRAINING PROCESS

train_datagen.flow_from_directory is the function that is used to prepare data from the train_dataset directory Target_size specifies the target size of the image. Test_datagen.flow_from_directory is used to prepare test data for the model and all is similar as above. fit_generator is used to fit the data into the model made above, other factors used are steps_per_epochs tells us about the number of times the model will execute for the training data.

1. **Data Collection and Preprocessing:**
- Gathering a representative dataset with sufficient diversity.
- Cleaning and preprocessing the data, including normalization, scaling, and handling missing values.
2. **Model Architecture:**
- Choosing an appropriate neural network architecture based on the nature of the problem (e.g., feedforward, convolutional, recurrent).
- Determining the number of layers, neurons, and activation functions for each layer.
3. **Loss Function:**

- Selecting a suitable loss function that measures the difference between predicted and actual values.
- Common loss functions include Mean Squared Error (MSE), Cross-Entropy Loss, and Hinge Loss, depending on the task (regression, classification, etc.).

4. **Optimizer Selection:**

- Choosing an optimization algorithm to update the model's parameters during training.
- Popular optimizers include Stochastic Gradient Descent (SGD), Adam, RMSprop, and Adagrad.

5. **Initialization:**

- Initializing the weights and biases of the neural network. Common methods include random initialization or using pre-trained weights (transfer learning).

6. **Training Loop:**

- Feeding input data into the network and obtaining predictions.
- Calculating the loss between predictions and ground truth.
- Backpropagating the error through the network to update weights and biases.
- Repeating this process iteratively for multiple epochs.

7. **Batching and Mini-Batch Gradient Descent:**

- Instead of updating weights after processing the entire dataset (batch gradient descent), using mini-batches of data for more efficient training.

8. **Regularization:**

- Applying techniques like dropout, L1/L2 regularization, or batch normalization to prevent overfitting and enhance generalization.

9. **Monitoring and Validation:**

- Evaluating the model's performance on a separate validation set to ensure it generalizes well to new data.
- Monitoring metrics such as accuracy, precision, recall, and F1-score.

10. **Hyperparameter Tuning:**

- Adjusting hyperparameters like learning rate, batch size, and regularization strength to optimize the model's performance.

11. **Early Stopping:**

- Introducing a mechanism to halt training if the model performance on the validation set ceases to improve, preventing overfitting.

12. **Model Evaluation:**

- Assessing the trained model on a test dataset to measure its generalization performance in real-world scenarios.

6.4 EPOCHS

It tells us the number of times model will be trained in forward and backward pass.

1. **Training Process:** During each epoch, the entire training dataset is fed forward through the neural network, and the resulting predictions are compared to the actual target values. The model's parameters are then adjusted using an optimization algorithm (such as gradient descent) to reduce the error.
2. **Multiple Epochs:** Neural networks often require multiple epochs to learn complex patterns within the data. Training for too few epochs may result in an underfit model that hasn't fully learned the underlying patterns, while training for too many epochs can lead to overfitting, where the model becomes too specific to the training data and performs poorly on new, unseen data.
3. **Batch Size:** In addition to epochs, the concept of batch size is important. During each epoch, the dataset is divided into batches, and the model's parameters are updated based on the average error over each batch. The batch size is a hyperparameter that affects the convergence speed and memory requirements during training.

4. **Validation Data:** It's common to reserve a portion of the dataset for validation. After each epoch, the model is evaluated on the validation set to monitor its performance on data it hasn't seen during training. This helps in detecting overfitting and guiding decisions on when to stop training.
5. **Early Stopping:** To prevent overfitting, a technique called early stopping is often employed. This involves monitoring the model's performance on the validation set and stopping the training process when the performance stops improving or starts to degrade.
6. **Learning Rate Scheduling:** The learning rate, which determines the size of the steps taken during parameter updates, can be adjusted during training. Learning rate scheduling can be applied to improve convergence and avoid overshooting the optimal parameter values.
7. **Computational Resources:** The number of epochs needed depends on factors such as the complexity of the model, the size of the dataset, and the computational resources available. Deep neural networks with large datasets might require more epochs, while simpler models on smaller datasets might converge faster.

6.5 VALIDATION PROCESS

Validation_data is used to feed the validation/test data into the model. Validation_steps denotes the number of validation/test samples.

1. **Data Splitting:**

- The dataset is typically divided into three subsets: training set, validation set, and test set.
- The training set is used to train the model, the validation set is used to fine-tune hyperparameters and monitor performance, and the test set is reserved for the final evaluation.

2. **Purpose of Validation:**

- The primary goal of validation is to prevent overfitting, where the model performs well on the training data but fails to generalize to new, unseen data.
- It helps in selecting the best-performing model during the training phase and provides insights into the model's ability to generalize.

3. **Hyperparameter Tuning:**

- During the training process, neural networks often have hyperparameters that need fine-tuning for optimal performance.
- Validation data helps in assessing the model's performance under various hyperparameter configurations, facilitating the selection of the best set.

4. **Early Stopping:**

- Validation can be used for early stopping, a technique where training is halted when the model's performance on the validation set stops improving.
- This prevents the model from overfitting and saves computational resources.

5. **Cross-Validation:**

- In situations where the dataset is limited, cross-validation can be employed. This involves splitting the dataset into multiple folds, training and validating the model on different combinations of these folds.
- It provides a more robust evaluation, especially when data is scarce.

6. **Performance Metrics:**

- Common metrics for evaluating neural network performance during validation include accuracy, precision, recall, F1 score, and area under the ROC curve (AUC-ROC).
- The choice of metrics depends on the nature of the problem and the desired model performance.

7. **Model Interpretability:**

- Validation results can offer insights into the model's behavior, helping to identify areas of improvement or potential biases.

- Visualization techniques, such as confusion matrices or learning curves, can aid in understanding the model's strengths and weaknesses.

8. **Validation Strategies:**

- Holdout Validation: A simple method where a portion of the dataset is randomly selected for validation.
- K-Fold Cross-Validation: The dataset is divided into K folds, and the model is trained and validated K times, with each fold serving as the validation set once.

CHAPTER 7

7 IMPLEMENTATION

7.1 ANACONDA NAVIGATOR

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows you to launch applications and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda.org or in a local Anaconda Repository.

Anaconda. Now, if you are primarily doing data science work, Anaconda is also a great option. Anaconda is created by Continuum Analytics, and it is a Python distribution that comes preinstalled with lots of useful python libraries for data science.

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment.

In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages and use multiple environments to separate these different versions.

The command-line program conda is both a package manager and an environment manager. This helps data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Navigator is an easy, point-and-click way to work with packages and conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages, and update them – all inside Navigator.

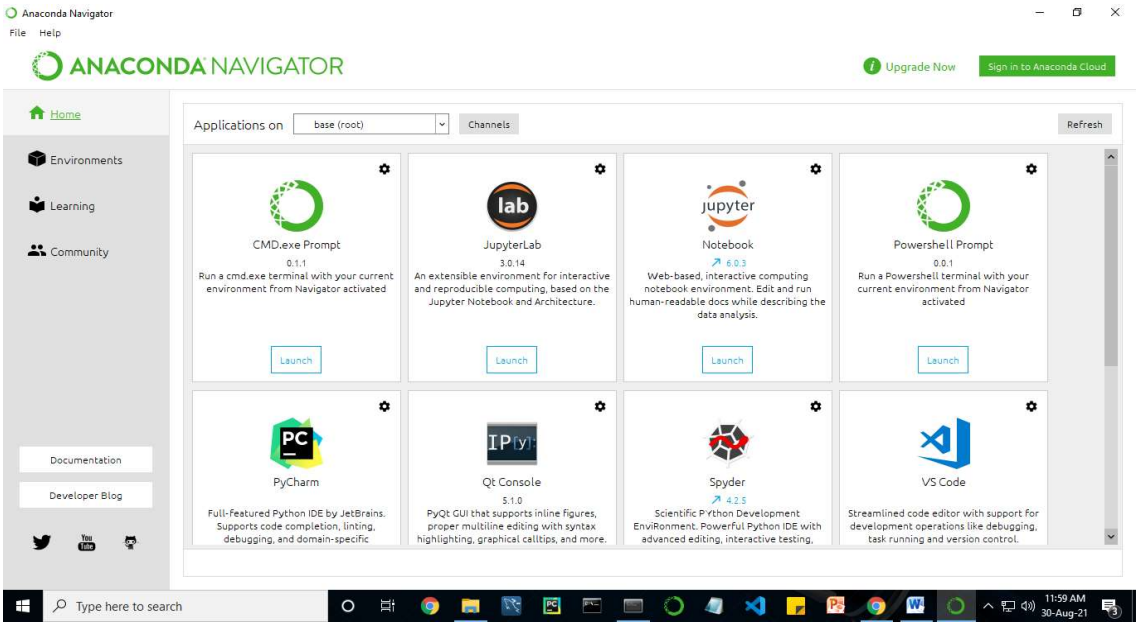

FIG 7.1 ANACONDA DASHBORAD_1

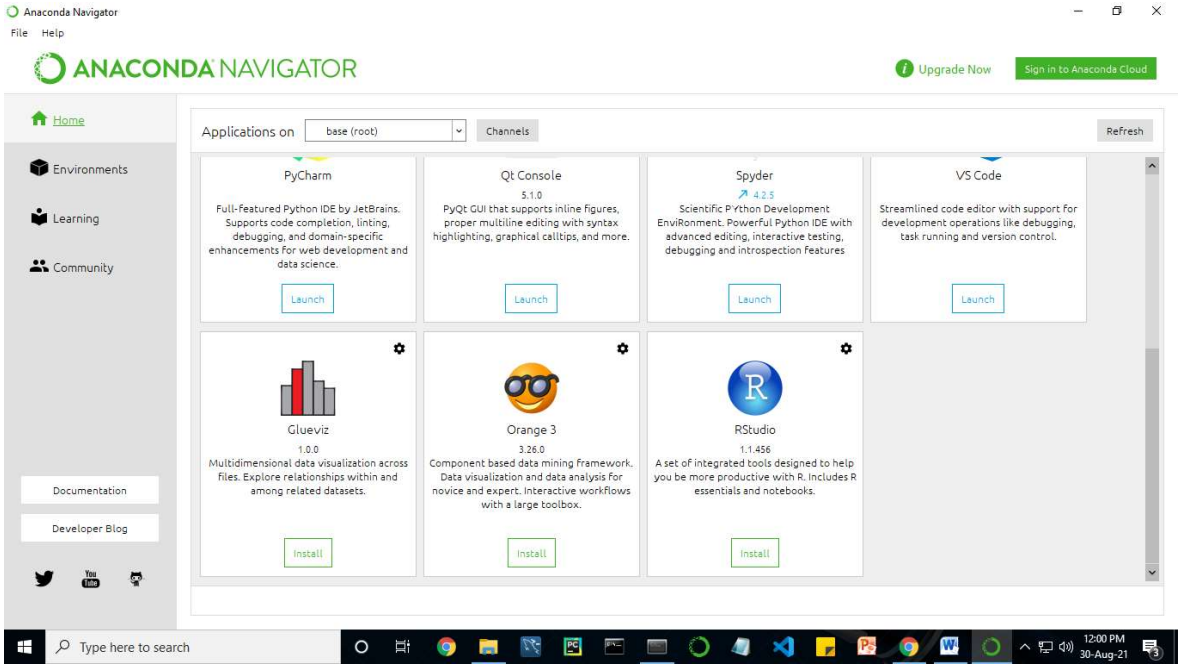

FIG 7.2 ANACONDA DASHBOARD_2

7.2 JUPYTER NOTEBOOK

This website acts as “meta” documentation for the Jupyter ecosystem. It has a collection of resources to navigate the tools and communities in this ecosystem, and to help you get started.

Project Jupyter is a project and community whose goal is to "develop open-source software, open-standards, and services for interactive computing across dozens of programming languages". It was spun off from IPython in 2014 by Fernando Perez.

Notebook documents are documents produced by the Jupyter Notebook App, which contain both computer code (e.g. python) and rich text elements (paragraph, equations, figures, links, etc.). Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc.) as well as executable documents which can be run to perform data analysis.

Installation: The easiest way to install the *Jupyter Notebook App* is installing a scientific python distribution which also includes scientific python packages. The most common distribution is called Anaconda

Running the Jupyter Notebook

Launching Jupyter Notebook App*:* The Jupyter Notebook App can be launched by clicking on the Jupyter Notebook icon installed by Anaconda in the start menu (Windows) or by typing in a terminal (cmd on Windows): “jupyter notebook”

This will launch a new browser window (or a new tab) showing the Notebook Dashboard, a sort of control panel that allows (among other things) to select which notebook to open.

When started, the Jupyter Notebook App can access only files within its start-up folder (including any sub-folder). No configuration is necessary if you place your notebooks in your home folder or subfolders. Otherwise, you need to choose a Jupyter Notebook App start-up folder which will contain all the notebooks.

Save notebooks: Modifications to the notebooks are automatically saved every few minutes. To avoid modifying the original notebook, make a copy of the notebook document (menu file -> make a copy…) and save the modifications on the copy.

Executing a notebook: Download the notebook you want to execute and put it in your notebook folder (or a sub-folder of it).

- Launch the jupyter notebook app
- In the Notebook Dashboard navigate to find the notebook: clicking on its name will open it in a new browser tab.
- Click on the menu *Help -> User Interface Tour* for an overview of the Jupyter Notebook App user interface.
- You can run the notebook document step-by-step (one cell a time) by pressing *shift + enter*.
- You can run the whole notebook in a single step by clicking on the menu *Cell -> Run All*.
- To restart the kernel (i.e. the computational engine), click on the menu *Kernel -> Restart*. This can be useful to start over a computation from scratch (e.g. variables are deleted, open files are closed, etc…).

Purpose: To support interactive data science and scientific computing across all programming languages.

File Extension: An **IPYNB** file is a notebook document created by Jupyter Notebook, an interactive computational environment that helps scientists manipulate and analyze data using Python.

JUPYTER Notebook App: The *Jupyter Notebook App* is a server-client application that allows editing and running notebook documents via a web browser. The *Jupyter Notebook App* can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet.

In addition to displaying/editing/running notebook documents, the *Jupyter Notebook App* has a "Dashboard" (Notebook Dashboard), a "control panel" showing local files and allowing to open notebook documents or shutting down their kernels.

kernel: A notebook *kernel* is a "computational engine" that executes the code contained in a Notebook document. The *ipython kernel*, referenced in this guide, executes python code. Kernels for many other languages exist (official kernels).

When you open a Notebook document, the associated *kernel* is automatically launched. When the notebook is *executed* (either cell-by-cell or with menu *Cell -> Run All*), the *kernel* performs the computation and produces the results. Depending on the type of computations, the *kernel* may consume significant CPU and RAM. Note that the RAM is not released until the *kernel* is shut-down

Notebook Dashboard: The *Notebook Dashboard* is the component which is shown first when you launch Jupyter Notebook App. The *Notebook Dashboard* is mainly used to open notebook documents, and to manage the running kernels (visualize and shutdown).

The *Notebook Dashboard* has other features similar to a file manager, namely navigating folders and renaming/deleting files

Working Process:

- Download and install anaconda and get the most useful package for machine learning in Python.
- Load a dataset and understand its structure using statistical summaries and data visualization.
- Machine learning models, pick the best and build confidence that the accuracy is reliable.

Python is a popular and powerful interpreted language. Unlike R, Python is a complete language and platform that you can use for both research and development and developing production systems. There are also a lot of modules and libraries to choose from, providing multiple ways to do each task. It can feel overwhelming.

The best way to get started using Python for machine learning is to complete a project.

- It will force you to install and start the Python interpreter (at the very least).
- It will give you a bird's eye view of how to step through a small project.
- It will give you confidence, maybe to go on to your own small projects.

When you are applying machine learning to your own datasets, you are working on a project. A machine learning project may not be linear, but it has a number of well-known steps:

- Define Problem.
- Prepare Data.
- Evaluate Algorithms.
- Improve Results.
- Present Results.

The best way to really come to terms with a new platform or tool is to work through a machine learning project end-to-end and cover the key steps. Namely, from loading data, summarizing data, evaluating algorithms and making some predictions. Here is an overview of what we are going to cover:

1. Installing the Python anaconda platform.
2. Loading the dataset.
3. Summarizing the dataset.
4. Visualizing the dataset.
5. Evaluating some algorithms.
6. Making some predictions.

7.3 PYCHARM

PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development. Code faster and with more easily in a smart and configurable editor with code completion, snippets, code folding and split windows support.

PyCharm Features

- **Intelligent Coding Assistance** - PyCharm provides smart code completion, code inspections, on-the-fly error highlighting and quick-fixes, along with automated code
- **Intelligent Code Editor** - PyCharm's smart code editor provides first-class support for Python, JavaScript, CoffeeScript, TypeScript, CSS, popular template languages and more. Take advantage of language-aware code completion, error detection, and on-the-fly code fixes!
- **Smart Code Navigation** - Use smart search to jump to any class, file or symbol, or even any IDE action or tool window. It only takes one click to switch to the declaration, super method, test, usages, implementation, and more.
- **Fast and Safe Refactorings** - Refactor your code the intelligent way, with safe Rename and Delete, Extract Method, Introduce Variable, Inline Variable or Method,

and other refactorings. Language and framework-specific refactorings help you perform project-wide changes.

- **Built-in Developer Tools** - PyCharm's huge collection of tools out of the box includes an integrated debugger and test runner; Python profiler; a built-in terminal; integration with major VCS and built-in database tools; remote development capabilities with remote interpreters; an integrated ssh terminal; and integration with Docker and Vagrant.
- **Debugging, Testing and Profiling** - Use the powerful debugger with a graphical UI for Python and JavaScript. Create and run your tests with coding assistance and a GUI-based test runner. Take full control of your code with Python Profiler integration.
- **VCS, Deployment and Remote Development** - Save time with a unified UI for working with Git, SVN, Mercurial or other version control systems. Run and debug your application on remote machines. Easily configure automatic deployment to a remote host or VM and manage your infrastructure with Vagrant and Docker.
- **Database tools** - Access Oracle, SQL Server, PostgreSQL, MySQL and other databases right from the IDE. Rely on PyCharm's help when editing SQL code, running queries, browsing data, and altering schemas.
- **Web Development** - In addition to Python, PyCharm provides first-class support for various Python web development frameworks, specific template languages, JavaScript, CoffeeScript, TypeScript, HTML/CSS, AngularJS, Node.js, and more.
- **Python Web frameworks** - PyCharm offers great framework-specific support for modern web development frameworks such as Django, Flask, Google App Engine, Pyramid, and web2py, including Django templates debugger, manage.py and appcfg.py tools, special autocompletion and navigation, just to name a few.
- **JavaScript & HTML -** PyCharm provides first-class support for JavaScript, CoffeeScript, TypeScript, HTML and CSS, as well as their modern successors. The

JavaScript debugger is included in PyCharm and is integrated with the Django server run configuration.

- **Live Edit** - Live Editing Preview lets you open a page in the editor and the browser and see the changes being made in code instantly in the browser. PyCharm auto-saves your changes, and the browser smartly updates the page on the fly, showing your edits.
- **Scientific Tools** - PyCharm integrates with IPython Notebook, has an interactive Python console, and supports Anaconda as well as multiple scientific packages including Matplotlib and NumPy.
- **Interactive Python console** - You can run a REPL Python console in PyCharm which offers many advantages over the standard one: on-the-fly syntax check with inspections, braces and quotes matching, and of course code completion.
- **Scientific Stack Support** - PyCharm has built-in support for scientific libraries. It supports Pandas, Numpy, Matplotlib, and other scientific libraries, offering you best-in-class code intelligence, graphs, array viewers and much more.
- **Conda Integration** - Keep your dependencies isolated by having separate Conda environments per project, PyCharm makes it easy for you to create and select the right environment.
- **Customizable and Cross-platform IDE** - Use PyCharm on Windows, Mac OS and Linux with a single license key. Enjoy a fine-tuned workspace with customizable color schemes and key-bindings, with VIM emulation available.
- **Customizable UI** - Enjoy a fine-tuned workspace with customizable color schemes and key-bindings.
- **Plugins** - More than 10 years of IntelliJ platform development gives PyCharm 50+ IDE plugins of different nature, including support for additional VCS, integrations with different tools and frameworks, and editor enhancements such as Vim emulation.

- **Cross-platform IDE** - PyCharm works on Windows, Mac OS or Linux. You can install and run PyCharm on as many machines as you have, and use the same environment and functionality across all your machines.

7.4 MATERIAL CLASSIFICATION

We give input image using keras pre-processing package. That input Image converted into array value using pillow and image to array function package. We have already classified materials in our dataset. It classifies what are the materials are available in our dataset then we have to predict our material.

The material recognition method is based on a two-channel architecture that is able to recognize the material. The material of the image parts are cropped and extracted and then used as the input into the inception layer of the ANN. The Training phase involves the feature extraction and classification using convolution neural network.

Libraries Required:

- **tensorflow**: Just to use the tensor board to compare the loss and adam curve our result data or obtained log.
- **keras**: To pre-process the image dataset.
- **matplotlib**: To display the result of our predictive outcome.
- **os**: To access the file system to read the image from the train and test directory from our machines.

TensorFlow:

TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow.

TensorFlow is a software library or framework, designed by the Google team to implement machine learning and deep learning concepts in the easiest manner. It combines the computational algebra of optimization techniques for easy calculation of many mathematical expressions.

Let us now consider the following important features of TensorFlow −

- It includes a feature of that defines, optimizes and calculates mathematical expressions easily with the help of multi-dimensional arrays called tensors.
- It includes a programming support of deep neural networks and machine learning techniques.
- It includes a high scalable feature of computation with various data sets.
- TensorFlow uses GPU computing, automating management. It also includes a unique feature of optimization of same memory and the data used.

Keras:

Keras runs on top of open source machine libraries like TensorFlow, Theano or Cognitive Toolkit (CNTK). Theano is a python library used for fast numerical computation tasks. TensorFlow is the most famous symbolic math library used for creating neural networks and deep learning models. TensorFlow is very flexible and the primary benefit is distributed computing. CNTK is deep learning framework developed by Microsoft. It uses libraries such as Python, C#, C++ or standalone machine learning toolkits. Theano and TensorFlow are very powerful libraries but difficult to understand for creating neural networks.

Keras is based on minimal structure that provides a clean and easy way to create deep learning models based on TensorFlow or Theano. Keras is designed to quickly define deep learning models. Well, Keras is an optimal choice for deep learning applications

Keras leverages various optimization techniques to make high level neural network API easier and more performant. It supports the following features −

- Consistent, simple and extensible API.

- Minimal structure - easy to achieve the result without any frills.
- It supports multiple platforms and backends.
- It is user friendly framework which runs on both CPU and GPU.
- Highly scalability of computation.

Matplotlib:

Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. Matplotlib is written in Python and makes use of NumPy, the numerical mathematics extension of Python. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonotTkinter. It can be used in Python and IPython shells, Jupyter notebook and web application servers also.
Matplotlib has a procedural interface named the Pylab, which is designed to resemble MATLAB, a proprietary programming language developed by MathWorks. Matplotlib along with NumPy can be considered as the open source equivalent of MATLAB.

Operating System:

The OS module in Python comes with various functions that enables developers to interact with the Operating system that they are currently working on. In this article we'll be learning mainly to create and delete a directory/folder, rename a directory and even basics of file handling.

Python OS module provides the facility to establish the interaction between the user and the operating system. It offers many useful OS functions that are used to perform OS-based tasks and get related information about operating system.

The OS comes under Python's standard utility modules. This module offers a portable way of using operating system dependent functionality.

CHAPTER 8

8 CONCLUSION AND FUTURE ENHANCEMENT

8.1 CONCLUSION

In conclusion, the utilization of Artificial Neural Networks (ANNs) for cyber attack detection represents a promising and effective approach. The ability of ANNs to analyze complex patterns in real-time data has shown great potential in enhancing the accuracy and speed of cyber threat identification. This method offers a proactive defense mechanism against evolving cyber threats, providing a robust layer of security. While challenges such as model interpretability and training data quality exist, ongoing research and advancements in neural network architectures continue to address these concerns. The integration of ANNs into cyber defense strategies is crucial for staying ahead in the constantly evolving landscape of cyber threats. As we move forward, further refinements and collaborative efforts between academia and industry will play a pivotal role in optimizing the performance and resilience of ANN-based cyber attack detection systems.

8.2 FUTURE ENHANCEMENTS

In future work, the refinement and optimization of artificial neural network architectures for cyber attack detection will be explored, aiming to enhance both accuracy and efficiency. Investigating the integration of advanced anomaly detection techniques and evolving the neural network models to adapt to emerging cyber threats will be a focal point. Additionally, research will focus on developing strategies for real-time threat identification and response, further strengthening the system's proactive defense capabilities. Continuous exploration of novel datasets and the incorporation of federated learning approaches may offer insights into improving the adaptability and robustness of the detection system. Ongoing efforts will be directed towards achieving a comprehensive and resilient cyber defense framework by embracing advancements in artificial intelligence and cybersecurity.

CHAPTER-9

9.CODING AND SCREENSHOTS
9.1 CODING

DATA PREPROCESSING AND DATA CLEANING

```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
df = pd.read_csv('A.csv')
del df['DATETIME']
del df['F_PU9']
del df['S_PU9']
del df['F_PU5']
del df['S_PU5']
del df['F_PU3']
del df['S_PU3']
del df['S_PU1']
df.head()
df.shape
df.size
df.columns
df.isnull()
df = df.dropna()
df['ATT_FLAG'].unique()
df.describe()
df.corr()
df.info()
pd.crosstab(df["L_T1"], df["L_T2"])
```

```
df.groupby(["L_T3","L_T5"]).groups
df["ATT_FLAG"].value_counts()
pd.Categorical(df["S_PU2"]).describe()
df.duplicated()
sum(df.duplicated())
```

DATA VISUALIZATION AND DATA ANALYSIS

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv('A.csv')
del df['DATETIME']
del df['F_PU9']
del df['S_PU9']
del df['F_PU5']
del df['S_PU5']
del df['F_PU3']
del df['S_PU3']
del df['S_PU1']
df.head()
df.columns
plt.figure(figsize=(15,5))
plt.subplot(1,2,1)
plt.hist(df['L_T1'],color='red')
plt.subplot(1,2,2)
plt.hist(df['L_T1'],color='blue')
df.hist(figsize=(15,55),layout=(15,4), color='green')
plt.show()
```

```
df['L_T3'].hist(figsize=(10,5),color='yellow')
sns.lineplot(df['L_T4'], color='brown') # scatter, plot, triplot, stackplot
sns.violinplot(df['L_T5'], color='purple')
df['P_J256'].plot(kind='density')
sns.displot(df['F_PU2'], color='purple')
# barplot, boxenplot, boxplot, countplot, displot, distplot, ecdfplot, histplot,
kdeplot, pointplot, violinplot, stripplot
sns.displot(df['F_PU1'], color='coral') # residplot, scatterplot
fig, ax = plt.subplots(figsize=(20,15))
sns.heatmap(df.corr(),annot = True, fmt='0.2%',cmap = 'autumn',ax=ax)
def plot(df, variable):
    dataframe_pie = df[variable].value_counts()
    ax = dataframe_pie.plot.pie(figsize=(9,9), autopct='%1.2f%%', fontsize = 10)
    ax.set_title(variable + ' \n', fontsize = 10)
    return np.round(dataframe_pie/df.shape[0]*100,2)
plot(df, 'ATT_FLAG')
```

ARTIFICIAL NEURAL NETWORK

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
df = pd.read_csv('A.csv')
del df['DATETIME']
del df['F_PU9']
del df['S_PU9']
del df['F_PU5']
```

```
del df['S_PU5']
del df['F_PU3']
del df['S_PU3']
del df['S_PU1']
df.head()
df=df.dropna()
df.head()
x1 = df.drop(labels='ATT_FLAG', axis=1)
y1 = df.loc[:,'ATT_FLAG']
import imblearn
from imblearn.over_sampling import RandomOverSampler
from collections import Counter

ros =RandomOverSampler(random_state=42)
x,y=ros.fit_resample(x1,y1)
print("OUR DATASET COUNT        : ", Counter(y1))
print("OVER SAMPLING DATA COUNT  : ", Counter(y))
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20,
random_state=42, stratify=y)
print("NUMBER OF TRAIN DATASET    : ", len(x_train))
print("NUMBER OF TEST DATASET     : ", len(x_test))
print("TOTAL NUMBER OF DATASET    : ", len(x_train)+len(x_test))
print("NUMBER OF TRAIN DATASET    : ", len(y_train))
print("NUMBER OF TEST DATASET     : ", len(y_test))
print("TOTAL NUMBER OF DATASET    : ", len(y_train)+len(y_test))
from sklearn.preprocessing import StandardScaler
# Standardize the features
```

```
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
# Create the neural network model
model = keras.Sequential([
    layers.Input(shape=x_train.shape[1]),
    layers.Dense(64, activation='relu'),
    layers.Dense(32, activation='relu'),
    layers.Dense(1, activation='sigmoid')
])

model.summary()
def graph():
    import matplotlib.pyplot as plt
    data=[a]
    alg="ARTIFICIAL NEURAL NETWORKE"
    plt.figure(figsize=(5,5))
    b=plt.bar(alg,data,color=("blue"))
    plt.title("THE ACCURACY SCORE OF ARTIFICIAL NEURAL
NETWORK IS\n\n\n")
    plt.legend(b,data,fontsize=9)
graph()
model.save("MODEL.h5")
```

PYTORCH

```
import pandas as pd
```

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import Dataset, DataLoader
import warnings
warnings.filterwarnings('ignore')
df = pd.read_csv('A.csv')
del df['DATETIME']
del df['F_PU9']
del df['S_PU9']
del df['F_PU5']
del df['S_PU5']
del df['F_PU3']
del df['S_PU3']
del df['S_PU1']
df.head()
df['ATT_FLAG'].value_counts()
df.head()
x1 = df.drop(labels='ATT_FLAG', axis=1).values
y1 = df.loc[:,'ATT_FLAG'].values
# Data Preprocessing
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
features = scaler.fit_transform(x1)
```

```
import imblearn
from imblearn.over_sampling import RandomOverSampler
from collections import Counter
ros =RandomOverSampler(random_state=42)
x,y=ros.fit_resample(x1,y1)
print("OUR DATASET COUNT        : ", Counter(y1))
print("OVER SAMPLING DATA COUNT  : ", Counter(y))
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20,
random_state=42, stratify=y)
print("NUMBER OF TRAIN DATASET    : ", len(x_train))
print("NUMBER OF TEST DATASET     : ", len(x_test))
print("TOTAL NUMBER OF DATASET    : ", len(x_train)+len(x_test))
print("NUMBER OF TRAIN DATASET    : ", len(y_train))
print("NUMBER OF TEST DATASET     : ", len(y_test))
print("TOTAL NUMBER OF DATASET    : ", len(y_train)+len(y_test))
# Step 3: Create DataLoader for batch processing
batch_size = 32
train_dataset = torch.utils.data.TensorDataset(train_features, train_labels)
test_dataset = torch.utils.data.TensorDataset(test_features, test_labels)
train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=batch_size, shuffle=False)

# Step 4: Define the neural network model
class MyModel(nn.Module):
    def __init__(self, input_size, hidden_size, num_classes):
        super(MyModel, self).__init__()
        self.fc1 = nn.Linear(input_size, hidden_size)
```

```
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(hidden_size, num_classes)
    def forward(self, x):
        out = self.fc1(x)
        out = self.relu(out)
        out = self.fc2(out)
        return out
from sklearn.metrics import accuracy_score
# Step 6: Training the model
for epoch in range(num_epochs):
    model.train()
    running_loss = 0.0
    num_batches = 0
    for i, (inputs, labels) in enumerate(train_loader):
        inputs = inputs.to(device)
        labels = labels.to(device)
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item()
        num_batches += 1
    average_loss = running_loss / num_batches
    # Calculate accuracy at the end of each epoch
    model.eval()  # Switch to evaluation mode
    with torch.no_grad():
```

```
        test_predictions = []
        for inputs, _ in test_loader:
            inputs = inputs.to(device)
            outputs = model(inputs)
            _, predicted = torch.max(outputs.data, 1)
            test_predictions.extend(predicted.cpu().numpy())
        # Calculate accuracy
        accuracy = accuracy_score(test_labels, test_predictions)
        epoch_accuracies.append(accuracy)
        epoch_losses.append(average_loss)
        print(f'Epoch [{epoch + 1}/{num_epochs}], Accuracy: {accuracy *
100:.2f}%, Loss: {average_loss:.4f}')
    model.train()  # Switch back to training mode
print('Training finished.')
```

9.2 EXECUTION

Sample-1:

Input Given: 2.4, 4.3, 5.7, 2.88, 7.11, 5.98, 8.5, 9.3, 6.6, 21.7, 8.9, 3.6, 8.5, 9.3, 7.4, 8.3, 12.3, 24.6, 8.5, 9.3, 6.8

Expected Output: The None of the Cyber Attacks detected in this case

Actual Output: The None of the Cyber Attacks detected in this case

Status: Passed

Sample-2:

Input Given: 8.5, 9.2, 3.24, 7.4, 9.11, 6.89, 4.89, 4.56, 2.4, 4.3, 5.7, 2.88, 7.11, 5.98, 8.5, 9.3, 6.6, 21.7, 8.9, 8.3

Expected Output: The None of the Cyber Attacks detected in this case

Actual Output: The None of the Cyber Attacks detected in this case

Status: Passed

Sample-3:

Input Given: 8.5, 9.2, 1, 7.4, 9.11, 1, 4.89, 4.56, 2.4, 1, 5.7, 2.88, 7.11, 5.98, 1, 1, 6.6, 21.7, 1, 8.3

Expected Output: The Cyber Attacks detected in this case

Actual Output: The Cyber Attacks detected in this case

Status: Passed

Sample-4:

Input Given: 8.5, 9.2, 1, 7.4, 9.11, 1, 4.89, 4.56, 1, 4.3, 5.7, 1, 7.11, 5.98, 8.5, 9.3, 6.6, 1, 8.9, 8.3

Expected Output: The Cyber Attacks detected in this case

Actual Output: The Cyber Attacks detected in this case

Status: Passed

9.3 SCREENSHOTS

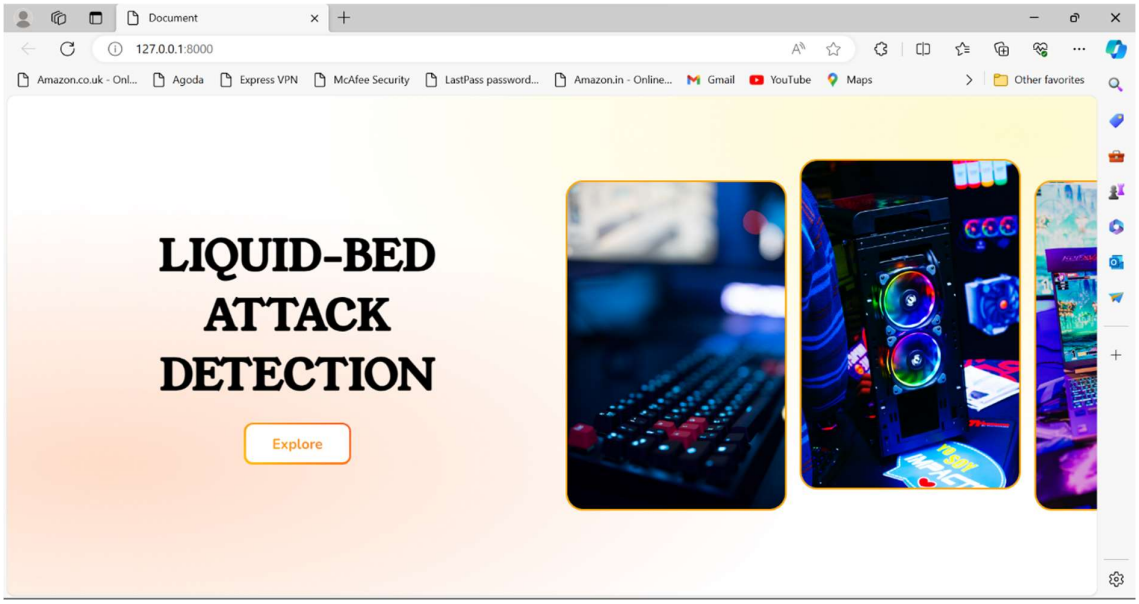

FIG 9.1 LANDING PAGE

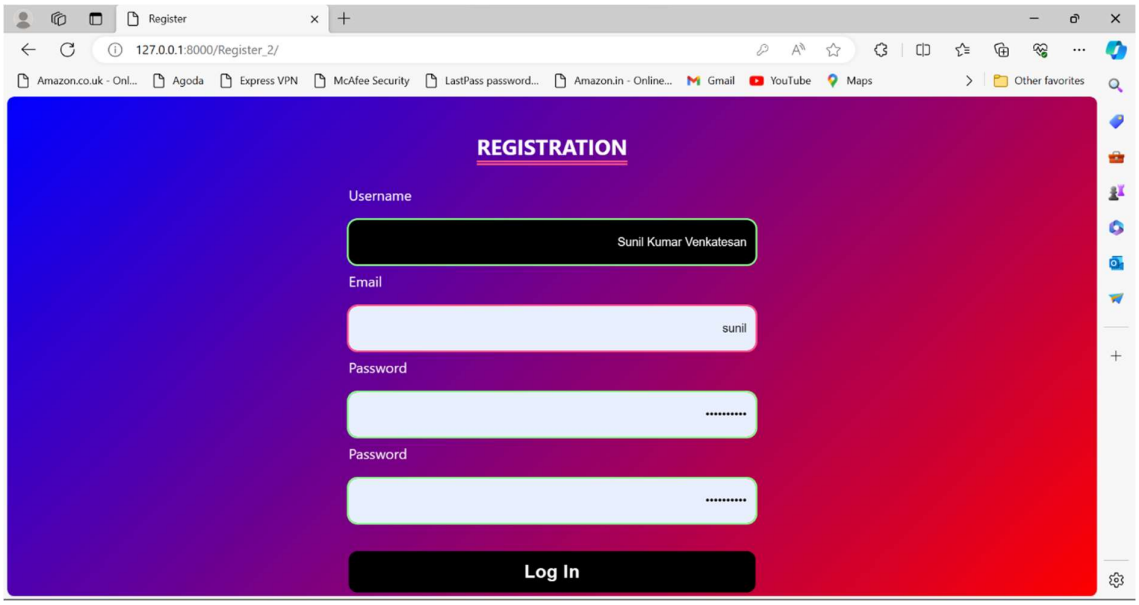

FIG 9.2 REGISTRATION PAGE

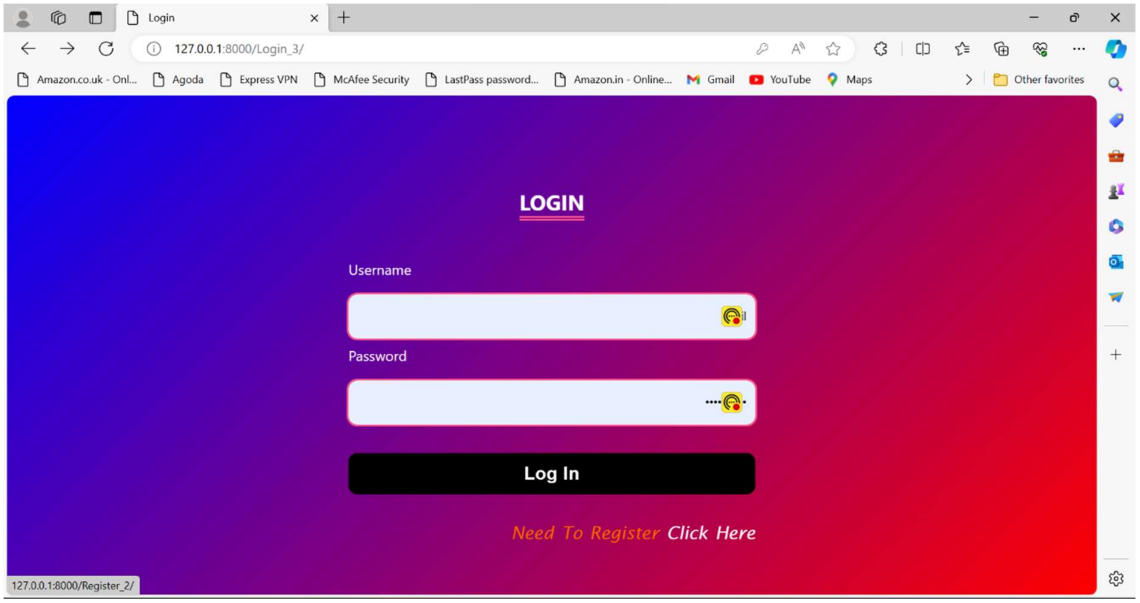

FIG 9.3 LOGIN PAGE

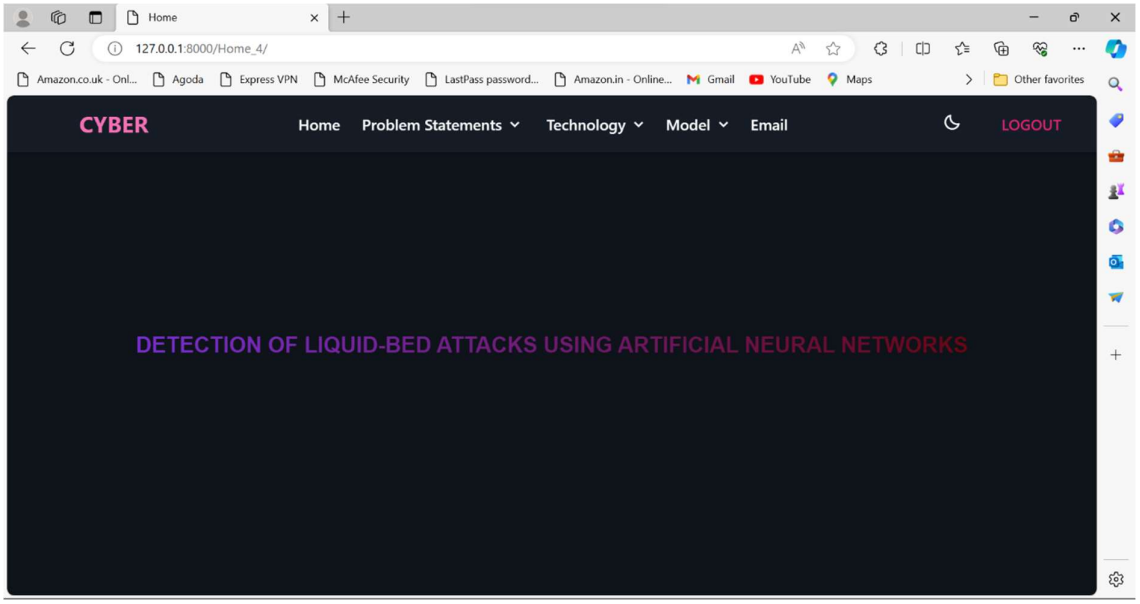

FIG 9.4 HOME PAGE

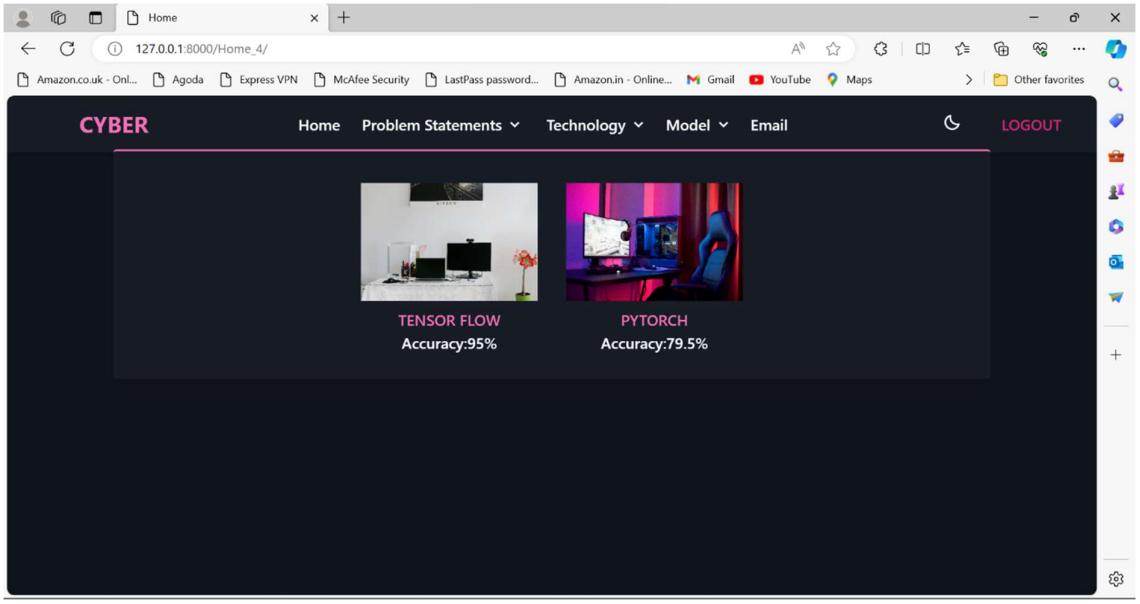

FIG 9.5 TECHNOLOGY STACKS

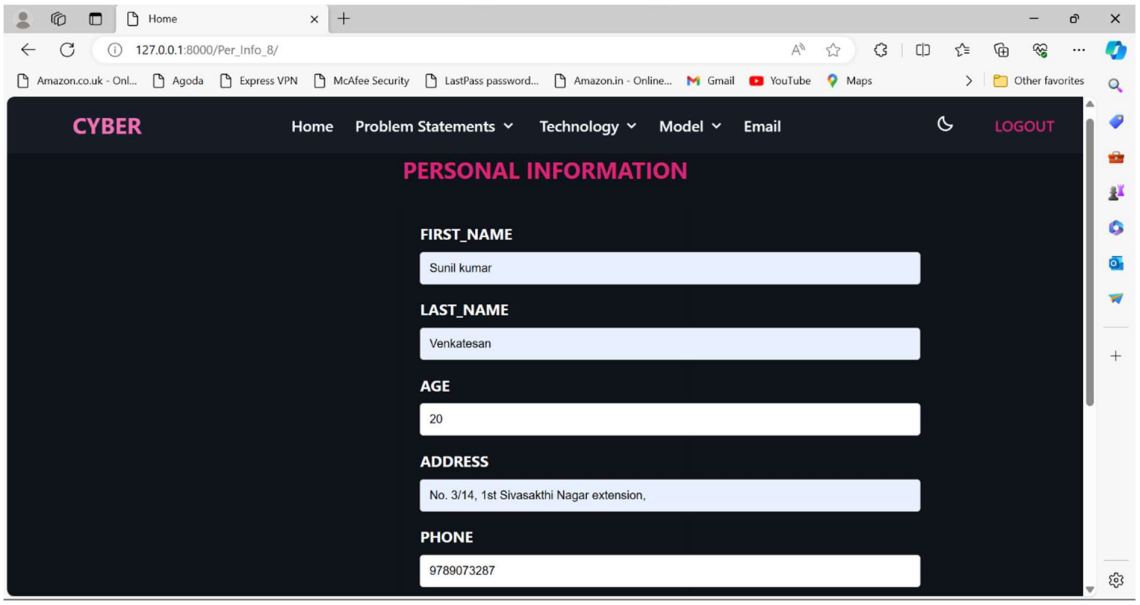

FIG 9.6 PERSONAL INFORMATION

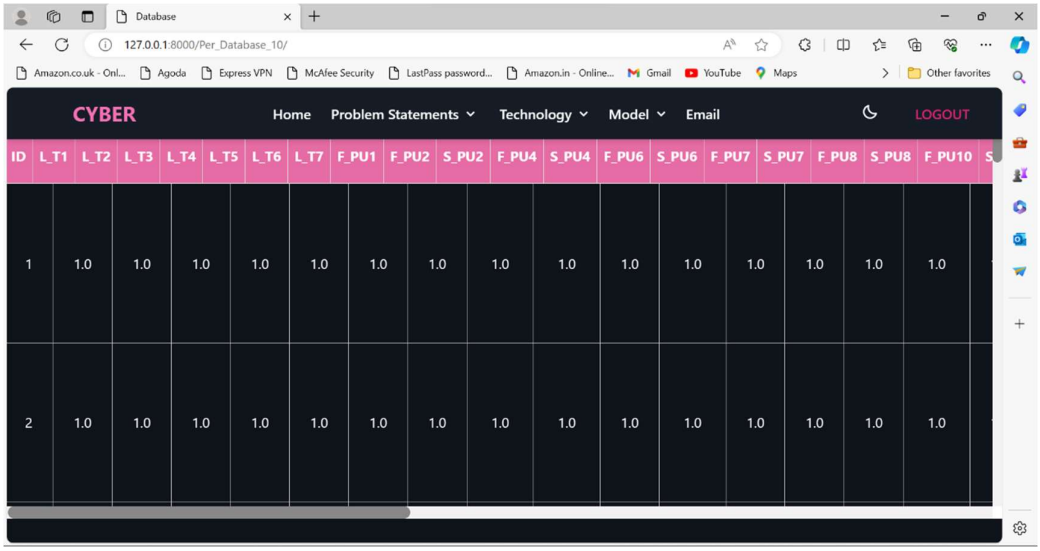

FIG 9.7 DATABASE

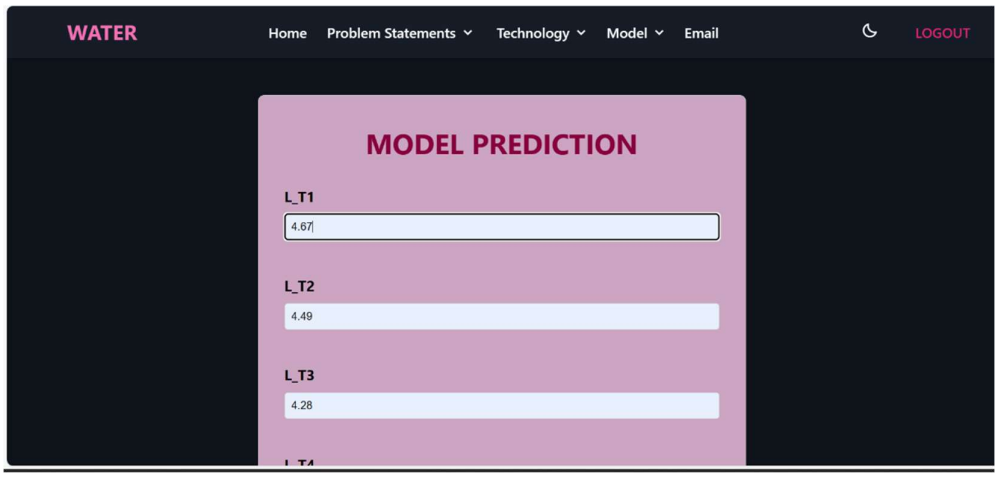

FIG 9.8 MODEL VIEW_1

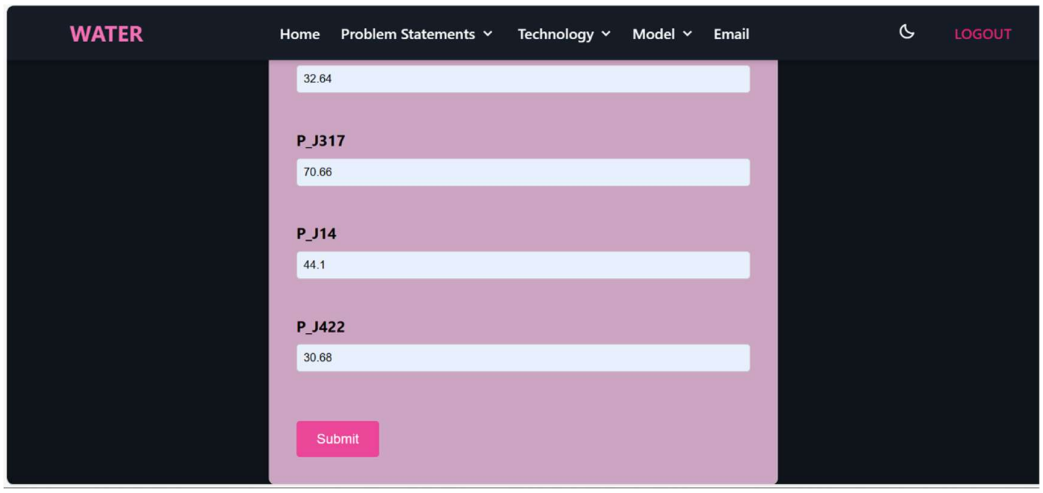

FIG 9.9 MODEL VIEW_2

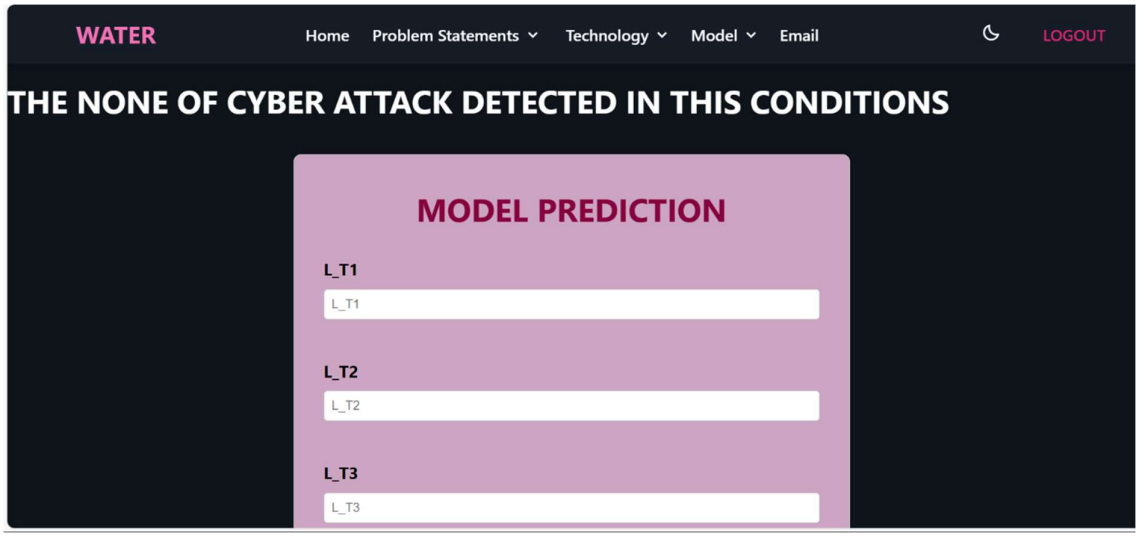

FIG 9.10 MODEL VIEW_3

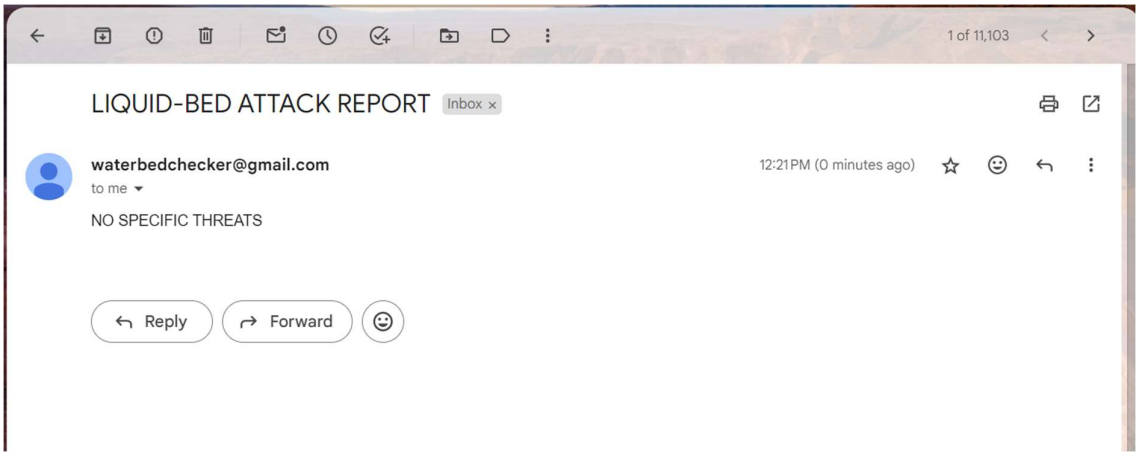

FIG 9.11 EMAIL ALERT

CHAPTER-10

10. REFERENCES

[1]Rajib Ranjan Maiti and Cheah Huei Young. “Mitigating Adversarial Attacks on Data -Driven invariant checkers for Cyber -Physical Systems”. IEEE/CAA Journal of Automatica Sinica 10.3 (2023): 637 -432 .

[2]Shamsun Nahar Edib and Yuzhang Lin. "Cyber Restoration of Power Systems: idea and approach for Resilient Observability". A survery. Computers & security, 78, 101567.

[3]Florian Klaus Kaiser, Uriel Dardik, Aviad Elitzur, Polina Zilberman and Nir Daniel. “Attack Hypotheses Generation Based on Threat Intelligence Knowledge Graph”. A survey. IEEE Transactions on Systems, Man, and Cybernetics: Systems , 11(1), 216 -450.

[4]Nikitha Reddy and Ugendar Reddy. “A Study of cyber security challenges and its emerging trends on Latest Technologies”. IEEE access , 5, 46375 - 14678.

[5]Yuchong Li and Qinghui Liu. “A comprehensive review study of cyber - attacks and cyber security; Emerging trends and recent developments”. IEEE Transactions on Neural Networks and Learning Systems (2021).

[6]Zhang, Jun, et al. "Deep learning based attack detection for cyber -physical system cybersecurity: A survey." IEEE/CAA Journal of Automatica Sinica 9.3 (2021): 377 -391.

[7]Zhang, J., Pan, L., Han, Q. L., Chen, C., Wen, S., & Xiang, Y. (2021). Deep learning based attack detection for cyber - physical system cybersecurity: A survey. IEEE/CAA Journal of Automatica Sinica, 9(3), 377-391.

[8]Boyaci, O., Umunnakwe, A., Sahu, A., Narimani, M. R., Ismail, M., Davis, K. R., & Serpedin, E. (2021). Graph neural networks based detection of stealth false data injection attacks in smart grids. IEEE Systems Journal, 16(2), 2946-2957.

[9]Fan, F. L., Xiong, J., Li, M., & Wang, G. (2021). On interpretability of artificial neural networks: A survey. IEEE Transactions on Radiation and Plasma Medical Sciences, 5(6), 741-760.

[10]Bhamare, D., Zolanvari, M., Erbad, A., Jain, R., Khan, K., & Meskin, N. (2020). Cybersecurity for industrial control systems: A survey. computers & security, 89, 101677.

[11]Corallo, A., Lazoi, M., & Lezzi, M. (2020). Cybersecurity in the context of industry 4.0: A structured classification of critical assets and business impacts. Computers in industry, 114, 103165.

[12]Lin, Y., Tu, Y., & Dou, Z. (2020). An improved neural network pruning technology for automatic modulation classification in edge devices. IEEE Transactions on Vehicular Technology, 69(5), 5703-5706.

[13]Gupta, M., Abdelsalam, M., Khorsandroo, S., & Mittal, S. (2020). Security and privacy in smart farming: Challenges and opportunities. IEEE access, 8, 34564-34584.

[14]Nguyen, T. T., & Reddi, V. J. (2021). Deep reinforcement learning for cyber security. IEEE Transactions on Neural Networks and Learning Systems.

[15]Zubaidi, S. L., Abdulkareem, I. H., Hashim, K. S., AlBugharbee, H., Ridha, H. M., Gharghan, S. K., ... & AlKhaddar, R. (2020). Hybridised artificial neural network model with slime mould algorithm: a novel methodology for prediction of urban stochastic water demand. Water, 12(10), 2692.

[16]Franceschini, S., Ambrosanio, M., Vitale, S., Baselice, F., Gifuni, A., Grassini, G., & Pascazio, V. (2020, September). Hand gesture recognition via radar sensors and convolutional neural networks. In 2020 IEEE Radar Conference (RadarConf20) (pp. 1-5). IEEE.

[17]Novak, M., Xie, H., Dragicevic, T., Wang, F., Rodriguez, J., & Blaabjerg, F. (2020). Optimal cost function parameter design in predictive torque control

(PTC) using artificial neural networks (ANN). IEEE Transactions on Industrial Electronics, 68(8), 7309-7319.
[18]Huang, Qiang, Makoto Yamada, Yuan Tian, Dinesh Singh, and Yi Chang. "Graphlime: Local interpretable model explanations for graph neural networks." IEEE Transactions on Knowledge and Data Engineering (2020).
[19]Lin, H., Wang, C., Hong, Q., & Sun, Y. (2020). A multistable memristor and its application in a neural network. IEEE Transactions on Circuits and Systems II: Express Briefs, 67(12), 3472-3476.
[20]Yohanandhan, R. V., Elavarasan, R. M., Manoharan, P., & Mihet-Popa, L. (2020). Cyber-physical power system (CPPS): A review on modeling, simulation, and analysis with cyber security applications. IEEE Access, 8,